Flexibilité dans la gestion des infrastructures informatiques distribuées

Jérôme Gallard

N° d'ordre : 4337 ANNÉE 2011

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Jérôme Gallard

préparée à l'unité de recherche 6074 – IRISA
Institut de Recherche en Informatique et Systèmes Aléatoires
Informatique - Electronique

Flexibilité dans la gestion des infrastructures informatiques distribuées

Thèse soutenue à Rennes
le 6 mai 2011
devant le jury composé de :

Bertil Folliot
Professeur, Université Paris 6 / Rapporteur

Eddy Caron
Maître de conférences, HDR, ENS Lyon / Rapporteur

Eric Gressier-Soudan
Professeur, CNAM / Examinateur

Adrien Lèbre
Chargé de recherche, École des Mines de Nantes / Examinateur

Gabriel Antoniu
Chargé de recherche, INRIA Rennes – Bretagne Atlantique / Examinateur

Christine Morin
Directrice de Recherche, INRIA Rennes – Bretagne Atlantique / Directeur de thèse

Remerciements

Merci à toi, cher lecteur, de considérer mes travaux et d'être prêt à lire ce document ! Je présente ici mes remerciements qui sont destinés à mes sphères professionnelles et privées.

Tout d'abord, je tiens à remercier les membres de mon jury de thèse pour l'attention qu'ils ont portée à mon travail. Merci à vous, Bertil Folliot et Eddy Caron, d'avoir été rapporteurs de mes travaux. Vos remarques et suggestions ont considérablement amélioré la qualité du travail dans son ensemble. Merci à vous, Eric Gressier-Soudan d'avoir été président du jury et de l'avoir animé de manière très conviviale. Merci à vous, Gabriel Antoniu et Adrien Lèbre, pour vos retours constructifs sur le manuscrit.

Christine, je voudrais également te remercier chaleureusement pour toutes ces années passées sous ta direction et qui ont démarré lors d'un stage d'été d'août 2004. Grâce à toi j'ai pu découvrir le monde de la recherche et évoluer dans un environnement très motivant. La fin de ma thèse n'était pas des plus faciles mais tu as toujours gardé confiance en moi et m'as guidé de ton mieux en toute circonstance. Merci !

Adrien, merci pour ce « coaching » intensif de première année de thèse ;-) et bien sûr d'avoir accepté de travailler avec moi tout au long de ces années. Je suis toujours sidéré par ta capacité à gérer 36 projets en même temps tout en parlant au téléphone et sur IRC. Je voudrais également remercier ta famille qui a été privée de ta présence, de nombreuses soirées et de nombreux week-ends, pour corriger ou terminer un papier ! Prochaine étape pour nous : organisation d'un workshop à la Réunion !

Pierre, merci d'avoir aussi bien géré l'approvisionnement en M&M's et autres confiseries en tout genre dans le bureau pendant ces années. On est bien d'accord pour dire que sans cela, le travail en aurait pâti. En plus de ça, merci pour ta disponibilité : toujours partant pour aider, que ce soit pour un service, une discussion scientifique, une aide technique ou autre. Il y en a qui "Google" pour avoir une info, moi j'ai eu la chance de pouvoir faire du "Pierre".

Merci à tous les membres de l'équipe PARIS/Myriads, Cyril, David, Eugen, Ghislain, Julien, Nikos, Pascal, Stefania, Sylvain, Thomas, Yvon d'avoir fait que ma vie sociale à l'IRISA soit très riche :-) Un gros merci à Maryse pour son aide dans toutes les formalités administratives ! J'ai une pensée particulière pour Françoise.

Une spéciale dédicace pour les "badmintoniens" Pascal, Cyril, Pierre, Ghislain. Une autre spéciale dédicace pour Eugen en rappel de l'incroyable séjour que l'on a passé en Inde ;-)

Merci Oana, Eliana, Archana et Sajith pour les fructueuses collaborations que nous avons pu avoir ensemble !

Un énorme merci à Yaya et Geoffroy pour leur accueil à Oak Ridge. C'était vraiment très bien. Ce fut un séjour très riche aussi bien professionnellement qu'amicalement. Je remercie aussi toute l'équipe qui m'a accueilli à l'ORNL, avec en particulier Stephen et Thomas.

Merci à ma famille et mes amis qui sont venus de loin (et de moins loin) assister à ma soutenance. Je pense en particulier à mes parents, Estelle, Fabrice, Emmanuel (ah, non, Emmanuel n'a pas pu venir, mais le cœur y était :-)).

Merci à Binôme d'être resté mon binôme malgré une 3eme année de thèse qui n'était pas des plus reposantes :-) Mais bon, tu as toujours su être là avec des billets pour le stade

Rennais qui rendaient cette rédaction beaucoup plus agréable :-)

Merci Fati, Denis de m'avoir soutenu sans discontinu et de le faire encore aujourd'hui en me faisant prendre l'air à Julouville et en me concoctant de bons petits plats marocains. Spéciale dédicace pour Denis : merci d'avoir relu mon état de l'art ;-)

Louis, Hema merci pour votre aide et votre soutien indéfectible (merci aussi Hema pour tes Amadeus, éléments culinaires indispensables à la bonne rédaction du manuscrit). Désolé d'avoir raté votre déménagement sous prétexte d'avoir un chapitre à terminer... :-)

Merci Pascal, mon grand frère et parrain, toujours là quand on a besoin de toi avec tes conseils rationnels pas tout le temps agréables à entendre, mais qui se révèlent toujours pertinents ;-) Merci beaucoup Magali également.

Merci Papa, Maman pour m'avoir toujours soutenu dans toutes mes entreprises. Sans votre appui, je n'aurais jamais pu faire cela. Spéciale dédicace pour Maman : merci d'avoir relu toute ma thèse ! ! !

Sont dans mon cœur, Grand-Père, Pépé, Mémé, Tato, Tonton Daniel, Tatie, Maryse.

Je vais terminer en remerciant Sylvie. On s'est rencontré au cours de ma première année de thèse. A ce moment là, elle ne réalisait pas ce que serait « la rédaction ». Et bien maintenant on sait. On a affronté tout ça ensemble : « Merci pour ta patience et ton aide au quotidien (ainsi que d'avoir relu ma thèse !). Taimheuuu. »

Il est maintenant temps pour moi de vous présenter ma vision de la flexibilité dans les infrastructures informatiques distribuées. Pour ce faire, je vous propose dans la suite de ce document un voyage à la *Saline* où on y fera une *Grillade* en admirant les pailles-en-queue (*Tropicbird*) planer haut dans un ciel bleu outre-mer... Oté la Réunion ! Là-dessus, je conclus ces remerciements en vous souhaitant une bonne et agréable lecture.

Table des matières

Table des figures

Introduction

La flexibilité est un élément clé de l'informatique, il suffit de reprendre les fondements de l'informatique posés par Turing pour s'en appercevoir [54]. La flexibilité extrême est celle de la machine de Turing universelle qui permet de simuler le comportement de n'importe quelle autre machine de Turing. En pratique, la flexibilité dans un système informatique revient à pouvoir exécuter une série d'instructions sur des données, ces instructions et ces données n'étant pas « codées » dans le système au moment de sa création, mais « chargées » au moment de son utilisation.

Ce document se concentre sur la flexibilité dans les infrastructures informatiques de type grappe, grille et centrale numérique (informatique en nuage [36], *Cloud Computing*). Ces infrastructures sont distribuées sur un ou plusieurs sites et sont gérées par des gestionnaires de ressources. Ce sont ces gestionnaires de ressources qui apportent la flexibilité requise pour la configuration et l'utilisation de ces infrastructures aussi bien pour les utilisateurs que pour les administrateurs. Prenons l'exemple d'une centrale numérique. Ce type de système propose aux utilisateurs de pouvoir déployer des machines virtuelles dans lesquelles ils ont des droits d'administration privilégiés. Ces machines virtuelles sont exécutées sur les nœuds de la centrale numérique. L'axe de flexibilité que l'on peut dégager ici est que, d'une part les administrateurs d'une centrale numérique définissent leurs politiques de gestion de l'infrastructure, et que d'autre part, les utilisateurs peuvent, sans intervention des administrateurs, définir le nombre de machines virtuelles qu'ils veulent en disposant d'un degré de personnalisation important. Ils ont en effet la possibilité de choisir le type de système d'exploitation et de configurer leur environnement d'exécution. Du point de vue des utilisateurs, ces opérations sont faites à la demande, c'est-à-dire immédiatement (ordre de la dizaine de secondes).

Le domaine de l'informatique est en constante évolution. On peut y remarquer un cercle vertueux : les besoins en service entraînent des avancées technologiques (logiciels et matériels) qui en retour apportent de nouvelles perspectives et mènent vers de nouveaux besoins [20, 27, 35]. Or, la mise en place et la maintenance d'infrastructures informatiques pour répondre à un besoin est complexe et coûteuse en terme de temps, et leur reconfiguration pour les étendre ou répondre à un besoin différent peut l'être encore plus. Dans ce contexte, la motivation d'utiliser des ressources le plus efficacement possible avec des systèmes extrêmement optimisés pour un besoin particulier, peut laisser place dans certains cas à celle d'avoir des systèmes flexibles pouvant se configurer et se reconfigurer pour s'adapter à une large majorité de cas d'utilisation, même si pour cela il faut faire des concessions sur les performances.

Cette thèse s'intéresse à *l'identification, la conception et la mise en œuvre d'axes de flexibilité pour la gestion et l'utilisation d'infrastructures informatiques distribuées.* Autrement dit, *dans quelle mesure est-il possible de concevoir et mettre en oeuvre un système générique pour gérer des infrastructures distribuées regroupant les axes de flexibilité préalablement identifiés ?* Pour y répondre nous proposons de diviser notre étude en

différentes parties, chacune correspondant à une contribution présentée dans ce document.

1 Démarche et contributions

Cette thèse se place dans le contexte du projet XtreemOS visant à concevoir et mettre en œuvre un système d'exploitation pour grille. Ce projet qui regroupe 19 partenaires académiques et industriels en Europe et en Chine, a pour objectif d'étendre le système Linux pour offrir un support natif aux organisations virtuelles. Le but recherché est d'offrir aux utilisateurs une interface permettant d'utiliser simplement et efficacement les ressources de la grille, et aux administrateurs, l'interface et les outils pour collaborer entre différents domaines d'administration, assurer la sécurité et le bon fonctionnement des ressources qu'ils mettent en partage.

Les systèmes de virtualisation dont les fondements datent des années 1970 sont de nos jours déclinés sous différentes formes de mise en œuvre. Ces systèmes de virtualisation découplent la vue logique des ressources qu'a l'utilisateur, de celle matérielle sous-jacente. C'est au cours des années 2000 que les systèmes de virtualisation prennent leur essor et les premières centrales numériques voient le jour à partir des années 2005. Ces systèmes permettant de créer des axes de flexibilité, c'est tout naturellement que nous avons démarré notre étude par ces derniers. Les centrales numériques peuvent être décrites comme des infrastructures informatiques mono-site (de type grappe) dont les nœuds sont gérés par un système de virtualisation. Dans ce contexte, nous nous sommes interrogés sur l'utilisation de système de virtualisation dans des infrastructures informatiques multi-sites (de type grille). Quels sont les avantages et inconvénients d'utiliser des systèmes de virtualisation à l'échelle d'une grille du point de vue de l'utilisateur, de son application et des administrateurs de la grille ? Quels sont les problèmes engendrés et quelles sont les solutions possibles ? Nous étudions et proposons des éléments de réponse à ces questions en concevant et mettant en œuvre Saline, un système gestion de collections de machines virtuelles sur grille pour l'exécution de tâches interruptibles.

1.1 Saline, gestion de collections de machines virtuelles sur grille pour l'exécution des tâches interruptibles

Les mécanismes de virtualisation permettent d'encapsuler les applications dans les machines virtuelles et de leur faire bénéficier de la flexibilité offerte par ces dernières. Cependant, alors que l'utilisation et la gestion d'une machine virtuelle sur un nœud peut être simple, la gestion de collections de machines virtuelles sur des nœuds répartis sur différents sites géographiques est complexe. Les principales raisons de cette complexité sont liées à la gestion des adresses MAC/IP ainsi qu'à la gestion des images des machines virtuelles.

Saline est un système conçu et mis en œuvre pour déployer et gérer des collections de machines virtuelles (un ensemble de machines virtuelles utilisé pour l'exécution d'une même tâche, parallèle ou distribuée) sur des infrastructures distribuées [82, 84, 85, 86]. Afin d'étudier un cas concret, nous avons développé Saline dans le but de gérer les tâches interruptibles (*best-effort jobs*) sur la plate-forme d'expérimentation à grande échelle Grid'5000 [53]. Les tâches interruptibles sont des applications de basse priorité qui sont démarrées lorsque des ressources sont inutilisées et peuvent être arrêtées de manière brutale, sans sauvegarde préalable, en fonction des tâches de plus haute priorité : une tâche interruptible peut ne pas être exécutée de manière efficace. C'est sur ce cas d'école que Saline se concentre. Pour ce faire, Saline encapsule les tâches interruptibles dans des ma-

chines virtuelles qu'il gère sur la grille. Saline propose des mécanismes de configuration transparente du réseau des machines virtuelles (adresses MAC et IP) : la configuration du réseau permet d'isoler des collections de machines virtuelles entre différents sous-réseaux et assure qu'une collection de machines virtuelles peut être déplacée d'un site à un autre de la grille sans engendrer de conflit d'adresse. Cela se fait de manière transparente pour l'application exécutée dans la collection considérée.

De plus Saline effectue des sauvegardes périodiques des collections de machines virtuelles afin de les restaurer le cas échéant. Lors d'une préemption des ressources, la tâche interruptible est arrêtée. Dans ce cas, si elle ne dispose pas de mécanismes internes lui permettant d'être restaurée à l'endroit où elle a été arrêtée (par exemple une application disposant de mécanismes de sauvegarde et restauration de points de reprise ou une application parallèle de type « sac de tâches » – *bag of tasks* –), son redémarrage se fait depuis le début, tous les calculs préalablement effectués sont perdus. L'encapsulation d'une tâche interruptible dans une machine virtuelle peut bénéficier des propriétés des machines virtuelles : une collection de machines virtuelles peut être arrêtée puis restaurée de manière transparente pour les applications exécutées. Ces fonctionnalités élargissent les tâches interruptibles à une plus large gamme de type d'application.

Enfin Saline est conçu pour être installé et utilisé sur une large gamme d'infrastructures informatiques sans nécessiter de modification de celles-ci. Saline s'interface avec le gestionnaire des ressources de l'infrastructure pour réserver les ressources matérielles et ensuite déployer et gérer les collections de machines virtuelles.

Ouvertures Saline est un système qui apporte de la flexibilité aux applications en gérant des collections de machines virtuelles dans des infrastructures informatiques distribuées de type grille. Nous nous sommes intéressés à étudier le rapprochement entre les mécanismes que nous avons conçus et mis en œuvre dans Saline et d'autres systèmes existants pour la gestion des grappes, grilles et centrales numériques.

En effet, comme nous l'étudions dans ce document, les systèmes de gestion de ressources pour grappe, grille et centrale numérique proposent des axes de flexibilité différents principalement dûs aux infrastructures et aux communautés d'utilisateurs visées qui diffèrent selon les cas. Se pose alors une question plus générale : quels seraient les avantages de faire un rapprochement entre les grappes, les grilles et les centrales numériques ? Un système pouvant gérer à la fois les grappes, les grilles et les centrales numériques est-il concevable et possible à mettre en œuvre ? Nous étudions et proposons des éléments de réponse à ces questions en concevant et mettant en œuvre Grillade, un système flexible de gestion des ressources pour grille et centrale numérique.

1.2 Grillade, vers un système de gestion pour grilles et centrales numériques

Nous continuons notre quête de flexibilité dans les infrastructures informatiques en étudiant plus en détail les axes de flexibilité offerts par les systèmes existants pour la gestion des grappes, des grilles et des centrales numériques [63, 128, 130]. De cette étude deux axes de flexibilité ont retenu notre attention : la possibilité d'avoir un système de gestion des collections de machines virtuelles sur une infrastructure matérielle répartie sur plusieurs sites et la possibilité d'étendre une infrastructure matérielle avec des ressources fournies par une centrale numérique. Ces axes de flexibilité ont été conçus et mis en œuvre dans le système Grillade, un système fondé sur le système d'exploitation pour grille XtreemOS. Grillade est composé de deux sous-systèmes, chacun des sous-systèmes traitant d'un axe de

flexibilité particulier. Nous nommons Grillade-CN (*Grillade-Centrale_Numérique*) le sous-système de Grillade traitant de l'axe de flexibilité orienté vers la gestion des collections de machines virtuelles à la manière des centrales numériques. Nous nommons Grillade-Ext (*Grillade-Extension*) le sous-système de Grillade traitant de l'axe de flexibilité orienté vers l'extension de l'infrastructure.

1.2.1 Grillade-CN

L'étude des axes de flexibilité présents dans les systèmes de gestion de grappes, grilles et centrales numériques nous montre trois aspects. Premièrement, une grappe gérée par un système à image unique permet l'agrégation de nœuds pour offrir la vision d'un nœud SMP bénéficiant des capacités physiques de l'ensemble des nœuds agrégés. Deuxièmement, une grille, grâce au service de gestion des organisations virtuelles (VO) permet à différents instituts (domaines d'administration) de partager des ressources situées sur différents sites. Troisièmement, une centrale numérique offre aux utilisateurs des machines virtuelles dont il est possible de choisir le système d'exploitation et de personnaliser l'environnement d'exécution.

Grillade-CN est un système pouvant gérer des collections de machines virtuelles à la manière des centrales numériques [83] sur une infrastructure de type grille. Dans un système de grille comme XtreemOS, les applications bénéficient de services de grille comme les mécanismes de collaboration (VO), des mécanismes de fédération de données (XtreemFS), des mécanismes de gestion des grappes (LinuxSSI) pour agréger la mémoire de différents nœuds d'une grappe.

Grillade-CN a pour objectif d'apporter une nouvelle granularité dans la gestion des tâches : en plus de la gestion des applications, le système est capable de gérer des machines virtuelles. Pour ce faire, les collections de machines virtuelles bénéficient des mécanismes élaborés dans Saline, mais également des fonctionnalités de grille offertes par XtreemOS comme par exemple, les VO, XtreemFS et LinuxSSI [87, 91].

1.2.2 Grillade-Ext

Les centrales numériques sont des systèmes « agiles » dans le sens où, le temps de création et de configuration d'une machine virtuelle demandé par un utilisateur est très bref (de l'ordre de la dizaine de secondes). Cette agilité permet d'avoir *en permanence* des ressources de calcul disponibles et prêtes à l'emploi. Cela a orienté notre réflexion vers l'axe de flexibilité suivant : extension d'une infrastructure informatique avec des ressources fournies par une centrale numérique.

Grillade-Ext est un système conçu pour étendre une infrastructure de grille de manière transparente sur des ressources fournies par une centrale numérique [164]. Grillade est capable, sur demande, de piloter la création et la configuration de machines virtuelles fournies par une centrale numérique afin de les utiliser de manière transparente au sein d'une grille existante. À la fin de leur configuration, ces machines virtuelles deviennent partie intégrante de Grillade et peuvent être utilisées comme n'importe quel autre nœud de la grille.

Ce cas d'utilisation peut être utile lors d'un afflux de demande en ressources : si aucune ressource n'est disponible, plutôt que de refuser ou de faire attendre les nouvelles tâches, il est possible de les exécuter sur des ressources fournies par une centrale numérique, et cela, de manière transparente pour les applications concernées.

Ouvertures Les deux premières contributions, Saline et Grillade, apportent de la flexibilité dans la gestion et l'utilisation des infrastructures informatiques distribuées sur plusieurs sites. Cette flexibilité est atteinte grâce à l'utilisation et à la combinaison de mécanismes de virtualisation, d'agrégation de nœuds (LinuxSSI), de système de fichiers distribués (XtreemFS), de partage de ressources entre des sites appartenant à des domaines d'administration différents (VO).

Si on appelle ces différents systèmes des outils, on se rend compte que plus on a d'outils (outils que l'on peut utiliser et combiner sur une infrastructure), plus la flexibilité offerte sur une infrastructure informatique est grande. La question qui se pose alors est de savoir s'il existe un moyen de formaliser ce qu'est « la flexibilité » dans les systèmes informatiques. À partir de cette formalisation, est-il possible de concevoir et mettre en œuvre un système capable de combiner ces outils, afin de les utiliser comme des Lego® et de pouvoir configurer et reconfigurer une infrastructure informatique à la demande ? Nous étudions et proposons des éléments de réponse à ces questions en concevant Tropicbird, un méta-système de gestion flexible des infrastructures informatiques, fondé sur le concept de plate-forme virtuelle.

1.3 Tropicbird, vers un méta-système de gestion des ressources pour plus de flexibilité dans les infrastructures informatiques

Le but recherché est de pouvoir dissocier complètement la vision « réelle » des ressources distribuées et la vision « logique » des ressources proposées à l'application [90, 91, 92, 169]. C'est ainsi que nous travaillons à la proposition d'un formalisme pour la notion de flexibilité. Ce formalisme s'appuie sur un raffinement de la théorie de Goldberg qui a été formulée au cours des années 1970. Ce raffinement, en plus de permettre de classer les différents types de système de virtualisation, permet d'étendre la théorie initiale de Goldberg à d'autres systèmes informatiques, comme par exemple des systèmes d'exploitation [88, 89, 90]. Ainsi, le formalisme proposé permet de décrire une vue logique des ressources indépendamment de l'infrastructure informatique sous-jacente. Cette vue logique que nous appelons plate-forme virtuelle, est le concept clé de notre vision de la flexibilité : une plate-forme virtuelle est décrite et mise en œuvre par le système sur des ressources matérielles.

Considérons les quatre ensembles suivants : les demandes des utilisateurs, les politiques des administrateurs, l'infrastructure matérielle et un ensemble d'outils (gestionnaires de ressources). Le but ultime recherché est de pouvoir mettre en relation ces quatre ensembles : les administrateurs définissant leurs politiques, Tropicbird doit selon les demandes des utilisateurs et en fonction des outils et de l'infrastructure informatique disponibles, être capable de « construire » la plate-forme virtuelle, c'est-à-dire, configurer l'infrastructure pour fournir à l'utilisateur une mise en œuvre de la vue logique des ressources demandées.

2 Organisation du document

Ce document présente notre vision de la flexibilité dans les infrastructures informatiques distribuées.

Le chapitre 1 décrit les infrastructures informatiques considérées dans ce document : les grappes, les grilles, les systèmes de virtualisation et les centrales numériques. Ce chapitre introduit les concepts clés nécessaires à la compréhension du problème que l'on traite : la flexibilité dans les systèmes informatiques.

Le chapitre 2 présente Saline, un système de gestion des machines virtuelles à l'échelle de la grille. Saline est conçu pour la gestion de tâches interruptibles, c'est-à-dire, des tâches qui peuvent être arrêtées de manière brutale. En encapsulant les tâches dans des machines virtuelles, Saline bénéficie des fonctionnalités de sauvegarde et restauration de ces dernières, pour les restaurer depuis leur dernière sauvegarde et non depuis le début de leur exécution si elles ont été interrompues.

Le chapitre 3 présente Grillade, une extension du système d'exploitation pour grille XtreemOS permettant de gérer nativement les machines virtuelles. Ce chapitre décrit deux types de mécanismes : des mécanismes permettant de gérer nativement des machines virtuelles à la manière d'une centrale numérique (Grillade-CN) et des mécanismes d'extension permettant à Grillade d'utiliser des ressources fournies par une centrale numérique externe (Grillade-Ext). Ce chapitre démontre qu'il est possible, grâce à différents outils (machines virtuelles, système à image unique, configuration du réseau), de s'abstraire des ressources matérielles et ainsi de différencier la vue qu'a l'utilisateur des ressources, de l'infrastructure informatique.

Le chapitre 4 présente notre raffinement de la théorie de Goldberg qui cible à l'origine les systèmes de virtualisation, en l'étendant à une plus large gamme de systèmes informatiques. Ainsi, nous proposons notre vision de la flexibilité dans les systèmes informatiques qui est fondée sur le concept de plate-forme virtuelle. Une plate-forme virtuelle représente la vision des ressources nécessaires à l'exécution d'une application. Une plate-forme virtuelle peut être « construite » sur des ressources matérielles grâce au formalisme que nous proposons dans le raffinement de la théorie de Goldberg, sans faire d'hypothèse sur ces ressources. Tropicbird est notre prototype de construction de plate-forme virtuelle fondé sur notre raffinement de la théorie de Goldberg.

Enfin nous concluons par une synthèse de nos contributions et une présentation des perspectives de recherche ouvertes par nos travaux.

Chapitre 1

Les infrastructures informatiques distribuées et la virtualisation

Les systèmes informatiques sont des systèmes capables de traiter des informations de manière automatisée. Ces systèmes existent sous différentes formes depuis des millénaires. L'invention des abaques ou bouliers permettant de faire des calculs mathématiques [45] datant du 3^{e} siècle avant Jésus-Christ en est un exemple. L'invention du transistor en 1947 fait apparaître de nouveaux types de systèmes informatiques dont les ordinateurs et les réseaux d'ordinateurs. Ces systèmes informatiques doivent être contrôlés par des systèmes dont le but principal est de réaliser l'interface avec les utilisateurs et gérer le matériel pour l'exécution des applications. Ces sytèmes informatiques qui étaient à l'origine principalement parallèles (super-ordinateurs) ont vu, grâce aux progrès techniques réalisés dans les réseaux d'ordinateurs, leur évolution s'orienter également dans les systèmes informatiques distribués. C'est ainsi que sont apparues les grappes et les grilles de calcul.

De plus, ces dernières années, les systèmes de virtualisation ont connu un regain d'intérêt et offrent de nouvelles possibilités quant à l'utilisation des ordinateurs avec par exemple, l'isolation des processus, la portabilité des applications, la consolidation des serveurs. Cet essor des systèmes de virtualisation a permis leur utilisation de manière distribuée : c'est ainsi qu'apparaissent les premières centrales numériques (*Cloud Computing*).

Ce chapitre présente un historique de l'évolution des systèmes informatiques en commençant par les ordinateurs et les réseaux d'ordinateurs. Ce premier point nous amène à présenter l'état de l'art des systèmes informatiques de type grappe et grille. Ensuite nous présentons les systèmes de virtualisation ainsi que l'utilisation de ce type de système de manière distribuée comme cela est le cas dans les centrales numériques (*Clouds*). Enfin, cette thèse traitant de la flexibilité dans la gestion des infrastructures informatiques distribuées, nous concluons ce chapitre par une discussion sur la capacité qu'ont ces systèmes à pouvoir être personnalisés à la demande par leurs administrateurs et utilisateurs.

1.1 Gestion des ressources informatiques

Les systèmes informatiques que nous étudions dans cette section sont : l'ordinateur, les grappes, les grilles. Ces systèmes matériels sont gérés par des systèmes logiciels pour accéder aux ressources et les partager (partage des ressources entre plusieurs utilisateurs).

1.1.1 Un ordinateur

Selon le Journal Officiel de la République Française [81], un ordinateur est un *équipement informatique de traitement automatique de données comprenant les organes nécessaires à son fonctionnement autonome* : le traitement automatique des données se fait selon des instructions exécutées par les unités de traitement qui constituent les organes de l'ordinateur. Ces unités de traitement sont gérées traditionnellement par un système d'exploitation.

Dans ce document, on appelle processus une instance d'un programme exécutée par le processeur. On appelle une tâche l'ensemble des processus d'une même application. De plus, dans la littérature, les termes « nœud », « machine physique » ou encore « ressource physique » pourront être utilisés à la place « d'ordinateur ». Dans ce document, on considère que ces termes ont la même signification et désignent la machine physique capable d'effectuer les calculs.

1.1.1.1 La vision matérielle

Historiquement, c'est Alan Turing qui a décrit en 1936 [166] un modèle de fonctionnement des appareils mécaniques de calcul. Un tel ordinateur, appelé machine de Turing, se compose d'un tableau infini de symboles finis (les données), d'une table d'action (les instructions), d'un registre d'état (l'état courant de la machine) et d'une tête de lecture/écriture (pour lire et écrire les données). Grâce à son modèle, Alan Turing montre qu'en codant la table d'action dans le tableau infini de symboles, il est possible à une machine de Turing de simuler le comportement de n'importe quelle autre machine de Turing. On appelle cette machine « la machine de Turing universelle ».

C'est à partir des travaux de Turing que John von Neumann a proposé dans les années 1940 une architecture dite de *von Neumann* [174] mettant en œuvre la machine universelle de Turing : une structure de stockage permet à une unité centrale de traitement de lire ses instructions ainsi que les données sur lesquelles elle doit agir. De plus, une unité d'entrée/sortie permet à l'ordinateur de « communiquer » avec l'extérieur. La figure 1.1 présente l'architecture simplifiée proposée par von Neumann et largement utilisée de nos jours.

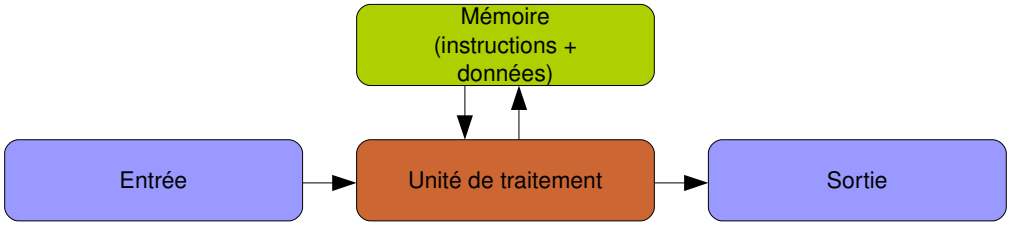


FIGURE 1.1 – Architecture simplifiée de von Neumann

La structure de stockage Avec cette architecture, l'ordinateur dispose de la même structure de stockage pour les instructions, pour les données à traiter et pour les données générées par le traitement des données. C'est ce que l'on appelle la *mémoire vive*. Cette mémoire vive d'accès rapide ne stocke les informations que lorsque l'ordinateur est électriquement alimenté. Par opposition, la *mémoire morte*, à accès lent garde les informations même si l'appareil est électriquement déconnecté.

L'unité centrale de traitement L'unité centrale de traitement (CPU, *Central Processing Unit*), autrement dit, un processeur, est capable d'exécuter les programmes informatiques en lisant les données contenues dans la mémoire vive et en y écrivant les données, résultats du traitement (ces données sont initialement chargées à partir de la mémoire morte, et stockées à la fin du traitement des données). De nos jours, les unités centrales de traitement sont composées de deux modes d'exécution des instructions : un mode privilégié et un mode non-privilégié [150]. Ainsi, les instructions non privilégiées peuvent être exécutées directement par l'application. Concernant les instructions privilégiées, une exception est générée par le système d'exploitation afin de pouvoir passer le processeur en mode « privilégié » et de lui permettre d'exécuter l'instruction. Enfin, il existe un type d'instruction dit « sensible », ce sont des instructions qui ne génèrent pas d'exception mais dont l'action peut interférer sur l'état de l'hyperviseur ou du système hôte.

Dans ce document on ne traitera que des processeurs dont le jeu d'instructions (ISA, *Instruction Set Architecture*) est x86. Le premier processeur fondé sur ce jeu d'instruction est l'*Intel 8086* datant de 1978. Même si depuis lors, de nombreuses modifications et améliorations ont été apportées à l'architecture originale, ce jeu d'instruction est toujours utilisé. Les processeurs de ce type possèdent 17 instructions sensibles.

L'unité d'entrée/sortie Une unité d'entrée/sortie permet à l'unité centrale de « communiquer » avec les autres périphériques de l'ordinateur mais aussi de communiquer avec des périphériques extérieurs à l'ordinateur.

Avec la réduction des coûts et de l'encombrement des ordinateurs, les premiers ordinateurs personnels pour le grand public apparaissent au début des années 1980. Parmi les premiers ordinateurs personnels connus, on peut citer l'Apple II et l'IBM PC.

1.1.1.2 Le système d'exploitation

Les systèmes d'exploitation (OS, *Operating System*) sont des systèmes permettant de gérer les ordinateurs en partageant ses capacités de calcul entre les applications et les utilisateurs. Ils offrent des interfaces aux applications ainsi qu'aux utilisateurs.

L'interface entre les ressources, les utilisateurs et les programmes Un système d'exploitation est un ensemble de programmes qui sert d'interface entre les ressources matérielles (par exemple, le processeur, la mémoire) de l'ordinateur et les programmes des utilisateurs à exécuter. Son rôle est de proposer une vue logique des unités constituant l'ordinateur : par exemple l'entité logique exécutée par un processeur est un processus, l'entité logique gérée par la mémoire est le segment mémoire.

Interaction Homme / machine Les premières interfaces étaient réalisées par carte perforée, puis sont venus les lignes de commandes qui ont été complétées par l'interface graphique. Linux, Windows ou Mac OS X sont des exemples de système d'exploitation pour ordinateur.

Le partage des ressources L'évolution des systèmes d'exploitation est liée à celle du matériel constituant les ordinateurs et aux usages. Les systèmes d'exploitation dédiés à un seul utilisateur à la fois (mono-utilisateur) et une seule application à la fois (mono-processus) ont laissé place à des systèmes d'exploitation multi-utilisateurs et multi-processus : en plus de l'abstraction du matériel, le rôle de l'OS est de gérer le partage des ressources entre les différents programmes des multiples utilisateurs.

Différentes politiques de partage des ressources peuvent être exécutées. Par exemple, la politique de temps partagé entre les processus est une politique interactive qui permet aux utilisateurs d'interagir avec les processus au fur et à mesure de leur exécution. Par opposition, la politique d'exécution différée est une politique de partage des ressources non-interactive, les processus s'exécutant de manière autonome et différée par rapport au moment de leur soumission.

La protection de l'exécution des programmes Le système d'exploitation assure la protection des programmes en étant le seul autorisé à exécuter des instructions privilégiées du processeur. Historiquement, c'est le système d'exploitation Multics [156] qui a été le premier à introduire la séparation des privilèges dans les systèmes d'exploitation (instructions privilégiées et non-privilégiées) à travers 8 niveaux de privilège (dans la littérature on trouve également l'expression « anneaux de privilèges » – *rings* –). Ce principe a été repris dans les systèmes informatiques actuels avec la définition de 4 niveaux de privilège, mais en pratique, 2 niveaux sont utilisés (le plus privilégié pour le système d'exploitation et le moins-privilégié pour les applications des utilisateurs). Ceci est illustré sur la figure 1.2 par une architecture à 4 niveaux de privilège.

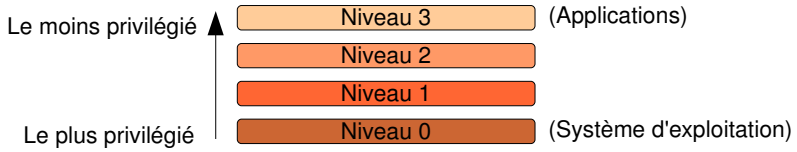


FIGURE 1.2 – Niveaux de privilège

1.1.2 Les réseaux

Les premiers réseaux apparaissent dès 1960. Ces premiers réseaux étaient des réseaux locaux (connexion à l'échelle d'une salle ou de salles voisines). Les premiers réseaux de réseaux (réseaux qui interconnectent d'autres réseaux) apparaissent dès 1970 aux États-Unis sous le nom d'ARPANET [149]. Ils permettent d'interconnecter des nœuds présents sur des lieux géographiques différents. Très vite, l'évolution des technologies des réseaux permet de connecter de plus en plus de nœuds à des distances de plus en plus grandes et avec des débits de communication toujours plus élevés. C'est grâce à ces réseaux qu'un peu plus tard, dans les années 1990, les grappes et les grilles voient le jour.

Une particularité des réseaux est que le nombre ainsi que la disponibilité des nœuds varient au cours du temps : un nœud peut être arrêté en raison d'une défaillance ou d'une maintenance et cela ne doit pas affecter le fonctionnement des autres nœuds. Cette particularité rend la gestion des réseaux de machines d'autant plus complexe : la probabilité de défaillance d'une ressource ou d'un lien réseau augmente avec le nombre de nœuds et le nombre de sites.

Les réseaux peuvent être filaires ou non et sont composés de grandes familles de types de connexion (dans le cas des réseaux filaires, on peut citer par exemple les réseaux Ethernet ou Token Ring – dans le cas des réseaux non-filaires, on peut citer par exemple les réseaux Wi-Fi ou Bluetooth). Ce document ne considère que des réseaux filaires pour les réseaux locaux et étendus.

1.1.2.1 Les réseaux locaux

Les réseaux locaux (LAN, *Local Area Network*) permettent la connexion de nœuds au sein d'une salle ou d'un bâtiment que l'on désigne sous le terme de site (la distance entre deux nœuds les plus éloignés au sein d'un site est de l'ordre du kilomètre). Ce type de réseau peut être décliné en réseau local à très haute performance, mais sur de plus courtes distances (généralement l'ordre de distance est d'une centaine de mètres, tous les nœuds se trouvant dans une même salle par exemple). On pourra également parler de SAN (*System Area Network*) afin de désigner des LAN orientés vers la haute performance. Alors que *Ethernet* ou *Token Ring* étaient des technologies très utilisées dans les années 1980, *Gigabit Ethernet* [9] est sans doute la technologie la plus utilisée de nos jours. D'autres technologies comme *Myrinet* [19, 52] ou *InfiniBand*^{TM} [14, 142] sont également répandues dans le domaine des communications à très haute performance.

1.1.2.2 Les réseaux étendus

Les réseaux étendus (WAN, *Wide Area Network*) permettent la connexion de différents sites géographiquement très éloignés (distance de l'ordre de centaines ou de milliers de kilomètres). Cependant cela se fait généralement au détriment des performances (en terme de latence par exemple).

La figure 1.3 présente sur un même schéma les réseaux locaux ainsi que les réseaux étendus.

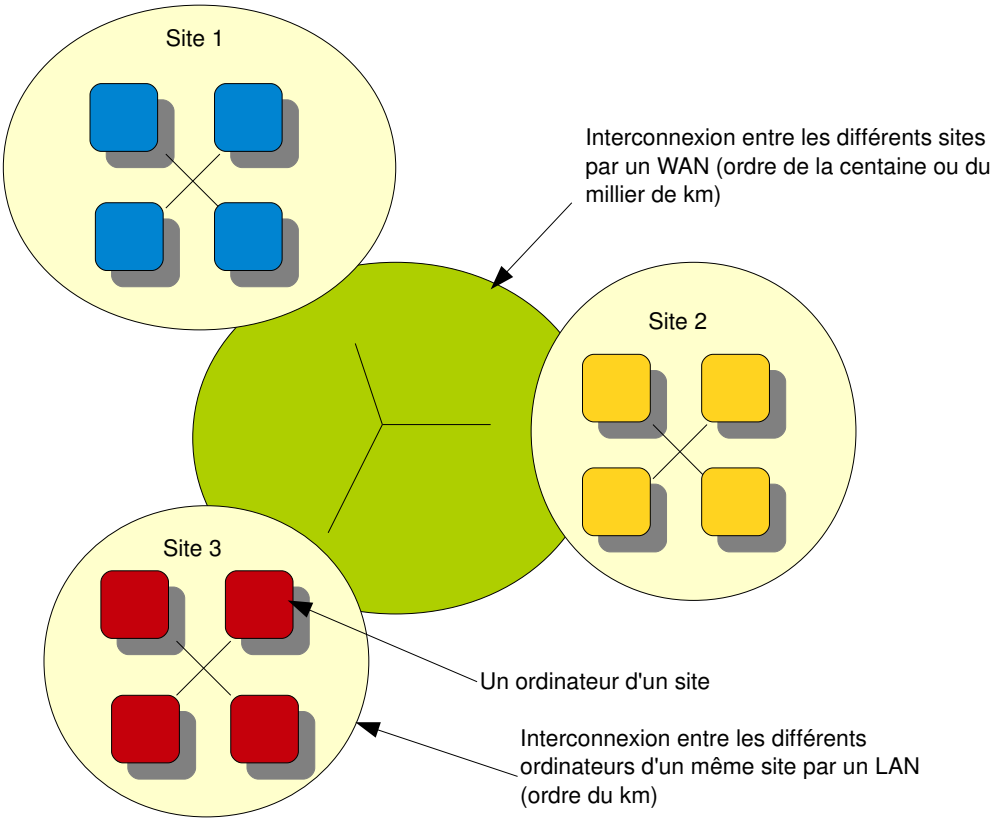


FIGURE 1.3 – Interaction entre LAN et WAN

1.1.2.3 Des normes pour les réseaux

Afin que les nœuds puissent communiquer entre eux, des normes ont été créées. Le modèle OSI (*Open System Interconnection*) divise la communication entre deux nœuds en sept couches en allant du niveau physique (les bits envoyés sur le support physique – câble électrique, fibre optique – constituant le réseau) au niveau applicatif (les données). Le modèle TCP/IP constitué de 4 couches, est un modèle dont les différentes couches ont une définition moins rigide que celle du modèle OSI. Quel que soit le modèle considéré, chaque couche apporte une abstraction de la couche sous-jacente à la couche supérieure. Dans le reste de ce document on se concentre sur le modèle TCP/IP. La figure 1.4 présente une comparaison entre les couches OSI et TCP/IP.

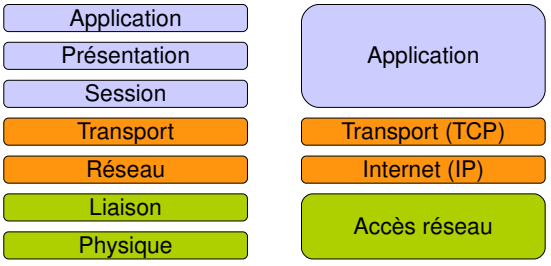


FIGURE 1.4 – Comparaison entre les couches OSI et TCP/IP

1.1.3 Les grappes de calcul

Les grappes (*cluster*) sont matériellement constituées de multiples nœuds, interconnectés par un réseau local à haute performance et souvent gérés par des systèmes à exécution par lots ou des systèmes d'exploitation de grappes, dits systèmes à image unique. Les nœuds d'une même grappe sont tous situés géographiquement dans une même salle.

1.1.3.1 Vision matérielle

Une grappe de calculateurs est un ensemble de nœuds homogènes interconnectés par un réseau local à haute performance dont la finalité est de réaliser conjointement un traitement.

La première grappe de calculateurs constituée d'ordinateurs personnels a été réalisée en 1994 dans le projet *Beowulf* de la NASA [163]. Cette grappe était composée de 16 nœuds, disposant chacun de 256 Mo de mémoire, reliés par un réseau *Ethernet* 10 Mbits. De nos jours, les grappes peuvent dépasser la centaine, voire le millier de nœuds en utilisant des technologies réseaux comme *Ethernet* 10 GBits, *Myrinet* ou *InfiniBand*^{TM}.

1.1.3.2 Introduction aux systèmes de gestion des grappes

Les grappes de calculateurs sont souvent employées dans le contexte du calcul à haute performance (HPC, *High Performance Computing*) qui a pour but la rapidité d'exécution des traitements. Il existe différents systèmes de gestion des grappes. Les plus connus sont les systèmes de traitement par lots et les systèmes à image unique.

La gestion traditionnelle avec un nœud maître : introduction aux systèmes à exécution par lots Les systèmes à exécution par lots (*batch scheduler*) nécessitent généralement l'utilisation d'un nœud maître qui possède une connaissance globale

de l'état des nœuds de la grappe. Les utilisateurs voulant exécuter leurs applications se connectent au nœud maître et soumettent leurs tâches (les tâches soumises à un système de traitement par lots sont généralement non-interactifs). PBS/TORQUE [31], IBM LoadLeveler [30], OAR [56] et sont des exemples de systèmes à exécution par lots.

La soumission des tâches se fait généralement avec l'utilisation d'un fichier de description qui peut être de type JSDL (*Job Submission Description Language*) [44]. Ce type de description permet d'expliciter où se situe le binaire de l'application à exécuter, où stocker les résultats de l'application, quel type de ressources doit être utilisé, etc.

Les systèmes à exécution par lots font de l'ordonnancement temporel et spatial en appliquant une politique définie par l'administrateur, afin d'exécuter les tâches sur les ressources au cours du temps : *premier entré, premier sorti, (FIFO – first in first out)* ou bien *dernier entré, premier sorti (LIFO – Last In, First Out)* en sont des exemples. La figure 1.5 présente une architecture de grappe gérée par un système à exécution par lots.

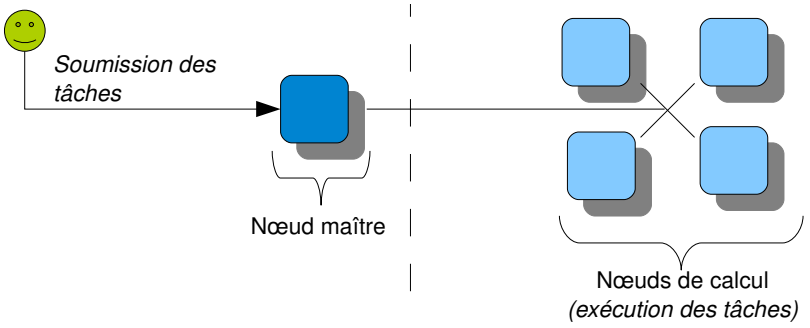


FIGURE 1.5 – Présentation de l'architecture d'une grappe gérée par un système à exécution par lots

Système d'exploitation pour grappes : les systèmes à image unique Les systèmes à image unique (SSI, *Single System Image*) permettent une gestion globale des processus s'exécutant sur une grappe. Certains systèmes à image unique comme GLUnix [94], BProc [104], CPlant [147] disposent d'un nœud maître. Pour accéder à la grappe, l'utilisateur doit passer par le nœud maître. La différence avec les systèmes à exécution par lots est qu'à partir du nœud maître, les processus de la grappe sont manipulables comme s'ils étaient locaux au nœud maître (l'utilisateur peut par exemple, exécuter un *ps*[1] et la liste de l'ensemble des processus exécutés sur l'ensemble des nœuds de calcul est retournée à l'utilisateur). L'ordonnanceur de ce type de système est situé sur le nœud maître, comme pour les systèmes à exécution par lots.

De plus, contrairement au système à exécution par lots, les systèmes à image unique permettent l'exécution de tâches interactives.

D'autres systèmes à image unique fournissent, en plus de la gestion globale des processus, une vue des ressources distribuées comme étant un nœud multiprocesseur virtuel. MOSIX [46], Kerrighed [129, 131] sont des exemples de systèmes à image unique dans lesquels il n'y a pas de nœud maître (c'est-à-dire que tous les nœuds sont équivalents). Les tâches peuvent être soumises à partir de n'importe quel nœud.

1. La commande *ps* des systèmes Unix permet de lister l'ensemble des processus présent sur le système.

Dans ce cas, l'ordonnanceur est distribué sur l'ensemble des nœuds. La figure 1.6 présente une architecture de grappe gérée par un système à image unique.

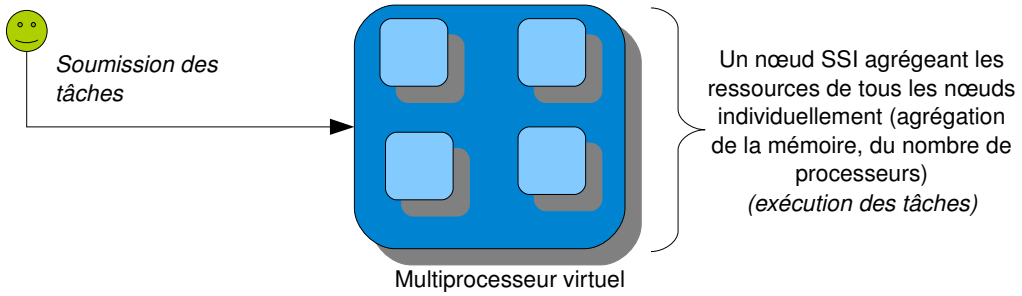


FIGURE 1.6 – Présentation de l'architecture d'une grappe gérée par un système à image unique

1.1.4 Les grilles

Une grille peut être vue comme la fédération de plusieurs grappes : des réseaux WAN interconnectent différents sites (voir figure 1.3). Ces différents sites sont gérés par différentes insitutions qui mettent en partage leurs ressources. Ce partage est fondé sur la notion d'organisation virtuelle (VO, *Virtual Organization*).

1.1.4.1 Présentation

Le terme *grille* apparaît à la fin des années 1990 [75, 77] et provient de l'analogie avec la distribution du courant électrique. En effet, comme il suffit de brancher son appareil électrique sur une prise de courant pour le faire fonctionner, il devrait suffire de brancher son ordinateur sur une prise de la grille pour lui fournir toute la puissance de calcul dont il a besoin. Cette puissance de calcul de la grille provient d'un ensemble de ressources hétérogènes géographiquement distribuées et pouvant appartenir à différents domaines d'administration.

Les systèmes de grille permettent de fédérer des ressources distribuées sur différents sites et appartenant à des domaines d'administration différents. Cette fédération est fondée sur la notion d'organisation virtuelle qui permet de gérer le partage des ressources de la grille et les droits d'accès des utilisateurs [40, 78, 79]. Il est alors possible aux utilisateurs appartenant à des organisations virtuelles d'exécuter des applications à très large échelle bénéficiant des ressources de calcul fournies par différents domaines d'administration.

1.1.4.2 Introduction aux systèmes de gestion des grilles

Les systèmes de grille ont pour objectif de fournir des services de gestion des ressources de la grille. Cinq principaux axes peuvent être mis en avant dans les systèmes de gestion des grilles :

- la collaboration (VO) : partage et utilisation de ressources de différents domaines d'administration par des utilisateurs appartenant à ces différents domaines,
- la sécurité : authentification des entités (utilisateurs et ressources) de la grille et contrôle d'accès aux ressources,
- la gestion des données : accès aux données entre les différents sites de la grille,

– la gestion de ressources hétérogènes : découverte et allocation des ressources pour leur utilisation,
– la gestion des applications : soumission, surveillance et contrôle de l'exécution des applications sur les ressources.

Selon les différents systèmes, d'autres fonctionnalités peuvent être fournies comme par exemple, les services de tolérance aux défaillances, d'isolation entre les applications, de comptabilisation de l'usage des ressources. Globus Toolkit™ [41, 76], UNICORE [32], Glite [11], mais aussi Legion [16, 100, 117], DIET [57], GridOS [99, 140], ou bien encore XtreemOS [68] sont des exemples de systèmes de gestion de grille.

1.2 La virtualisation des ressources informatiques

La définition ainsi que les premières mises en œuvre des machines virtuelles (VM, *Virtual Machine*) ont été proposées dès le début des années 1970 par Popek et Goldberg [97, 145]. Un nœud (système hôte) peut exécuter un programme (hyperviseur) qui est capable d'exécuter des machines virtuelles (système invité) chacune étant indépendante des autres. À l'origine, les systèmes de virtualisation étaient conçus pour les super-ordinateurs afin de pouvoir les manipuler avec une plus grande souplesse. Ce n'est qu'au cours des dix dernières années que la virtualisation a suscité un fort intérêt dans la gestion des ordinateurs individuels et des grappes : l'arrivée sur le marché d'ordinateurs très performants à bas coût dotés de processeurs multi-cœurs qui dans le cas des processeurs récents supportent la virtualisation de manière matérielle, a permis de démocratiser les systèmes de virtualisation.

1.2.1 Principe de la virtualisation

Une machine virtuelle s'exécute sur un nœud grâce à un programme que l'on appelle *hyperviseur*. Goldberg donne une définition de l'hyperviseur comme un programme disposant des trois propriétés suivantes :

Équivalence : un programme utilisateur s'exécutant dans une machine virtuelle contrôlée par un hyperviseur doit avoir un comportement identique à celui de l'exécution du même programme directement sur le matériel.

Contrôle des ressources : l'hyperviseur doit avoir le contrôle complet de l'ensemble des ressources du nœud hôte.

Efficacité : une fraction importante des instructions de la machine virtuelle doit être exécutée directement sur le nœud sans intervention de l'hyperviseur.

De nos jours, on ne peut pas dire que tous les hyperviseurs respectent ces trois points. En effet, le point le plus délicat à mettre en œuvre reste l'efficacité de l'hyperviseur. Ce point est en lien direct avec le processeur du nœud hôte. Comme vu dans la section 1.1.1.1 les processeurs disposent d'instructions non-privilégiées, privilégiées et sensibles. Ainsi :
– lorsqu'une machine virtuelle veut exécuter une instruction non-privilégiée, cette instruction peut être directement exécutée par le processeur,
– lorsqu'une machine virtuelle veut exécuter une instruction privilégiée, une exception est générée et l'hyperviseur peut l'intercepter et traiter la requête.
– lorsqu'une machine virtuelle veut exécuter une instruction sensible, des mécanismes spécifiques doivent être mis en place afin de traiter ce cas : c'est précisément cela qui peut engendrer des problèmes de performance.

On dit qu'un processeur est virtualisable s'il ne possède pas d'instructions sensibles. Or, les processeurs de type x86 possèdent 17 instructions sensibles. Les processeurs de type x86 ne sont pas virtualisables, il faut donc traiter les instructions sensibles d'une manière spéciale. Plusieurs méthodes sont possibles :

La para-virtualisation (solution logicielle) : c'est un procédé de virtualisation dans lequel on modifie le système invité afin qu'il prévienne directement l'hyperviseur lorsqu'il a besoin d'exécuter une instruction sensible. Ce procédé nécessite donc une modification du système invité.

L'interprétation à la volée (solution logicielle) : c'est un procédé dans lequel l'hyperviseur surveille chaque instruction du système invité et peut donc intercepter les instructions sensibles et agir en conséquence.

La virtualisation matérielle (solution matérielle – *full-virtualization*) : c'est un procédé dans lequel l'hyperviseur n'est pas au niveau 0 des privilèges, mais est au niveau -1 (encore plus privilégié). Ce niveau « matériellement » conçu permet d'exécuter les système invités des machines virtuelles au niveau 0 de manière transparente et sans modifications de celles-ci.

1.2.1.1 Les procédés de virtualisation

Deux principales approches sont couramment utilisées : la virtualisation axée sur le système et la virtualisation axée sur les processus. La virtualisation axée sur le système offre au système invité une vue totale ou partielle des ressources dont est composé le système hôte : il est possible d'installer un système d'exploitation personnalisé dans le système invité qui est différent de celui présent sur le système hôte. La virtualisation axée sur les processus offre au système invité une vue totale ou partielle du système d'exploitation hôte : le système invité utilise le système d'exploitation hôte, il n'est pas possible d'y installer un autre système d'exploitation.

La virtualisation axée sur les systèmes complets Dans le cadre de la virtualisation axée sur les systèmes complets (*System-Level Virtualization*), il est question de virtualiser un ordinateur complet sur un nœud (l'interface offerte par ce type de système est un jeu d'instructions ISA).

À cause de la propriété de *contrôle des ressources* définie par Goldberg, une machine virtuelle ne peut exécuter aucune instruction processeur privilégiée, ni même accéder aux ressources du nœud directement. Pour résoudre ce problème, les machines virtuelles passent par un intermédiaire exécuté dans le système hôte qui, lui, possède les privilèges nécessaires pour répondre à la requête.

De plus, plusieurs machines virtuelles peuvent s'exécuter en même temps sur un nœud et leur exécution est ordonnancée par l'hyperviseur. L'hyperviseur a également la charge de gérer l'exécution des instructions privilégiées (accès à la table des pages mémoire par exemple) et de résoudre ou transférer la requête au système hôte (généralement, l'hyperviseur gère la mémoire des machines virtuelles directement et transfère tous les autres types d'instructions privilégiées au système hôte).

Goldberg a identifié deux types d'hyperviseurs pour la virtualisation axée sur les systèmes : les hyperviseurs de *Type-I* et ceux de *Type-II*.

Les hyperviseurs de *Type-I* Goldberg a défini les hyperviseurs de *Type-I* comme étant des hyperviseurs s'exécutant directement sur le nœud. Les différents

systèmes invités ainsi que le système hôte s'exécutent *au-dessus* de l'hyperviseur. Xen [47] et Hyper-V [172] sont des exemples de systèmes de virtualisation de *Type-I*. La figure 1.7 présente une architecture d'hyperviseur de Type-I sur un nœud. L'*OS privilégié* est le système hôte. Les systèmes invités sont *VM1* et *VMn*.

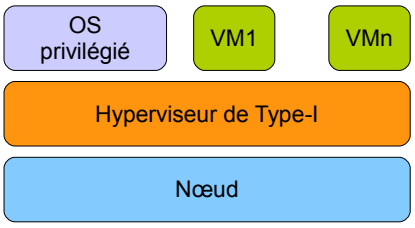


FIGURE 1.7 – Représentation de l'architecture d'un hyperviseur de Type-I

Les hyperviseurs de *Type-II* Les hyperviseurs de *Type-II* sont exécutés *au-dessus* du système hôte, et les machines virtuelles *au-dessus* de l'hyperviseur. QEMU [49], KVM [114] et VMware Server [34] sont des exemples de système de virtualisation de *Type-II*. La figure 1.8 présente une architecture d'hyperviseur de Type-II sur un nœud. L'*OS privilégié* est le système hôte qui exécute l'hyperviseur. Les systèmes invités sont *VM1*, *VM2* et *VMn*.

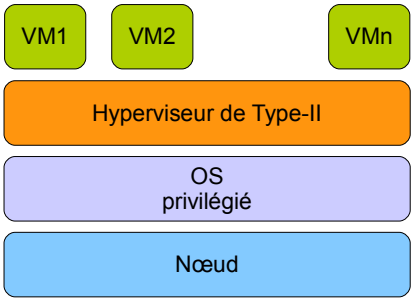


FIGURE 1.8 – Représentation de l'architecture d'un hyperviseur de Type-II

La virtualisation axée sur les processus Quand Goldberg propose sa classification dans les années 1970, il s'intéresse uniquement à la virtualisation axée sur les systèmes. La virtualisation axée sur les processus (*Process-Level Virtualization*), qui ne fait donc pas partie directement de la classification de Goldberg, est principalement connue sous le nom de virtualisation par conteneur (*container*). Les conteneurs permettent de faire de la virtualisation au sein d'un même système hôte, c'est-à-dire que différents processus s'exécutant en même temps sur un système hôte vont avoir chacun une vision différente des ressources disponibles (et des autres processus en cours d'exécution simultanément). Notons qu'avec ce type de virtualisation le système invité dans les conteneurs est le même que celui du système hôte, et dans ce cas-ci, le système hôte et l'hyperviseur sont, l'un et l'autre, généralement intégrés. OpenVZ [24], *chroot* [6], *Control Group* (*cgroup*), *Namespace* sont des exemples de conteneurs pour les systèmes Linux. La figure 1.9 présente une architecture d'hyperviseur axée sur les processus sur un nœud.

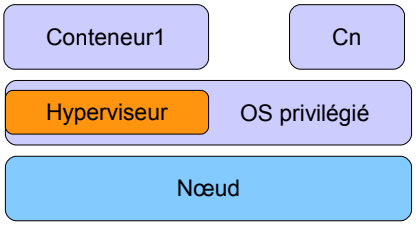


FIGURE 1.9 – Représentation de l'architecture d'un hyperviseur axée sur les processus

Les machines langages D'autres systèmes peuvent être considérés comme des systèmes de virtualisation. Les machines virtuelles Java (JVM, *Java Virtual Machine*) [119] en sont un exemple.

Il est possible de comparer une JVM à un émulateur qui est capable de transformer du code Java en *bytecode*. Ce *bytecode* est ensuite lu et exécuté par le JRE (*Java Runtime Environment*) sur un nœud. Un élément clé des JVM est la portabilité qu'elle offre au code Java : il est possible d'exécuter du code Java sur une JVM quelle que soit l'architecture de la ressource matérielle, à condition qu'il existe une JVM pour l'environnement matériel concerné.

1.2.2 Intérêt de la virtualisation sur un nœud

Les systèmes de virtualisation proposent des fonctionnalités et des propriétés utiles dans le contexte de l'exécution d'applications sur un nœud.

1.2.2.1 Les fonctionnalités d'une machine virtuelle

Les fonctionnalités élémentaires d'une machine virtuelle quand elle est exécutée sur un nœud sont décrites dans les paragraphes suivants.

En cours d'exécution, une machine virtuelle peut être :

Arrêtée L'arrêt d'une machine virtuelle correspond au bouton « arrêt » d'un nœud.

Suspendue Cette fonctionnalité permet de suspendre l'exécution d'une machine virtuelle (donc des applications s'exécutant dans la machine virtuelle), puis de la reprendre plus tard et cela, de manière transparente pour les applications.

Sauvegardée La sauvegarde de l'état des machines virtuelles (*snapshot*) est une fonctionnalité qui permet de sauvegarder l'état d'une machine virtuelle lorsqu'il est suspendu, sur un périphérique de stockage afin de pouvoir le restaurer ultérieurement. Certains systèmes de virtualisation peuvent gérer différentes versions des sauvegardes, cela permet de revenir à une version antérieure si nécessaire.

La sauvegarde d'une machine virtuelle peut être combinée avec la fonctionnalité d'arrêt : dans un cas, la sauvegarde peut avoir lieu sans arrêter la machine virtuelle, dans l'autre cas, une fois la sauvegarde terminée la machine virtuelle est arrêtée.

Dans ce document, pour simplifier le discours, lorsque nous parlons de sauvegarde d'une machine virtuelle, cela signifie que l'on parle de la sauvegarde de l'état de la machine virtuelle et de son arrêt.

Lorsqu'une machine virtuelle est à l'arrêt ou suspendue, elle peut être :

Allumée L'allumage d'une machine virtuelle correspond au bouton « marche » d'un nœud. Une machine virtuelle ne peut être allumée que si elle n'a jamais été allumée ou qu'elle a été précédemment éteinte.

Restaurée La restauration d'une machine virtuelle s'effectue lorsque celle-ci a été préalablement suspendue ou sauvegardée et arrêtée. Dans ce cas, la machine virtuelle reprend son exécution là où elle a été suspendue ou sauvegardée.

1.2.2.2 Les propriétés de la virtualisation

Les systèmes de virtualisation possèdent des propriétés qui peuvent être utilisées dans la gestion des infrastructures de calcul et des applications [173].

Isolation : C'est le degré d'isolation d'une machine virtuelle par rapport au système hôte et aux autres machines virtuelles. Une machine virtuelle complètement isolée ne peut compromettre les autres machines virtuelles, l'hyperviseur, ou le système hôte en cas d'attaque malicieuse.

L'isolation peut-être définie selon trois aspects [62, 116, 178] :
- *La sécurité (Security Isolation)* : un élément extérieur ne peut accéder à la machine virtuelle sans autorisation.
- *La protection du système (Software Fault Isolation)* : un élément exécuté dans la machine virtuelle ne peut volontairement ou involontairement accéder au système hôte ou à d'autres machines virtuelles.
- *L'isolation des ressources (Full Resource Containment and Control)* : Les ressources (CPU, mémoire...) sont attribuées aux différentes machines virtuelles selon les règles définies : une machine virtuelle ne peut utiliser plus de ressources que celles qui lui sont allouées.

Par construction, les machines virtuelles apportent un degré d'isolation et de sécurité pour les applications (ce degré étant variable d'un type de machine virtuelle à un autre). L'isolation est assurée par le fait que les machines virtuelles sont gérées individuellement par l'hyperviseur. La sécurité provient du fait que l'hyperviseur gérant les machines virtuelles a un code suffisamment concis pour pouvoir être vérifié formellement [162]. Ainsi, une attaque dans une machine virtuelle par quelqu'un de mal intentionné ne concerne que cette machine virtuelle et non l'ensemble du système. Cependant à notre connaissance, parmi les systèmes de virtualisation les plus cités (par exemple Xen, et QEMU-KVM), aucun n'a été vérifié formellement. De plus, les hyperviseurs étant de plus en plus perfectionnés, ceux-ci comptent de plus en plus de lignes de code [2].

Émulation : Selon Goldberg, un système fait de l'émulation lorsqu'un micro-code (*firmware*) intercepte chaque instruction de la machine virtuelle pour l'exécuter sur le nœud. Bien qu'un surcoût peut être engendré par l'utilisation d'un micro-code, l'émulation apporte cependant une très grande flexibilité en apportant une complète abstraction des ressources sous-jacentes. Avec l'émulation, l'architecture des ressources matérielles peut être différente de celle de l'application exécuté dans la machine virtuelle, le micro-code étant chargé de faire la correspondance entre les deux architectures.

Exécution d'applications sans modification : C'est la capacité de pouvoir exécuter une application dans une machine virtuelle sans la modifier. En d'autres termes, c'est

2. Par exemple l'hyperviseur de Xen 4.0 est constitué de 250 931 lignes de code (relevé avec l'utilitaire *sloccount*). De même, l'hyperviseur QEMU-KVM dans sa version 0.13.03 contient 558 156 lignes de code.

la capacité d'adapter la machine virtuelle, via *émulation* par exemple, à l'application et non le contraire.

De plus, de part les propriétés d'isolation, il est possible de préciser un environnement de travail spécifique par application (et non un environnement standard pour toutes les applications). Un même nœud peut exécuter plusieurs systèmes d'exploitation spécialisés pour différentes applications.

1.2.3 Intérêt de la virtualisation en environnement distribué

Les systèmes de virtualisation peuvent être utilisés en environnement distribué. Par rapport à une utilisation sur un seul nœud, l'utilisation des systèmes de virtualisation en environnement distribué apporte de nouvelles fonctionnalités et propriétés.

1.2.3.1 Les fonctionnalités

Les fonctionnalités élémentaires des machines virtuelles en environnement distribué sont les mêmes que celles présentées sur un nœud (voir section 1.2.2.1) complétées par d'autres tirant profit de la distribution des ressources.

En cours d'exécution, une machine virtuelle peut être :

Déplacé Cette fonctionnalité permet de déplacer (migrer) une machine virtuelle d'un nœud à un autre et cela de manière transparente pour les applications exécutées dans cette machine virtuelle.

Cette fonctionnalité peut-être combinée avec celle d'arrêt : en combinant ces deux fonctionnalités, la machine virtuelle initiale est arrêtée après sa migration, dans un second cas la migration entraîne un clonage de la machine virtuelle (la machine virtuelle d'origine n'est pas arrêtée après sa migration). Dans ce document on considère que les migrations entraînent l'arrêt des machines virtuelles d'origine.

Optimisation : la migration à chaud Des techniques dites de migration à chaud (*Live Migration*) permettent de déplacer une machine virtuelle d'un nœud à un autre avec un temps d'indisponibilité de la machine virtuelle comparable au temps que peut prendre une perturbation du réseau [59] : du point de vue des connexions réseau, le temps de migration est généralement inférieur au temps limite de décision de terminaison de la connexion (*Time Out*).

1.2.3.2 Les propriétés de la virtualisation

En environnement distribué les systèmes de virtualisation possèdent les mêmes propriétés que celles en environnement mono-nœud (voir section 1.2.2.2) complétées par d'autres.

La consolidation de serveurs : C'est la capacité de changer à la demande les ressources allouées à une machine virtuelle, et cela de manière transparente pour les applications exécutées dans la machine virtuelle. Cette propriété est réalisable efficacement grâce aux propriétés de migration à chaud.

Par exemple, lorsqu'un nœud est ajouté à une grappe, selon les politiques mises en place, une ou plusieurs machines virtuelles peuvent être migrées sur ce nouveau nœud afin d'équilibrer la charge de la grappe. De la même manière, avant de retirer un nœud de la grappe, il est possible de déplacer les machines virtuelles sur d'autres nœuds.

Par exemple, IVS [58], VOVO [143] et Entropy [105] font de la consolidation de serveurs. Cependant, à notre connaissance, ces solutions ne considèrent pas la grille, et font uniquement de la gestion des machines virtuelles à l'échelle de la grappe.

La portabilité : C'est la capacité de déplacer une machine virtuelle d'un nœud à un autre. Généralement la portabilité impose que les nœuds source et de destination soient de la même architecture avec les mêmes environnements logiciels (système d'exploitation, hyperviseur).

Économie d'énergie : Grâce aux fonctionnalités de migration et de suspension et restauration des machines virtuelles, il est possible de gérer plus finement la charge des nœuds. Ainsi, il est possible d'économiser de l'énergie en allumant et en éteignant les nœuds selon la charge de travail à exécuter [58, 133, 143].

Gestion d'une collection de machines virtuelles : De part la fonctionnalité de sauvegarde/restauration d'une machine virtuelle, il est possible en cas de défaillance d'un ou plusieurs nœuds, de restaurer une collection de machines virtuelles (grappe virtuelle) depuis la dernière sauvegarde et non depuis le début de son exécution. Cette opération se fait de manière transparente du point de vue des communications réseau si leur sauvegarde et restauration se font de manière synchronisée [70].

Haute disponibilité : Lors d'une maintenance planifiée, il est possible de faire de la migration à chaud des machines virtuelles et ainsi continuer d'assurer l'exécution des applications s'exécutant en leur sein.

1.2.3.3 Les centrales numériques

Les premières définitions de centrales numériques (*Clouds*) [106] apparaissent à partir des années 2005. Eric Schmidt, président de Google, donne en août 2006 une définition des centrales numériques :

It starts with the premise that the data services and architecture should be on servers. We call it cloud computing – they should be in a Cloud somewhere. And that if you have the right kind of browser or the right kind of access, it doesn't matter you have a PC or a Mac or a mobile phone or a Black-Berry or what you have – or new devices to be developed – you can get access to the cloud...

Depuis, les recherches et les technologies progressant, d'autres définitions plus précises, mais toujours en évolution sont apparues. Par exemple, le NIST (*National Institute of Standards and Technology*) donne la définition suivante des centrales numériques [125] :

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

De ces définitions, nous retenons que les centrales numériques fournissent, à la demande, un ensemble de services personnalisables avec lesquels l'utilisateur peut stocker des informations, interagir avec ces informations et cela à partir de n'importe quel terminal disposant d'un accès à Internet. L'accès aux services se fait de manière authentifiée et l'utilisateur paye pour l'utilisation de ces services. Les services et données fournis à l'utilisateur se situent quelque part dans la centrale numérique : « l'utilisateur ne sait pas où exactement, mais il ne veut pas le savoir non plus ».

C'est ce dernier point, à savoir *le fait que les services des centrales numériques sont masqués à l'utilisateur* qui marque la principale différence avec les grappes et les grilles de calcul [80].

Grappes, grilles et centrales numériques Les grappes, grilles et centrales numériques sont des systèmes informatiques distribués qui sont différents de part leur mise en œuvre. En effet, les grappes sont des systèmes initialement dédiés à l'exécution des applications distribuées et parallèles. Elles peuvent être gérées par des systèmes à exécution par lots et par des systèmes à image unique. Dans le cas des systèmes à image unique les nœuds de la grappe sont agrégés pour donner l'illusion à l'utilisateur d'utiliser un unique nœud SMP. La fédération des grappes, la grille, apporte la notion de collaboration entre différentes institutions désirant partager leurs ressources informatiques.

Les centrales numériques apportent une abstraction supplémentaire dans la gestion et l'utilisation des ressources matérielles en utilisant des systèmes de virtualisation [103, 176]. Ainsi, le modèle d'accès aux ressources des centrales numériques est différent de celui des grilles : les centrales numériques utilisent le paradigme d'utilisateur/fournisseur dans lequel la facturation se fait sur la consommation des ressources faites par l'utilisateur, alors que le modèle des grilles est plus orienté sur le principe de collaboration, de partage des ressources entre différentes institutions. Nous détaillons les différences existant entre centrales numériques et grilles dans la section 3.1.

Les différents types de centrale numérique Les centrales numériques peuvent être décomposées en trois types de service qu'elles fournissent : des services (SaaS, *Software as a Service*), des plates-formes (PaaS, *Platform as a Service*) et des infrastructures (IaaS, *Infrastructure as a Service*).

Dans ce document nous nous focalisons sur les centrales numériques de type IaaS.

L'infrastructure vue comme un service Les centrales numériques orientées infrastructures offrent une nouvelle manière de concevoir la gestion des ressources informatiques. Le principal service offert par une centrale numérique de type IaaS est un environnement informatique constitué généralement de machines virtuelles. En effet, une entreprise, plutôt que d'investir dans du matériel et de gérer son cycle de vie (installation, entretien, maintenance, remplacement), peut faire appel à des fournisseurs de ressources sur Internet qui facturent leur utilisation (*pay as you go*). Cela rend possible l'externalisation des infrastructures et services informatiques, et permet de s'affranchir de l'hétérogénéité des ressources.

Amazon EC2 [2] et AppNexus [3] sont des exemples de centrale numérique commerciale. Les clients de ces centrales peuvent louer des instances et les utiliser. Différents autres services peuvent être proposés. Par exemple Amazon fournit un service de stockage permanent (S3) et permet de redimensionner le nombre d'instances en cours d'exécution en fonction de leur charge. Nimbus [111], Eucalyptus [135] et OpenNebula [161] sont des exemples de système de gestion de centrale numérique permettant de créer des centrales numériques privées. Ils proposent également leurs propres services de stockage et de gestion des images des machines virtuelles.

Les interfaces d'accès aux centrales numériques Les centrales numériques offrent une interface d'accès aux utilisateurs leur permettant par exemple d'importer/sélectionner une image de machine virtuelle, de la démarrer, de l'éteindre.

L'interface sans doute la plus connue est celle EC2 (*Elastic Compute Cloud*) d'Amazon. C'est une interface web qui permet d'accéder facilement aux options de contrôle des instances (création, démarrage, arrêt, état). Il est également possible d'interagir avec ces instances en utilisant directement des utilitaires en ligne de commande. Enfin, une API est proposée aux utilisateurs pour qu'ils puissent développer eux-mêmes leurs utilitaires de gestion des instances.

Vers la standardisation des interfaces La définition et l'élaboration de ces interfaces sont encore sujets à discussion. Par exemple, des groupes de travail comme l'OCCI (*Open Cloud Computing Interface*) [22] ou l'OCC (*Open Cloud Consortium*) [21] cherchent à standardiser les interfaces d'accès des centrales numériques et d'une manière plus générale leur interopérabilité.

1.2.3.4 La fédération de centrales numériques

Avec l'émergence des centrales numériques, c'est tout naturellement que sont apparues les premières propositions de fédération des centrales numériques [113] (*Sky Computing*). Le but de ces fédérations, à la manière des grilles, est de pouvoir bénéficier du potentiel de fournisseurs d'infrastructures différents (comme vu dans la section 1.1.4, dans le cas des grilles c'est le potentiel de différents sites que l'on cherche à fédérer).

De plus l'utilisation de systèmes de virtualisation à l'échelle de plusieurs sites permet, de la même manière qu'il est possible sur un site de faire de la consolidation de serveurs, de faire de la consolidation de sites : gérer les machines virtuelles de manière transparente en pouvant les déplacer d'un site à un autre.

Le domaine des fédérations de centrales numériques tout comme celui des centrales numériques sont encore en pleine expansion : leur définition et les technologies utilisées ne sont pas encore complètement établies. De multiples problèmes concernant par exemple l'interopérabilité et la gestion du cycle de vie des collections de machines virtuelles, doivent être traités [148].

1.3 Discussion

Les infrastructures informatiques sont de différents types. L'élément unitaire d'une infrastructure est un nœud. Les grappes sont des infrastructures interconnectant des nœuds géographiquement proches. Les grappes sont gérées par des systèmes à exécution par lots ou des systèmes à image unique. Les grilles sont des infrastructures interconnectant des grappes géographiquement éloignées. Enfin, une centrale numérique peut être définie comme ayant l'infrastructure matérielle d'une grappe, mais est gérée par un système de virtualisation. De même, une multi-centrale numérique peut être décrite comme une grille qui est gérée par un système de virtualisation.

Cette manière de présenter les infrastructures informatiques montre que pour une infrastructure matérielle donnée, son système de gestion de ressources orientera son type d'utilisation. Par exemple, une infrastructure matérielle de type grappe gérée par un système à exécution par lots sera qualifiée de grappe de calcul, alors que la même infrastructure matérielle avec un système de virtualisation distribué sera qualifiée de centrale numérique. Ces systèmes de gestion des ressources sont installés et configurés par l'administrateur de l'infrastructure qui définit des politiques d'usage des ressources. Les utilisateurs peuvent selon les politiques et les axes de flexibilité offerts par le système de gestion des ressources avoir un degré de personnalisation de l'environnement qui leur est offert plus ou moins grand. C'est à partir du constat que pour une infrastructure matérielle donnée il existe différentes manières de l'exploiter, que nous avons orienté nos recherches : de quelle manière ce cadre pourrait-il être suffisamment flexible pour que dans une infrastructure donnée, selon les politiques définies par les administrateurs, les utilisateurs peuvent s'ils le veulent bénéficier d'un large degré de personnalisation de l'infrastructure considérée ? Finalement, on peut discerner différents rôles : l'utilisateur a un besoin, le système doit apporter une réponse adaptée à son besoin, l'administrateur précise les politiques qui cadrent les règles

d'utilisations de l'infrastructure.

1.3.1 Composition des systèmes informatiques et degrés de flexibilité

Quel que soit le type d'infrastructure considéré (grappe, grille ou centrale numérique) ces systèmes informatiques peuvent être décomposés en trois niveaux :

L'environnement d'exécution L'environnement d'exécution contient les bibliothèques et toutes les dépendances nécessaires à une application pour être exécutée, par exemple, un environnement MPI.

Gestionnaire de ressources Le gestionnaire de ressources réalise l'interface entre l'environnement d'exécution et le matériel sous-jacent. Par exemple, un système d'exploitation est un gestionnaire de ressources.

Infrastructure L'infrastructure contient l'ensemble des ressources matérielles. Par exemple, l'infrastructure matérielle peut être de type grappe ou grille.

Chacun de ces niveaux offre un axe de flexibilité selon le gestionnaire de ressources utilisé. Afin de mieux comprendre intuitivement ce qu'est la flexibilité nous avons regroupé dans un même tableau (cf 1.1) les systèmes informatiques que nous traitons dans ce document, avec pour chacun d'eux, le degré de flexibilité qu'ils offrent à chaque niveau, selon que l'on est administrateur ou utilisateur.

Environnement d'exécution Dans le cas des grappes gérées avec un système à exécution par lots, de celles gérées par un système à image unique et des grilles, le degré de personnalisation des environnements d'exécution des utilisateurs est limité. Les environnements sont généralement préconfigurés et pour l'installation ou la modification d'une configuration il faut faire une demande auprès de l'administrateur de l'infrastructure considérée. Les administrateurs peuvent installer et configurer les environnements d'exécution.

Dans le cas des centrales numériques (et multi-centrales) l'utilisateur peut personnaliser son environnement directement, sans avoir besoin de l'intervention d'un administrateur. L'utilisateur dispose de droits d'administration privilégiés et peut installer et configurer son environnement d'exécution lui même.

Gestionnaire de ressources Dans le cas des grappes et des grilles, l'utilisateur ne peut changer de gestionnaire de ressources. Par exemple, il ne lui est pas possible d'installer un système d'exploitation personnalisé sur la grappe ou la grille. De même, les administrateurs sont liés au gestionnaire de ressources utilisé. Par exemple, dans une grappe gérée par Torque, les systèmes d'exploitation exécutés sur les nœuds d'exécution doivent être de type Linux (il n'est pas possible d'installer des nœuds avec un système d'exploitation de type Windows).

Dans le cas des centrales numériques, l'utilisateur peut choisir le système d'exploitation qu'il peut utiliser.

Infrastructure L'utilisateur d'une grappe ou d'une grille ne peut pas modifier son infrastructure (il ne peut pas y ajouter ou y enlever des nœuds par exemple). L'administrateur peut changer la configuration de l'infrastructure de la grappe ou la grille qu'il gère.

Dans le cas des centrales numériques, l'utilisateur dispose d'une certaine forme de flexibilité. Par exemple, il peut choisir le type de machines virtuelles qu'il désire en fonction d'un catalogue contenant les différents types de machines virtuelles que la centrale numérique est capable d'exécuter. La flexibilité ici est que, une fois le type de

machine virtuelle choisi, l'utilisateur les obtient en s'affranchissant de l'infrastructure sous-jacente : il ne connaît ni quelle est la capacité matérielle du nœud qui exécute sa machine virtuelle, ni même le système de virtualisation utilisé. Par exemple, il peut avoir la vision d'une machine virtuelle équipée de 10 Go de mémoire vive et 4 processeurs alors qu'en réalité cette machine virtuelle est exécutée sur un nœud équipé de 48 Go de mémoire avec 8 CPU.

Finalement, de cette première analyse sur la flexibilité dans les systèmes informatiques, on peut retenir que les infrastructures matérielles, leur gestionnaire de ressources et leur environnement d'exécution peuvent être personnalisés à différents degrés selon que l'on est utilisateur ou administrateur. Ce degré de personnalisation est en lien direct avec la flexibilité que nous définissons dans la section suivante.

1.3.2 La flexibilité

La flexibilité d'une manière générale dans les systèmes informatiques peut être définie ainsi :

Définition 1.1 (La flexibilité) *La flexibilité d'un système informatique est sa capacité à pouvoir être configuré et reconfiguré à la demande.*

La définition de la flexibilité peut être affinée selon que l'on se place du point de vue de l'utilisateur ou de l'administrateur :

Définition 1.2 (La flexibilité pour l'utilisateur) *La flexibilité d'un système informatique pour l'utilisateur est sa capacité à pouvoir choisir le degré de personnalisation, à la demande, qu'il veut avoir. Cette opération doit se faire sans intervention directe d'un administrateur.*

Par exemple, dans le cas d'une centrale numérique, les administrateurs définissent des politiques de répartition des machines virtuelles sur les nœuds. Ils remplissent également le catalogue de machines virtuelles disponibles. C'est ensuite en fonction du besoin de l'utilisateur et des politiques des administrateurs que le système va déterminer par exemple, le placement d'une machine virtuelle donnée sur un nœud précis.

Définition 1.3 (La flexibilité pour l'administrateur) *La flexibilité pour l'administrateur d'un système informatique est sa capacité à pouvoir définir les politiques qui définissent les axes selon lesquels, pour répondre à un besoin d'un utilisateur le système informatique peut reconfigurer l'infrastructure de manière automatique, contrôlée et autorisée sans risque pour la sécurité globale du système.*

Dans ce contexte, cette thèse s'intéresse à *l'identification, la conception et la mise en œuvre d'axes de flexibilité pour la gestion et l'utilisation d'infrastructures informatiques distribuées*. Pour traiter cette problématique nous divisons notre étude en trois parties, chacune correspondant à une contribution présentée dans ce document.

La virtualisation sur la grille La virtualisation sur des infrastructures distribuées de type grappe permet essentiellement aux administrateurs de faire de la consolidation de serveurs et aux utilisateurs de bénéficier d'un plus grand degré de personnalisation des environnements d'exécution qui sont créés à la demande (à la manière des centrales numériques). Se pose alors la question de l'utilisation de ce type de système sur des infrastructures de grille. En effet, les grilles sont composées de ressources hétérogènes réparties sur différents sites. Grâce à cette répartition des ressources, il

Niveaux informatiques	utilisateur / administrateur	Grappe batch	Grappe SSI	Grille	Centrale numérique	Multi-centrale numérique
Environnement d'exécution	utilisateur	Limité : l'environnement est préconfiguré	Limité : l'environnement est préconfiguré	Limité : l'environnement est préconfiguré	OK	OK
	administrateur	OK	OK	OK	OK	OK
Gestionnaire de ressources	utilisateur	NO	NO	NO	OK	OK
	administrateur	NO	NO	NO	OK	OK
Infrastructure	utilisateur	NO	Limité : l'infrastructure est préconfigurée	NO	Limité : réseau d'une grappe virtuelle par exemple	Limité : réseau d'une grappe virtuelle par exemple
	administrateur	OK	OK	OK	OK	OK

TABLE 1.1 – Degrés de personnalisation des différentes couches (application / bibliothèques – système de gestion des ressources – infrastructure) pour l'utilisateur et l'administrateur selon les infrastructures (grappes / grilles / centrales numériques) considérées

est intéressant d'étudier dans quelle mesure les systèmes de virtualisation peuvent apporter plus de flexibilité dans la gestion des tâches, par exemple, en les déplaçant d'un site à un autre. Quels sont les problèmes engendrés par l'utilisation de ce type de système, par exemple du point de vue de la configuration du réseau ou de la gestion des images des machines virtuelles, et quelles sont les solutions possibles ? Nous étudions et proposons des éléments de réponse à ces questions dans le chapitre 2.

Le rapprochement grappe / grille / centrale numérique Comme nous venons de le présenter dans ce chapitre, les grappes, grilles et centrales numériques sont des systèmes informatiques qui proposent des axes de flexibilité différents. Par exemple, une grappe gérée par un système à image unique a un axe de flexibilité orienté vers l'agrégation des nœuds. Une grille a un axe de flexibilité orienté vers la gestion des ressources hétérogènes distribuées entre différents domaines d'administration. Enfin, les centrales numériques proposent un axe de flexibilité orienté vers la « création » de ressources à la demande et la personnalisation des environnements d'exécution. La question que l'on se pose est de savoir s'il est possible de faire un rapprochement entre ces différents systèmes. Un système « générique » pouvant offrir à la fois la flexibilité des grappes, des grilles et des centrales numériques est-il concevable et possible à mettre en œuvre ? Nous étudions et proposons des éléments de réponse à ces questions dans le chapitre 3.

La flexibilité d'une manière générale Le bilan des contributions précédentes nous montre qu'il est possible de combiner des axes de flexibilité différents, ce qui ouvre de nouvelles perspectives pour la gestion et l'utilisation des infrastructures distribuées. Nous proposons une méthode de formalisation de « la flexibilité » qui permet de décrire des règles de combinaison entre les différents axes de flexibilité. À partir de cette formalisation, nous proposons un système qui intègre les axes de flexibilité étudiés et ouvre la perspective de la configuration d'une infrastructure virtuelle aux besoins des utilisateurs. Cette infrastructure virtuelle, que nous appelons plate-forme virtuelle, est « construite » sur l'infrastructure matérielle dans le respect des règles d'utilisation et de sécurité définies par l'administrateur. Cette construction se fait grâce à la configuration et reconfiguration de l'infrastructure matérielle par la combinaison, à la manière de Lego®, des gestionnaires de ressources qui offrent les axes de flexibilité adéquats. Nous présentons notre réflexion sur ces différents points dans le chapitre 4.

Chapitre 2

Saline : gestion de collections de machines virtuelles sur une grille pour l'exécution des tâches interruptibles

Les systèmes de virtualisation dont les fondements datent de 1970, sont très utilisés depuis les années 2000. Ces systèmes découplent la vue logique des ressources qu'a l'utilisateur de l'infrastructure matérielle. Ils ont été étudiés principalement dans le contexte des infrastructures informatiques de type grappe et des centrales numériques. Dans ce chapitre, nous nous intéressons à la flexibilité apportée par l'utilisation de techniques de virtualisation dans les infrastructures distribuées de type grille.

Ces travaux ont été effectués en collaboration avec Adrien Lèbre, chargé de recherche à l'École des Mines de Nantes et Pierre Riteau, doctorant dans l'équipe Myriads. Ils ont donné lieu à deux stages que j'ai encadrés : le stage de master de Oana Goga d'une durée de 6 mois effectué en 2008, le stage d'été d'Archana Nottamkandath d'une durée de 3 mois effectué en 2010. Saline a fait l'objet de plusieurs publications [82, 84, 85, 86].

2.1 Les tâches interruptibles dans les grilles

Dans une première partie, nous motivons notre approche en prenant le cas d'étude des tâches interruptibles. Dans une seconde partie nous présentons les mécanismes existants en fonction des axes de flexibilité qu'ils offrent aux tâches interruptibles.

2.1.1 Motivation

Les utilisateurs des grappes et des grilles ont généralement accès aux ressources de ces dernières par l'intermédiaire d'un système à exécution par lots pour l'exécution de leurs applications appelées tâches. Ces systèmes permettent de gérer l'attribution des tâches sur les ressources en fonction de politiques [160].

FCFS Un exemple de politique est celle de FCFS (*First Come First Served*) dans laquelle les tâches n'ont pas de priorité : la première arrivée est la première qui est servie. Cette politique permet d'ordonnancer les tâches sur l'infrastructure distribuée dans l'espace (en sélectionnant les nœuds adéquats pour l'exécution des tâches) et dans le temps (les tâches

sont ordonnancées sur les nœuds disponibles au fur et à mesure). Avec cette politique, il est possible d'avoir des nœuds disponibles si la tâche suivante dans la file nécessite plus de nœuds qu'il y en a de disponible. Cela peut conduire à un gaspillage de l'utilisation des ressources, si par exemple, dans la file d'attente il y a des plus petites tâches qui s'accomoderaient du nombre de nœuds disponibles. Une méthode pour y remédier est d'utiliser la technique du *backfilling*.

Backfilling Un autre exemple de politique est celui de FCFS-FF (*First Come First Served, First Fit*) qui est une version modifiée de FCFS. Elle y intègre, par exemple, la notion de *backfilling* [158]. Dans ce cas, les tâches sont exécutées selon leur ordre d'arrivée dans la file d'attente. Cependant, si des tâches « plus petites » disposent de ressources suffisantes pour être exécutées sans retarder le démarrage des autres tâches dans la file d'attente, alors elles sont exécutées. Ce type de politique est destinée à gérer des tâches dont il est possible de savoir à l'avance le temps d'exécution.

Les tâches interruptibles Ce chapitre est centré sur l'étude des tâches interruptibles (*best-effort jobs*). Les tâches interruptibles sont des tâches qui sont exécutées lorsque les ressources sont disponibles, mais qui peuvent être détruites à tout moment, si une tâche prioritaire devait être exécutée. La politique d'exécution des tâches interruptibles peut être considérée comme une version assouplie de la politique de *backfilling*. En effet, ces deux politiques autorisent que des tâches arrivées en dernier dans la liste des tâches soient exécutées sur des ressources disponibles en « doublant » les autres tâches de la file. Leur différence est que, dans le cas du *backfilling*, les tâches autorisées à « doubler » les autres tâches doivent avoir un temps d'exécution qui est au maximum égal au temps de disponibilité des nœuds (il n'y a pas de décalage de l'heure de démarrage prévue de la tâche qui a été « doublée »). Dans le cas des tâches interruptibles, l'assouplissement se porte sur la contrainte de temps : les tâches dans la file sont autorisées à « doubler » les autres tâches quelle que soit leur durée prévue, par contre, ces tâches peuvent être détruites à tout moment (et bien sûr, au moment du démarrage prévu des tâches qui ont été « doublées »).

Les tâches interruptibles par rapport au *backfilling* permettent d'augmenter le taux d'utilisation des ressources, cependant, elles restreignent ce type de tâches aux applications disposant de mécanismes de sauvegarde de points de reprise ou des applications de type sac de tâches (*bag of tasks*), qui peuvent être interrompues à tout moment et être redémarrées sans perte des calculs précédemment effectués. En effet, si une tâche à exécuter selon la politique de la file d'attente des tâches ne peut démarrer à cause de tâches interruptibles qui occupent tout ou partie des nœuds, le gestionnaire de ressources va détruire les tâches interruptibles afin de libérer des nœuds et exécuter la tâche prioritaire. Cela peut conduire à un gaspillage de l'utilisation des ressources de calcul étant donné que dans le cas de tâches qui n'intègrent pas de mécanismes de sauvegarde de leur état, elles sont détruites et doivent être redémarrées par la suite, depuis le début de leur exécution.

Par exemple, si un utilisateur soumet une tâche interruptible d'une durée de 10 heures et qu'une tâche prioritaire doit être exécutée à la 9^{e} heure de l'exécution de la tâche interruptible, alors la tâche interruptible sera détruite et, à moins que des mécanismes spécifiques aient été mis en place pour sauvegarder ou déplacer les données obtenues au cours des 9 premières heures du calcul, celles-ci seront perdues, si la tâche de plus forte priorité ne peut s'exécuter sur d'autres ressources disponibles.

En conclusion, les tâches interruptibles permettent à une application d'utiliser les ressources de l'infrastructure lorsqu'elles sont disponibles. Cependant, pour être efficace, les tâches interruptibles doivent intégrer une méthode de sauvegarde pour être restaurées

depuis leur dernière sauvegarde et non depuis le début de leur exécution.

2.1.2 Des approches pour l'exécution de tâches interruptibles

Nous nous intéressons à la gestion des tâches interruptibles qui peuvent être arrêtées à tout moment et de manière brutale pour allouer des ressources à une tâche prioritaire dans les infrastructures distribuées.

Nous considérons deux types d'applications : les applications qui par conception peuvent être interrompues (et redémarrées à l'endroit où elles ont été arrêtées) et les autres. Les applications conçues pour être interrompues exécutent des calculs sur des jeux de données qui sont sauvegardés au fur et à mesure de leurs exécutions. Les jeux de données sont indépendants et les calculs faits sur ces jeux de données sont suffisamment courts (en terme de temps) pour que lors d'une défaillance, les calculs peuvent reprendre depuis les derniers jeux de données calculés.

Les applications qui ne sont pas conçues pour être interrompues peuvent bénéficier de techniques de sauvegarde et restauration de leur « état » qui sont décrites dans les paragraphes suivants.

Sauvegarde et restauration de points de reprise de processus Dans les environnements d'exécution, les mécanismes de sauvegarde et restauration de points de reprise (*checkpoint/restart*) ou de migration nécessitent généralement l'utilisation de bibliothèques spécifiques et/ou d'environnements d'exécution particuliers.

Libckpt [144] et DejaVu [154] sont des exemples de systèmes permettant de sauvegarder une application. Dans ce cas, du point de vue du système d'exploitation, c'est l'image du processus qui est sauvegardée. Cependant, il est nécessaire de modifier l'application considérée en la liant avec les bibliothèques du système de sauvegarde de points de reprise utilisé.

Une autre approche consiste à utiliser un système d'exploitation spécifique permettant d'effectuer des sauvegardes de points de reprise de manière transparente. Par exemple, BLCR [102] qui est mis en œuvre sous forme de module du noyau permet de sauvegarder et restaurer des processus. Kerrighed [131] est un exemple de système d'exploitation distribué offrant des mécanismes de sauvegarde et restauration de points de reprise. Cette approche offre plus de flexibilité par rapport aux applications car elle ne nécessite pas de recompilation de ces dernières. Cependant, elle nécessite des modules noyau ou des systèmes d'exploitation qui ne sont pas forcément disponibles sur la grille.

Une variante de l'approche précédente consiste à encapsuler les applications dans des conteneurs (à la manière des conteneurs décrits dans la section 1.2.1.1). OpenVZ [24] est un système de conteneur qui permet d'isoler les processus en leur définissant leurs propres espaces de nommage. Ce type de système permet de sauvegarder l'état d'un conteneur (et donc, de tous les processus exécutés dans le conteneur). Cette approche dispose des mêmes avantages et inconvénients que l'utilisation d'un système d'exploitation spécifique, à savoir : elle offre une plus grande flexibilité par rapport aux applications (pas besoin de recompilation), mais elle reste limitée dans le degré de personnalisation de l'environnement d'exécution offert aux applications (l'environnement d'exécution est forcément le même que celui du nœud hôte).

Sauvegarde et restauration de points de reprise de tâches Les techniques présentées pour la sauvegarde et restauration de points de reprise de processus peuvent s'appliquer pour les tâches distribuées : utilisation de systèmes spécifiques, de système d'exploitation spécifique ou de conteneur.

DejaVu [154], BLCR [102], OpenVZ [24] ou Kerrighed [131] sont capables de sauvegarder et restaurer des applications distribuées en interceptant les appels systèmes liés à la gestion des *sockets* TCP.

Le problème de ces systèmes est principalement lié à la flexibilité offerte aux applications. Ces systèmes sont conçus pour des environnements d'exécution spécifiques s'exécutant sur des systèmes d'exploitation spécifiques, laissant peu de marge dans le degré de personnalisation offert aux utilisateurs.

Mécanismes de virtualisation Une autre approche consiste à utiliser les mécanismes de virtualisation pour sauvegarder et restaurer les applications durant leur exécution. Par exemple, SGE (*Sun Grid Engine*, maintenant OGE, *Oracle Grid Engine*) a été étendu avec XGE (*Xen Grid Engine*) [72] pour gérer les machines virtuelles. Les utilisateurs peuvent définir leur propre environnement d'exécution dans les machines virtuelles. De même Moab [18] a été mis à jour afin d'intégrer les fonctionnalités de virtualisation.

Cependant, une fois encore, ces travaux se concentrent principalement sur un problème particulier (par exemple, la gestion des réseaux, le démarrage et l'arrêt des machines virtuelles) et ne traitent pas des problèmes de la gestion complète (déploiement, configuration, surveillance, contrôle et redéploiement) des machines virtuelles à l'échelle de la grille.

Finalement, les mécanismes et outils présentés ont des axes de flexibilité orientés vers la gestion des processus, des tâches ou des machines virtuelles. Cependant ces mécanismes ne traitent pas de l'exécution des tâches interruptibles dans les grilles en utilisant les axes de flexibilité fournis par les machines virtuelles et en pouvant s'intégrer à une infrastructure existante sans modification ou reconfiguration de son gestionnaire de ressources.

2.2 Objectifs

Nos objectifs se placent dans la conception d'un système utilisant la flexibilité offerte par les technologies de virtualisation pour la gestion d'applications distribuées et interruptibles qui répond aux critères suivants :

Viser des infrastructures distribuées sur plusieurs sites Le système que l'on conçoit doit s'adapter à des infrastructures distribuées sur plusieurs sites. Pour ce faire, Saline doit pouvoir configurer les infrastructures lorsque cela est possible (comme dans Grid'5000) ou utiliser les environnements préconfigurés des nœuds d'exécution.

Viser la transparence à l'intégration dans l'infrastructure ciblée Le système conçu ne doit pas nécessiter de modifications de l'infrastructure qu'il doit gérer. Pour ce faire, il doit s'interfacer avec le gestionnaire de ressources de l'infrastructure.

Être non intrusif par rapport aux applications Nous voulons étendre la gamme des tâches interruptibles, généralement liée aux tâches de type sac de tâches (*bag of tasks*) à un plus grand ensemble de tâches grâce aux fonctionnalités offertes par les systèmes de virtualisation (personnalisation des environnements d'exécution, possibilité de sauvegarde et restauration des machines virtuelles).

Ces objectifs majeurs, mènent à des objectifs de mise en œuvre (liés à l'utilisation des mécanismes de machines virtuelles sur une infrastructure distribuée de type grille) qui sont :

La gestion du cycle de vie d'une collection de machines virtuelles sur la grille Les applications sont encapsulées dans des collections de machines virtuelles. Ces

collections de machines virtuelles sont gérées sur la grille. Saline doit utiliser les mécanismes des machines virtuelles (sauvegarde/restauration et migration) pour gérer le cycle de vie des machines virtuelles.

La configuration du réseau Les collections de machines virtuelles doivent être indépendantes les unes des autres. Il faut configurer les adresses MAC et IP des machines virtuelles en fonction de leur appartenance à une collection donnée. De plus, cette configuration doit se faire sans engendrer de conflits avec les autres collections de machines virtuelles, mais aussi avec l'infrastructure matérielle.

La gestion des images des machines virtuelles D'un point de vue du système de fichiers, les machines virtuelles sont représentées par des fichiers dont les tailles peuvent varier de la centaine de méga-octets à la dizaine de giga-octets. Le système conçu doit gérer efficacement les images des collections de machines virtuelles.

2.2.1 Infrastructures et tâches ciblées par Saline

Les paragraphes suivants présentent les infrastructures et les types de tâches ciblées dans nos travaux.

2.2.1.1 Infrastructures de calcul

La figure 2.1 présente le type d'infrastructures considérées dans les travaux présentés dans ce chapitre. Celles-ci sont des grilles composées de grappes appartenant à différents sites et interconnectées par un réseau étendu. Ces grappes sont gérées par un système à exécution par lots avec un nœud maître qui permet de gérer les ressources à l'échelle de la grille et de pouvoir déployer des systèmes de virtualisation comme KVM ou Xen, sur les nœuds de la grappe. Le nœud maître possède également les fonctionnalités de nœud de stockage pour les images des machines virtuelles.

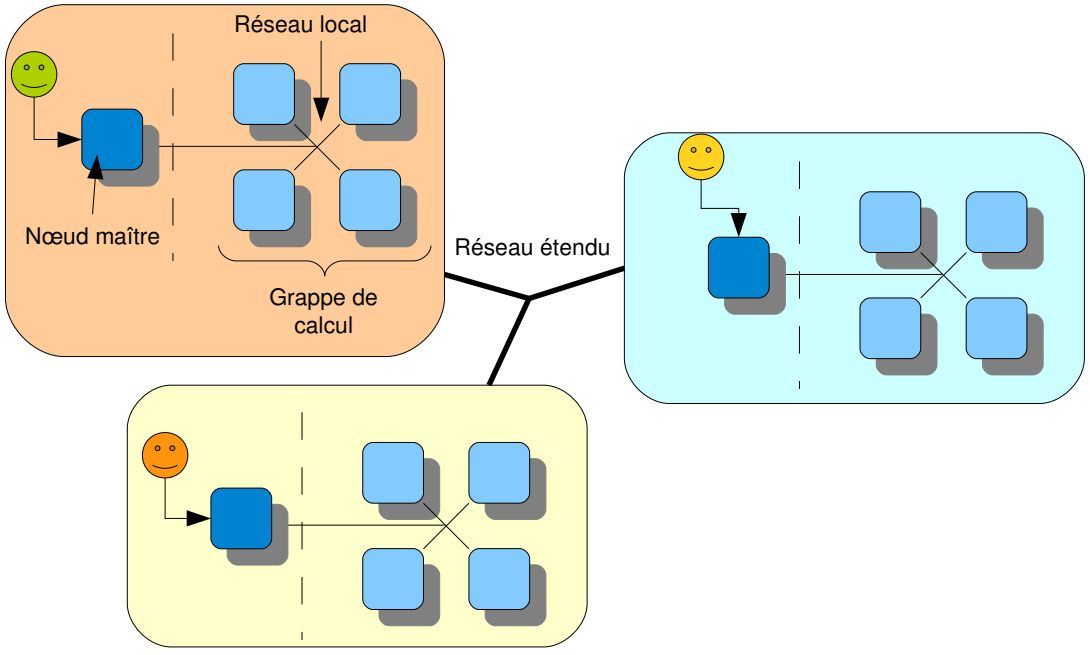


FIGURE 2.1 – Architecture de grille

Validation de notre approche sur Grid'5000 Grid'5000 [1, 53] est une grille scientifique constituée de 9 sites situés en France dont le but est d'offrir à l'utilisateur un environnement qui peut être totalement configuré et reconfiguré avec des droits d'accès administrateur sur les nœuds dont il dispose. Grid'5000 est constitué d'un ensemble de nœuds offrant un total de plus de 5 000 processeurs pour les expérimentations de systèmes distribués à large échelle. Cette plate-forme composée de nœuds hétérogènes est exploitée par de multiples utilisateurs simultanément qui peuvent réserver des nœuds et qui exécutent les tâches non-interruptibles et interruptibles. Les tâches non-interruptibles sont des tâches exécutées pour un temps déterminé sur les nœuds réservés pour cette tâche et qui ne peuvent être interrompues avant la fin du temps alloué pour leur exécution. Les tâches interruptibles sont des tâches exécutées sur des nœuds disponibles (non attribués à une réservation) et peuvent être interrompues à tout moment.

La gestion des ressources de Grid'5000 repose sur le système de gestion des ressources (*Resource Manager*) OAR [56] ainsi que sur la suite logicielle de déploiement Kadeploy [53]. Un utilisateur peut trouver des ressources sur la grille, les réserver et déployer une pile logicielle personnalisée directement sur ces nœuds. Cependant, aucune fonctionnalité permettant de faire une sauvegarde de l'état de l'environnement déployé en cours d'exécution afin de le restaurer plus tard n'est fournie. Il n'est donc pas possible de sauvegarder l'environnement complet d'une tâche interruptible au cours de son exécution afin de le redémarrer plus tard depuis la dernière sauvegarde.

2.2.1.2 Intégration de manière transparente aux gestionnaires de ressources existants sur les infrastructures ciblées

Saline est conçu pour gérer des tâches interruptibles dans des infrastructures distribuées à large échelle. Pour ce faire, Saline interagit avec le gestionnaire de ressources de l'infrastructure distribuée permettant ainsi de l'étendre de manière transparente.

Dans le cadre des tâches interruptibles, deux types de gestionnaire de ressources peuvent être considérés : ceux récupérant les nœuds dont ils ont besoin de manière « brutale », dans ce cas, les tâches ne sont pas prévenues avant d'être arrêtées, et ceux récupérant les nœuds en envoyant un signal au préalable, cela permet à la tâche interruptible d'anticiper son arrêt imminent. Le système que l'on propose interagit avec le gestionnaire de ressources de la grille dans les deux cas :

- Dans le cas où aucun signal n'est émis, cas étudié dans la section 2.3, des mécanismes de sauvegarde des machines virtuelles permettent de les redéployer par la suite depuis leur dernière sauvegarde. Dans ce cas, Saline interagit avec le gestionnaire de ressources de l'infrastructure en utilisant les fonctionnalités suivantes : *découverte de ressources disponibles sur l'infrastructure* et *réservation de ressources*.
- Dans le cas où un signal est émis, cas étudié dans la section 2.4, il suffit de l'intercepter afin de le prendre en compte et de le traiter en ayant recours à des mécanismes de sauvegarde des machines virtuelles ou de migration des tâches interruptibles. Dans ce cas, Saline interagit avec le gestionnaire de ressources de la grille en utilisant les mêmes fonctionnalités que dans le cas précédent, plus celle d'*attente de notification*.

2.2.1.3 Les tâches

Les types de tâche que nous ciblons sont les applications distribuées qui peuvent être interrompues (une fois interrompue, les tâches interruptibles peuvent être redémarrées à l'endroit où elles ont été interrompues) et qui peuvent être exécutées sur une grappe.

De plus, on considère que les processus constituant une tâche ne peuvent communiquer qu'entre eux et non avec des processus extérieurs à la tâche.

2.3 Gestion du cycle de vie des collections de machines virtuelles dans Saline

Cette section présente l'architecture ainsi que la mise en œuvre de Saline. Saline est un système générique pouvant interagir avec le gestionnaire de ressources de la grille sans modification de celui-ci. Saline soumet et gère les tâches interruptibles jusqu'à la fin de leur exécution en les encapsulant dans des machines virtuelles pouvant communiquer les unes avec les autres et cela de manière transparente par rapport à la tâche à exécuter et à l'utilisateur ayant soumis la tâche.

Pour ce faire, nous utilisons les mécanismes de sauvegarde et restauration des machines virtuelles et de migration qui permettent de déplacer les machines virtuelles entre des nœuds d'une grille [82, 86].

2.3.1 Les propriétés du protocole TCP utilisées par Saline

Saline est conçu pour gérer des machines virtuelles communiquant en utilisant un protocole de communication fiable comme TCP [42, 139]. Le protocole de communication TCP assure ne perdre aucun message et les délivre dans l'ordre d'émission au destinataire. Pour ce faire, des acquittements (ACK) sont envoyés par le destinataire à l'émetteur. Si certains messages ne sont pas reçus, l'émetteur les renvoie. Ainsi, dans tous les cas, s'il y a une perte des messages émis, ou des messages d'acquittement renvoyés par le destinataire, ceux-ci sont retransmis plusieurs fois pendant un laps de temps qui peut généralement être configuré. Celui-ci, selon les configurations, peut varier de plusieurs secondes à plusieurs minutes.

Ainsi, grâce au protocole TCP et aux mécanismes de synchronisation des horloges des nœuds, il est possible de suspendre et déplacer d'un site à un autre une collection de machines virtuelles en gardant un état cohérent du réseau au sein de cette collection [70]. De cette manière, les opérations d'infrastructure (suspension et migration de la collection) se font de manière transparente pour la tâche interruptible en cours d'exécution.

2.3.2 Architecture de Saline

L'architecture de Saline s'articule autour de différents modules représentés dans la figure 2.2, que nous explicitons dans les paragraphes suivants.

2.3.2.1 La liste des tâches interruptibles

Cette liste contient l'ensemble des tâches interruptibles (dans ce contexte une tâche correpond à une collection de machines virtuelles qui exécute une application). Pour chaque tâche son état est mémorisé : *jamais exécutée*, *en cours d'exécution*, *suspendue en attente de ressources*. Cette liste est gérée par le module de configuration initiale des machines virtuelles, par le gestionnaire de placement des machines virtuelles, et par le gestionnaire d'infrastructure. Afin d'éviter tout conflit entre les accès concurrents à la liste, des verrous sont définis assurant à chaque instant qu'un seul gestionnaire puisse accéder à la liste. La liste des tâches est définie à l'échelle d'un site. Dans une grille, chaque site dispose donc de sa propre liste des tâches à exécuter.

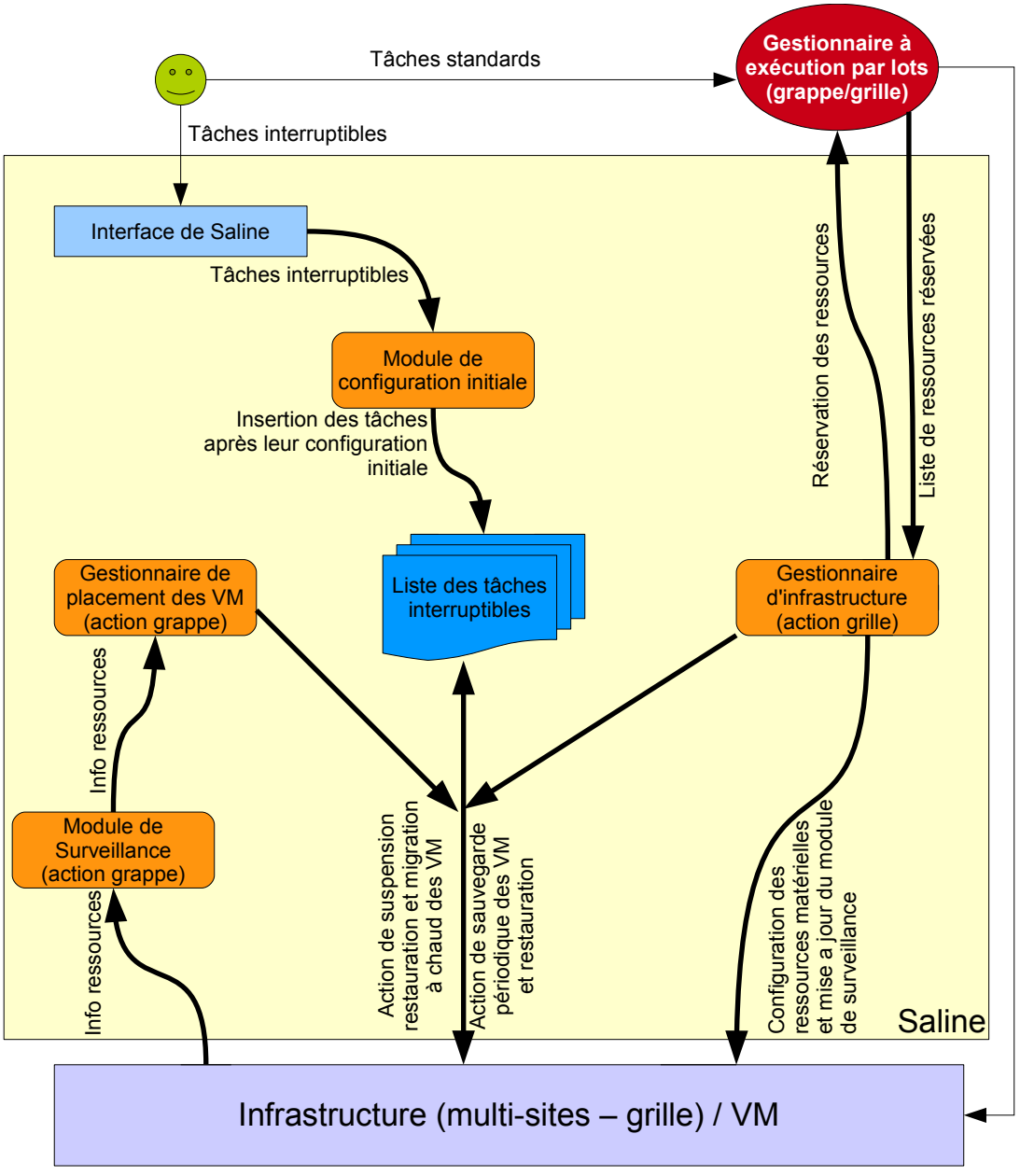


FIGURE 2.2 – Architecture de Saline

2.3.2.2 Module de configuration initiale

Le module de configuration initiale des machines virtuelles a pour rôle de réceptionner les nouvelles tâches interruptibles, de les encapsuler dans les machines virtuelles et de configurer les machines virtuelles pour qu'elles puissent être mises dans la liste des tâches avec l'état *jamais exécutée*. La configuration initiale doit de plus configurer l'interface réseau des machines virtuelles (adresses MAC et IP) pour que les machines virtuelles d'une collection puissent communiquer entre-elles, et pour qu'elles puissent être déplacées d'un site à un autre sans risque de conflit du point de vue du réseau.

2.3.2.3 Gestionnaire de placement des machines virtuelles

Le gestionnaire de placement des machines virtuelles a la charge d'exécuter au mieux les tâches interruptibles sur les ressources disponibles. Ce gestionnaire a son périmètre d'action à l'échelle de la grappe, il place et déplace les machines virtuelles de manière optimisée (en fonction de la charge des nœuds, par exemple) sur la grappe grâce aux informations récupérées par le module de surveillance.

2.3.2.4 Module de surveillance des machines virtuelles

Le module de surveillance interagit avec le module de placement des machines virtuelles. Il surveille en permanence l'état des ressources et des machines virtuelles afin de signaler une sur-utilisation ou une sous-utilisation des ressources au module de placement des machines virtuelles qui peut alors prendre les décisions nécessaires pour optimiser le placement des machines virtuelles. Ce gestionnaire a son périmètre d'action à l'échelle de la grappe.

2.3.2.5 Gestionnaire d'infrastructure

Le gestionnaire d'infrastructure est en charge de deux missions : sauvegarder périodiquement les images des machines virtuelles et surveiller l'état de la liste des tâches interruptibles pour réserver des nœuds sur la grille.

Sauvegarde périodique des machines virtuelles Le gestionnaire d'infrastructure s'occupe périodiquement de faire des sauvegardes des machines virtuelles afin de pouvoir les redémarrer plus tard en cas de besoin.

Contrôle de l'état des nœuds Le gestionnaire scrute périodiquement l'état de la liste des tâches interruptibles. Si des tâches sont dans la liste (état *jamais exécutée* ou *suspendue en attente de ressources*), cela signifie qu'il n'y a pas suffisamment de place sur les ressources pour permettre leur exécution. Dans ce cas, le gestionnaire d'infrastructure soumet une requête au gestionnaire de ressources de la grille afin d'obtenir de nouveaux nœuds (cela se fait en réalisant une requête standard de réservation des ressources auprès du gestionnaire de ressources). Ce module a son périmètre d'action à l'échelle de la grille, il est capable de déplacer des tâches interruptibles d'un site à un autre, et cela de manière transparente pour les applications (la migration n'a lieu que si les ressources obtenues appartiennent à un autre site).

2.3.3 Choix de mise en œuvre

Cette section discute et présente les choix effectués pour la mise en œuvre de l'architecture présentée dans la section précédente. Les points abordés portent sur :

- la mise en place de l'environnement nécessaire pour l'exécution des machines virtuelles sur les ressources matérielles,
- la création de l'image de la machine virtuelle et l'installation et la configuration de la tâche interruptible,
- la gestion du stockage des images des machines virtuelles,
- la configuration du réseau des machines virtuelles pour assurer leur mobilité sur la grille,
- la surveillance des machines virtuelles au cours de leur exécution,
- le placement intelligent des machines virtuelles en fonction de la charge des ressources matérielles,
- la terminaison de la tâche interruptible, la sauvegarde des résultats et la libération des ressources.

2.3.3.1 Gestion de l'environnement d'exécution des machines virtuelles sur les ressources

Saline propose une couche générique permettant l'installation et la configuration d'un hyperviseur sur les nœuds cibles si ceux-ci en sont dépourvus. Saline étant conçu pour gérer les tâches interruptibles sur Grid'5000, pour configurer les ressources de la grille, Saline s'appuie sur l'infrastructure logicielle existante de Grid'5000, à savoir les outils de la famille Kadeploy [53]. Ainsi, en utilisant Kadeploy, Saline peut déployer et configurer sur une grappe l'environnement nécessaire à l'exécution des machines virtuelles : installation et configuration d'un système d'exploitation ainsi que d'un système de virtualisation (un hyperviseur).

Cependant, l'actuelle évolution de l'utilisation des machines virtuelles dans les grilles nous laisse penser que, de plus en plus, les ressources bénéficieront par défaut d'une pile logicielle incluant des mécanismes de gestion des machines virtuelles : l'utilisation de l'hyperviseur KVM intégré directement dans la plupart des distributions Linux récentes en est un exemple. Ainsi, des discussions au comité de direction de Grid'5000 sont en cours afin d'évaluer l'impact de l'utilisation des systèmes de virtualisation sur les environnements par défaut déployés sur les ressources. Une telle évolution ne nécessiterait plus de déployer toute la pile logicielle en commençant par le système d'exploitation sur les ressources.

2.3.3.2 Création de l'image des machines virtuelles et installation de l'application

Saline utilise et configure les images des machines virtuelles selon les spécificités des tâches interruptibles. Deux cas peuvent être considérés : la tâche interruptible peut s'exécuter dans des environnements d'exécution standard (par exemple sur un système d'exploitation Linux) ou, la tâche interruptible nécessite un environnement particulier (par exemple des librairies spécifiques ne faisant pas partie des distributions standard). Si le premier cas peut être traité de manière simple en utilisant des images déjà configurées avec un environnement standard (*Virtual Applicances*) [15, 23, 25, 69], le second cas implique la création d'une image particulière. Différents outils permettent de créer des images de machines virtuelles personnalisées [26, 33]. C'est pourquoi, dans cette thèse, nous n'étudions pas plus en détail ce point et nous nous limitons à l'utilisation d'images proposant un environnement d'exécution standard avec des tâches interruptibles pouvant s'installer avec des outils standard (par exemple *APT* pour les distributions Linux de type Debian).

Gestion du disque dur virtuel d'une machine virtuelle avec la technique dite de la copie sur écriture La technique de la copie sur écriture (COW, *Copy-On-Write*) [51] s'applique sur le disque dur virtuel d'une machine virtuelle : cette technique consiste à sauvegarder chaque modification de l'image dans un fichier distinct de celui de l'image d'origine (voir la figure 2.3). Cette technique représente chaque disque virtuel avec une image de référence (*ref*) et une image contenant toutes les modifications par rapport à cette image de référence (*diff*). Un disque dur virtuel créé avec cette méthode nécessite l'image de référence et le fichier de modifications pour être cohérent.

L'intérêt majeur de cette technique réside dans le gain de place économisé lorsqu'on compare la taille d'une collection de machines virtuelles sans cette technique ($nb\_ordi\_virt \times taille\_ref$) et celle d'une collection de machines virtuelles avec cette technique ($taille\_ref + \sum_{i=1}^{nb\_ordi\_virt} taille\_fichier\_diff_i$). Nous présentons dans la section 2.5.2 une évaluation de cette économie réalisée.

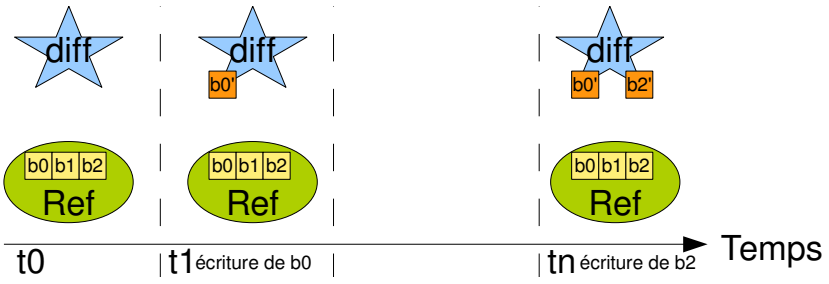


FIGURE 2.3 – Représentation de la technique de copie sur écriture
À l'origine, le disque dur virtuel de la machine virtuelle est le même que l'image de référence (disque dur virtuel = ref). Au fur et à mesure que le temps s'écoule, à chaque écriture dans le disque dur virtuel, cette écriture est faite dans un fichier séparé (disque dur virtuel = *diff* + *ref*).

Dans le reste de ce chapitre, sauf indication contraire, on considère l'utilisation de la technique de la copie sur écriture pour toutes les machines virtuelles. Cela implique qu'une fois sauvegardée, une machine virtuelle est composée : d'une image de référence (*ref*), d'un fichier de modifications spécifique généré par les copies sur écritures successives (*diff*), et d'un fichier spécifique comprenant l'état de sa mémoire et de ses registres CPU et des périphériques virtuels (*état*).

2.3.3.3 Stockage des images des machines virtuelles

La gestion des machines virtuelles concerne principalement la manière dont sont gérées leurs images. Deux techniques peuvent être considérées : les machines virtuelles sont stockées sur les nœuds d'exécution ou les machines virtuelles sont stockées sur un nœud dédié au stockage. La section 2.3.4 présente deux mises en œuvre de Saline, l'une dans laquelle les machines virtuelles sont stockées sur les nœuds, et l'autre dans laquelle le stockage des machines virtuelles est fait sur un nœud de stockage.

Quel que soit le type de solution de stockage utilisé, il est nécessaire de s'intéresser aux points suivants :

- le déploiement initial des images des machines virtuelles,
- la sauvegarde des images des machines virtuelles,
- le redéploiement des images préalablement sauvegardées,
- la gestion à chaud du placement des machines virtuelles sur les ressources.

a) Les machines virtuelles sont stockées localement sur les nœuds d'exécution

a1) Déploiement initial des images des machines virtuelles Le déploiement des machines virtuelles consiste à déployer et configurer une image de référence contenant l'environnement d'exécution des machines virtuelles. Cela revient à déployer l'image de référence (ref) d'un nœud de stockage vers n nœuds d'exécution. Deux approches peuvent être considérées : les images des machines virtuelles sont stockées sur les nœuds d'exécution en permanence ou, les images des machines virtuelles sont copiées depuis un nœud de stockage dédié vers les nœuds d'exécution lors d'un déploiement.

- Quand les images sont stockées de manière permanente sur les ressources, « il suffit » de localiser l'image à déployer sur les ressources afin de la configurer et l'exécuter. Ce type d'approche pour le déploiement initial a l'avantage d'être performant, l'accès aux données se faisant depuis l'espace de stockage des nœuds d'exécution. Cependant, la gestion des mises à jour des images de référence peut se révéler complexe. Enfin, des problèmes de sécurité sont à traiter : il faut s'assurer que les images présentes sur les disques locaux des nœuds d'exécution ne puissent être corrompues malicieusement ou non. Dans le cas de Grid'5000, les utilisateurs peuvent déployer et redéployer des environnements d'exécution sur les nœuds en écrasant toutes les données présentes sur ces nœuds. Cette solution ne peut convenir.
- La copie efficace consiste à déployer les images d'un nœud de stockage sur l'ensemble des nœuds d'exécution en réalisant des copies optimisées. Ainsi, le nœud de stockage copie l'image de référence vers un sous-ensemble de nœuds d'exécution qui exécutent la même opération vers d'autres sous-ensembles, etc. De cette manière, les images sont copiées en parallèle entre les différents sous-ensembles.

  Le principal avantage de cette technique est qu'il n'est pas nécessaire de faire d'hypothèse sur un stockage permanent des images des machines virtuelles sur les ressources : les images sont copiées lors du déploiement. La seule hypothèse que nous avons à faire est celle de disposer d'un nœud de stockage pour chaque site : hypothèse validée par la présence d'un nœud maître (*frontend*) disposant d'un espace de stockage sur chaque site de Grid'5000. C'est cette technique que nous retenons pour Saline.

a2) Sauvegarde des images des machines virtuelles Pour rendre notre système générique et non-intrusif, des sauvegardes des images des machines virtuelles sont faites de manière périodique afin de pouvoir les redémarrer depuis leur dernière sauvegarde lorsque leur exécution a été interrompue et cela indépendamment des choix d'attribution et de préemption des ressources du gestionnaire de ressources de la grille (OAR dans le cas de Grid'5000). Pour toutes les machines virtuelles d'une même collection, deux éléments doivent être considérés : le disque dur virtuel (dans le cas de la technique de la copie sur écriture, celui-ci est constitué de l'image de référence – *ref* – et du fichier contenant les modifications – *diff* –) et l'état courant de la machine virtuelle, principalement, la mémoire et les registres (état). Ces deux éléments (*diff* + *état*) doivent être périodiquement sauvegardés pour chaque machine virtuelle de la tâche interruptible.

Comme dans le cas précédent, deux approches peuvent être considérées : les sauvegardes sont faites localement et stockées en permanence localement sur

chaque nœud, ou, une fois les sauvegardes faites, elles sont transférées vers un nœud de stockage.

- La sauvegarde des images localement permet de garder les images des machines virtuelles sur l'espace de stockage local des nœuds d'exécution. De cette manière, les machines virtuelles peuvent être restaurées lorsque le nœud est de nouveau disponible ou être copiées sur d'autres nœuds disponibles. Cette technique a l'avantage d'être performante du point de vue du temps de la sauvegarde de l'image.
  Cependant, une défaillance d'un nœud peut entraîner la perte des images contenues sur celui-ci et ainsi entraîner la perte de la tâche interruptible. Enfin, sauvegarder les images localement nécessite de pouvoir accéder aux données stockées sur ce nœud indépendamment de l'état du nœud : cela implique qu'une partie du nœud soit allouée exclusivement aux images des machines virtuelles afin que même dans le cas d'une préemption des ressources, ces images restent disponibles. Comme présenté dans le cas précédent, cette dernière hypothèse ne peut être vérifiée dans le cadre d'une utilisation standard de Grid'5000. Nous ne retenons pas cette solution.
- La copie des images des nœuds d'exécution vers un nœud de stockage des images peut amener à des problèmes de goulot d'étranglement du réseau sur le nœud de stockage dédié dans le cas de l'utilisation d'un grand nombre de machines virtuelles. Cependant, cette technique étant moins intrusive que celle d'un stockage permanent, c'est celle-ci que nous choisissons. De plus, afin d'éviter les problèmes liés au goulot d'étranglement du réseau sur le nœud de stockage, nous concevons et mettons en œuvre dans Saline un algorithme de gestion efficace des sauvegardes des machines virtuelles présenté dans la figure 2.6 et décrit dans la section 2.4.

a3) Redéploiement des images préalablement sauvegardées L'opération de déploiement des images préalablement sauvegardées consiste à redéployer les images (*ref* + *diff* + *état*) sur des ressources nouvellement acquises. Ce problème ne peut se comparer au déploiement initial des images des machines virtuelles. En effet, deux cas sont à étudier : les nœuds d'exécution disposent déjà des images des machines virtuelles à redémarrer (cas du stockage permanent des sauvegardes de machines virtuelles sur les ressources), ou différentes images doivent être transmises d'un nœud de stockage à n nouveaux nœuds d'exécution appartenant potentiellement à un autre site.

- Dans le cas où les nœuds d'exécution disposent déjà des images des machines virtuelles à redémarrer, les machines virtuelles sont restaurées depuis l'espace de stockage local. L'avantage de cette technique est la performance au redémarrage, les images se trouvant directement sur l'espace de stockage local. Son inconvénient est que sans mécanisme de migration des images sauvegardées il n'est pas possible de redémarrer les machines virtuelles sur d'autres ressources que celles où elles ont été sauvegardées. Au vu de cet inconvénient nous écartons cette stratégie.
- Dans le cas où un nœud de stockage est utilisé, alors il est nécessaire de redéployer les disques dur virtuels des machines virtuelles. Cela se fait en deux étapes : tout d'abord l'image de référence (à la manière d'un déploiement initial en utilisant une technique de copie efficace sur le fichier *ref*) est copiée sur les nœuds d'exécution, puis les fichiers spécifiques (fichiers des différences – *diff* –) à chaque machine virtuelle le sont également (ces fichiers étant tous

différents, ils doivent être copiés un par un, sans optimisation particulière pour gagner en efficacité). De même les fichiers d'état (état) sont copiés un par un. L'avantage de cette technique est que, mises à part les contraintes liées au nœud de stockage, aucune hypothèse n'est faite quand à la fiabilité de l'accès aux données des nœuds : les machines virtuelles sont stockées sur un nœud et copiées uniquement lors de leur redéploiement.

a4) L'ordonnancement des machines virtuelles Des mécanismes avancés permettant de faire de la migration à chaud des machines virtuelles, de la sauvegarde ainsi que du redémarrage des machines virtuelles, des optimisations selon certains critères (comme les performances, la consommation d'énergie, ...) sont autant d'éléments à prendre en compte pour une gestion efficace des machines virtuelles.

Cependant, ces techniques ont des contraintes particulières. Par exemple, la migration à chaud nécessite que l'image de la machine virtuelle se situe sur le nœud de départ et également sur celui de destination (c'est le protocole de migration à chaud qui impose cela). Ainsi, dans le cas où les images de machines virtuelles se situent directement sur les nœuds, avant de pouvoir faire une migration à chaud d'une machine virtuelle entre deux nœuds, il faut s'assurer qu'une copie de l'image de la machine virtuelle est faite sur le nœud de destination. Cela complexifie grandement l'opération de migration à chaud et nécessite d'arrêter la machine virtuelle, de copier son image, et de la restaurer afin de garder la cohérence de l'image entre les deux nœuds (ce qui revient à faire une migration à froid). Pour cette raison nous choisissons de mettre en œuvre dans la version de Saline utilisant l'espace de stockage local des nœuds, un gestionnaire de placement des machines virtuelles ne réalisant pas de migration à chaud des machines virtuelles.

La figure 2.4 résume les différents choix pris au cours de cette discussion sur l'utilisation d'un nœud de stockage et de mécanismes de copie efficace pour le stockage des machines virtuelles.

b) Les machines virtuelles sont stockées sur un système de fichiers distribué Les systèmes de fichiers distribué sont très utilisés dans le déploiement de système d'exploitation sur différentes ressources. Elle consiste à utiliser un système de fichiers distribué (on peut penser au traditionnel système de fichiers NFS [155] ou à des solutions plus récentes comme Lustre [157], Panasas [177], Parallax [175] ou BlobSeer [134]) qui sert d'espace de stockage pour les images des machines virtuelles. Dans ce cas, les nœuds peuvent exécuter une machine virtuelle « simplement » en utilisant le système de fichiers distribué sur lequel est stocké son image.

L'inconvénient de cette technique par rapport à l'objectif de transparence à l'installation de Saline sur l'infrastructure matérielle, est qu'elle est plus complexe à mettre en œuvre. Il est nécessaire de disposer de droits d'administration privilégiés sur des nœuds de stockage afin de les configurer. De même, il est nécessaire de configurer de manière adéquate les différents nœuds d'exécution.

Dans cette section, nous prenons le choix d'utiliser un serveur NFS qui est un compromis entre la relative simplicité et transparence de l'installation (exige un nœud de stockage fiable) et des performances qui diminuent plus le nombre de machine virtuelle est grand.

b1) Déploiement initial des images des machines virtuelles Du point de vue du déploiement initial, l'avantage principal du stockage sur système de

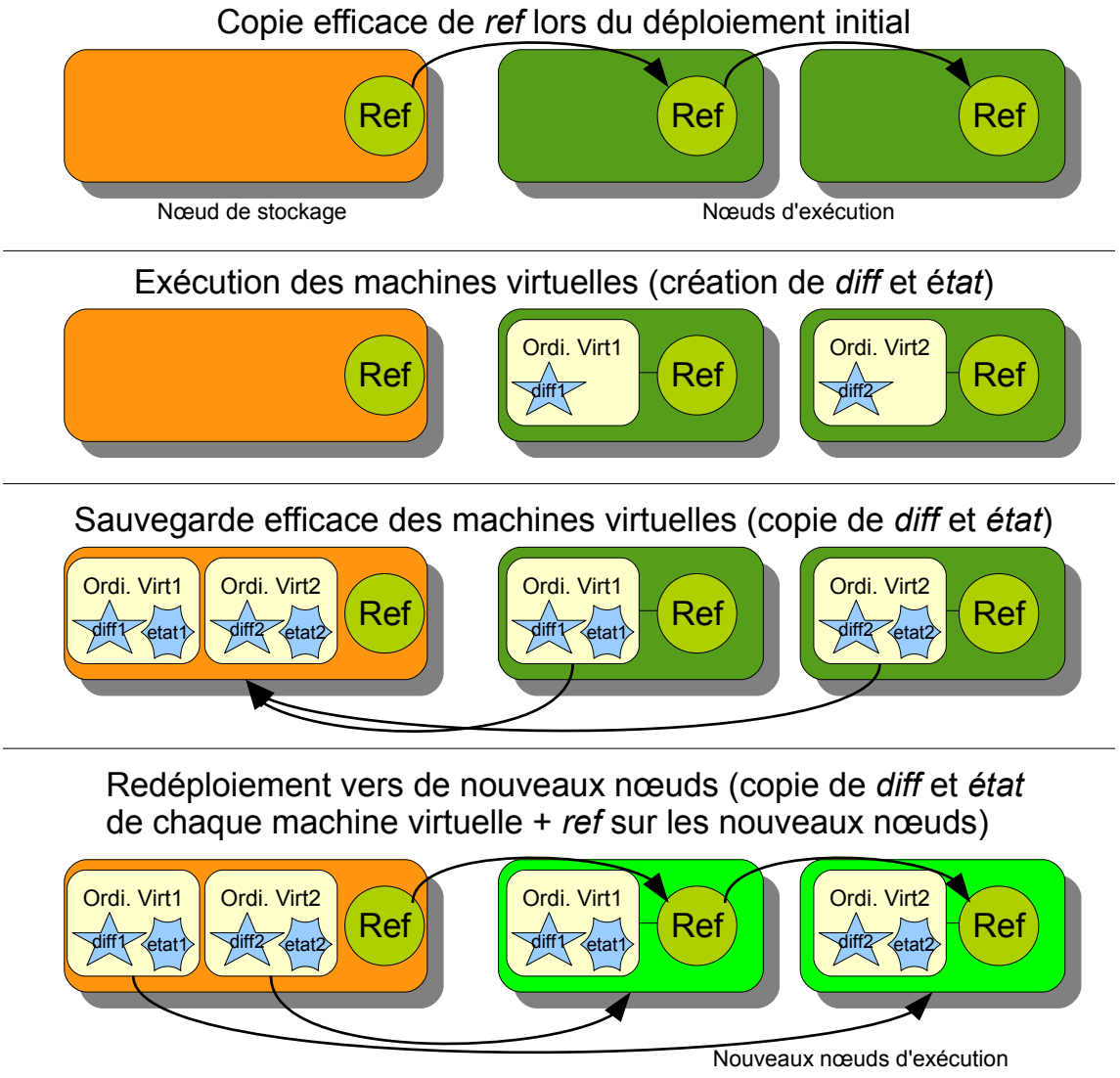


FIGURE 2.4 – Techniques de stockage utilisées : utilisation d'un nœud de stockage et des techniques de copie efficace
Les images de références (*ref*) sont copiées efficacement d'un nœud de stockage vers les nœuds d'exécution. Avec l'utilisation de la copie sur écriture, l'exécution des machines virtuelles crée un fichier de différence (*diff*) localement. Lors d'une sauvegarde d'une machine virtuelle, un fichier d'état (*état*) est également créé localement. Ces fichiers *diff* et *état* sont ensuite copiés efficacement sur le nœud de stockage. Lors d'un redéploiement, uniquement les fichiers *diff* et *état* sont copiés vers les nouveaux nœuds (le fichier *ref* est copié avec une technique de copie efficace). De plus, les nouveaux nœuds peuvent se trouver sur le même site (grappe) ou sur un site distant (grille).

fichiers distribué est que la mise à jour de l'image de référence (*ref*) peut se faire simplement sans avoir à gérer manuellement les multiples copies. Les inconvénients résident principalement dans la complexité de mise en œuvre et le passage à l'échelle aussi bien en nombre de nœuds que de machines virtuelles devant lire cette image de référence simultanément lors du démarrage des machines virtuelles.

b2) Sauvegarde des images des machines virtuelles L'utilisation d'un système de fichiers distribué amène à des limitations pour le passage à l'échelle : par exemple dans le cas de NFS où un nœud de maître partage son espace de stockage avec un ensemble de nœuds clients, de fortes limitations peuvent apparaître lors de la sauvegarde de multiples machines virtuelles (*état*) en entraînant un goulot d'étranglement pour les connexions réseaux du nœud maître.

b3) Déploiement des images préalablement sauvegardées L'utilisation d'un système de fichiers distribué dans ce cas dispose des mêmes avantages et inconvénients que dans celui du déploiement initial, à savoir, l'avantage de pouvoir déployer une image sur un nœud quelconque sans avoir à copier l'image préalablement (*ref* + *diff*), et l'inconvénient de ne fonctionner que pour une grappe pour un nombre limité de machines virtuelles.

b4) L'ordonnancement des machines virtuelles L'utilisation d'un système de fichiers distribué a l'avantage considérable de permettre des migrations à chaud des images des machines virtuelles de manière simple. C'est la raison principale pour laquelle nous proposons dans la section 2.3.4.2 une mise en œuvre de Saline s'appuyant sur une infrastructure avec un système de fichiers distribué. Cependant, comme décrit précédemment, son principal inconvénient reste le passage à l'échelle.

La figure 2.5 résume les différents choix pris au cours de cette discussion sur l'utilisation d'un système de fichiers distribué pour le stockage des machines virtuelles.

2.3.3.4 La configuration du réseau et la mobilité des machines virtuelles

La configuration du réseau des machines virtuelles doit être faite en considérant que les tâches interruptibles peuvent être déplacées d'un site à un autre de la grille. Cette opération consiste à assigner une unique adresse MAC et IP à chaque machine virtuelle lors de sa configuration initiale, cela afin d'éviter tout conflit futur entre les différentes adresses MAC et IP des autres nœuds ou machines virtuelles en cours d'exécution. De plus, afin de ne pas « perturber » la tâche interruptible considérée, ces adresses MAC et IP doivent être attribuées pour toute sa durée de vie.

a) Le protocole de configuration automatique des adresses IP : DHCP, *Dynamic Host Configuration Protocol* Le protocole DHCP permet la configuration des adresses IP des nœuds dans les grappes de calcul. Pour ce faire, un nœud maître reçoit les requêtes des nœuds d'exécution et leur attribue une unique adresse IP pendant une période déterminée, un bail. À la fin du bail, celui-ci peut être renouvelé, ou clôturé (l'adresse IP est alors de nouveau disponible). On peut ainsi penser au même protocole pour la configuration des machines virtuelles : les machines virtuelles envoient une requête de configuration à un nœud maître disposant du protocole de configuration automatique des adresses IP afin de recevoir la configuration nécessaire exactement comme le ferait un nœud. Cependant, ce protocole

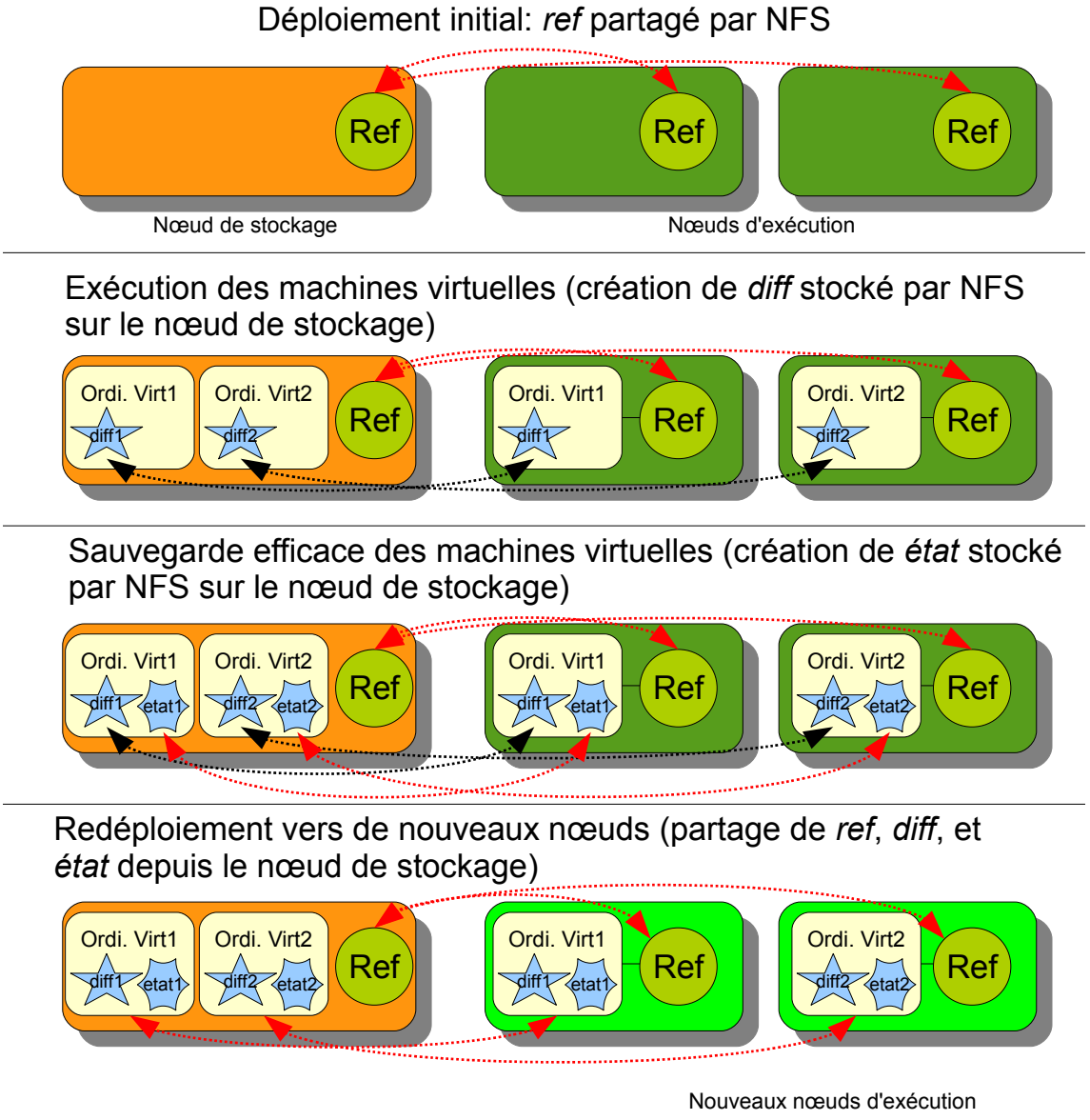


FIGURE 2.5 – Techniques de stockage utilisées avec un système de fichiers distribué
Les images de références (*ref*) sont partagées depuis le nœud de stockage par NFS. Avec l'utilisation de la copie sur écriture, l'exécution des machines virtuelles créent un fichier de différence (*diff*) sauvegardé en temps-réel via NFS sur le nœud de stockage. Lors d'une sauvegarde d'une machine virtuelle, un fichier d'état (*état*) est également créé et sauvegardé via NFS sur le nœud de stockage. Lors d'un redéploiement, les fichiers *ref*, *diff* et *état* sont partagés (toujours via NFS) vers les nouvelles ressources. Les nouveaux nœuds peuvent se trouver sur le même site (grappe) ou sur un site distant (grille). Dans ce cas, une copie préalable des fichiers *diff* et *état* sera nécessaire sur le serveur NFS du site distant (on considère que l'image *ref* est disponible sur tous les sites, ce n'est donc pas nécessaire de la copier).

est adapté à l'échelle de la grappe avec des nœuds fixes (n'ayant pas de contrainte de mobilité). D'autres solutions doivent être trouvées pour gérer les machines virtuelles pouvant être déplacées d'un site à un autre, à l'échelle de la grille. De plus, comme son nom l'indique, le protocole de configuration automatique des adresses IP configure automatiquement les adresses IP des nœuds, la configuration de leur adresse MAC restant à la charge d'un autre mécanisme (ce qui est compréhensible dans le cas de nœuds car l'adresse MAC est attribuée matériellement sur la carte réseau et n'a pas besoin d'être attribuée par un protocole spécial).

b) Proposition d'utilisation d'un protocole de configuration automatique des adresses MAC/IP à l'échelle de la grille pour machines virtuelles Pour résoudre le problème de l'attribution des adresses MAC et IP aux machines virtuelles à l'échelle de la grille, nous proposons l'utilisation d'un serveur d'adresses MAC et IP distribué. Cette approche nécessite un nœud maître par site qui est chargé d'assigner les adresses MAC et IP aux machines virtuelles appartenant à une même tâche interruptible. La configuration voulue est la suivante :

- toutes les machines virtuelles d'une même collection peuvent communiquer directement les unes avec les autres, c'est-à-dire que les machines virtuelles d'une collection doivent appartenir à un même sous-réseau,
- deux collections de machines virtuelles ne doivent pas pouvoir communiquer directement, c'est-à-dire que deux collections doivent avoir des sous-réseaux différents.

Pour cela, on définit un vecteur de 16 bits (*networkmapID*) qui correspond à 65 536 entrées (de 0 à 65 535). Chacune de ces entrées correspond à un identifiant de sous-réseau (*subnetID*) qui permet d'identifier les différentes collections de machines virtuelles. De plus, on définit également un vecteur de 8 bits (*vmmapID*) qui correspond à 254 entrées (de 1 à 254). Chacune de ces entrées correspond à un identifiant de machine virtuelle (*vmID*), qui permet au sein d'une collection de pouvoir identifier une machine virtuelle.

Quand une nouvelle tâche interruptible est démarrée, chaque machine virtuelle de la collection récupère un identifiant *subnetID* (commun à toutes les machines virtuelles de la collection) et un identifiant *vmID* (unique pour chaque machine virtuelle d'une collection).

b1) Configuration des adresses MAC Les adresses MAC sont composées de 48 bits (voir le tableau 2.1). Chaque carte réseau, virtuelle ou non, est dotée d'une adresse MAC dans laquelle les 24 premiers bits (MAC1) sont utilisés par le constructeur de la carte en tant qu'identifiant et les 24 derniers bits (MAC2) sont uniques à la carte réseau. MAC1, aussi appelé *Organizationally Unique Identifier* (OUI) est attribué aux constructeurs de manière centralisée par l'IEEE [13] : il n'y a donc pas de risque d'avoir de conflit sur MAC1. Dans l'idéal, il faudrait faire une demande auprès de l'IEEE afin d'obtenir une adresse MAC1 pour Saline, cependant, pour les besoins de notre prototype, nous choisissons dans la liste des attributions d'adresses MAC aux constructeurs une adresse MAC1 disponible.

MAC2 ($\approx$ 16 millions d'adresses distinctes) est utilisé pour différencier les différentes cartes réseau d'un même constructeur. Nous composons les premiers 16 bits de MAC2 avec le *subnetID* proposé par le nœud maître. Les 8 derniers bits de MAC2 sont définis avec le *vmID*. Par ce mécanisme de construction de l'adresse MAC, on assure son unicité.

48 bits		
24 bits MAC1 : Identifiant du fabricant	24 bits MAC2 : unique à la carte réseau	
	16 bits *subnetID*	8 bits *vmID*

TABLE 2.1 – Représentation de la configuration de l'adresse MAC

b2) Configuration des adresses IP Pour répondre aux objectifs de Saline, à savoir un minimum d'intrusion et de modification de l'infrastructure cible, nous nous concentrons uniquement sur la version d'IP la plus répandue au moment de l'écriture de ces lignes qui est IPv4 [138]. L'attribution des adresses IP aux machines virtuelles doit se faire en conformité avec les standards de l'IANA [12] en utilisant des adresses privées. Différentes plages d'adresses privées sont disponibles, nous choisissons celle nous permettant d'avoir un plus grand nombre de collections de machines virtuelles : la plage d'adresse 10.0.0.0/8 [146]. Bien entendu, si certaines adresses de cette plage sont réservées par l'infrastructure ciblée, il est possible de retirer ces adresses du *networkmapID* afin d'éviter tout conflit.

Chaque machine virtuelle se voit attribuer une configuration IP qui dépend des deux identifiants (*subnetID* et *vmID*) reçus par le nœud maître (voir le tableau 2.2). Les adresses IP sont composées de 32 bits. Dans une classe réseau de type 10.0.0.0/8, les 8 premiers bits (IP1) sont assignés par l'IANA. Les 24 bits suivants sont utilisés par Saline. Dans ces 24 bits, les 16 premiers bits (IP2) sont paramétrés avec le *subnetID* et les 8 derniers bits (IP3) sont paramétrés avec le *vmID*. Cela permet à notre architecture de choisir parmi 65 536 sous-réseaux (de 10.0.0/24 à 10.254.254/24) avec 254 machines virtuelles par sous-réseau (de 1 à 254). Si le nombre de machines virtuelles composant une tâche interruptible est plus grand que 254, il est possible d'agréger plusieurs sous-réseaux consécutifs.

32 bits		
8 bits IP1 : assigné par l'IANA	24 bits (utilisés par Saline)	
	IP2 : 16 bits *subnetID*	IP3 : 8 bits *vmID*

TABLE 2.2 – Représentation de la configuration de l'adresse IP

2.3.3.5 Surveillance des nœuds d'exécution et des machines virtuelles

Afin de déterminer quelles sont les actions à exécuter (migration, suspension, reprise) pour optimiser l'exécution des machines virtuelles, Saline doit avoir connaissance des informations sur l'état des ressources qu'il gère.

Dans certains cas, ces informations peuvent être données par le gestionnaire de ressources de la grille, mais dans d'autres cas, Saline déploie ses propres outils de surveillance. Différents outils existent dans le domaine de la surveillance de ressources : la commande UNIX élémentaire *ping* ou les systèmes évolués de surveillance comme Ganglia [124],

Nagios [48] en sont des exemples. Dans les mises en œuvre de Saline présentées dans la section 2.3.4, nous utilisons le mécanisme de *ping* ainsi que le système de surveillance Ganglia.

2.3.3.6 Placement intelligent des machines virtuelles

Le placement des machines virtuelles sur les nœuds de l'infrastructure se fait par l'action du gestionnaire de placement des machines virtuelles. Son périmètre d'action est sur la grappe. Pour répondre aux objectifs de simplicité et de transparence d'intégration par rapport à l'infrastructure matérielle, nous mettons en œuvre un algorithme de placement des machines virtuelles en *round-robin*. Cette politique, qui a le mérite d'être simple à mettre en œuvre sur l'infrastructure matérielle, ne permet cependant pas de placer et déplacer les machines virtuelles en fonction de la charge des nœuds.

Il est également possible de faire interagir Saline avec un outil spécialisé dans le placement des machines virtuelles sur les ressources physiques comme Entropy [105]. Cet outil est capable de gérer de manière efficace le placement des machines virtuelles sur les nœuds en les déplaçant à chaud. Pour ce faire, Entropy nécessite l'utilisation d'un outil de surveillance des ressources (comme ceux présentés dans la section 2.3.3.5), et en fonction des informations reçues, il est capable de prendre des décisions en accord avec les politiques définies et de planifier les différentes étapes nécessaires afin d'obtenir une configuration optimale du placement des machines virtuelles sur les nœuds.

2.3.3.7 Terminaison d'une tâche interruptible, sauvegarde des résultats et libération des ressources

La terminaison des machines virtuelles consiste à sauvegarder les calculs effectués, à libérer les ressources de la tâche interruptible et libérer les adresses MAC et IP préalablement attribuées. Aucun contrôle de succès de l'application n'est effectué : Saline apporte de la tolérance aux fautes à l'infrastructure, et non directement à l'application. Enfin, des mécanismes permettant de contrôler qu'une tâche ne boucle pas indéfiniment sont à mettre en œuvre : une politique élémentaire serait de définir un temps maximum d'exécution par collection de machines virtuelles.

2.3.4 Présentation des deux mises en œuvre proposées

Deux mises en œuvre de Saline ont été effectuées. La première est fondée sur le stockage des machines virtuelles localement sur les nœuds (voir figure 2.4) et est totalement non-intrusive par rapport à l'infrastructure matérielle ciblée. La seconde, plus intrusive, consiste à utiliser un système de fichiers, un système évolué de gestion des machines virtuelles à l'échelle de la grappe, ainsi qu'un système évolué de surveillance des nœuds (voir figure 2.5).

2.3.4.1 Mise en œuvre fondée sur un stockage local des machines virtuelles

Cette section présente la mise en œuvre de Saline, non-intrusive, fondée sur un stockage des machines virtuelles sur l'espace de stockage local des nœuds. Dans cette mise en œuvre, une instance de Saline est exécutée sur chaque site.

a) Le module d'initialisation des machines virtuelles Il réceptionne les tâches interruptibles et les « encapsule » dans des machines virtuelles. Dans cette mise en œuvre, chaque site exécute une instance de Saline, ce qui signifie que chaque site dispose de son module d'initialisation. Ces différents modules communiquent entre

eux pour attribuer de manière coordonnée les adresses MAC et IP aux machines virtuelles.

a1) Mise en œuvre de la gestion des applications L'application ainsi que son environnement sont installés lors de la phase de création de l'image de la machine virtuelle. Lorsqu'un utilisateur exécute une tâche interruptible qu'il confie à Saline, Saline exécute cette tâche dans les machines virtuelles. Les machines virtuelles de la collection sont configurées avec un jeu de clés *ssh* : il est possible à une machine virtuelle de la collection de se connecter à une autre machine virtuelle, ce qui est utile par exemple dans le cas de l'exécution d'une application MPI.

Concrètement, l'utilisateur donne en entrée à Saline un script. Saline exécute ce script dans la machine virtuelle dont le *vmID* est égal à 1. C'est donc ce script qui contient toutes les informations de configuration de l'application distribuée à exécuter. De plus, ce script est instrumenté de manière à surveiller l'état de la tâche en cours d'exécution. Une fois la tâche terminée, la machine virtuelle numéro 1 fait une requête auprès de l'instance de Saline en charge de sa gestion. Saline récupère les résultats copiés dans un répertoire spécifique des machines virtuelles (*/tmp/res*). Enfin, Saline détruit l'ensemble des machines virtuelles de la collection et libère le *subnetID* correspondant à la collection dans la table des *networkmapID*.

a2) Configuration des adresses MAC/IP Nous avons fait l'hypothèse qu'il existe un nœud maître par site. Saline s'exécute sur les nœuds maîtres des différents sites. Une instance de Saline, qui correspond à la gestion d'une collection de machines virtuelles s'exécute sur le nœud maître du site sur lequel les machines virtuelles s'exécutent.

La gestion des adresses MAC/IP repose sur la gestion de la table *networkmapID* : à chaque création d'une nouvelle collection de machines virtuelles, l'instance de Saline en charge de sa création interroge le *networkmapID* afin d'obtenir un *subnetID* pour sa collection. Nous proposons de mettre en œuvre le *networkmapID* par un fichier texte contenant l'ensemble des adresses de sous-réseaux autorisés (*subnetID*). Le *networkmapID* doit être partagé entre tous les nœuds maîtres. Pour ce faire, nous utilisons une solution centralisée : un site est en charge de la gestion du *networkmapID* pour lui-même et pour tous les autres sites.

Bien entendu, cette preuve de concept, bien que fonctionnelle, mène à un point unique de défaillance (SPOF, *Single Point Of Failure*) dans la gestion du *networkmapID*. Dans une prochaine version, nous mettrons en œuvre une approche distribuée : en effet, dans ce cas d'utilisation, le nombre de requêtes faites simultanément sur le *networkmapID* est faible. Cela nous permettrait d'utiliser un protocole à consistance forte entre les lectures et les écritures dans la table des *networkmapID* qui serait distribuée sur les différents sites.

Dans sa mise en œuvre, Saline utilise la bibliothèque Libvirt [17] (interface de programmation des machines virtuelles qui gère une dizaine d'hyperviseur dont KVM et Xen) pour manipuler les machines virtuelles. Le fonctionnement de Libvirt implique l'utilisation d'un fichier de description pour chaque machine virtuelle. Ce fichier de configuration écrit en XML permet par exemple de définir le nom de la machine virtuelle, le chemin d'accès à son disque dur virtuel, son adresse MAC. C'est lors de l'initialisation de la collection de machines virtuelles

que Saline configure ce fichier de configuration.

La configuration des adresses IP se fait lors du premier démarrage des machines virtuelles en récupérant les identifiants *subnetID* et *vmID*. Une méthode simple pour obtenir ces identifiants est d'exécuter un script automatiquement dans la machine virtuelle qui lors de son premier démarrage lit son adresse MAC.

b) Le gestionnaire de placement des machines virtuelles Dans la mise en œuvre fondée sur un stockage local, du fait que les images des machines virtuelles ne sont pas partagées entre les nœuds, il est difficile d'utiliser des techniques comme la migration à chaud. Ainsi, on attribue les machines virtuelles sur les nœuds en *round-robin*, sans optimisation. Le déploiement initial des images de référence se fait en utilisant TakTuk [60], un outil efficace de copie de fichiers (pour rappel, avec la technique de la copie sur écriture, le déploiement initial des machines virtuelles consiste à copier l'image de référence du nœud de stockage vers les nœuds d'exécution).

c) Le gestionnaire d'infrastructure Le gestionnaire d'infrastructure gère la sauvegarde des images des machines virtuelles, et contacte le gestionnaire de ressources de la grille en cas de préemption des ressources utilisées. Le gestionnaire de ressource est en charge de fournir au gestionnaire d'infrastructure une liste de nœuds disponibles localement ou sur un autre site. Si les nouveaux nœuds sont disponibles localement, les images des machines virtuelles sont déployées sur ces nœuds. Par contre, si les nouveaux nœuds sont disponibles sur un site distant, le gestionnaire d'infrastructure copie les images des machines virtuelles sur le nœud de stockage du nouveau site qui sera en charge de les déployer sur les nouveaux nœuds d'exécution.

De plus, périodiquement, les sauvegardes (fichiers de type différence et fichiers de type état) sont rapatriées des nœuds d'exécution vers le nœud maître du site à l'aide d'un mécanisme efficace, décrit dans la figure 2.6, permettant d'éviter un goulot d'étranglement pour le nœud maître. Le processus de sauvegarde ne s'intéresse qu'au fichier créé par la copie sur écriture, l'image de référence étant déjà sur le nœud maître.

De plus, c'est le gestionnaire d'infrastructure qui est en charge de redéployer les sauvegardes en cas de notification par le système de surveillance qu'une défaillance a eu lieu. Pour ce faire, des nouvelles ressources sont réservées (localement ou sur un autre site) et la collection est redémarrée depuis son dernier point de sauvegarde (le redéploiement de l'image de référence se fait à la manière d'un déploiement initial, en utilisant TakTuk, et le redéploiement des fichiers de différence et d'état se font de manière élémentaire avec la commande UNIX *scp*).

d) Surveillance des nœuds Le module de surveillance contrôle l'état des ressources en faisant périodiquement des *ping*. En cas de défaillance d'un nœud, une alerte est levée pour que le module de sauvegarde des machines virtuelles et de gestion des ressources puisse redéployer la collection touchée par la défaillance sur de nouveaux nœuds.

Pour valider cette mise en œuvre nous avons installé Saline sur les nœuds maître de Grid'5000. L'interface avec le gestionnaire de ressources de Grid'5000, OAR, est faite grâce aux commandes d'OAR (*oarsub*). Saline a pu gérer des collections de machines virtuelles sur la grille sans apporter de modification à l'infrastructure de la grille et en ayant à aucun moment des droits d'administration privilégiés sur les nœuds maître (*frontend*) de Grid'5000. Ceci valide que notre approche est non-intrusive.

2.3.4.2 Mise en œuvre fondée sur un système de fichiers distribué

Cette seconde mise en œuvre de Saline, plus intrusive, a pour but de vérifier nos hypothèses sur le fait qu'il est possible d'avoir des collections de machines virtuelles gérées localement sur un site de manière efficace et qui peuvent être déplacées d'un site à un autre de manière transparente.

a) Surveillance des ressources Ce module surveille l'état des ressources grâce au système de surveillance Ganglia. Ganglia donne une vue détaillée de l'état des nœuds et des machines virtuelles (taux d'utilisation du CPU, consommation mémoire, ...). Cependant, dans sa configuration actuelle, Ganglia ne gère qu'un sous-réseau.

b) Le gestionnaire de placement des machines virtuelles Dans cette mise en œuvre, Entropy est utilisé pour placer les machines virtuelles intelligemment sur les nœuds d'un site. Lorsque la charge des machines virtuelles est trop importante par rapport à la capacité des nœuds, Entropy arrête des machines virtuelles, et les remet en file d'attente.

c) Le module de configuration initial Le module de configuration initial réceptionne les tâches interruptibles et les « encapsule » dans des machines virtuelles. Dans sa mise en œuvre actuelle, cette solution est limitée par le fait que Ganglia est configuré pour ne gérer qu'un seul sous-réseau, ce qui, dans la terminologie de Saline signifie que cela ne correspond qu'à une seule tâche interruptible.

De plus, la mise en œuvre actuelle de Kentropy (outil automatisant le déploiement d'Entropy sur Grid'5000) repose sur le service DHCP de Grid'5000 pour récupérer les adresses IP des machines virtuelles. Cela signifie que les tâches que l'on exécute ne peuvent être distribuées dans différentes machines virtuelles : lors d'une migration d'un site à un autre, rien n'assure que les machines virtuelles ne changeront pas d'adresse IP.

d) Le gestionnaire d'infrastructure Le gestionnaire d'infrastructure gère la sauvegarde des images de machines virtuelles, et contacte l'ordonnanceur de la grille lorsque le nombre de nœuds est insuffisant.

Pour valider cette mise en œuvre, n'ayant pas les droits d'administrateur sur les nœuds maîtres de Grid'5000, nous avons réservé sur chaque site un nœud que nous avons configuré en nœud maître. Nous avons alors pu installer Entropy et Ganglia. Cette validation de notre prototype a permis de gérer efficacement des machines virtuelles sur une grappe, en étant capable au besoin de les déplacer sur d'autres sites de la grille.

2.4 Saline et les ordonnanceurs émettant un signal avant préemption des ressources

Cette section présente le cas d'utilisation de Saline avec un ordonnanceur émettant un signal avant préemption des ressources. Ce cas d'utilisation est identique à celui présenté dans la section 2.3 à l'exception du fait, qu'une sauvegarde forcée peut être exécutée lors de la réception du signal avant préemption des ressources.

En effet, certains ordonnanceurs peuvent fournir un signal informant de l'arrêt imminent et brutal de la tâche interruptible. Quand ce type d'évènement intervient, Saline demande à l'ordonnanceur d'autres ressources sur la grille. Si d'autres ressources sont trouvées (i), alors les machines virtuelles sont déplacées vers les nouveaux nœuds. Sinon (ii), les machines virtuelles doivent être suspendues sur un nœud de stockage dédié.

Dans tous les cas, le défi consiste à libérer les nœuds impactés en un minimum de temps. Si le cas (i) peut être résolu en copiant directement les images du nœud d'origine à celui de destination, le cas (ii) nécessite un mécanisme avancé et efficace permettant de sauvegarder les images des machines virtuelles sur le nœud de stockage distant. Dans ce second cas, le problème est que la copie de n nœuds d'exécution vers 1 nœud de stockage entraîne une saturation du réseau du nœud de stockage, ce qui peut sensiblement augmenter le temps nécessaire à la sauvegarde des machines virtuelles et donc celui nécessaire à la libération des nœuds d'exécution. De plus, des copies « naïves » ne considèrent pas le fait qu'il faut libérer le maximum de nœuds en un minimum de temps. Par exemple, certains nœuds peuvent être retirés de la réservation et alloués à des tâches de priorité supérieure pendant que les autres continuent de fonctionner pour le compte de la tâche interruptible. En d'autres termes, nous devons (i) libérer tous les nœuds le plus rapidement possible en régulant la copie des machines virtuelles vers le nœud de stockage, mais aussi (ii) libérer le plus de nœuds le plus rapidement possible en regroupant les machines virtuelles sur des sous-ensembles de nœuds. Pour faire cela, en considérant que les nœuds disposent de suffisament d'espace de stockage localement, nous proposons l'algorithme présenté dans la figure 2.6 et illustré par la figure 2.7.

Tant qu'il y a au moins un nœud qui attend de transférer une image sur le nœud de stockage, alors faire :
- copier une image depuis le nœud contenant le moins d'images vers le nœud de stockage (si tous les nœuds contiennent exactement le même nombre d'images, alors choisir un nœud au hasard).
- depuis chaque autre nœud, copier une image vers un nœud sauf si :
  * le nœud de destination est déjà en train d'envoyer ou de recevoir une image,
  * le nœud de destination a moins d'images que l'émetteur,
  * le nœud courant est déjà en train d'envoyer ou de recevoir une image.

FIGURE 2.6 – Algorithme de copie efficace de l'état des collections de machines virtuelles depuis leurs nœuds d'exécution vers le nœud de stockage
Dans cet algorithme, pour plus de clarté, nous utilisons le terme *image* pour désigner la sauvegarde d'une machine virtuelle (fichier de différence (*diff*) créé au moment de la copie sur écriture + fichier contenant l'état de la machine virtuelle (*état*) – voir la section 2.3.3.2).

Extension possible de l'algorithme L'algorithme présenté gère la récupération des machines virtuelles des différents nœuds vers un nœud de stockage de manière ordonnée. Deux extensions peuvent être proposées. La première est d'avoir plusieurs nœuds de stockage : la copie des machines virtuelles serait alors répartie entre les différents nœuds de stockage. La seconde serait de considérer la taille des machines virtuelles plutôt que leur nombre afin de prendre des décisions de copie.

Mise en œuvre de l'algorithme Nous avons mis en œuvre cet algorithme (sans ses extensions) par un programme écrit en langage C qui s'exécute de manière centralisée depuis le nœud de stockage qui se connecte sur les différents nœuds pour leur donner des instructions. C'est cet algorithme qui a été mis en œuvre pour la récupération des

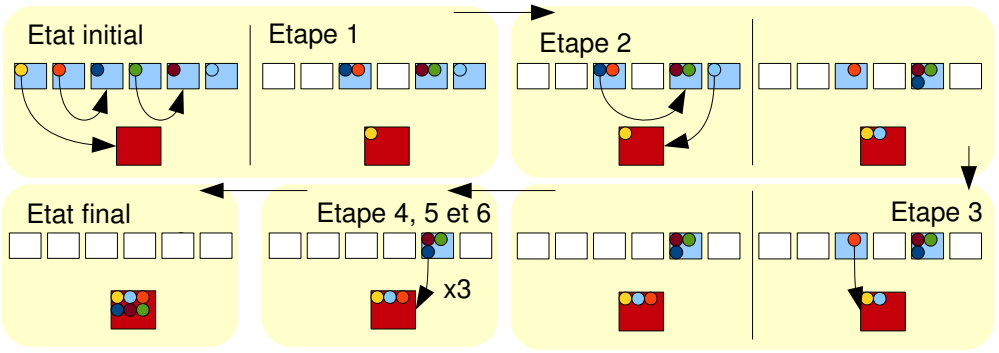


FIGURE 2.7 – Principe de fonctionnement de l'algorithme de récupération efficace des sauvegardes de machines virtuelles des nœuds d'exécution vers le nœud de stockage
Les cercles représentent des machines virtuelles à sauvegarder. Les carrés de couleur bleu représentent des nœuds possédant au moins une machine virtuelle. Les carrés de couleur blanche représentent des nœuds libérés de toute machine virtuelle et pouvant être réattribués pour d'autres tâches. Le carré rouge représente le nœud de stockage.

sauvegardes de machines virtuelles présentée dans la section 2.3.4 traitant de la mise en œuvre de Saline.

2.5 Évaluations

Les évaluations présentées dans cette section sont liées à la mise en œuvre de Saline en utilisant les ressources de stockage locales des nœuds d'exécution (cette série d'expérience n'a pas pour but de tester Saline dans le contexte d'un système de fichiers distribué de type NFS). Les expérimentations que nous avons conduites s'articulent autour de cinq axes principaux :

1. évaluer le coût de configuration des nœuds d'exécution,
2. évaluer le gain obtenu par l'utilisation de la technique de la copie sur écriture,
3. évaluer le temps de déploiement des machines virtuelles,
4. évaluer le temps de sauvegarde et de récupération des images des machines virtuelles ainsi que la capacité de Saline à libérer les nœuds en un minimum de temps,
5. évaluer le temps de restauration des machines virtuelles à partir des sauvegardes.

Toutes les expériences sont réalisées sur la plate-forme Aladdin/G5K. Nous donnons plus de détails sur l'environnement utilisé au fur et à mesure de la présentation des expérimentations. De plus, dans cette section, nous considérons pour plus de clarté qu'une seule machine virtuelle est déployée par nœud d'exécution.

2.5.1 Coût de configuration des nœuds d'exécution

Les nœuds d'exécution de Grid'5000 disposent par défaut d'un environnement d'exécution de production. Cet environnement qui est sous Linux dispose de certains logiciels et bibliothèques. Les utilisateurs peuvent donc utiliser ces nœuds pour exécuter leurs applications directement. Cependant, dans ce cas d'utilisation, il n'est pas possible de devenir administrateur des nœuds, et donc d'installer par exemple des logiciels supplémentaires. Pour remédier à ce problème, il existe un autre mode d'utilisation des

nœuds de Grid'5000 qui consiste à deployer sur ces nœuds un système d'exploitation ainsi qu'une pile logicielle personnalisée. Il devient alors possible de devenir administrateur des nœuds d'exécution.

Dans notre cas, Saline utilise Libvirt pour gérer des machines virtuelles de type KVM. Ces logiciels ne sont pas disponibles dans l'environnement de production des nœuds de Grid'5000. Il nous faut donc déployer notre propre environnement afin de pouvoir les installer. Libvirt et KVM sont disponibles dans la version stable de Debian (Debian Lenny). C'est donc cette version de Linux que nous déployons sur les nœuds. Le déploiement des nœuds se fait grâce à la suite logicielle Kadeploy. Nous utilisons la version de Kadeploy 3.1-3.

La configuration de la pile logicielle de 64 nœuds appartenant à un même site nécessite en moyenne 13m30s (moyenne calculée avec les sites de Sophia, Lyon et Bordeaux). Ce résultat qui est lié à l'infrastructure de Grid'5000 pourrait être réduit à 0 si la librairie Libvirt et le système de virtualisation KVM étaient disponibles sur les environnements de production des nœuds de Grid'5000.

2.5.2 Gain obtenu par l'utilisation de la technique de la copie sur écriture

Nous évaluons l'impact de la technique de la copie sur écriture d'une collection de machines virtuelles en cours d'exécution. Pour ce faire, nous déployons des machines virtuelles qui exécutent une application distribuée MPI. Cette application est MrBayes, une application de calcul phylogénique qui utilise la méthode de Monte-Carlo pour le calcul des résultats (cette méthode nécessitant l'utilisation de nombreuses ressources de calcul et de mémoire pendant de nombreuses heures, l'application MrBayes est tout à fait adaptée aux expérimentations que nous menons). L'application MrBayes est préinstallée dans l'image de référence des machines virtuelles utilisée. La taille de l'image de référence utilisée est de 1 Go.

La figure 2.8 nous montre le bénéfice de l'utilisation de la technique de la copie sur écriture. L'axe des ordonnées présente la taille des collections de machines virtuelles. L'axe des abscisses présente le nombre de machines virtuelles par collection. Cette figure montre que plus le nombre de machines virtuelles augmente, et plus la taille de la collection (la taille de l'ensemble des images des machines virtuelles de la collection) est grande dans les deux cas (avec ou sans utilisation de la technique de la copie sur écriture). Cependant, pour 64 machines virtuelles, la taille de la collection de machines virtuelles sans la technique de la copie sur écriture atteint 64 Go, alors que la taille de la collection de machines virtuelles utilisant la technique de la copie sur écriture atteint 155,2 Mo (1,2 Go en comptant également l'image de référence). Bien entendu, ces résultats peuvent varier selon le type d'application exécutée : une application faisant énormément d'écriture sur le disque n'obtiendra pas les mêmes résultats qu'une application n'écrivant jamais sur le disque, mais cette expérience montre que la technique de la copie sur écriture peut être très avantageuse.

2.5.3 Temps de déploiement des machines virtuelles

Nous mesurons le temps nécessaire au déploiement des images des machines virtuelles. Comme nous l'avons déjà souligné, cela consiste à copier une image de référence d'un nœud maître vers n nœuds d'exécution. Pour ce faire, Saline utilise TakTuk [60], un outil de copie efficace. TakTuk propage les fichiers sous forme d'arbre de manière efficace : la

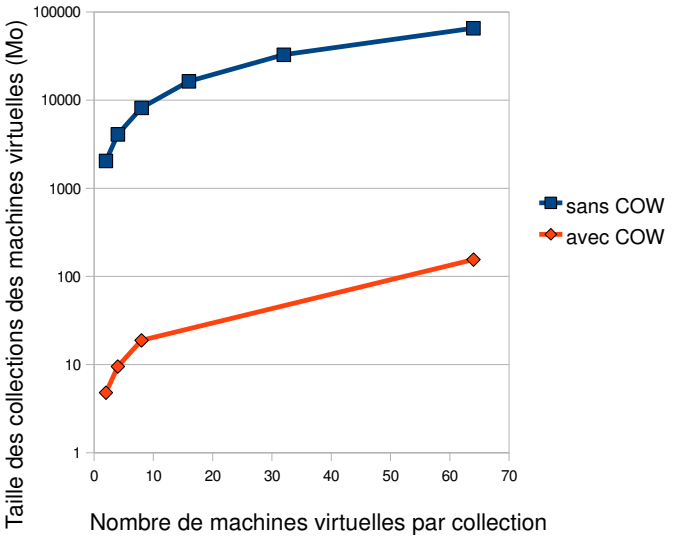


FIGURE 2.8 – Impact de la technique de la copie sur écriture sur les collections de machines virtuelles
Avec copie sur écriture = avec COW. Sans copie sur écriture = sans COW.

copie des fichiers se fait vers un ensemble de nœuds qui à leur tour copient également les fichiers vers d'autres ensembles de nœuds.

La figure 2.9 présente une comparaison du temps (axe des ordonnées) du déploiement de n machines virtuelles (indiqué en axe des abscisses – 1 machine virtuelle par nœud) sur le même nombre de nœuds (une machine virtuelle par nœud) avec le mécanisme de copie élémentaire d'UNIX *scp* et l'outil de copie efficace TakTuk. Les résultats montrent la pertinence d'utiliser un outil de déploiement efficace comme TakTuk. Grâce à ses copies de sous-ensemble en sous-ensemble, parallèlement entre les sous-ensembles, le temps de déploiement dépend peu du nombre de machines virtuelles, alors que pour *scp* le nombre de machines virtuelles fait varier très significativement le temps de déploiement.

2.5.4 Temps de sauvegarde et de récupération des images des machines virtuelles

La sauvegarde des machines virtuelles se fait de manière périodique (sans notification). Dans le cas de notifications, Saline réalise la sauvegarde au moment précis de la réception de la notification. Dans les deux cas (avec ou sans notification) le temps de sauvegarde des machines virtuelles et le temps de récupération des images des machines virtuelles à la même importance (plus ce temps sera court, mieux cela sera). Cependant, dans le cas de la notification, il est important de connaître la vitesse à laquelle les nœuds se libèrent (pour les réattribuer aux gestionnaires de ressources de l'infrastructure le plus rapidement possible).

2.5.4.1 Temps de sauvegarde et de récupération des machines virtuelles

Nous étudions le temps nécessaire à la sauvegarde et à la récupération des images de machines virtuelles sur le nœud de stockage (fichiers de différence *diff* et fichiers d'états des machines virtuelles *état*). Concrètement, cela revient à faire la copie de n images de machines virtuelles qui sont stockées sur n nœuds d'exécution vers 1 nœud de stockage.

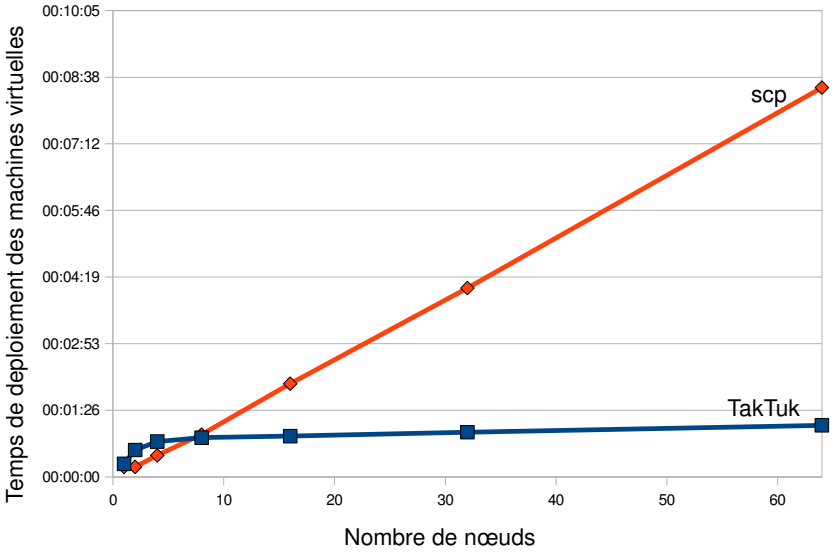


FIGURE 2.9 – Comparaison du temps de déploiement de machines virtuelles avec TakTuk et *scp*

Nous comparons deux méthodes : l'utilisation de `scp` et de Saline (selon l'algorithme présenté dans la figure 2.6).

Les résultats visibles sur la figure 2.10 montrent que Saline a des performances similaires à `scp` pour un nombre de nœuds inférieur à 16, et que ses performances sont meilleures pour un nombre de nœuds supérieur à 16. En effet, Saline est capable de gérer la ressource réseau vers le nœud maître et ainsi évite la création d'un goulot d'étranglement du nœud maître, améliorant ainsi les performances pour un nombre de nœuds élevé.

2.5.4.2 Libération d'un grand nombre de nœuds le plus rapidement possible

Dans le cas de notifications, il est important de libérer le plus grand nombre de nœuds le plus rapidement possible (les nœuds rapidement libérés peuvent être réattribués au gestionnaire de ressources). L'algorithme de Saline présenté dans la figure 2.6 a été conçu en ce sens.

Pour cette expérimentation, nous avons utilisé 64 nœuds et nous avons mesuré pour chacun d'eux le temps nécessaire avant qu'il se libère, à savoir, le temps nécessaire, avant que toutes les sauvegardes de points de reprise aient été copiées sur le nœud maître ou sur un nœud de calcul voisin.

La figure 2.11 présente le pourcentage de nœuds libérés au cours du temps. L'axe des abscisses représente le temps, et celui des ordonnées représente le pourcentage de nœuds libérés. Les résultats montrent que Saline est capable de libérer 80% des nœuds en moins de 2 minutes alors que *scp* arrive au même résultat après 17 minutes.

2.5.5 Redéploiement des machines virtuelles sauvegardées

Nous mesurons le temps nécessaire à la restauration des machines virtuelles. Elle se déroule en deux phases : tout d'abord il convient de déployer l'image de référence avec TakTuk, puis il faut déployer les fichiers de différence et d'état avec *scp*. Concernant le

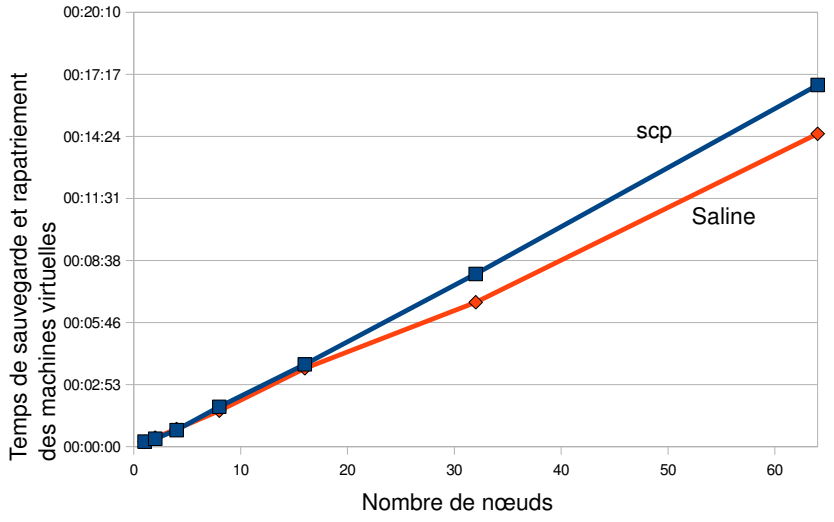


FIGURE 2.10 – Comparaison du temps nécessaire à la copie de machines virtuelles des nœuds d'exécution vers un nœud de stockage entre `scp` et Saline.
L'algorithme utilisé dans Saline est celui présenté dans le tableau 2.6.

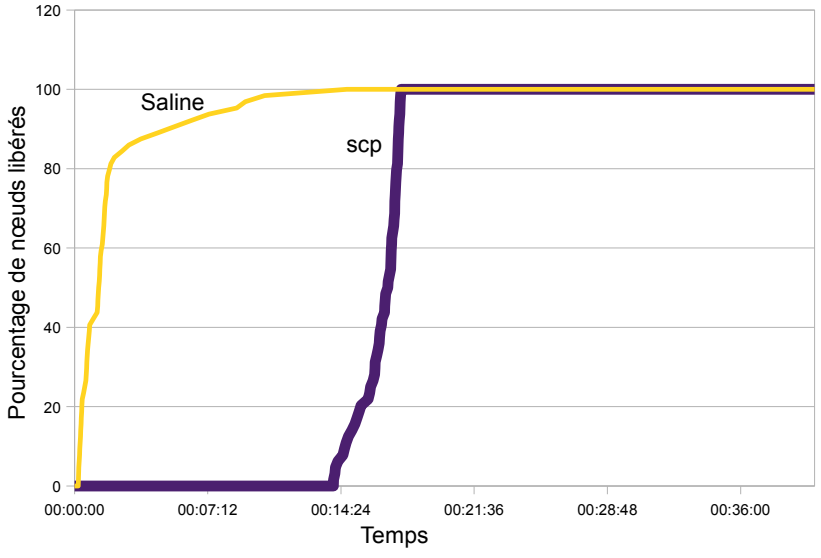


FIGURE 2.11 – Détail du pourcentage de nœuds libérés au cours du temps lors d'une sauvegarde

déploiement de l'image de référence, cela revient à faire un déploiement initial (expérience déjà réalisée dans la section 2.5.3).

Nous exploitons ainsi plusieurs commandes `scp` pour effectuer le transfert des machines virtuelles sauvegardées depuis le nœud de stockage vers les nœuds d'exécution.

La figure 2.12 présente les résultats de l'expérience mesurant le temps nécessaire au déploiement des machines virtuelles sauvegardées. Les résultats montrent que plus le nombre de nœuds d'exécution est important, plus le temps de déploiement est long. Cependant, dans le cas de tâches interruptibles, le temps de déploiement des machines virtuelles (approximativement de l'ordre de la minute) comparé au temps de redémarrage depuis le début de la tâche interruptible est négligeable.

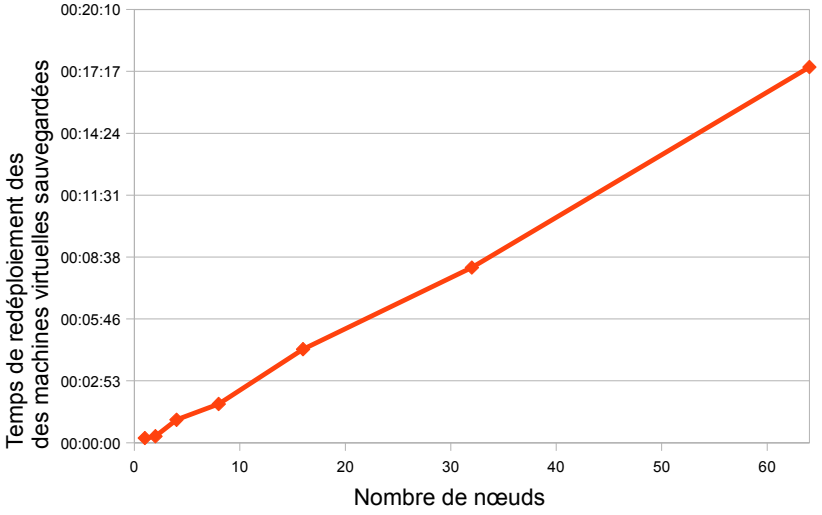


FIGURE 2.12 – Temps de redéploiement des machines virtuelles sauvegardées

2.6 Conclusion et ouverture vers d'autres outils

L'exploitation de la virtualisation a beaucoup été étudiée dans le domaine des infrastructures de type grappe et des centrales numériques. Les machines virtuelles apportent un axe de flexibilité par rapport au degré de personnalisation des environnements d'exécution, mais également dans la gestion de l'infrastructure matérielle (possibilité de faire de la consolidation de serveurs). Ce chapitre étudie les mécanismes de gestion des machines virtuelles sur une infrastructure informatique de type grille. Pour ce faire, nous prenons le cas de la gestion des tâches interruptibles. Les tâches interruptibles sont des tâches exécutées lorsque les ressources sont disponibles et peuvent être détruites lors de l'arrivée de tâches prioritaires. Les tâches interruptibles pouvant être détruites à tout moment, ce type de gestion des tâches est utilisée pour des applications disposant de mécanismes pour redémarrer depuis leur dernière interruption (par exemple des applications de type sac de tâches). En étudiant les axes de flexibilité offerts par les systèmes de virtualisation sur une grille et en les appliquant aux tâches interruptibles, nous avons élargi la gamme de tâches pouvant être gérée de manière interruptible. En effet, l'encapsulation des tâches interruptibles dans des machines virtuelles permet de leur faire bénéficier des fonctionnalités de migration et de sauvegarde et restauration des machines virtuelles.

Nous avons présenté Saline, un système de gestion des collections de machines virtuelles

conçu et mis en œuvre dans le contexte de Grid'5000 pour gérer les tâches interruptibles. Saline s'intègre de manière transparente aux infrastructures existantes. De plus, Saline ne nécessite pas de lourdes manipulations pour son installation et sa configuration : Saline est constitué de scripts écrits en langages *bash* et *C* pour un total de 1 550 lignes de code[1]. La conception modulaire de Saline lui permet d'interagir facilement avec le gestionnaire de ressources de la grille en lui apportant ainsi une extension de ses fonctionnalités. De même, le gestionnaire de placement des machines virtuelles de Saline, qui par défaut met en œuvre une politique simple de *round-robin* peut interagir avec des outils plus évolués de placement des machines virtuelles comme Entropy.

Pour gérer les machines virtuelles, deux points principaux ont été traités : la gestion des images des collections de machines virtuelles et la configuration du réseau de ces machines virtuelles. Concernant la gestion des images, Saline propose une gestion complète du cycle de vie des collections de machines virtuelles. De plus, ces machines virtuelles ont leurs adresses MAC et IP qui sont configurées au moment de leur création afin de ne pas entrer en conflit avec les autres machines virtuelles ou les autres ressources appartenant à l'infrastructure matérielle.

Ouvertures Nous avons validé Saline dans le cadre de tâches interruptibles constituées d'une application MPI (MrBayes). D'autres validations et expérimentations restent à effectuer. En particulier il faudra étudier le surcoût complet engendré par Saline sur l'exécution d'une application (temps total de déploiement et impact des sauvegardes périodiques).

De plus, il faudra étudier de manière plus précise le temps optimum de périodicité des sauvegardes des machines virtuelles.

Les paragraphes suivants présentent des perspectives de travail. La combinaison d'outils avec Saline apporterait plus de flexibilité dans la gestion des infrastructures de calcul en y apportant une plus grande dissociation entre l'infrastructure matérielle et la vue logique des ressources qui est donnée aux applications.

2.6.1 Les outils de déploiement de système d'exploitation : indépendance par rapport à l'infrastructure ciblée

Nous avons utilisé Kadeploy dans le cadre de l'utilisation de Saline sur Grid'5000. Kadeploy déploie une pile logicielle complète sur des nœuds. Cependant, l'installation des paquets et la configuration de l'environnement d'exécution ne sont pas prévues. D'autres systèmes de déploiement et de configuration des infrastructures existent comme Rocks [141] et OSCAR [132] qui permettent l'installation et la configuration de paquets adéquats pour un environnement d'exécution souhaité.

OSCAR est une suite logicielle permettant d'installer et de configurer des grappes de calculateurs pour le calcul à haute performance développé à l'ORNL (Oak Ridge National Laboratory). OSCAR s'utilise sur des grappes ayant un nœud maître et dispose d'une bibliothèque de logiciels empaquetés pour être déployés sur des grappes. Ces logiciels sont empaquetés de telle manière qu'OSCAR peut les installer sur le nœud maître et sur les nœuds d'exécution. OSCAR permet d'installer et configurer pratiquement tout type de distribution Linux. Grâce à une collaboration dans le cadre de l'équipe associée INRIA SER-OS, nous avons pu avec Geoffroy Vallée, chercheur associé à l'ORNL, combiner l'utilisation de Saline et d'OSCAR. La solution proposée, en cours de développement au moment de l'écriture de ce document, permettra à terme d'apporter deux axes de flexibilité

1. Le calcul du nombre de ligne de code a été réalisé à l'aide de l'utilitaire *sloccount*.

dans la gestion des infrastructures de calcul : le premier axe est que OSCAR peut déployer et configurer des infrastructures informatiques composées de grappes hétérogènes avec des environnements d'exécution complets. Le second axe est qu'une fois les machines virtuelles déployées par Saline, OSCAR peut également configurer ces machines virtuelles.

Le but de la configuration des nœuds d'exécution des infrastructures est de pouvoir, quel que soit le type de ressource et le type de système d'exploitation (Fedora, CentOS, Debian, ...), installer (si les paquets adéquats existent) un système de virtualisation (par exemple KVM) ainsi qu'une librairie de manipulation des machines virtuelles (par exemple Libvirt). Une fois l'installation des nœuds terminée, OSCAR installe Saline, si ce n'est pas déjà fait, et ordonne à Saline la création de machines virtuelles. Saline crée les machines virtuelles qui sont configurées grâce à OSCAR (installation de l'OS dans les machines virtuelles, installation des paquets, configuration des applications). Cette double flexibilité (flexibilité par rapport aux nœuds et flexibilité par rapport aux machines virtuelles) permet aux utilisateurs d'avoir une vue de leurs applications et de leurs environnements d'exécution indépendamment des ressources matérielles sous-jacentes. La figure 2.13 représente la combinaison de Saline et OSCAR.

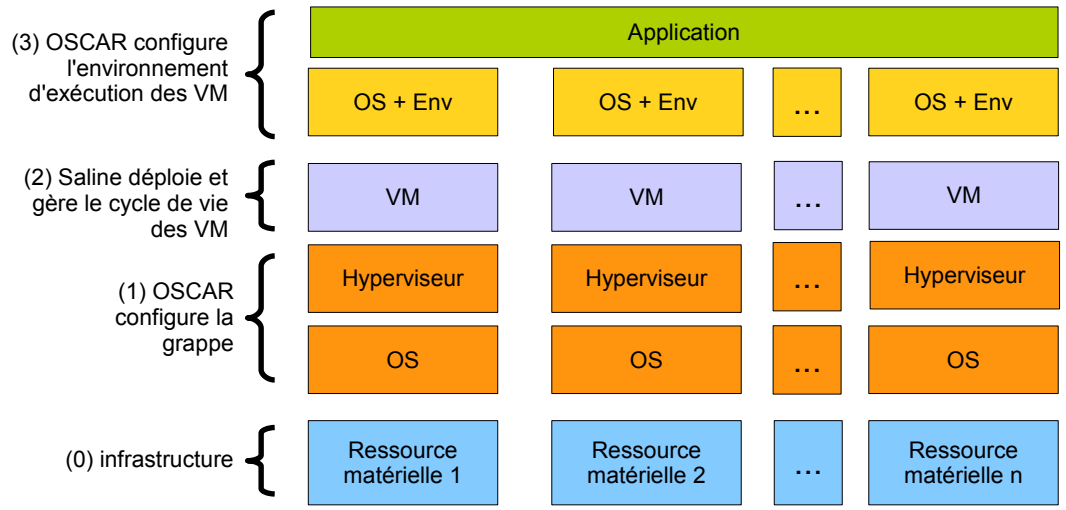


FIGURE 2.13 – Exemple de combinaison de Saline et OSCAR

L'utilisation d'outils capables de configurer les environnements d'exécution combinés à Saline apporte deux points de flexibilité intéressants :

Synthèse 1 : Saline combiné à des outils de déploiement de systèmes d'exploitation peut déployer des collections de machines virtuelles sur des ressources hétérogènes sans faire d'hypothèse sur ces dernières. Cela permet à Saline d'être indépendant de l'architecture matérielle des ressources.

Synthèse 2 : L'utilisation d'un outil de déploiement et de configuration des environnements d'exécution permet à Saline de déployer des environnements spécifiques et configurés dans les machines virtuelles déployées. Cela permet à Saline de pouvoir configurer les environnements d'exécution des applications dans les machines virtuelles.

2.6.2 Les outils de gestion des réseaux virtuels : indépendance par rapport à la localisation géographique des nœuds

L'utilisation d'un outil de virtualisation des réseaux offre une indépendance par rapport à la localisation géographique des nœuds. Deux approches peuvent être considérées : les solutions purement logicielles et la reconfiguration du matériel physique à la demande.

2.6.2.1 Un exemple de solution purement logicielle : les réseaux privés virtuels

Un réseau privé virtuel (VPN, *Virtual Private Network*) est un réseau qui utilise une infrastructure réseau qui peut être publique, comme Internet, pour fournir à l'utilisateur un réseau privé. Son but est de mettre en place des canaux de communication privés en utilisant des canaux de communication publics. Pour ce faire, les données des réseaux privés sont encapsulées dans des paquets sécurisés et sont transférées au destinataire à travers le réseau public.

2.6.2.2 Un exemple de solution fondée sur la reconfiguration du matériel, la configuration à la demande des routeurs, Kavlan

Kavlan est un outil développé dans le contexte de Grid'5000. Sa fonctionnalité est de pouvoir gérer des sous-réseaux (VLAN) dans Grid'5000. L'avantage d'une telle solution est d'avoir une isolation complète au niveau 2 de la couche OSI (voir figure 1.4). En effet, Saline, dans sa mise en œuvre actuelle, attribue des sous-réseaux différents par collection : deux collections ne peuvent pas communiquer l'une avec l'autre. Cependant, cette séparation se fait au niveau IP (niveau 3) de la couche OSI. Les utilisateurs pouvant accéder aux machines virtuelles dans lesquelles ils peuvent être administrateur peuvent alors changer ces adresses et ainsi un utilisateur malicieux pourrait « écouter » et s'introduire dans une collection ne lui appartenant pas.

Une isolation complète au niveau 2 de la couche OSI a l'avantage d'être configurée au niveau de l'infrastructure, c'est-à-dire au niveau du matériel. Ainsi, même avec des droits administrateur dans les machines virtuelles, un changement d'adresse IP ne peut interférer sur les autres collections.

2.6.2.3 Synthèse

L'utilisation d'outils de virtualisation du réseau permet à Saline de déployer des collections de machines virtuelles appartenant à différents sous-réseaux, chaque collection étant isolée des autres (niveau 2 et 3).

2.6.3 Les outils de manipulation des machines virtuelles : flexibilité par rapport aux systèmes de virtualisation utilisés

Il existe des outils permettant de s'abstraire des technologies utilisées par les machines virtuelles. Par exemple, Saline utilise Libvirt [17], une bibliothèque permettant de gérer des machines virtuelles quel que soit le type d'hyperviseur utilisé. Actuellement Libvirt gère une dizaine d'hyperviseurs dont KVM, QEMU, Xen. Cela signifie que Saline, bien que nous n'avons pas encore eu l'occasion de le tester, grâce à Libvirt peut gérer une dizaine d'hyperviseurs différents.

L'utilisation de Libvirt pourrait être complétée par d'autres outils permettant par exemple de convertir des images pour machine virtuelle d'un type d'hyperviseur à un

autre. Par exemple *qemu-img* est un outil capable de convertir des images de machines virtuelles VMware vers des images de machines virtuelles QEMU.

Synthèse Saline combiné à des outils de manipulation de machines virtuelles peut gérer les machines virtuelles sans se soucier du système de virtualisation sous-jacent.

Chapitre 3

Grillade : vers un système de gestion flexible des infrastructures de grilles et centrales numériques

Le chapitre précédent qui combine les techniques de virtualisation sur des infrastructures de type grille, nous amène à réfléchir sur le rapprochement qu'il y a entre les infrastructures de grappe, grille et centrale numérique. En effet, comme décrit dans le tableau 1.1, ces systèmes proposent des axes de flexibilité différents. Dans ce chapitre, nous identifions ces axes de flexibilité et étudions la possibilité et l'intérêt, pour les administrateurs et les utilisateurs, de leur combinaison au sein d'un même système de gestion des ressources de l'infrastructure matérielle.

Nous présentons Grillade, une extension du système de grille XtreemOS qui gère des infrastructures distribuées multi-sites en intégrant les axes de flexibilité identifiés dans les grappes, les grilles et les centrales numériques. Grillade est composé de deux sous-systèmes. Le premier, Grillade-CN (Grillade-Centrale-Numérique) gère les collections de machines virtuelles sur la grille à la manière des centrales numériques. La contribution de Grillade-CN est de permettre la gestion de collections de machines virtuelles sur une infrastructure distribuée constituée de plusieurs domaines d'administration en utilisant et en adaptant les services de grille offert par XtreemOS (par exemple, le système de fichiers de grille XtreemFS et le service de gestion des organisations virtuelles). Le second, Grillade-Ext (Grillade-Extension), gère l'extension de la grille avec des machines virtuelles fournies par une centrale numérique. C'est à partir du constat que les centrales numériques, en créant des ressources à la demande peuvent être vues comme un réservoir de ressources, que nous avons étudié les mécanismes utiles pour l'extension d'une grille. La contribution de Grillade-Ext est de pouvoir étendre une infrastructure informatique avec des ressources fournies par une centrale numérique de manière transparente pour l'exécution des applications.

Ces travaux ont été effectués en collaboration avec Eliana-Dina Tîrşa, en thèse à l'Université Politechnica de Bucarest, et Pierre Riteau, doctorant dans l'équipe Myriads. Ils ont donné lieu au stage d'été 2010 de Sajith Kalathingal (3 mois), étudiant en master à l'Université d'Amsterdam Vrije (VUA). Ces travaux ont fait l'objet de plusieurs publications [63, 83, 87, 91, 128, 130, 164].

3.1 Grilles et centrales numériques : un rapprochement à étudier

Les grappes, les grilles et les centrales numériques sont des systèmes informatiques distribués qui diffèrent par leur système de gestion des ressources [171].

Les grilles sont des infrastructures réparties sur plusieurs sites. Ces différents sites appartiennent généralement à différentes institutions (différents domaines d'administration). Les systèmes de gestion des grilles apportent des mécanismes de partage des ressources entre les différents domaines d'administration. Ces ressources peuvent être utilisées par tous les participants de la grille. Il n'y a pas de facturation explicite par rapport à l'utilisation des ressources. Les utilisateurs peuvent déployer leurs applications dans des environnements d'exécution généralement déjà pré-configurés.

Les centrales numériques sont des infrastructures mono-site appartenant à un seul domaine d'administration [109]. Elles permettent aux utilisateurs de déployer leurs applications dans des environnements d'exécution personnalisés. Ces environnements d'exécution sont créés dans des machines virtuelles à la demande. Les clients d'une centrale numérique payent pour les ressources qu'ils consomment (modèle économique du *pay-as-you-go*). Une centrale numérique peut être vue comme une grappe gérée par un système de virtualisation distribué sur l'ensemble des nœuds. Les centrales numériques peuvent exécuter des tâches interactives ou non-interactives.

De cette première analyse, voici les critères de flexibilité que nous retenons :

Multi-domaines d'administration et partage des ressources : Capacité du système à proposer un service permettant de partager des nœuds répartis sur différents sites appartenant à des domaines d'administration différents.

Flexibilité de l'environnement d'exécution : Capacité du système à créer, configurer et reconfigurer à la demande les environnements d'exécution pour l'exécution des applications des utilisateurs.

Flexibilité de l'utilisation des ressources : Capacité du système à configurer et reconfigurer à la demande les ressources matérielles (par exemple, reconfigurer un ensemble de ressources pour les mettre dans un sous-réseau spécifique).

Granularité : Capacité du système à gérer des tâches et/ou des machines virtuelles.

Capacité d'extension : Capacité du système à s'étendre sur des ressources fournies par une ou plusieurs centrales numériques externes.

Le tableau 3.1 présente les principales caractéristiques des grilles et des centrales numériques en mettant en valeur leurs différences sur les critères que nous venons d'énumérer.

Dans les paragraphes suivants nous présentons et étudions les différents points d'intérêts à combiner les services de grilles avec ceux des centrales numériques. Pour ce faire, nous décrivons d'abord les cas d'utilisation élémentaires des centrales de ressources ainsi que des systèmes de grille, puis nous les combinons dans un système nommé Grillade et présentons l'intérêt de notre démarche [128, 130].

3.1.1 Quand les grilles et les centrales numériques se rencontrent

Nous avons mené une réflexion sur l'intérêt de combiner des grilles et des centrales numériques.

Trois points de vue doivent être étudiés : celui de l'administrateur des centrales de ressources (mono-domaine d'administration – un administrateur par centrale), celui des

	Domaines d'administration, partage des ressources	Flexibilité des environnements d'exécution	Flexibilité de gestion de l'infrastructure	Granularité	Extension de l'infrastructure
Grilles	multi-domaines d'administration, organisation virtuelle	environnements préconfigurés	infrastructure préconfigurée	tâches (groupe de processus)	non
Centrales numériques (IaaS)	mono-domaine d'administration, non	oui	partiel : catalogue de choix possible (possibilité de choisir le nombre de CPU par exemple)	machines virtuelles (système d'exploitation)	oui

TABLE 3.1 – Comparaison de fonctionnalités élémentaires entre grilles et centrales numériques

administrateurs de la grille (multi-domaines d'administration – multi-administrateurs) et celui des utilisateurs.

Les cas d'utilisation élémentaires Les cas *1.* et *2.* sont les cas de base d'utilisation des centrales numériques et d'un système d'exploitation pour grille (voir figure 3.1).

- Le cas *1.* présente une centrale numérique qui met à disposition de ses utilisateurs, généralement sous forme contractuelle, des machines virtuelles. Du point de vue de l'utilisateur, les ressources dont il dispose sont des machines virtuelles. Il bénéficie du choix de l'environnement d'exécution de son application (système d'exploitation, bibliothèques) dans les machines virtuelles : l'utilisateur dispose généralement d'un catalogue de systèmes d'exploitation disponibles et de droits privilégiés pour configurer complètement l'environnement d'exécution.
- Le cas *2.* présente trois fournisseurs de ressources qui assurent l'administration de leur domaine. Ces fournisseurs décident de fédérer leurs ressources en utilisant un système de grille. Du point de vue des administrateurs de la grille, les organisations virtuelles (VO) permettent de gérer le partage des ressources des différents domaines d'administration entre les différents utilisateurs. L'utilisateur dispose d'environnement d'exécution préconfiguré et peut exécuter une application distribuée sur les ressources de la grille indépendamment de leur localisation et des domaines d'administration en bénéficiant de services dédiés à l'exécution d'application distribuée sur la grille.

Un système flexible de grille pour centrales numériques La figure 3.2 présente une centrale numérique créée et gérée par un service flexible de grille. Cette centrale numérique tire profit des ressources distribuées, réparties sur trois domaines d'administration différents. Du point de vue des administrateurs de Grillade, cela permet de fédérer et gérer de multiples centrales de ressources, à la manière des multi-centrales

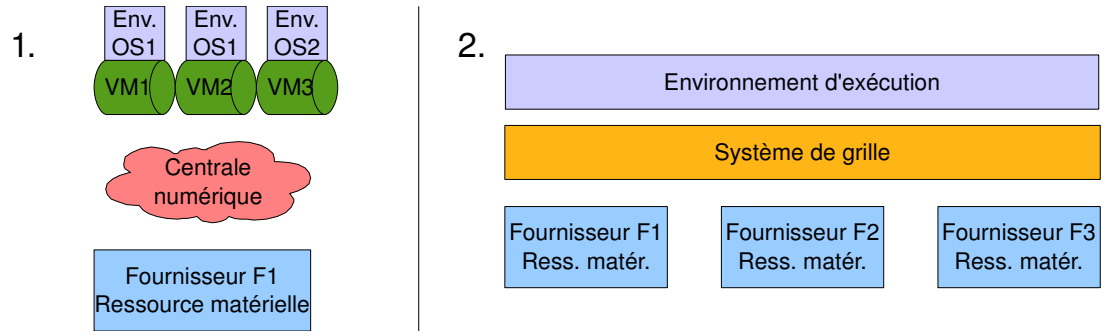


FIGURE 3.1 – Représentation des cas élémentaires d'utilisation des centrales numériques (cas 1) et des systèmes de grille (cas 2)

de ressources (*Sky Computing* – voir 1.2.3.4). Du point de vue de l'utilisateur, il peut bénéficier à la fois des avantages des systèmes de grilles et des centrales : l'utilisateur peut utiliser les ressources matérielles avec leurs environnements d'exécution préconfigurés, comme cela est proposé nativement par des mécanismes de type grille (voir le cas *2.*), mais il peut également utiliser des machines virtuelles proposées par des mécanismes de type centrale numérique afin de pouvoir par exemple, déployer et utiliser son propre système d'exploitation (voir le cas *1.*).

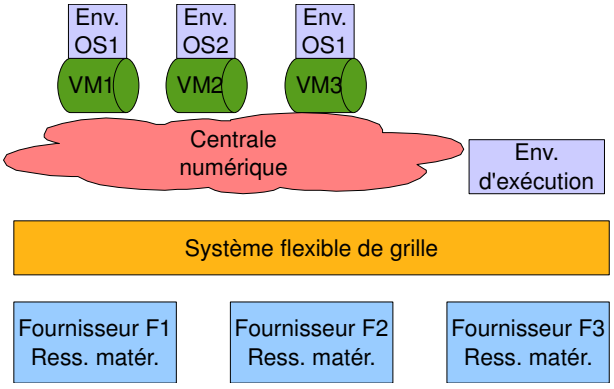


FIGURE 3.2 – Représentation d'un système flexible de grille gérant à la fois des machines virtuelles (avec leurs environnements personnalisés) à la manière des centrales numériques et des environnements d'exécution préconfiguré sur les ressources, à la manière des systèmes de grille standard

Extension d'une grille avec des ressources de centrales numériques La figure 3.3 présente un système flexible de grille installé sur des nœuds répartis sur deux sites mais qui selon les besoins peut s'étendre sur des ressources fournies par deux centrales numériques (le système flexible de grille est exécuté dans les machines virtuelles fournies par les centrales numériques). Du point de vue des administrateurs du système flexible de grille, ils peuvent agrandir leur grille temporairement grâce à des ressources d'une ou plusieurs centrales numériques. Du point de vue de l'utilisateur, il peut tirer avantage de la flexibilité offerte par le système flexible de grille en bénéficiant des outils et services proposés par ce dernier, de manière transparente.

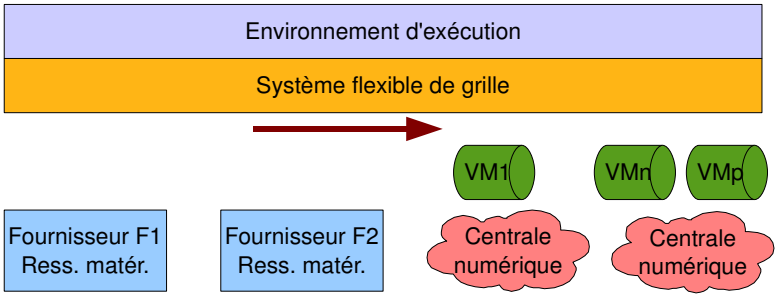


FIGURE 3.3 – Représentation de l'extension d'une infrastructure de grille avec des ressources fournies par des centrales numériques (le système flexible de grille est exécuté dans les machines virtuelles fournies par les centrales numériques)

3.1.2 Autres variantes

Différents autres cas peuvent être envisagés. Cependant, ces cas s'assimilent aux cas ou à une combinaison des cas présentés précédemment.

Par exemple, on peut imaginer un cas particulier de celui de l'extension (voir figure 3.3) dans lequel le système flexible de grille est complètement déployé sur des machines virtuelles de plusieurs centrales numériques. L'utilisateur peut dans ce cas bénéficier d'un environnement d'exécution qui est en fait exécuté sur des ressources fournies par des centrales numériques, alors que les administrateurs des centrales n'ont pas eux-mêmes établis de partage de ressources entre leurs centrales. Cependant, mis à part des problèmes de configuration du réseau à régler entre les ressources obtenues des différents fournisseurs, ce cas d'utilisation est très similaire à l'utilisation d'un système de grille directement sur des ressources matérielles (cas élémentaire numéro *2.*). De plus, dans le cas d'XtreemOS qui est le système de grille que nous utilisons, son utilisation sur un ensemble de machines virtuelles a déjà été démontrée [67], c'est pourquoi nous faisons le choix de ne pas étudier ce cas plus en détail dans ce chapitre.

Un autre exemple pourrait être celui où, les ressources fournies par les centrales numériques ne seraient pas des machines virtuelles, mais des ressources matérielles. On pense ainsi à la gestion des nœuds à la manière de Grid'5000, dans laquelle, les nœuds disposent de console de service leur permettant d'être configurés et reconfigurés pour l'installation du système flexible de grille (cas d'une extension) ou d'un autre système d'exploitation (cas de l'environnement d'exécution personnalisé).

Ces cas, qui dépassent le cadre de ce document, feront l'objet d'une étude approfondie ultérieurement.

3.1.3 Cas d'étude retenus

Les axes de flexibilité retenus de l'analyse précédente sont les suivants : gestion des collections de machines virtuelles sur une infrastructure matérielle de type grille (voir figure 3.2) permettant de personnaliser les environnements d'exécution et extension d'une infrastructure matérielle avec des ressources fournies par une centrale numérique (cas *5*). Grillade est le système de gestion flexible des infrastructures réparties sur plusieurs sites, issu de la réflexion sur les différentes interactions possibles entre les systèmes de grille et les centrales numériques. Le tableau 3.2 présente les objectifs fixés pour Grillade.

Grillade est fondé sur le système d'exploitation pour grille XtreemOS. Nous présentons dans la section 3.2 les caractéristiques d'XtreemOS pertinentes pour la gestion des ma-

	Domaines d'administration, partage des ressources	Flexibilité des environnements d'exécution	Flexibilité de gestion de l'infrastructure	Granularité	Extension de l'infrastructure
Grilles	multi-domaines d'administration, organisation virtuelle	environnements préconfigurés	infrastructure préconfigurée	tâches (groupe de processus)	non
Centrales numériques (IaaS)	mono-domaine d'administration, non	oui	partiel : catalogue de choix possible (possibilité de choisir le nombre de CPU par exemple)	machines virtuelles (système d'exploitation)	oui
Grillade	**multi-domaines d'administration, organisation virtuelle**	**oui**	**oui**	**tâches + machines virtuelles**	**oui**

TABLE 3.2 – Présentation des objectifs fixés pour Grillade
Ce tableau complète celui 3.1.

chines virtuelles. Le cas *4.*, étudié dans la section 3.3, présente la conception et la mise en œuvre des mécanismes nécessaires pour gérer les collections de machines virtuelles dans le système d'exploitation pour grille XtreemOS. Le cas *5.*, étudié dans la section 3.4, présente la conception et la mise en œuvre des mécanismes permettant d'étendre une grille XtreemOS avec des ressources de centrales numériques.

3.2 XtreemOS, un système d'exploitation pour grille

XtreemOS [38, 65, 68, 127] est un projet, financé par la commission européenne, qui a débuté en juin 2006. Son but est de concevoir, mettre en œuvre, évaluer et promouvoir un système d'exploitation pour grille, du même nom XtreemOS, disposant nativement de mécanismes de gestion des organisations virtuelles et pouvant s'exécuter sur des ordinateurs individuels (PC), des grappes, et des périphériques mobiles (PDA, téléphone mobile). Ce système mis en œuvre au-dessus du système d'exploitation Linux est générique, évolutif, et s'appuie sur des mécanismes robustes et sûrs du système Linux qui de plus, sont connus des administrateurs système et leur permettent un minimum de temps d'adaptation dans la prise en main d'XtreemOS.

Architecture générale d'XtreemOS L'architecture générale d'XtreemOS est constitué de deux abstractions : XtreemOS-F et XtreemOS-G. XtreemOS-F est l'ensemble des systèmes d'exploitation Linux configuré pour XtreemOS et adapté à l'infrastructure matérielle visée (PC et grappes – dans ce document nous ne considérons pas les périphériques mobiles). XtreemOS-G s'appuie sur XtreemOS-F pour fournir les services de grille indépendamment du type de ressources considérées. La figure 3.4 présente l'architecture générale simplifiée d'XtreemOS.

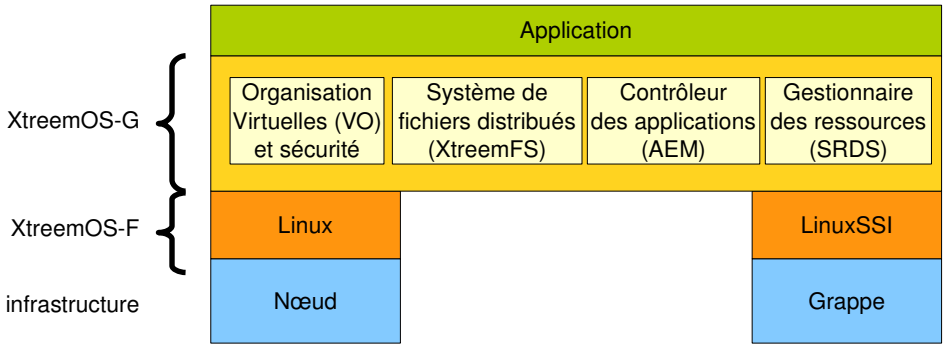


FIGURE 3.4 – Architecture simplifiée d'XtreemOS

Installé sur chaque ressource participant à la grille, XtreemOS fournit à l'échelle de la grille des services similaires à ceux qu'offre un système d'exploitation pour un ordinateur : abstraction du matériel, partage des ressources entre différents utilisateurs. Une manière de présenter XtreemOS est de le décomposer selon quatre axes : la gestion des ressources, la gestion des utilisateurs, la gestion des applications et la gestion des données.

Concernant la gestion des ressources, XtreemOS dispose de mécanismes de découverte et de réservation des ressources selon les besoins définis par les utilisateurs et les politiques d'accès aux ressources. Les ressources de la grille peuvent joindre ou quitter la grille à tout moment et cela, que ce soit à cause d'une défaillance, ou lorsqu'un arrêt/ajout planifié est réalisé.

La gestion des utilisateurs est faite par le service de gestion des VO, un utilisateur ne pouvant utiliser que les ressources d'une VO dans laquelle il est lui même enregistré.

La gestion des applications est faite par le service d'exécution des applications (AEM, *Application Execution Management*) qui gère le déploiement et l'exécution des applications sur les ressources.

Enfin, la gestion des données est faite par XtreemFS, le système de fichiers pour grille d'XtreemOS qui permet de fédérer des espaces de stockage situés dans différents domaines d'administration.

Composition d'une grille XtreemOS Une grille XtreemOS est composée de trois types de ressources : les nœuds de service, les nœuds d'exécution et les nœuds client. Les nœuds de service exécutent les services utiles au fonctionnement d'une grille XtreemOS (par exemple, le service de distribution des certificats). Les nœuds d'exécution exécutent les applications. Les nœuds client sont les points d'accès depuis lesquels un utilisateur peut accéder à la grille.

Pour des raisons de clarté, dans ce document, nous focalisons notre étude uniquement sur les nœuds de service et les nœuds d'exécution en considérant que les actions à exécuter depuis les nœuds client peuvent être faites depuis un nœud de l'un des deux autres types. La figure 3.5 présente un exemple de grille XtreemOS constituée de trois sites.

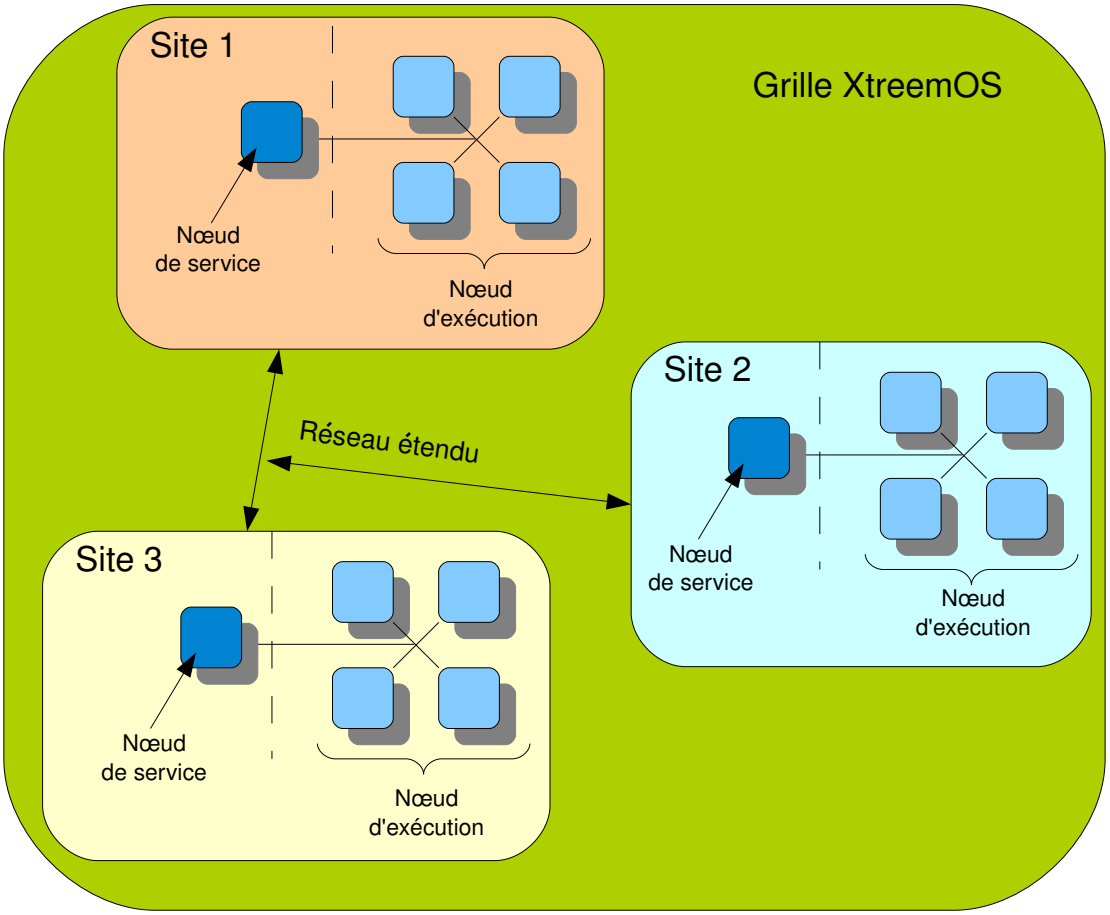


FIGURE 3.5 – Une grille XtreemOS composée de trois sites

Présentation détaillée des services offerts par XtreemOS Les sections suivantes présentent les services offert par XtreemOS qui sont pertinents pour nos travaux. Tout d'abord nous présentons le gestionnaire de grappe LinuxSSI (XtreemOS-F). Puis nous présentons les fonctionnalités de haut niveau (XtreemOS-G) que sont les organisations virtuelles (VO), le système de fichier distribué (XtreemFS), le gestionnaire de découverte des ressources (SRDS) et le gestionnaire d'exécution des applications (AEM).

3.2.1 LinuxSSI

XtreemOS gère les grappes de calculateurs avec un système à image unique, LinuxSSI. Un système à image unique donne l'illusion à l'utilisateur d'utiliser une grappe comme un nœud multi-processeur (SMP). Dans XtreemOS, la mise en œuvre de LinuxSSI est faite par l'utilisation du système à image unique Kerrighed [129, 131].

Kerrighed est une extension du noyau Linux et offre des fonctionnalités comme la gestion globale des processus, le partage et l'agrégation de la mémoire, un seul espace de communication (IPC, *Inter-Process Communication*) pour tous les processus, la tolérance aux fautes avec la possibilité de sauvegarder et restaurer des points de reprise d'applications parallèles.

Par exemple, considérons une grappe constituée de quatre nœuds, chacun disposant d'un processeur et de 1 Go de mémoire. L'agrégation de ces nœuds par Kerrighed donnera à l'utilisateur l'illusion d'utiliser un ordinateur SMP composé de 4 processeurs et 4 Go de mémoire.

Comme nous l'avons décrit dans la figure 3.4, XtreemOS-F est capable de gérer des PC individuels, mais également des grappes avec LinuxSSI. Du point de vue de l'infrastructure de services, une grappe gérée par LinuxSSI est vue comme s'il s'agissait d'un unique nœud.

3.2.2 Les organisations virtuelles

XtreemOS propose nativement un service de création et de gestion des VO dans une grille [127]. Une VO permet à des institutions ou des organisations de gérer le partage des ressources dont elles disposent entre les utilisateurs appartenant aux différents domaines d'administration qui travaillent conjointement vers un même objectif. Les administrateurs des différentes institutions ou organisations disposent de mécanismes assurant la sécurité des ressources qu'ils partagent ainsi que d'autres leur permettant de décrire des politiques de gestion des ressources et des utilisateurs.

3.2.2.1 La gestion des politiques

Deux niveaux de politiques peuvent être définies pour le partage des ressources : les politiques locales à la ressource, et les politiques globales à la VO. Les politiques locales sont définies par les administrateurs des ressources (il peut ne pas y avoir de politiques locales). Les politiques globales sont définies par les administrateurs de la VO.

Le service VOPS (*VO Policy Service*) offre le support nécessaire à l'administration des ressources. Selon les politiques globales définies dans la VO, VOPS vérifie et valide l'allocation des ressources. L'≪ utilisation des ressources uniquement la nuit ≫ est un exemple de politique de VO.

De plus, XtreemOS étant un système d'exploitation pour grille, la gestion des comptes utilisateurs doit se faire à l'échelle de la grille. Un utilisateur de la grille appartient à un domaine d'administration participant à la grille. Cet utilisateur possède probablement un compte utilisateur sur les ressources de son domaine d'administration, mais n'en possède

vraisemblablement pas sur les ressources des autres domaines, ce qui pose le problème d'exécution des applications sur des ressources partagées par d'autres domaines d'administration. Pour y remédier, XtreemOS propose la notion de compte utilisateur de la grille. Ainsi, XtreemOS dispose de mécanismes permettant de faire la correspondance entre les comptes des utilisateurs de la grille avec les comptes locaux des nœuds (projection de l'identifiant d'utilisateur de la grille sur un compte local à un nœud). Cela se fait grâce au service *XOS Node* qui est exécuté sur chaque ressource de la grille. Ce service est chargé de créer des comptes temporaires pour l'exécution d'application d'utilisateurs n'ayant pas de comptes locaux sur les ressources. Une fois l'exécution terminée, ces comptes sont supprimés.

3.2.2.2 Le contrôle du taux d'utilisation des ressources

Les VO disposent de mécanismes de comptabilisation du taux d'utilisation des ressources (*accounting*). Ces informations sont disponibles aux administrateurs des VO.

3.2.2.3 La distribution des certificats aux utilisateurs

Le service CDA (*Certificate Distribution Authority*) est en charge, sur requête des utilisateurs, de leur fournir leur certificat. Ce certificat est ensuite utilisé pour authentifier et vérifier les droits d'un utilisateur dans la grille. Le certificat de l'utilisateur est sa clef d'accès à la grille, lui permettant d'être identifié dans une VO afin de pouvoir utiliser des ressources.

3.2.2.4 La distribution des certificats aux ressources

XtreemOS dispose d'un service d'enregistrement des ressources appelé RCA (*Resource Certification Authority*) [64]. Un serveur RCA s'exécute sur un nœud de service. Chaque site dispose d'un serveur RCA en charge de distribuer des certificats aux ressources du site.

L'enregistrement d'une ressource dans une VO de la grille se fait en deux étapes. Tout d'abord la ressource doit s'identifier auprès du serveur RCA pour la VO considérée. Celui-ci confirme l'identification. À partir de ce moment, la ressource est dans l'état « identification confirmée », elle peut faire une requête d'obtention de certificat. Cette seconde étape consiste à faire une requête de certificat au RCA. Ce dernier signe le certificat correspondant à la ressource et les certificats sont installés. À partir de ce moment, la ressource fait partie de la grille.

3.2.3 XtreemFS

XtreemFS est le système de fichiers pour grille d'XtreemOS [107]. XtreemFS permet de fédérer des espaces de stockage se situant sur différents domaines d'administration. Ce système de fichiers distribué gère des répliques de données afin de fournir aux utilisateurs un service sûr et efficace. Les membres d'une VO et les applications exécutées pour leur compte peuvent accéder aux données depuis n'importe quel nœud de la grille quel que soit le site où elles se trouvent. Enfin XtreemFS répond à la norme POSIX.

3.2.4 La découverte de ressources

Les ressources peuvent être recherchées de deux manières. Les utilisateurs peuvent formuler une requête précise concernant les ressources qu'ils veulent, par exemple, « je

veux toutes les ressources de la grappe de calcul nommée *grappe1* », ou bien, ils peuvent spécifier des requêtes beaucoup plus imprécises, par exemple, « je souhaite 3 ressources ayant chacune 2 Go de mémoire vive». Ces requêtes sont formulées dans un fichier de type JSDL (*Job Submission Description Language*) ce qui permet de décrire la tâche à exécuter ainsi que les ressources nécessaires à son exécution. Pour procéder à la découverte des ressources XtreemOS dispose d'un service de découverte des ressources (SRDS, *Service Resource Discovery System*) qui est composé de deux services [61, 66] : l'ADS (*Application Directory Service*) et le service RSS (*Resource Selection Service*).

Le service RSS fait une présélection des ressources à allouer à une application en se focalisant sur des attributs statiques (par exemple, 2 Go de mémoire, 1 processeur de 2 GHz). Le service RSS est mis en œuvre de manière complètement distribuée en utilisant des protocoles de type pair-à-pair, ce qui lui permet de pouvoir découvrir les ressources recherchées parmi des centaines, des milliers d'autres. Le service RSS produit une liste de ressources correspondant aux critères de recherche, cette liste contient un nombre plus important de ressources que celui demandé par l'utilisateur.

Le service ADS, mis en œuvre également de manière distribuée, reçoit la présélection faite par le service RSS et affine la sélection en se focalisant sur des attributs dynamiques (par exemple, l'espace mémoire disponible sur une ressource, le pourcentage d'utilisation des CPU). Finalement, une liste de ressources correspondant aux critères de recherche est retournée.

3.2.5 Le contrôleur d'exécution d'applications

XtreemOS dispose d'un contrôleur d'exécution des applications (AEM, *Application Execution Manager*) [65]. Le contrôleur d'exécution d'applications est en charge de la gestion des applications sur la grille. Par exemple, c'est au contrôleur d'exécution d'applications de sélectionner (en fonction de la liste de ressources fournie par le service de découverte des ressources) et d'allouer les ressources pour une application spécifique, puis de démarrer et surveiller l'application considérée.

Le contrôleur d'exécution d'applications est composé de différents services dont les principaux, liés à nos travaux, sont présentés dans les paragraphes suivants.

3.2.5.1 Le gestionnaire de réservation des ressources

XtreemOS propose des mécanismes de réservation des ressources (*Reservation Manager*). Les ressources peuvent être spécifiées par l'utilisateur ou trouvées par le service de découverte des ressources d'XtreemOS. La réservation peut être faite de manière explicite (c'est l'utilisateur qui spécifie les ressources qu'il veut) ou être implicite (l'utilisateur soumet une application et c'est XtreemOS qui se charge de réserver les ressources appropriées).

3.2.5.2 Le gestionnaire des tâches

Le gestionnaire des tâches (*Job Manager*) est le point de contact pour interagir avec une tâche, répondre à une requête concernant l'état d'une tâche, exécuter les tâches, coordonner les sauvegardes de points de reprise, décider de la migration d'une tâche, etc.

Le gestionnaire des tâches est mis en œuvre de manière distribuée. Chaque instance du gestionnaire des tâches contient les informations relatives à une partie des tâches en cours d'exécution sur la grille.

3.2.5.3 Le répertoire d'accès au gestionnaire des tâches

Le répertoire d'accès au gestionnaire des tâches (*Job Directory*) permet de déterminer et localiser le gestionnaire des tâches en charge d'une tâche donnée. Ce répertoire est mis à jour au fur et à mesure de l'exécution des tâches (création, terminaison).

3.2.5.4 Le contrôleur de ressources

Le contrôleur de ressources (*Resource Manager*) est exécuté sur chaque ressource faisant partie de la grille. Ce contrôleur est l'interface entre la ressource et les services exécutés sur la grille. Par exemple, c'est ce module qui est en charge d'envoyer les informations de contrôle de la ressource (charge de la mémoire, du processeur...), mais c'est également lui qui reçoit et négocie les requêtes d'exécution de tâches sur la ressource.

3.2.5.5 Le gestionnaire d'exécution

Le gestionnaire d'exécution (*Execution Manager*) est exécuté sur chaque ressource. Ce gestionnaire assure l'exécution des tâches envoyées par le gestionnaire des tâches et lui renvoie des informations concernant l'état d'exécution d'une tâche lorsqu'il est sollicité.

3.2.5.6 L'interface de communication XATI

XATI est l'interface de communication utilisée par XtreemOS qui permet les interactions entre les applications et les services du contrôleur d'exécution des applications.

3.2.6 Les fonctionnalités d'XtreemOS

Grâce aux services présentés dans les paragraphes précédents le système XtreemOS offre des fonctionnalités de haut niveau que nous présentons dans les paragraphes suivants.

3.2.6.1 La découverte des ressources

La fonctionnalité de découverte des ressources utilise le service de découverte des ressources présenté dans la section 3.2.4. Les utilisateurs d'XtreemOS peuvent chercher des ressources XtreemOS correspondant à certaines caractéristiques comme le décrit la figure 3.6. Cela se fait via l'interface XOSAGA par laquelle l'utilisateur fournit une description des ressources dont il a besoin. Cette description est transmise à l'ADS qui retransmet la requête au RSS. Après recherche, le service RSS retourne une liste préliminaire de ressources pouvant convenir à la description. Cette liste est raffinée par l'ADS qui retourne à l'utilisateur une liste de ressources correspondant aux critères de recherche.

3.2.6.2 La réservation des ressources

Un utilisateur d'XtreemOS peut réserver un ensemble de ressources. La figure 3.7 présente le procédé de réservation des ressources. Initialement, l'utilisateur peut connaître précisément les ressources dont il a besoin, par exemple, la grappe de calculateurs nommée « grappe1 », ou, il peut s'appuyer sur le service de découverte des ressources d'XtreemOS précédemment décrit. Avec la description des ressources dont il a besoin, l'utilisateur contacte le gestionnaire de réservation par l'intermédiaire de l'interface XOSAGA. Le gestionnaire de réservation négocie avec VOPS. Cette négociation consiste principalement à vérifier que la réservation n'enfreint pas les politiques globales de la VO et locales des ressources concernées. Puis, la réservation est faite en contactant le gestionnaire

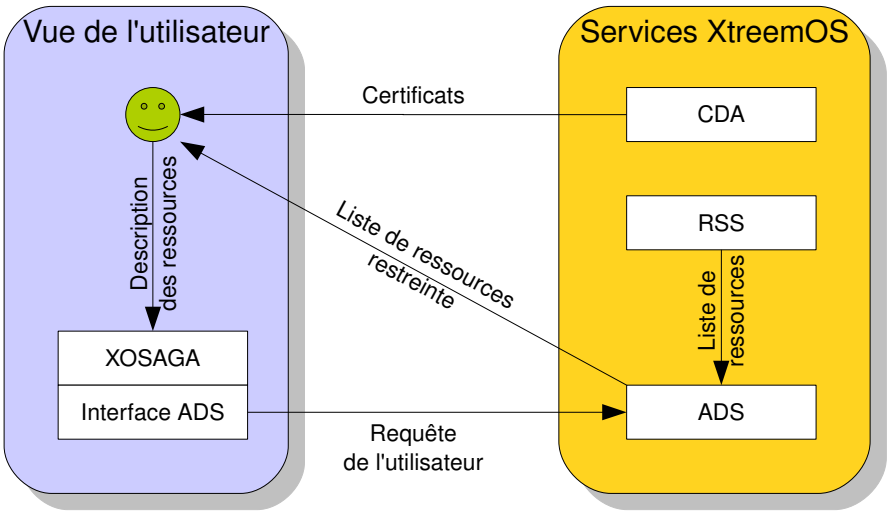


FIGURE 3.6 – Mécanisme de découverte des ressources.

de ressources des nœuds d'exécution concernés. À la fin de l'opération, l'identifiant de réservation (IDréservation) est retourné à l'utilisateur.

3.2.6.3 La soumission des tâches

Un utilisateur d'XtreemOS peut soumettre des tâches de deux manières. La première méthode consiste à réserver des ressources et ensuite à soumettre une tâche sur les ressources réservées. La seconde méthode incorpore la description des ressources requises dans la description de la tâche et laisse XtreemOS s'occuper de la réservation et de l'ordonnancement de la tâche.

La figure 3.8 présente le procédé de soumission d'une tâche sur un ensemble de ressources déjà réservé. L'utilisateur soumet une tâche via l'interface XOSAGA. Cette soumission est transmise au gestionnaire des tâches par le bus de communication XATI. Le gestionnaire des tâches obtient la liste des ressources réservées grâce au gestionnaire de réservation en utilisant le numéro d'identifiant de la réservation. Les tâches sont ensuite envoyées par le gestionnaire des tâches aux différents gestionnaires d'exécution de chaque ressource impliquée dans la réservation. La tâche peut alors démarrer. Les fichiers exécutables, les librairies ainsi que toutes les autres données nécessaires à l'exécution des tâches sont rendus disponibles par le système de fichiers XtreemFS.

La figure 3.9 présente la soumission d'une tâche sans réservation préalable. Dans ce cas, XtreemOS sélectionne et réserve les ressources requises puis, ordonnance l'exécution de la tâche sur les ressources.

3.3 Grillade-CN : gestion des centrales numériques sur une grille XtreemOS

Les systèmes de grille offrent nativement des mécanismes permettant de fédérer et manipuler des ressources réparties sur différents domaines d'administration. Ces mécanismes, tout d'abord conçus pour gérer des tâches, peuvent être étendus afin de pouvoir gérer et manipuler des collections de machines virtuelles, à la manière des centrales numériques.

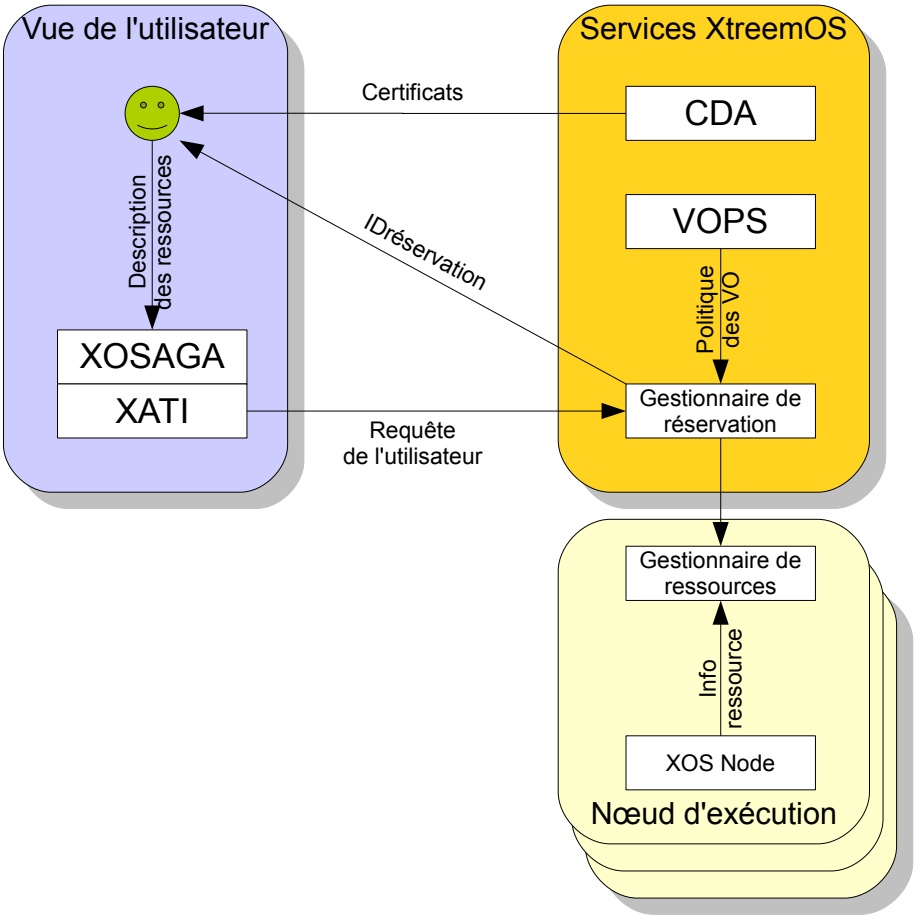


FIGURE 3.7 – Mécanisme de réservation

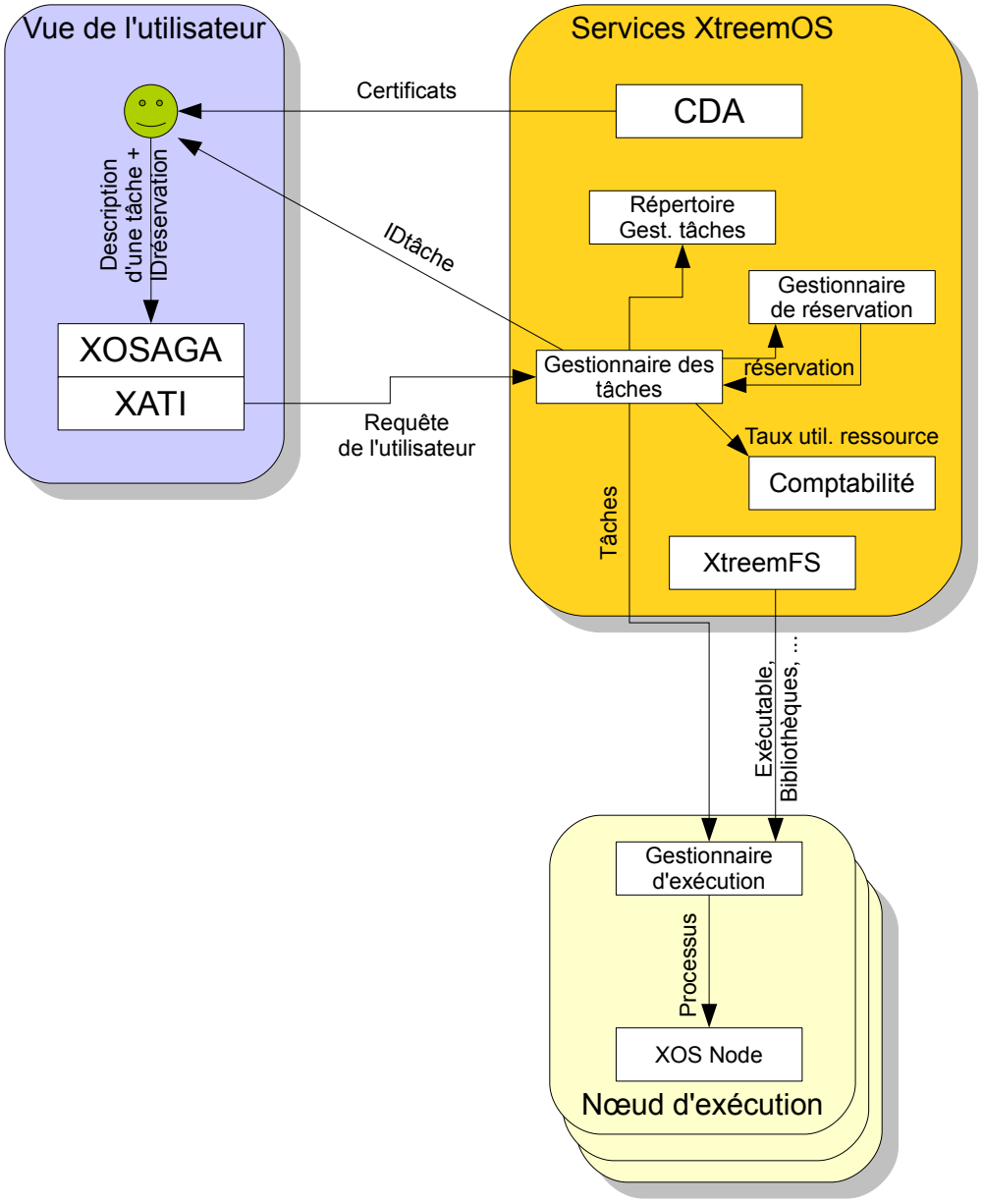


FIGURE 3.8 – Mécanisme de soumission d'une tâche avec une réservation existante

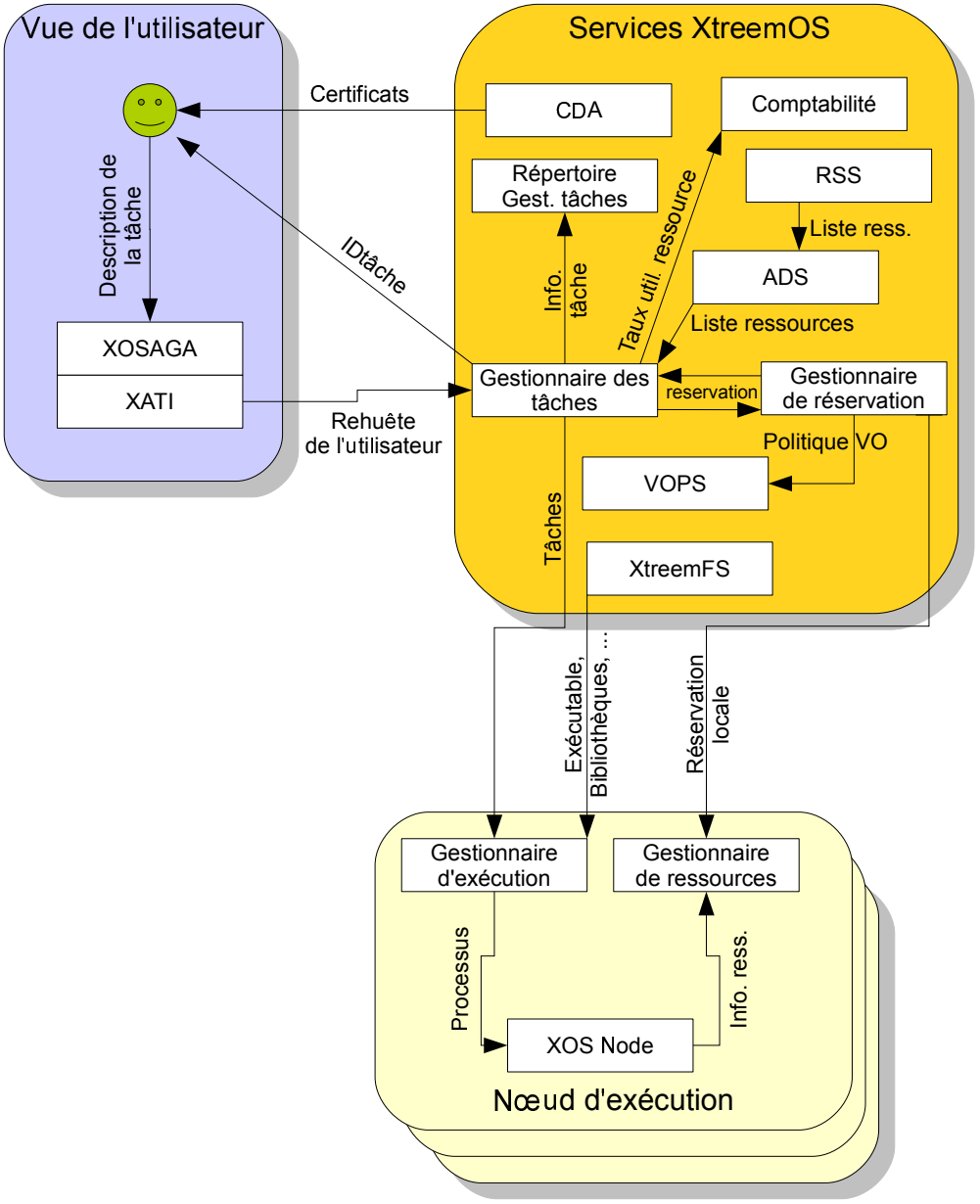


FIGURE 3.9 – Mécanisme de soumission d'une tâche sans réservation existante
Figure obtenue en combinant les figures 3.7 et 3.8.

Grillade met à la disposition des utilisateurs des machines virtuelles dans lesquelles ils peuvent déployer leur propre système d'exploitation. Ces machines virtuelles sont gérées par les mécanismes de Grillade de manière transparente pour les utilisateurs.

L'objectif de Grillade-CN en tant que gestionnaire de centrale numérique est d'apporter une abstraction supplémentaire dans la gestion et l'utilisation des ressources de la grille pour les administrateurs et les utilisateurs [63, 83, 91, 130]. Ce cas d'étude correspond à celui présenté sur la figure 3.2. Les points suivants résument les objectifs que nous nous fixons pour Grillade et que nous avions énoncés dans le tableau 3.2.

Gestion multi-domaines d'administration avec partage des ressources : créer et gérer des machines virtuelles tirant profits des nœuds distribués sur différents sites pouvant appartenir à des domaines d'administration différents et les gérer de manière transparente pour les utilisateurs ;

Flexibilité des environnements d'exécution : pouvoir offrir aux utilisateurs des environnements d'exécution différents de celui de Grillade ;

Flexibilité dans la gestion des ressources de l'infrastructure : pouvoir agréger les capacités des nœuds pour créer des machines virtuelles de capacité plus importante ;

Granularité : bénéficier des fonctionnalités de grille standard qu'offre XtreemOS pour la gestion des applications en ajoutant la capacité de gérer des machines virtuelles.

3.3.1 Sélection des mécanismes d'XtreemOS utiles pour la gestion des collections de machines virtuelles

Dans cette section, nous sélectionnons les fonctionnalités d'XtreemOS utiles dans la gestion des collections de machines virtuelles sur une grille. Parmi elles, on peut noter le service de gestion des organisations virtuelles (VO), le système de fichiers pour grille XtreemFS, le système à image unique LinuxSSI, et enfin, le contrôleur d'exécution d'applications. Les paragraphes suivants détaillent ces fonctionnalités en présentant leur pertinence par rapport aux objectifs à atteindre pour Grillade-CN.

3.3.1.1 Les organisations virtuelles

Les mécanismes de VO permettent de partager les ressources réparties sur plusieurs domaines d'administration. De plus, les VO permettent de définir des politiques de gestion des ressources et des utilisateurs. Par exemple, les administrateurs de Grillade peuvent définir des classes d'utilisateurs. Enfin, l'utilisation des VO permet la comptabilisation du taux d'utilisation des ressources, afin par exemple, de facturer à l'utilisateur sa consommation des ressources. Ce dernier point est d'ailleurs très important dans le contexte des centrales numériques étant donné que le client paye pour les ressources qu'il consomme.

Comme exemple, nous pouvons prendre le cas d'une centrale numérique gérée par Grillade disposant de ressources situées sur deux sites. Il est alors possible de définir des classes de clients appartenant à des VO. En supposant que le site 1 dispose de ressources très performantes de toute dernière génération et que le site 2 dispose de ressources d'une génération antérieure, moins performantes que celles du site 1, il est par exemple possible de définir 3 VO :

- VO1 : intègre les ressources du site 1 pour les clients soucieux de performance mais avec une facturation élevée,
- VO2 : intègre les ressources du site 2 pour les clients peu soucieux des performances et dont la facturation est moins importante,

– VO3 : intègre les ressources du site 1 et 2, exécute les applications des utilisateurs uniquement la nuit (20h - 4h) si les ressources sont disponibles (c'est à dire qu'elles ne sont pas utilisées par des applications de VO1 ou VO2), et est destinée aux clients voulant exécuter des tâches interruptibles nécessitant de nombreuses ressources.

3.3.1.2 Le système de fichiers pour grille XtreemFS

Dans le contexte de nos travaux, XtreemFS permet de stocker les images des collections de machines virtuelles ainsi que leurs fichiers de configuration aisément, et cela à l'échelle de la grille. Le processus transparent de réplication des fichiers d'XtreemFS permet à tous les nœuds de la grille disposant des droits d'accès de pouvoir lire et modifier ces images, et cela, même si le nœud qui les a créées venait à disparaître, par exemple à cause d'une défaillance. Il devient ainsi aisé de déplacer une collection de machines virtuelles d'un site à un autre : une fois sauvegardées dans XtreemFS, les machines virtuelles peuvent être démarrées depuis n'importe quel site.

3.3.1.3 Le système à image unique LinuxSSI

L'utilisation du système à image unique LinuxSSI dans XtreemOS pour les grappes de calculateurs, apporte une propriété unique à Grillade : pouvoir déployer des machines virtuelles dont les caractéristiques « matérielles », comme la taille de la mémoire ou le nombre de processeurs, sont plus grandes que les caractéristiques matérielles des ressources de calcul [87, 91]. En effet, les systèmes à image unique permettent d'agréger les capacités des ressources. Par exemple, avec LinuxSSI, deux ressources équipées chacune de 1 processeur et de 1 Go de mémoire vive seront « transformées » en une ressource constituée de deux processeurs et de 2 Go de mémoire.

Cette propriété est très pertinente dans le contexte des machines virtuelles. Par exemple, si un utilisateur demande la création d'une machine virtuelle équipée de 2 Go de mémoire, mais que seuls des nœuds de 1 Go sont disponibles, alors Grillade va utiliser LinuxSSI pour agréger deux ressources de 1 Go et « créer » une ressource équipée de 2 Go de mémoire. La machine virtuelle qui est ensuite exécutée sur cette agrégation de nœuds peut bénéficier des 2 Go de mémoire de manière transparente.

3.3.1.4 Le contrôleur d'exécution d'applications

L'action du contrôleur d'exécution d'application est nécessaire pour toutes les opérations relatives à la découverte, réservation, configuration et gestion des ressources de la grille. Les utilisateurs définissent leurs besoins par exemple en terme de taille mémoire, de nombre de processeurs, dans un fichier JSDL qui est transmis et interprété par XATI.

3.3.2 Conception de mécanismes de gestion des centrales numériques intégrés à XtreemOS

Les paragraphes suivants présentent les modifications apportées au système XtreemOS pour déployer et gérer des collections de machines virtuelles à l'échelle de la grille. Pour cela, il est nécessaire d'étendre les mécanismes de découverte ainsi que de réservation des ressources d'XtreemOS. Une fois les ressources découvertes et réservées, il faut déployer les collections de machines virtuelles et assurer la gestion de leur cycle de vie. Par défaut, la version PC d'XtreemOS est déployée sur tous les nœuds de la grille. Lors d'une requête, si nécessaire, Grillade reconfigure certains de ces nœuds pour les agréger en y déployant la version LinuxSSI d'XtreemOS. La figure 3.10 présente un exemple de collection des

machines virtuelles en cours d'exécution distribuées sur différents nœuds. La machine virtuelle VM1 est exécutée sur le nœud 1, la machine virtuelle VM2 est exécutée sur l'agrégation des nœuds 2 et 3 et enfin les machines virtuelles VM4 et VM5 sont exécutées sur le nœud 5. Lors de la terminaison de la collection (sa destruction), les nœuds 2 et 3 reviennent dans leur état par défaut (version PC d'XtreemOS sans agrégation) pour répondre à de nouvelles requêtes.

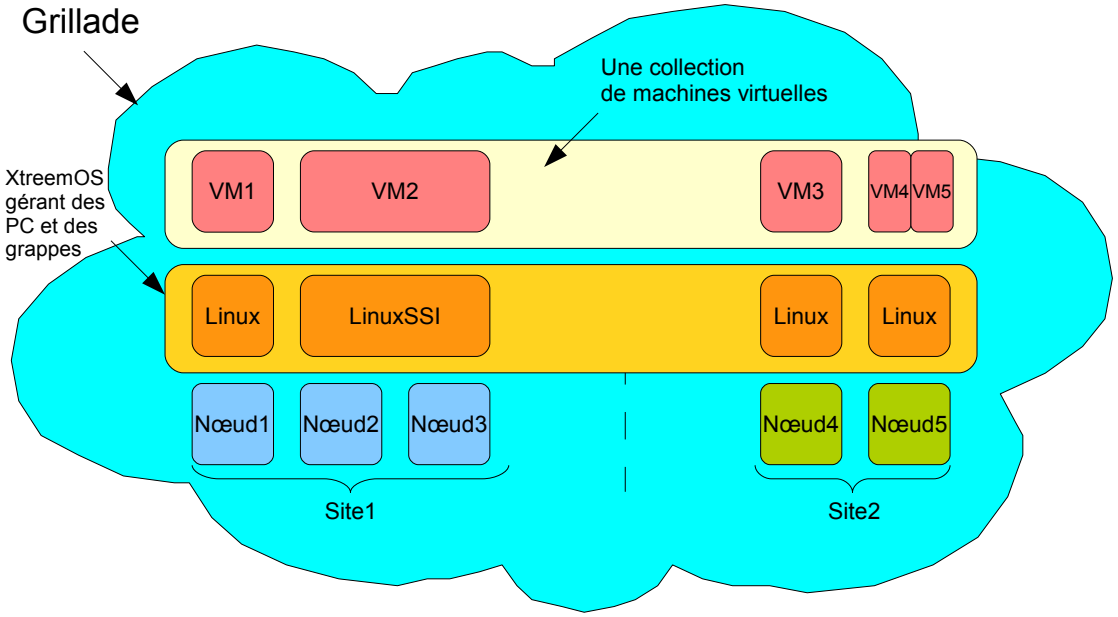


FIGURE 3.10 – Exemple de collection de machines virtuelles distribuée sur les ressources de deux sites

3.3.2.1 Cas idéal : gestion des collections de machines virtuelles directement intégrées à XtreemOS

La figure 3.11 présente une modification de la figure 3.9 décrivant la soumission des tâches dans XtreemOS, en y introduisant les modifications nécessaires pour assurer la gestion des collections des machines virtuelles.

Le gestionnaire de tâches et de machines virtuelles Le gestionnaire de tâches et de machines virtuelles est chargé de gérer deux types de requêtes : l'exécution d'une tâche, ce qui correspond au cas d'utilisation standard d'XtreemOS, mais également de gérer le cycle de vie des collections de machines virtuelles. L'exécution de tâches dans XtreemOS étant un cas standard d'utilisation déjà décrit dans la section 3.2.5.2, nous focalisons notre présentation sur la gestion des machines virtuelles.

Dans ce cas, la description des ressources que l'utilisateur veut obtenir est faite par un fichier JSDL (l'interface d'utilisation du fichier JSDL pourrait être étendue avec des interfaces standard aux centrales numériques comme EC2, c'est un choix lié à la mise en œuvre que nous ne détaillons pas plus ici). La requête de l'utilisateur est envoyée par l'intermédiaire de l'interface de communication XATI au gestionnaire de tâches et de machines virtuelles. Ce dernier initie une négociation auprès de l'ADS et du RSS afin de trouver des ressources adéquates pour l'exécution de la collection des machines virtuelles demandée. Cette négociation concerne la recherche de ressources disposant de capacité suffisante pour

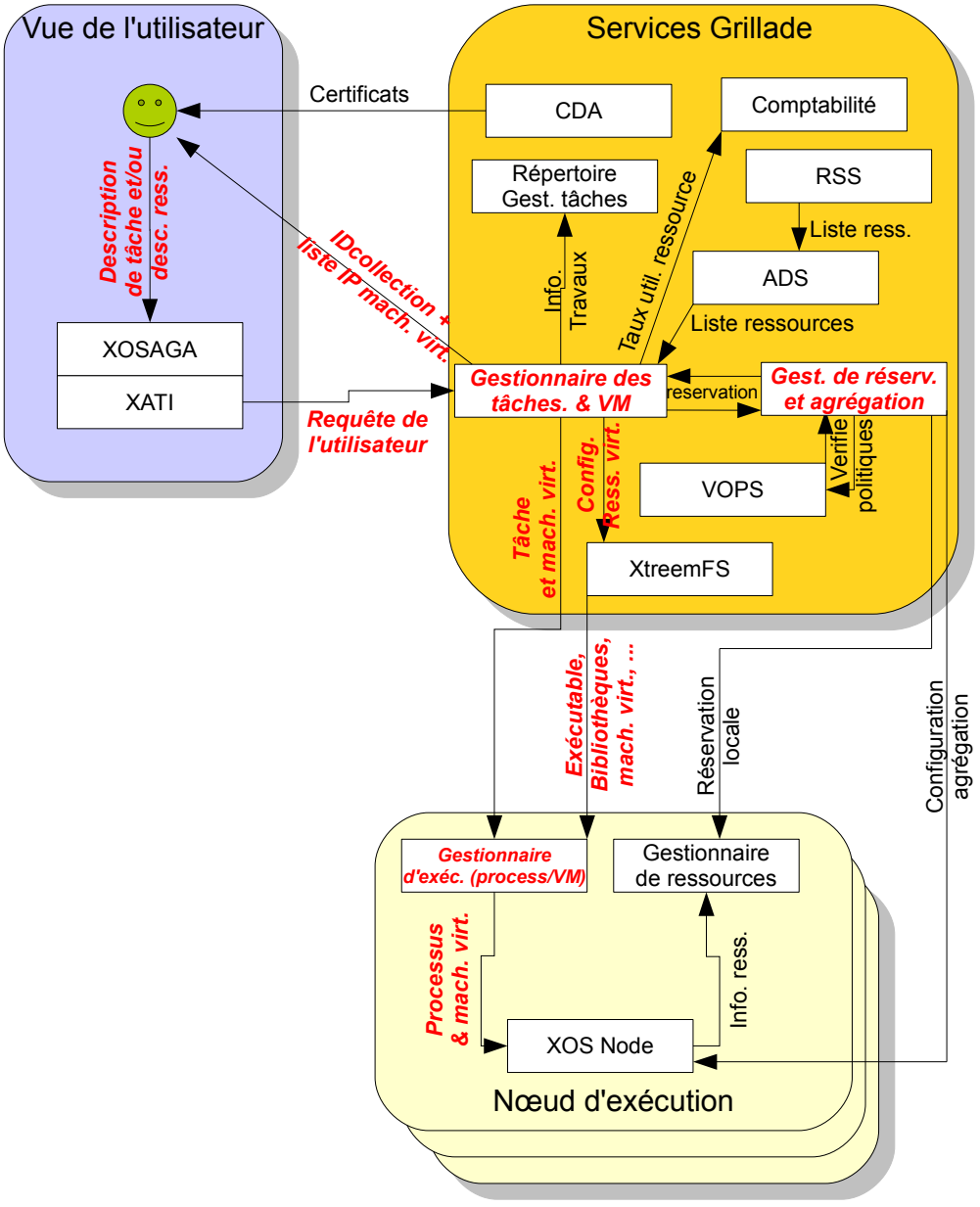


FIGURE 3.11 – Architecture de Grillade : modifications d'XtreemOS pour gérer nativement les collections de machines virtuelles
Cette figure est une modification de la figure 3.9 traitant de la soumission des tâches dans XtreemOS

exécuter des machines virtuelles, mais également si nécessaire, des ressources pouvant être agrégées afin de « créer » des ressources de capacité plus importantes.

Une fois la recherche effectuée, les nœuds sont réservés et agrégés si nécessaire par le gestionnaire de réservation et d'agrégation. À la suite de cette opération, le gestionnaire de tâches et de machines virtuelles crée les fichiers de configuration liés aux machines virtuelles dans XtreemFS. Cette étape consiste essentiellement en la configuration des adresses MAC et IP des machines virtuelles ainsi qu'à la configuration de leurs images.

Enfin la requête de démarrage des machines virtuelles peut être faite auprès du gestionnaire d'exécution de chaque ressource impliquée dans l'exécution de la collection des machines virtuelles. Finalement, l'utilisateur reçoit l'identifiant de la collection (*IDcollection*) qui a été créée ainsi que la liste des adresses IP de l'ensemble des machines virtuelles démarrées.

Le gestionnaire de réservation et d'agrégation Le gestionnaire de réservation et d'agrégation est chargé de réserver les ressources demandées par le gestionnaire de tâches et de machines virtuelles. Il vérifie que la réservation n'enfreint pas les politiques globales définies par les administrateurs à l'aide du service VOPS et réserve les ressources.

De plus, lorsqu'une agrégation de ressources est nécessaire, le gestionnaire de réservation et d'agrégation configure les nœuds avec LinuxSSI. Une fois l'agrégation faite, les différentes ressources agrégées sont vues par Grillade comme une seule et unique ressource. Une collection de machines virtuelles peut être exécutée de manière distribuée sur des nœuds et des nœuds agrégés.

Le gestionnaire d'exécution Le gestionnaire d'exécution est chargé de démarrer les machines virtuelles sur les ressources lorsque le gestionnaire de tâches et de machines virtuelles le sollicite. C'est également lui qui contrôle la bonne exécution des machines virtuelles.

3.3.2.2 Prise en compte des contraintes de mise en œuvre

Lors de la réalisation de ces travaux, la version stable la plus récente d'XtreemOS était la version 2.1. Cependant à cette même période, la nouvelle version stable d'XtreemOS, la version 3.0, était sur le point d'être publiée. Le problème que nous avions à prendre en compte est que la version 3.0 d'XtreemOS intègre des changements significatifs dans la mise en œuvre du contrôleur d'exécution d'application. Ces changements auraient rendu un portage de nos travaux de la version 2.1 à 3.0 très compliqué. C'est pour cela que nous avons choisi de dissocier le gestionnaire des tâches de celui des machines virtuelles.

3.3.3 Raffinement du cas idéal : conception de mécanismes de gestion des centrales numériques intégrés à XtreemOS en tenant compte des contraintes de mise en œuvre

Cette section présente un raffinement du cas idéal présenté dans la section précédente. Dans ce raffinement nous proposons un gestionnaire de nœuds, un gestionnaire d'agrégation, et un gestionnaire de machines virtuelles. De plus, nous proposons également une interface de gestion des centrales qui fait le lien entre les demandes de l'utilisateur et Grillade. Cette nouvelle architecture s'appuie sur la soumission et la gestion des tâches standard à XtreemOS. Ainsi, lorsqu'un utilisateur veut exécuter une tâche standard, il utilise les mécanismes d'XtreemOS standard, mais quand il veut exécuter des machines virtuelles, il utilise les nouveaux mécanismes que nous apportons à XtreemOS.

Dans la suite de cette section, nous ne développons que le cas d'un utilisateur souhaitant exécuter des machines virtuelles à la manière d'une centrale numérique.

La figure 3.12 présente l'architecture prenant en compte les contraintes de mise en œuvre liées à XtreemOS pour gérer les machines virtuelles à la manière des centrales numériques.

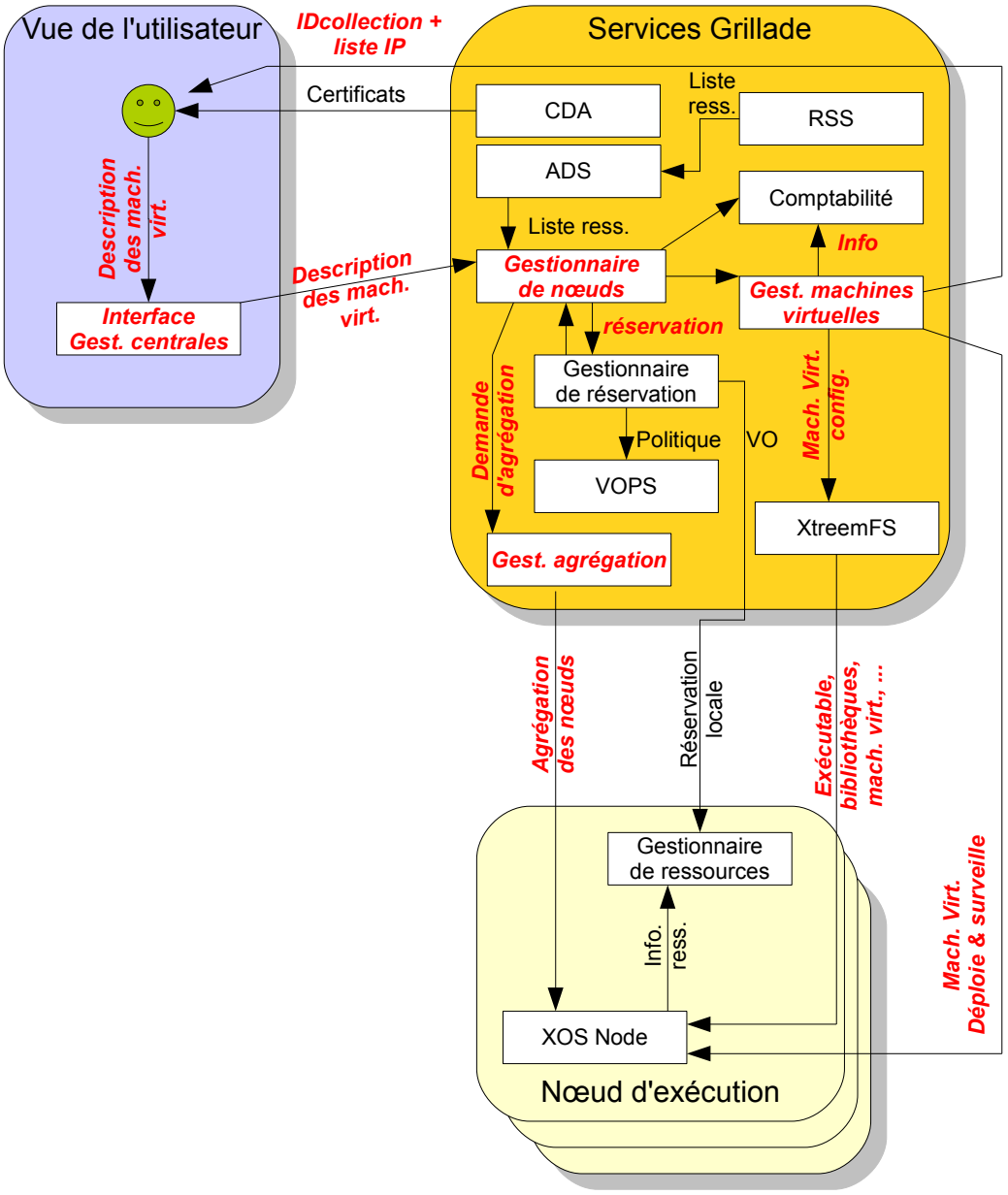


FIGURE 3.12 – Architecture de Grillade tenant compte des contraintes de mises en œuvre liées à XtreemOS pour le déploiement et la gestion des collections des machines virtuelles. Cette figure est un raffinement de la figure 3.11.

3.3.3.1 L'interface utilisateur

L'interface permet à l'utilisateur de décrire les ressources dont il a besoin (taille mémoire, nombre de processeurs, image à déployer, durée d'utilisation). Cette descrip-

tion est ensuite transmise au gestionnaire de centrale.

Par exemple, l'utilisateur spécifie sa requête ainsi : 1 machine virtuelle avec 3 Go de mémoire vive, 2 processeurs, utilisant l'environnement d'exécution disponible sur */images/environnement_linux.img*, pendant 3 heures.

3.3.3.2 Le gestionnaire de nœuds

Le gestionnaire de nœuds est chargé de trouver, réserver, configurer les ressources nécessaires pour exécuter les machines virtuelles.

a. La recherche des nœuds La recherche des nœuds consiste à trouver les nœuds pouvant exécuter la collection de machines virtuelles demandée par l'utilisateur. La recherche se fait via les services ADS et RSS. Tout d'abord, le gestionnaire de nœuds recherche des ressources dont les caractéristiques matérielles suffisent à l'exécution des machines virtuelles demandées. Si un nombre insuffisant de ressources est trouvé, le gestionnaire de nœuds relance une recherche avec des caractéristiques moins ambitieuses, dans le but d'agréger les ressources trouvées.

De plus, en l'état actuel des technologies les systèmes d'agrégation ne peuvent agréger que des ressources se trouvant sur un même réseau local. Des techniques de réseaux virtuels (VPN) peuvent être mises en place entre des ressources appartenant à différents sites pour simuler un même réseau local. Cependant, cette complexité de mise en œuvre ainsi que les performances très modestes obtenues font que nous prenons le parti de considérer que les systèmes d'agrégation ne peuvent être utilisés que sur des ressources appartenant à un réseau local d'un même site.

- Recherche de nœuds disposant des capacités matérielles suffisantes Ce cas de recherche permet de trouver des ressources de la grille disposant d'un système de virtualisation avec les capacités suffisantes pour exécuter les machines virtuelles. Les machines virtuelles peuvent alors être créées directement sur ces ressources, par exemple, une machine virtuelle avec 2 Go de mémoire pourra être exécutée sur une ressource disposant de 4 Go de mémoire libre.

- Recherche de nœuds pouvant être reconfigurés avec LinuxSSI Ce cas de recherche permet de trouver des ressources de la grille qui peuvent être configurées en tant que ressources de LinuxSSI. En effet, si aucun nœud ne dispose des capacités suffisantes pour exécuter une machine virtuelle, il est alors possible d'agréger plusieurs nœuds avec LinuxSSI afin de créer des ressources de plus grande capacité. Par exemple, une machine virtuelle avec 2 Go de mémoire, pourra être exécutée sur une grappe LinuxSSI constituée de deux ressources ayant chacune 1 Go de mémoire libre.

b. La réservation des nœuds Une fois les ressources trouvées, celles-ci doivent être réservées par l'intermédiaire du gestionnaire de réservation qui effectue les vérifications d'autorisation de réservation des ressources avec le service VOPS. La réservation des ressources est identique au procédé de réservation déjà présenté dans la section 3.2.6.2.

c. L'agrégation des nœuds Une fois les nœuds réservés, si nécessaire, le gestionnaire de nœuds interagit avec le gestionnaire d'agrégation pour déployer LinuxSSI sur l'ensemble ou une partie de l'ensemble des ressources trouvées.

Finalement, une liste de ressources, agrégées ou non, est envoyée au gestionnaire de machines virtuelles qui va pouvoir procéder à la configuration et au déploiement de la collection de machines virtuelles.

3.3.3.3 Gestionnaire d'agrégation des nœuds

Le gestionnaire d'agrégation configure LinuxSSI sur un ensemble de nœuds fourni par le gestionnaire de nœuds. C'est également lui qui est en charge, en cas d'erreur lors de la configuration du système d'agrégation (défaillance d'un nœud par exemple) de prévenir le gestionnaire de nœuds pour libérer les ressources et exécuter une nouvelle requête d'agrégation sur de nouvelles ressources.

Lors de la terminaison d'une collection de machines virtuelles, sur demande du gestionnaire nœuds, le gestionnaire d'agrégation reconfigure les ressources dans leur état d'origine, c'est-à-dire sans LinuxSSI.

La configuration des ressources est faite par Grillade qui dispose des droits d'administration nécessaires pour configurer ses propres ressources. Ces droits d'administration sont gérés en interne par Grillade et n'ont pas besoin d'être donnés à l'utilisateur de la machine virtuelle créée. Dans ce chapitre, nous n'abordons pas les problèmes de sécurité liés à une utilisation malicieuse de Grillade.

3.3.3.4 Le gestionnaire de machines virtuelles

Le gestionnaire de machines virtuelles a la charge, sur requête du gestionnaire de nœuds, de configurer les fichiers relatifs aux machines virtuelles dans XtreemFS avant de procéder à leur démarrage et ensuite à leur surveillance tout au long de leur cycle de vie.

Configuration des machines virtuelles La configuration des machines virtuelles consiste principalement à leur attribuer des adresses MAC et IP uniques. Le système d'attribution de MAC et IP utilisé est le même que celui présenté dans le chapitre relatif à Saline, en section 2.3.3.4.

Premier démarrage des machines virtuelles Une fois la phase de configuration terminée, le gestionnaire de machines virtuelles démarre les machines virtuelles. Les adresses IP de l'ensemble des machines virtuelles ainsi qu'un identifiant unique caractérisant la collection de machines virtuelles (*IDcollection*) sont retournés à l'utilisateur. Ce dernier peut alors se connecter directement à ses machines virtuelles.

Surveillance et gestion des collections de machines virtuelles La surveillance et la gestion des machines virtuelles sont assurées par le gestionnaire de machines virtuelles.

Les machines virtuelles peuvent être exploitées par les utilisateurs de manière non-interactive ou interactive. Dans le cas d'une tâche non-interactive, des opérations de migration et de sauvegarde et redémarrage d'un point de reprise peuvent être faites de manière transparente pour l'utilisateur (cela se fait grâce aux mécanismes de Saline présentés au chapitre 2 et au système de fichiers pour grille XtreemFS qui permet de stocker les images des machines virtuelles et de les restaurer sur un site différent de celui d'origine).

Dans le cas d'une tâche interactive, les migrations ainsi que les sauvegardes et restaurations des machines virtuelles doivent être contrôlées et prévues, afin que l'utilisateur ne perde pas le bénéfice de l'interactivité à cause des interruptions de service liées à l'utilisation de ces mécanismes. Dans ce document, nous considérons que les tâches non-interactives peuvent être déplacées et sauvegardées de manière totalement automatique,

et que les tâches interactives peuvent être déplacées et sauvegardées uniquement sur demande de l'utilisateur ou de l'administrateur, pour prévenir une défaillance matérielle par exemple. Dans tous les cas, ces mécanismes sont les mêmes que ceux conçus dans Saline et présentés dans la section 2.3.3.5.

Enfin, lors de la terminaison d'une collection de machines virtuelles, le gestionnaire de machines virtuelles interagit avec le gestionnaire de nœuds afin de libérer les nœuds et de les reconfigurer dans leur état initial.

3.3.4 Mise en œuvre de Grillade-CN

Cette section présente les éléments de mise en œuvre de Grillade-CN. Ceux-ci s'appuient sur les mécanismes de gestion des collections de machines virtuelles que nous avons décrits dans le chapitre 2 sur Saline. Le prototype présenté permet de déployer et gérer des machines virtuelles avec des caractéristiques définies en terme de nombre de processeurs et de taille de la mémoire pour une durée déterminée. Dans une prochaine version du prototype, il serait intéressant d'intégrer d'autres critères comme par exemple, l'espace disque disponible ou le taux d'utilisation de bande passante du réseau. La figure 3.13 présente les mécanismes de gestion des centrales numériques dans Grillade.

3.3.4.1 L'interface utilisateur

Comme décrit précédemment, l'interface utilisateur transfère la requête de l'utilisateur au gestionnaire de nœuds. Pour plus de clarté, nous décrivons les mécanismes utilisés pour la création et la gestion d'une machine virtuelle. Le principe reste le même pour la création et la gestion de collections de machines virtuelles.

3.3.4.2 Le gestionnaire de nœuds

Le gestionnaire de nœuds a la charge de rechercher des ressources et de les configurer en vue du démarrage de la machine virtuelle demandée.

La recherche des nœuds (étapes 1bis. à 4. de la figure 3.13) XOSAGA propose une interface de soumission des tâches à l'aide de fichiers JSDL (*Job Submission Description Language*) qui décrit la tâche à effectuer ainsi que les ressources nécessaires à son exécution. Lors d'une requête de création de machines virtuelles le gestionnaire de nœuds transforme la requête en un fichier JSDL. Ce fichier JSDL contient une description de tâche « vide » et une description des ressources correspondant aux caractéristiques matérielles de la machine virtuelle demandée. La description des ressources se fait en spécifiant différents paramètres permettant de décrire précisément les ressources recherchées de manière individuelle, ressource par ressource, ou bien de manière globale pour un ensemble de ressources. Par exemple, les paramètres *IndividualCPUCount* ou *IndividualPhysicalMemory* indiquent le nombre de processeurs ou la taille de la mémoire requis pour une ressource. D'autres paramètres comme *TotalCPUCount* ou *TotalPhysicalMemory* permettent de spécifier un nombre total de processeurs ou de mémoire pour un ensemble de ressources. Cependant, dans la version 2.1 d'XtreemOS, les paramètres de recherche globaux ne sont pas pris en compte : il n'est donc pas possible de faire une recherche utilisant les paramètres *TotalCPUCount* ou *TotalPhysicalMemory*.

Cette limitation étant, le gestionnaire de nœuds fait une recherche en commençant par déterminer s'il existe au moins un nœud capable d'exécuter la machine virtuelle requise. Pour ce faire, il lance une requête de recherche auprès des services ADS et RSS. En retour,

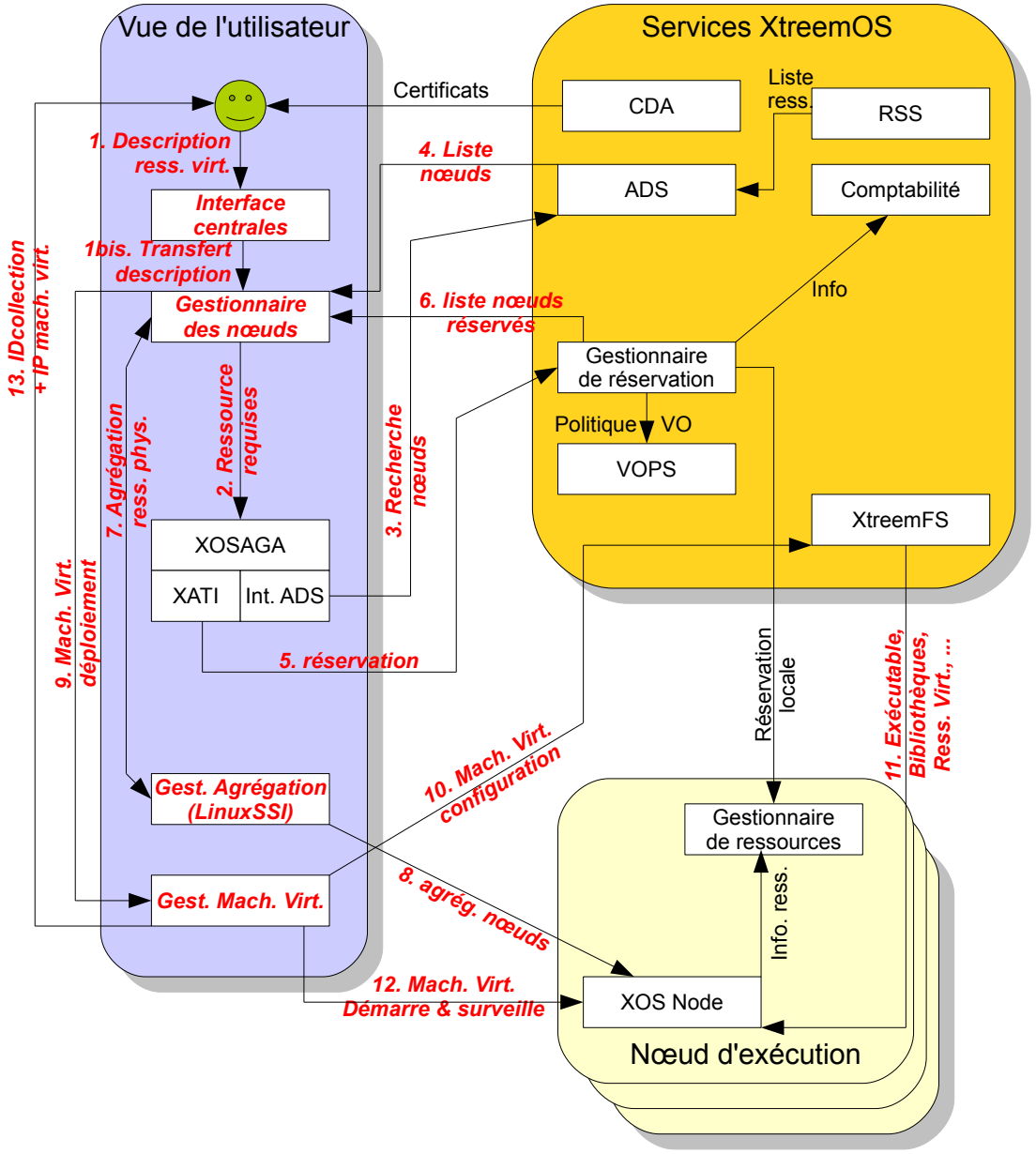


FIGURE 3.13 – Architecture de mise en œuvre de la gestion des centrales numériques dans Grillade.

si au moins un nœud correspondant à la requête est trouvé, le gestionnaire de nœuds le réserve et envoie une requête de déploiement de machines virtuelles au gestionnaire de machines virtuelles. Sinon, le gestionnaire des nœuds relance une recherche, mais cette fois-ci en demandant plusieurs ressources possédant des critères de capacité plus petits, dans le but de les agréger. Lorsque la liste des ressources est constituée, elle est transmise au gestionnaire d'agrégation, et ensuite au gestionnaire de machines virtuelles pour déploiement des machines virtuelles.

La réservation des nœuds (étapes 5. et 6. de la figure 3.13) La réservation des nœuds se fait selon le mode de réservation standard d'XtreemOS que nous avons déjà détaillé dans la section 3.2.6.2.

L'agrégation des nœuds (étapes 7. et 8. de la figure 3.13) Si aucun nœud n'est capable d'exécuter la machine virtuelle demandée, une liste de nœuds dont l'agrégation est susceptible de répondre au besoin est formée. Cette liste est transmise au gestionnaire d'agrégation pour configurer les ressources avec Kerrighed (LinuxSSI).

Ce service est en charge de configurer les ressources à agréger. Cela revient à vérifier que le noyau spécifique à Kerrighed est en cours d'exécution. Si ce n'est pas le cas, le service redémarre les nœuds à agréger afin qu'ils démarrent sur le noyau spécifique à Kerrighed avec les paramètres de démarrage appropriés. La configuration de Kerrighed entre toutes les ressources nouvellement redémarrées peut ensuite se faire.

Démarrage des machines virtuelles, surveillance et terminaison (étapes 9. à 13. de la figure 3.13) Une fois que la configuration des nœuds ressources est faite, le gestionnaire de nœuds envoie une requête au gestionnaire de machines virtuelles qui est en charge de la gestion du cycle de vie des machines virtuelles.

3.3.4.3 Le gestionnaire d'agrégation des nœuds

Dans XtreemOS, la mise en œuvre de LinuxSSI est faite avec le système d'exploitation à image unique Kerrighed. Par défaut, les ressources d'XtreemOS exécutent un noyau Linux standard. Or, Kerrighed nécessite l'utilisation d'un noyau Linux spécial avec des paramètres spéciaux comme un identifiant unique à la grappe, sur l'ensemble des ressources à agréger. Le gestionnaire d'agrégation configure les ressources en remplaçant le noyau Linux par défaut par celui de Kerrighed pour toutes les ressources à agréger (étapes 7. et 8. de la figure). Ensuite, le gestionnaire d'agrégation redémarre l'ensemble des nœuds. Lors de leur premier démarrage, ces nœuds, grâce aux fonctionnalités de configuration automatique de Kerrighed, vont se découvrir mutuellement et s'agréger. À la fin de cette étape, la grappe Kerrighed est vue par XtreemOS comme une ressource unique correspondant à l'agrégation de plusieurs ressources. À la fin de l'opération, un acquittement est alors envoyé au gestionnaire de nœuds.

Si le déploiement de Kerrighed échoue, les nœuds sont libérés, le gestionnaire de nœuds est prévenu, et c'est à lui qu'il revient de démarrer une nouvelle procédure de recherche et configuration des ressources.

3.3.4.4 Le gestionnaire de machines virtuelles

Le gestionnaire de machines virtuelles est chargé de la configuration et du déploiement des machines virtuelles (étapes 9. à 13. de la figure 3.13). Le prototype actuel déploie des machines virtuelles de type QEMU sans accélération matérielle afin d'être le plus

indépendant possible par rapport à Kerrighed dont la mise en œuvre actuelle ne gère pas les systèmes de virtualisation à accès matériel.

Les fichiers de configuration ainsi que les images des machines virtuelles sont stockés dans XtreemFS.

La configuration des machines virtuelles, principalement liée à la configuration de leurs adresses MAC et IP, de même que la surveillance du cycle de vie des machines virtuelles se font grâce aux mécanismes de Saline, décrit dans le chapitre 2. Notons que grâce à XtreemFS, il n'est pas nécessaire de copier les images d'une ressource à une autre pour les sauvegarder ou les déplacer.

Une fois la collection de machines virtuelles déployée, l'ensemble de leurs adresses IP est retourné à l'utilisateur pour qu'il puisse s'y connecter. De plus, dans la mise en œuvre de notre prototype, nous tirons avantage des fonctionnalités VNC (*Virtual Network Computing*) offertes par QEMU pour permettre aux utilisateurs de se connecter graphiquement à leurs machines virtuelles même si elles ne disposent pas de connexion de type *ssh* par exemple.

Lors de la fin d'exécution d'une machine virtuelle, les résultats des tâches sont sauvegardés sur XtreemFS, celle-ci est arrêtée et ses adresses MAC et IP sont libérées selon les mêmes mécanismes que ceux présentés dans Saline (voir sections 2.3.3.7 et 2.3.4.1).

3.3.5 Validation et expérimentations

Le prototype de Grillade-CN est mis en œuvre en langage de programmation *bash* et interagit avec XtreemOS 2.1 grâce à l'interface XOSAGA. Ce prototype a été mis en œuvre et validé dans le contexte de Grid'5000. Nous avons mené différentes expérimentations afin de valider chaque mécanisme individuellement.

Tout d'abord, nous évaluons le cas d'une requête de création d'une machine virtuelle dont les caractéristiques matérielles permettent de l'exécuter directement sur une ressource disponible. Puis nous évaluons le cas d'une requête de création d'une machine virtuelle dont les caractéristiques nécessitent l'agrégation de plusieurs nœuds.

De plus, nous évaluons le temps de reconfiguration de LinuxSSI, le temps de migration d'une machine virtuelle d'un nœud à un autre (intra-site), l'impact de l'utilisation de LinuxSSI et enfin, nous validons l'utilisation du système de fichiers XtreemFS pour stocker les images des machines virtuelles avant de présenter un cas d'utilisation des machines virtuelles et des VO.

3.3.5.1 Temps de reconfiguration des ressources

Dans cette expérience nous présentons le temps nécessaire à Grillade pour configurer des ressources XtreemOS avec Kerrighed (la mise en œuvre de LinuxSSI dans XtreemOS) afin de les agréger. Dans notre expérimentation nous utilisons une grille composée d'un nœud de service et de plusieurs nœuds ressource (de type PC). Le temps de configuration mesuré correspond au temps nécessaire pour configurer les ressources et les redémarrer de manière « agrégées ». Le temps de dé-configuration des ressources consiste au temps nécessaire pour redémarrer les nœuds sans Kerrighed.

Le graphique 3.14 présente les résultats de cette expérience. D'une manière générale, le temps d'agrégation varie très peu quel que soit le nombre de ressources (en moyenne 351 secondes – 5min51s). Cela est dû au fait que les actions à réaliser pour la configuration et le redémarrage des nœuds se font en parallèle sur les nœuds concernés. De plus, ce temps de configuration est principalement lié au temps de redémarrage des nœuds. La configuration de Kerrighed nécessite de donner des arguments au noyau Linux (par exemple l'identifiant

de session utilisé par toutes les ressources d'une même grappe Kerrighed). Ces arguments ne peuvent être attribués en cours d'exécution des ressources. Il est donc nécessaire de redémarrer les ressources pour leur donner les arguments noyau concernés. De plus, à cause d'un problème de mise en œuvre lié à la gestion des identifiants de session dans Kerrighed, les nœuds doivent être redémarrés une seconde fois. Ce problème qui sera réglé dans les nouvelles versions permettra de s'affranchir d'un second redémarrage des nœuds et ainsi d'avoir un temps d'agrégation plus court.

La même remarque peut être faite concernant le temps de dés-agrégation des ressources : le temps de dés-agrégation (en moyenne 149 secondes – 2min29s) varie peu par rapport au nombre de ressources parce que les actions à réaliser sont faites en parallèle.

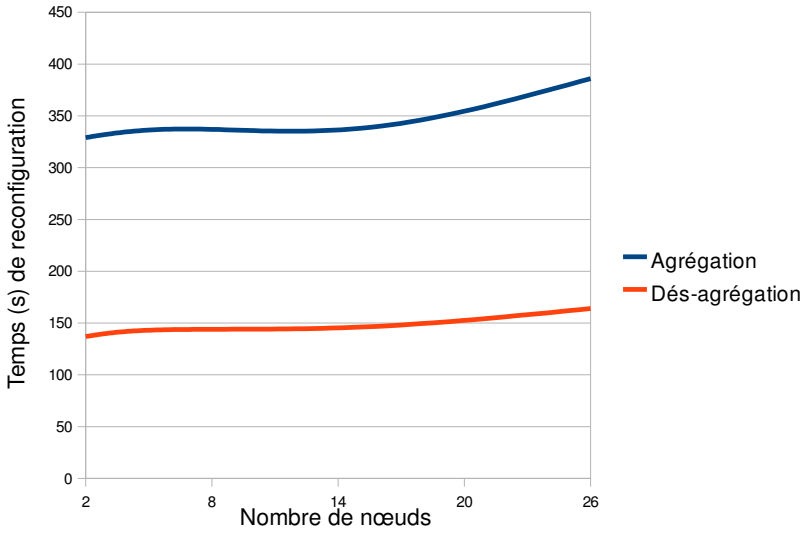


FIGURE 3.14 – Temps d'agrégation et de dés-agrégation des nœuds

Notons que la différence de temps existant entre le temps de configuration des ressources lors de leur agrégation et celui de dés-agrégation est principalement dûe au fait que l'opération de dés-agrégation ne nécessite qu'un redémarrage des ressources contrairement à celle d'agrégation qui en nécessite deux.

3.3.5.2 Temps de déploiement

Cette expérience présente le temps de déploiement des machines virtuelles qui consiste à mettre en place les images des machines virtuelles pour leur démarrage. Utilisant des images profitant de la technique de la copie sur écriture, deux fichiers doivent être considérés : l'image de référence et le fichier contenant les différences créé et mis à jour au fur et à mesure des écritures successives. Dans notre cas, l'image de référence est située dans XtreemFS. Le déploiement d'une collection de machines virtuelles consiste à créer les différents fichiers de différences dans XtreemFS et ensuite de démarrer les machines virtuelles.

Nous avons réalisé nos tests sur un ensemble de 2 à 21 nœuds. Pour des raisons de clarté, nous avons limité le déploiement des machines virtuelles à une machine virtuelle par nœud. Les résultats obtenus montrent que, quel que soit le nombre de nœuds, le temps de déploiement des machines virtuelles est constant de l'ordre de *2s*. Ce temps s'explique par le fait que l'opération de création du fichier de différence de la machine virtuelle est très faible (la création d'un fichier de différence revient à créer un fichier vide).

3.3.5.3 Temps de migration à froid (intra-site)

Cette expérience présente l'impact de la migration d'une machine virtuelle entre deux nœuds ressource du même site en utilisant XtreemFS, le système de fichiers pour grille d'XtreemOS.

Le temps de migration d'une machine virtuelle à froid peut être décomposé de la manière suivante :

1. faire une sauvegarde de l'état de la machine virtuelle dans XtreemFS,
2. restaurer l'état sauvegardé sur un nœud distant à partir d'XtreemFS.

Nous proposons d'étudier le temps de migration des machines virtuelles pendant leur phase de démarrage. En effet, la phase de démarrage d'une machine virtuelle est une phase qui allie calcul CPU, accès disque, accès mémoire : ce sont des opérations qui sont coûteuses. Lors de nos expérimentations, nous avons réalisé une migration (sauvegarde et redémarrage) à la 30^{e} seconde de la phase de démarrage des machines virtuelles.

1) Temps de sauvegarde Le temps de sauvegarde des machines virtuelles correspond au temps nécessaire pour sauvegarder leur état courant, état à partir duquel elles peuvent être redémarrées par la suite, si besoin.

Dans notre cas, l'image de référence ainsi que les images de différence étant déjà stockées dans XtreemFS, aucune opération supplémentaire n'est nécessaire pour les sauvegarder. Ainsi, l'opération de sauvegarde consiste à stocker l'état de la mémoire et des registres des machines virtuelles en cours d'exécution. La version 0.9.1 de QEMU que nous utilisons sauvegarde dans un même fichier, appelé fichier COW, les éléments que dans ce document nous nommons fichier de différence et fichier d'état contenant la mémoire et les registres. Dans nos expérimentations le fichier COW d'une machine virtuelle créé à la 30^{e} seconde après son démarrage est de l'ordre de 30 Mo.

Les résultats présentés sur le graphique 3.15 montrent que plus le nombre de machines virtuelles augmente, et plus le temps de sauvegarde s'allonge. Ces résultats s'expliquent de part la configuration d'XtreemFS que nous avons utilisée. En effet, XtreemFS est organisé en trois parties : un répertoire d'accès aux informations (DIR), un gestionnaire de métadonnées (MRC) et un gestionnaire d'objets (OSD). C'est le gestionnaire d'objets qui stocke réellement les données. Notre configuration, qui est celle par défaut, utilise un seul OSD à la fois. Ainsi, lors de la sauvegarde, toutes les machines virtuelles copient simultanément leur fichier COW vers le même OSD et créent un goulot d'étranglement du réseau du côté de l'OSD. Cependant, dans un contexte de flexibilité et en se détachant des performances, XtreemFS permet de sauvegarder les machines virtuelles de manière simple.

2) Temps de restauration Le temps de restauration d'une machine virtuelle consiste à lire les données préalablement sauvegardées dans XtreemFS. Cela se fait de manière parallèle et non bloquante. Ainsi, quel que soit le nombre de machines virtuelles à restaurer, leur redémarrage depuis la sauvegarde se fait dans la seconde comme le montre la figure 3.15. Cependant, le temps de restauration que nous mesurons ici n'évalue pas les performances de la machine virtuelle juste après son redémarrage (différentes machines virtuelles faisant parallèlement des lectures vers un même OSD peut créer un goulot d'étranglement). Ce point fait l'objet d'une autre expérimentation présentée dans la section 3.3.5.5.

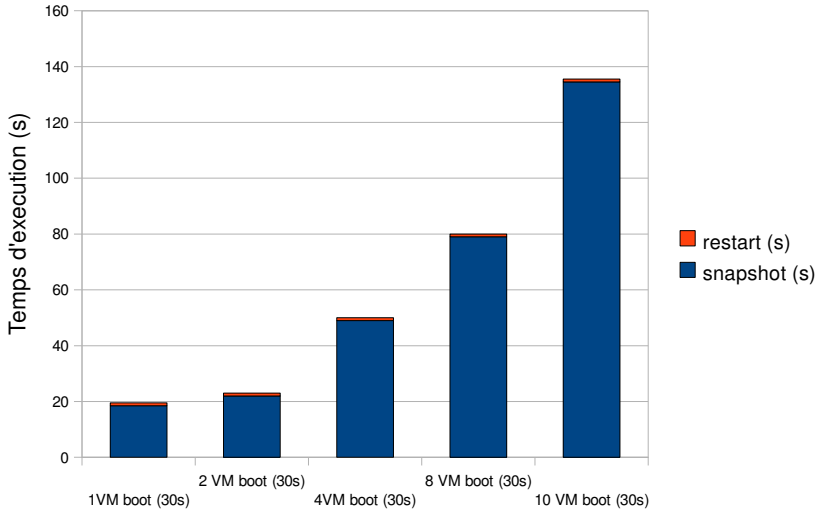


FIGURE 3.15 – Temps de migration de collections de machines virtuelles avec XtreemFS

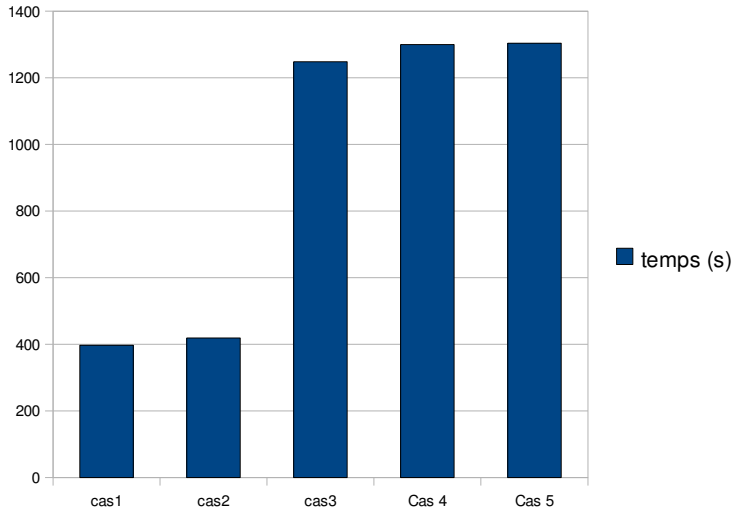
3.3.5.4 Impact de Kerrighed sur l'agrégation mémoire

Cette expérimentation évalue l'impact en terme de performance de l'exécution d'une machine virtuelle sur un ou plusieurs nœuds agrégés. Cette machine virtuelle exécute une application qui écrit en mémoire en permanence. Pour ces expérimentations nous avons utilisé des nœuds équipés de 8 Go de mémoire vive dont nous avons, selon les cas expliqués ci-dessous, limité la taille.

Voici les différents cas que nous avons étudiés et qui sont présentés sur la figure 3.16 :

1. Dans un premier cas, celui de référence, nous déployons une machine virtuelle de 1,5 Go sur un nœud sans Kerrighed disposant de 2 Go de mémoire vive. L'application que l'on déploie dans la machine virtuelle utilise 1 Go de sa mémoire.
2. Dans un second cas, nous évaluons l'impact de Kerrighed sur l'utilisation de la mémoire. Dans ce cas, nous déployons une machine virtuelle de 1,5 Go de mémoire sur un nœud de 2 Go de mémoire locale géré par Kerrighed et disposant de 4 Go de mémoire (2 nœuds). L'application utilisée fait 1 Go.
3. Dans un troisième cas, nous évaluons l'impact de l'utilisation de la mémoire distante avec Kerrighed. La machine virtuelle utilisée occupe 3,5 Go, l'agrégation mémoire des nœuds est de 6 Go ($2 \times 3\ Go$) et l'application utilise 3 Go de mémoire.
4. Dans un quatrième cas, nous déployons une machine virtuelle de 3,5 Go sur un nœud sans Kerrighed disposant de 8 Go de mémoire. L'application exécutée est de 3 Go.
5. Dans un cinquième cas, nous déployons une machine virtuelle de 3,5 Go sur un nœud agrégée de 6 Go ($3 \times 2\ Go$) et une application de 3 Go.

En conclusion, nous notons que l'impact de l'utilisation de Kerrighed lorsque la mémoire distante n'est pas utilisée est faible (397s sans Kerrighed – cas 1 – contre 419s avec Kerrighed – cas 2 –, soit une perte de 5%). De plus, l'impact de l'utilisation de Kerrighed lorsque la mémoire distante est utilisée est négligeable également (1248s pour Kerrighed agrégeant 2 ressources – cas 3 –, 1300s pour une ressource sans Kerrighed – cas 4 – et 1304s pour Kerrighed agrégeant 3 ressources – cas 5 –, soit un écart maximum de 4%). L'utilisation de Kerrighed comme système d'agrégation pour fournir plus de flexibilité dans la



	Nb ress. agrégées	Taille mem. agrégée	Taille mem. une ress. phys.	Taille mem. mach. virt.	Taille mem. appli.
Cas 1	0	N/A	2 Go	1,5 Go	1 Go
Cas 2	2	4 Go	2 Go	1,5 Go	1 Go
Cas 3	2	6 Go	3 Go	3,5 Go	3 Go
Cas 4	0	N/A	8 Go	3,5 Go	3 Go
Cas 5	3	6 Go	2 Go	3,5 Go	3 Go

FIGURE 3.16 – Impact de l'utilisation de la mémoire agrégée (partagée entre différentes ressources) sur le temps d'exécution d'une application consommatrice de mémoire

gestion des ressources est validée, et cette solution, d'un point de vue mémoire n'entraîne pas de perte significative de performance pour l'application exécutée.

3.3.5.5 Impact de l'utilisation d'XtreemFS

Cette expérimentation évalue l'impact de l'utilisation du système de fichiers distribué XtreemFS lors du démarrage des machines virtuelles. Pour ce faire, nous démarrons une collection de machines virtuelles composée de 1 à 20 machines virtuelles et notons le temps nécessaire au démarrage de la machine virtuelle de la collection qui prend le plus de temps à démarrer. La figure 3.17 présente les résultats de cette expérimentation.

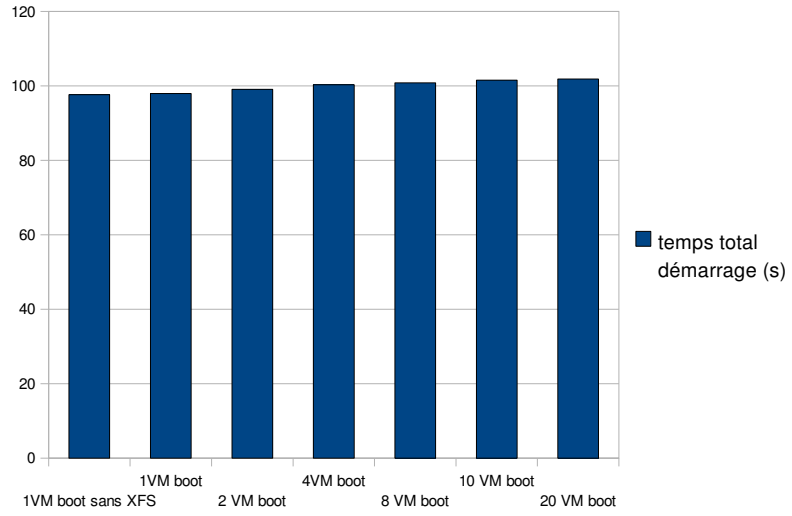


FIGURE 3.17 – Impact de l'utilisation d'XtreemFS lors du démarrage des machines virtuelles

Ces résultats montrent que le nombre de machines virtuelles n'influe pas sur le temps que mettent les machines virtuelles à démarrer : moins de 5% de différence de temps entre le démarrage d'une machine virtuelle sur disque local et le démarrage de 20 machines virtuelles réparties sur 20 nœuds dont les images sont stockées dans XtreemFS. Cette expérimentation montre qu'il n'existe pas pour une collection allant jusqu'à 20 machines virtuelles, avec l'utilisation de la technique de la copie sur écriture, de ralentissement significatif entraînant une baisse des performances en utilisant XtreemFS. Cela valide le fait qu'XtreemFS apporte de la flexibilité dans la gestion des ressources en permettant de gérer les images des machines virtuelles à l'échelle d'une grille.

3.3.5.6 Utilisation des organisations virtuelles dans la gestion des collections de machines virtuelles

Cette expérience présente un cas d'utilisation de Grillade en tant que centrale numérique exploitant des ressources situées sur différents domaines d'administration. Nous considérons le scénario suivant. Les dirigeants d'une centrale numérique (*centrale 1* – par exemple en Europe –) décident de s'associer avec une autre (*centrale 2* – par exemple aux États-Unis –) afin de partager leurs ressources et de pouvoir offrir à leurs clients un plus large choix de caractéristiques de machines virtuelles. Ils définissent différentes classes de clients : les clients qui veulent des ressources situées géographiquement le plus proche

d'eux, les clients qui veulent des machines virtuelles avec des contraintes de qualité particulière ne pouvant être exécutées que sur les meilleures ressources de la *centrale 1* et de la *centrale 2*, et enfin, les clients sans contrainte particulière. Chacune de ces catégories peut être associée à une tarification spécifique.

Les centrales concernées installent Grillade sur l'ensemble de leurs ressources et utilisent des VO pour gérer les différentes classes de clients :

- une VO (*vo-centrale1*) contenant toutes les ressources de la *centrale 1*,
- une VO (*vo-centrale2*) contenant toutes les ressources de la *centrale 2*,
- une VO (*vo-gold*) contenant les meilleures ressources des *centrales 1* et *2*,
- une VO (*vo-besteffort*) contenant toutes les ressources des *centrales 1* et *2*.

À des fins de validation, nous avons réalisé cette expérimentation entre les sites de Rennes et Sophia de Grid'5000. Nous avons validé qualitativement le comportement suivant :

- les utilisateurs de *vo-centrale1* et *vo-centrale2* ont leurs machines virtuelles déployées sur les ressources des *centrales 1* et *2*,
- les utilisateurs de *vo-gold* ont leurs machines virtuelles déployées sur les meilleures ressources de la *centrale 1* ou de la *centrale 2*, en fonction des ressources disponibles au moment de la soumission des tâches,
- les utilisateurs de *vo-besteffort* ont leurs machines virtuelles déployées sur les ressources des *centrales 1* et *2* en fonction des ressources disponibles au moment de la soumission des tâches.

En conclusion, l'utilisation des VO permet de gérer les politiques d'allocation des ressources. D'autres expérimentations avec des politiques plus évoluées seront à évaluer. Par exemple, on peut imaginer une politique dans laquelle les ressources peuvent changer dynamiquement de VO selon les politiques des administrateurs. En reprenant le cas présenté, une gestion dynamique des ressources dans les VO permettrait d'assigner des nœuds appartenant à *vo-centrale1* ou *2* à *vo-gold* s'il en venait à manquer (due à une charge de travail trop importante par exemple).

3.3.6 Conclusion

Dans cette première partie du chapitre nous avons présenté Grillade-CN : une extension du système pour grille XtreemOS permettant de gérer nativement les collections de machines virtuelles, à la manière des centrales numériques.

L'architecture générale d'XtreemOS est fondée sur XtreemOS-F et XtreemOS-G. XtreemOS-F contient l'ensemble des systèmes d'exploitation Linux configurés et adaptés aux infrastructures pour PC et grappe. XtreemOS-G s'appuie sur XtreemOS-F pour fournir l'infrastructure de service de la grille. Grillade-CN bénéficie de services et mécanismes de XtreemOS-F et XtreemOS-G pour la gestion des collections de machines virtuelles.

XtreemOS-F permet d'exploiter des grappes avec un système d'exploitation à image unique nommé LinuxSSI. LinuxSSI offre des fonctionnalités permettant d'agréger des nœuds. Cette agrégation de nœuds permet de « construire » des ressources dont les capacités physiques sont celles de l'agrégation des nœuds. Dans le cadre des machines virtuelles, cette fonctionnalité permet aux utilisateurs de faire des requêtes de machines virtuelles en s'affranchissant des capacités des nœuds : si aucun nœud disponible de la grille n'est capable d'exécuter la machine virtuelle demandée, Grillade-CN agrègera des nœuds disponibles de manière à pouvoir répondre à la requête.

XtreemOS-G fournit des services de haut niveau utiles à la gestion des collections

de machines virtuelles. En s'appuyant sur les mécanismes de VO, Grillade permet aux administrateurs de différents domaines d'administration de définir des classes d'utilisateurs et de nœuds. Ces ressources peuvent être utilisées par les utilisateurs pour déployer des machines virtuelles munies de leur propre système d'exploitation. Ces machines virtuelles bénéficient alors du système de fichiers pour grille XtreemFS. XtreemFS permet de gérer les machines virtuelles à l'échelle de la grille. De plus, le service de gestion des ressources permet de découvrir et allouer des nœuds en fonction des requêtes demandées par les utilisateurs pour y déployer des machines virtuelles.

Enfin, Grillade-CN étant fondé sur XtreemOS, il propose les services de gestion d'XtreemOS pour la gestion des tâches standard (processus). Ainsi, Grillade-CN propose une double granularité de gestions des tâches : la granularité des processus (avec un environnement d'exécution préconfiguré) et la granularité des machines virtuelles (avec un environnement d'exécution personnalisable).

Le prototype mis en œuvre a été testé et validé sur Grid'5000. Ce prototype apporte une grande souplesse et flexibilité dans la gestion des ressources en apportant un degré supplémentaire de dissociation entre la vue de l'utilisateur des machines virtuelles qu'il manipule et l'infrastructure matérielle sous-jacente (par exemple un nœud d'exécution ou une agrégation de nœuds).

3.4 Grillade-Ext : extension d'une grille XtreemOS avec des ressources provenant de centrales numériques

Dans la partie précédente, nous présentons nos travaux traitant de la conception et de la mise en œuvre de mécanismes dans Grillade permettant de gérer des collections de machines virtuelles à la manière des centrales numériques, sur des sites appartenant à des domaines d'administration différents. Les mécanismes résultants offrent plus de flexibilité dans la gestion des ressources et dans la gestion des environnements d'exécution des applications.

Cette seconde partie s'oriente vers l'extension d'une infrastructure distribuée existante avec des ressources fournies par des centrales numériques. Ce cas d'utilisation peut être utilisé par exemple, lors d'une extension temporaire des ressources : par exemple, pour les fêtes de fin d'année « l'interface web de commande liée à la base de données de gestion des stocks » peut avoir besoin de plus de ressources que le reste de l'année. Ces ressources peuvent être apportées temporairement par un prestataire externe, et retirées à la fin des fêtes de fin d'année, lorsque le taux de commande retourne à la normale.

Cette seconde partie du chapitre traite des mécanismes de Grillade permettant à un système de grille XtreemOS de s'étendre automatiquement sur des ressources fournies par une centrale numérique, selon les besoins, et de manière transparente pour les utilisateurs [164] (cf objectifs de Grillade présenté dans le tableau 3.2). Ce cas d'étude correspond à la figure 3.3.

3.4.1 Modification du système XtreemOS pour l'approvisionnement dynamique des ressources

Afin d'étendre automatiquement une grille XtreemOS avec des ressources d'une centrale numérique, il est nécessaire d'étendre les services de découverte et de réservation des ressources.

En effet, le mécanisme de découverte des ressources doit être étendu pour être capable de découvrir des ressources fournies par une centrale numérique. Le mécanisme de

réservation doit quant à lui pouvoir réserver les machines virtuelles, puis interagir avec la centrale numérique pour les configurer et intégrer à la grille les ressources qu'elle fournit.

3.4.1.1 Découverte des ressources

Nous faisons l'hypothèse que les centrales numériques proposent un catalogue permettant à l'utilisateur de savoir quels types de machines virtuelles peuvent être créés. Par exemple, Amazon EC2 propose un catalogue d'instances pouvant être exécutées sur sa centrale.

Dans Grillade, la particularité de découvrir des ressources appartenant à une centrale numérique réside dans le fait, qu'au moment de la recherche, les machines virtuelles n'existent pas encore. Le service de découverte de ressources est modifié en conséquence pour prendre en compte la recherche de ressources dans un catalogue de centrales. Ce catalogue de centrales contient l'ensemble des centrales qu'il est possible de contacter, ainsi que pour chaque centrale, l'ensemble des caractéristiques des machines virtuelles qu'il est possible de créer. Ce catalogue peut être rempli manuellement par les administrateurs, ou automatiquement en interagissant avec les centrales pour obtenir les caractéristiques des instances de machines virtuelles qu'elles peuvent créer.

La figure 3.18 présente les modifications apportées au système de découverte des ressources d'XtreemOS pour gérer des machines virtuelles fournies par des centrales numériques.

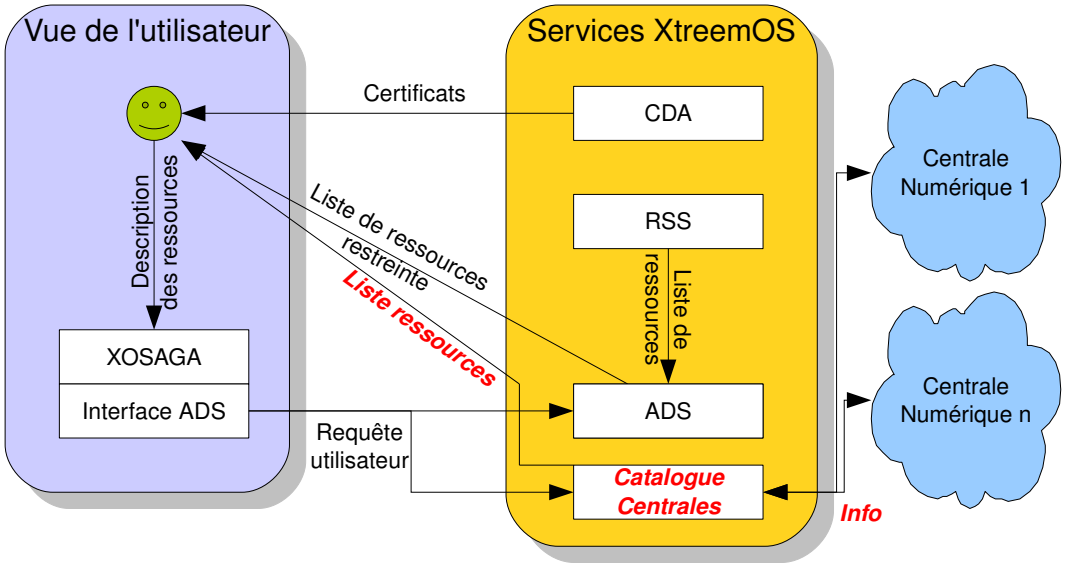


FIGURE 3.18 – Extension du service de découverte des ressources
Cette figure est une modification de la figure 3.6.

Lors de l'exécution d'une requête de découverte de ressources, celle-ci est envoyée parallèlement à l'ADS, pour une recherche standard dans les ressources de la grille et au catalogue de centrales, pour repérer si des ressources appartenant à des centrales numériques pourraient convenir. Finalement, l'utilisateur qui dans notre cas est Grillade lui-même, reçoit deux listes de description de ressources : celle qui contient les ressources de la grille, et celle qui contient une liste de centrales pouvant créer des machines virtuelles dont les caractéristiques correspondent à la recherche.

3.4.1.2 Création, ajout et réservation des machines virtuelles

Lorsque des machines virtuelles ont des caractéristiques pouvant convenir à la requête, Grillade-Ext envoit à la centrale concernée une requête de création. Une fois les ressources créées, elles sont ajoutées à la grille et peuvent être ensuite utilisées comme n'importe qu'elle autre ressource de la grille. Pour ce faire, le service de réservation des ressources d'XtreemOS est modifié pour contrôler la configuration et la gestion des ressources externes. Ces modifications représentées sur la figure 3.19 introduisent le gestionnaire des centrales qui est l'interface de communication entre le gestionnaire de réservation et la centrale numérique.

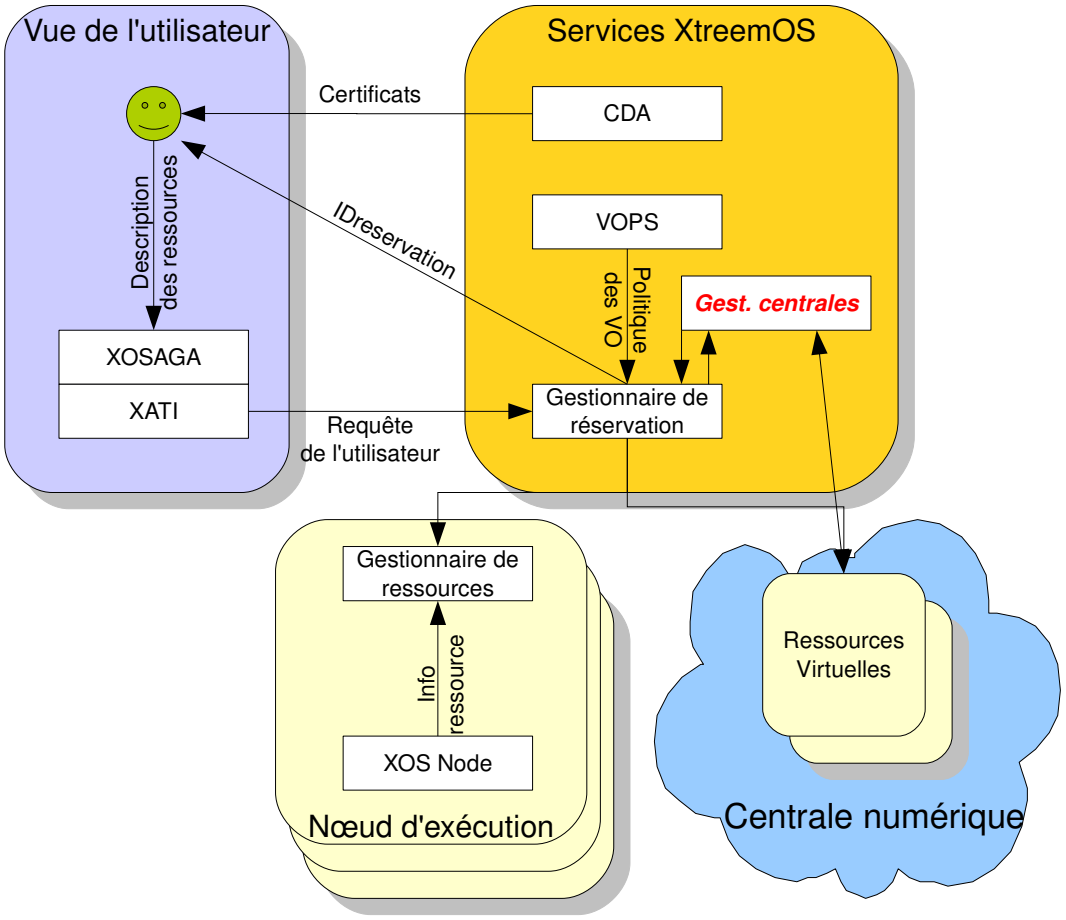


FIGURE 3.19 – Réservation des ressources de la grille et des machines virtuelles d'une centrale numérique
Cette figure est une modification de la figure 3.7.

Création de ressources XtreemOS virtualisées Le gestionnaire de centrales numériques est l'interface entre le système de grille et celui de la centrale. Lorsque le gestionnaire de réservation envoie une requête de réservation de ressources au gestionnaire de centrales, celui-ci doit tout d'abord demander la création des ressources à la centrale. Cette requête peut être faite en utilisant une interface standard de communication, par exemple, Amazon EC2 WSDL.

Lors de la requête de création, des paramètres d'initialisation de la machine virtuelle

sont transmis à la centrale. Ces paramètres configurent la machine virtuelle de telle manière que, lors de son premier démarrage, elle puisse contacter le nœud de service de Grillade-Ext qui a exécuté la requête de création de machines virtuelles, afin de se configurer pour intégrer la grille.

À la fin de cette opération, la centrale numérique retourne au gestionnaire des centrales un identifiant unique de réservation ainsi qu'une liste d'adresses IP des machines virtuelles créées. Ces identifiants sont sauvegardés pour permettre la vérification ultérieure de l'authenticité d'une ressource demandant à être ajoutée à la grille.

Ajout de machines virtuelles à une grille XtreemOS Une fois les machines virtuelles créées, elles doivent être configurées et ajoutées à la liste des ressources de la grille.

Dans ce document, nous considérons l'utilisation de centrales numériques privées ou publiques. Dans le cas de centrales privées, les machines virtuelles de la centrale sont accessibles directement depuis les services de Grillade, par exemple, par un réseau local. Dans le cas de centrales publiques, nous considérons que la centrale fournit des ressources avec des adresses IP publiquement accessibles depuis Internet. Par exemple, la centrale numérique Amazon EC2 offre cette fonctionnalité.

Le service RCA, présenté dans la section 3.2.2.4, permet de distribuer des certificats à des ressources afin de les intégrer à la grille (chaque domaine d'administration dispose d'un RCA). Ainsi, lors d'une procédure standard d'ajout d'une ressource à XtreemOS, cette ressource doit s'enregistrer auprès du RCA du site auquel il appartient. Le RCA confirme alors la requête d'enregistrement. Suite à cette opération, la ressource peut faire une requête d'obtention de son certificat. À la fin de l'opération, la nouvelle ressource est intégrée à la grille.

Cette procédure d'authentification est conservée dans Grillade pour enregistrer les machines virtuelles fournies par une centrale numérique. Cependant, un mécanisme supplémentaire est introduit afin d'assurer l'authenticité des machines virtuelles faisant une requête d'obtention d'un certificat. En effet, dans une centrale numérique, la liste des adresses IP est partagée entre tous les utilisateurs de la centrale numérique, c'est-à-dire qu'un utilisateur A peut recevoir une adresse IP précédemment allouée à un utilisateur B. Considérant ce fait, il est nécessaire de s'assurer que les machines virtuelles demandant à être ajoutées à la grille sont bien celles attendues. Pour résoudre ce problème, à chaque interaction entre une machine virtuelle et le nœud de service de Grillade, ce dernier interroge la centrale pour vérifier que l'adresse IP de la machine virtuelle fait bien partie de la liste des adresses IP liées à l'identifiant de réservation précédemment obtenue lors de la demande de création des machines virtuelles. Si l'adresse IP est bien liée à l'identifiant de réservation, c'est qu'il s'agit de la machine virtuelle attendue, sinon, cela signifie que pour une raison malicieuse ou technique, il ne s'agit pas d'une ressource légitime, aucun certificat ne lui sera délivré.

La figure 3.20 présente le mécanisme d'ajout de machines virtuelles fournies par une centrale numérique dans Grillade-Ext.

Réservation des ressources La réservation des ressources, une fois qu'elles sont intégrées à la grille se fait comme pour les autres ressources de la grille, par l'action du gestionnaire de réservation présenté dans la section 3.2.6.2.

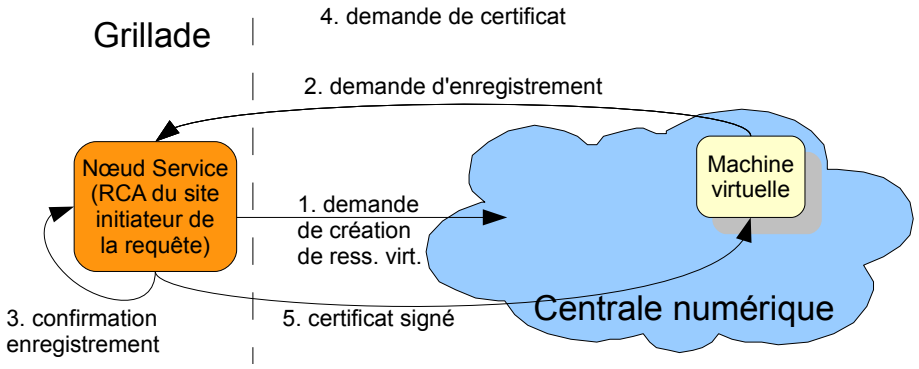


FIGURE 3.20 – Fonctionnement des mécanismes d'extension de Grillade.

3.4.1.3 Utilisation des machines virtuelles

Les machines virtuelles sont gérées dans les VO comme les autres ressources de la grille : les politiques des VO sont appliquées aux nœuds comme aux machines virtuelles. Les applications peuvent être déployées sur ces machines virtuelles et tirent profit des services de grille fournis par XtreemOS. Par exemple, XtreemFS, le système de fichiers pour grille d'XtreemOS, permet aux applications de lire leurs données d'entrée et d'écrire leurs données de sortie sur un volume accessible depuis toutes les ressources. De plus, ces données restent disponibles, même si les ressources ayant produit ces données disparaissent : un calcul effectué par des ressources d'une centrale externe sauvegardé dans XtreemFS reste disponible pour les autres ressources de la VO même après l'arrêt des machines virtuelles fournies par la centrale numérique.

La figure 3.21 présente l'extension d'une grille XtreemOS avec les ressources d'une centrale numérique.

3.4.1.4 Surveillance et retrait des machines virtuelles de la grille

À la fin de l'utilisation des machines virtuelles, celles-ci doivent être retirées de la grille. Pour ce faire, le RCA révoque les certificats des ressources concernées et le gestionnaire de centrales envoie une requête de terminaison des machines virtuelles à la centrale. Ensuite, l'identifiant unique de réservation de la centrale ainsi que la liste des adresses IP des ressources liées à cet identifiant sont supprimés.

L'état des machines virtuelles est vérifié périodiquement afin de dés-enregistrer une machine virtuelle qui s'est terminée de manière anormale prématurément. Dans ce cas, une nouvelle demande de machines virtuelles pourra être faite par le gestionnaire de centrales.

3.4.2 Mise en œuvre de Grillade-Ext

Cette section présente les éléments de mise en œuvre des mécanismes d'extension de la grille avec des ressources fournies par une centrale numérique. La centrale numérique qui correspond à nos besoins et que nous avons choisi d'exploiter est Nimbus. Nous considérons que l'image des machines virtuelles XtreemOS est déjà disponible et prête à l'emploi dans le catalogue d'instance de Nimbus (concrètement, nous avons créé une image XtreemOS en réalisant les opérations d'installation standard à XtreemOS [67], puis nous avons copié cette image sur la centrale Nimbus – d'autres méthodes concernant la création des images de machines virtuelles sont présentées dans la section 2.3.3.2).

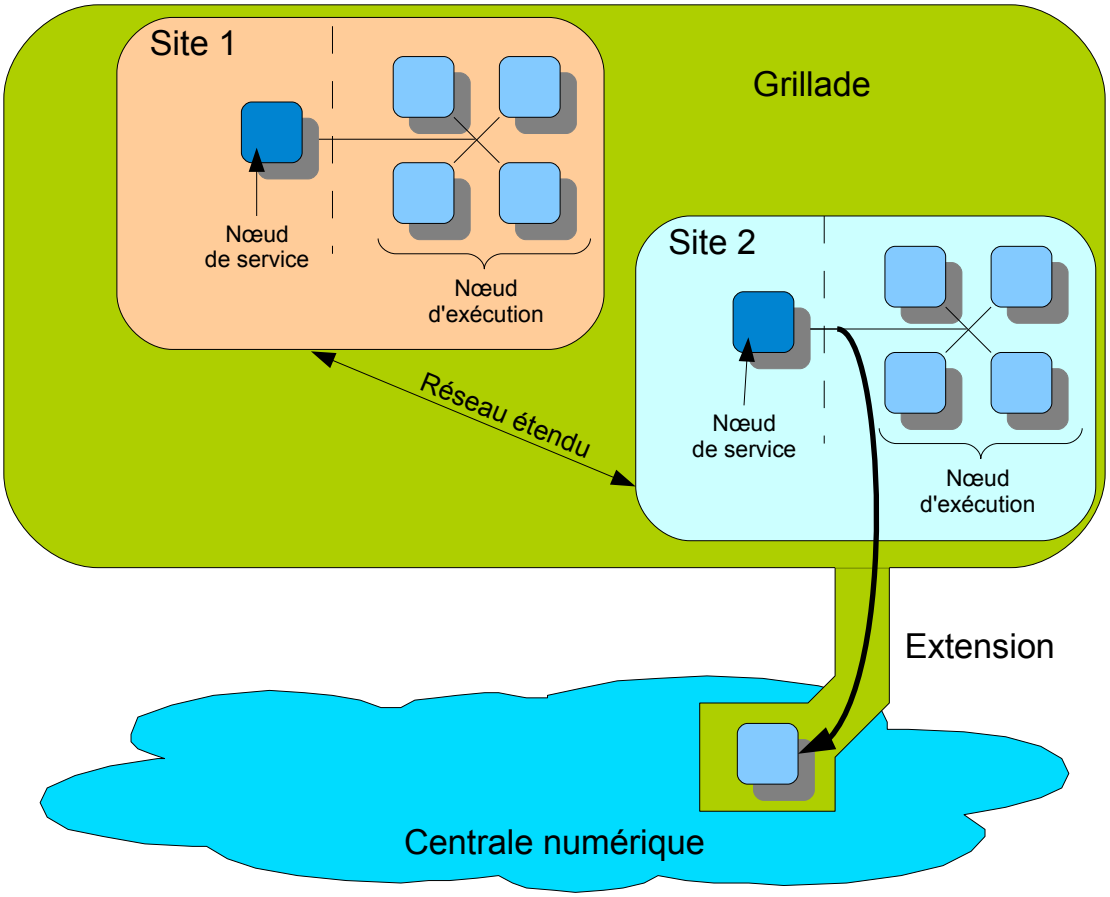


FIGURE 3.21 – Extension d'une grille XtreemOS avec des ressources d'une centrale numérique

3.4.2.1 Création de nouvelles ressources XtreemOS dans Nimbus

Nimbus propose deux types d'interface utilisateur : l'interface standard EC2 et une interface privée conçue spécifiquement pour Nimbus (*client cloud*). Dans sa mise en œuvre, Grillade-Ext peut utiliser l'une ou l'autre de ces interfaces.

Le gestionnaire de centrales numériques fait une requête de création de machines virtuelles à la centrale Nimbus en spécifiant une durée (en heure) ainsi que le nom de l'image à déployer. Cette image est celle d'un nœud d'exécution d'XtreemOS. À la fin de la création des machines virtuelles, les adresses IP des machines virtuelles déployées sont retournées au gestionnaire de centrales numériques.

3.4.2.2 Ajout de nouvelles ressources à Grillade

Lors du premier démarrage d'une nouvelle ressource, celle-ci exécute l'outil de configuration automatique d'XtreemOS. Cet outil développé par le consortium XtreemOS est appelé `xosautoconfig`, développé en langage de programmation *bash*. Cet outil de configuration nécessite différents arguments spécifiés lors de son exécution : des variables locales (IP et nom d'hôte de la ressource), globales (IP et nom d'hôte du nœud de service) et spécifiques à un site (par exemple l'adresse d'un serveur d'une passerelle pour que les nœuds aient accès à Internet) qui sont définies à l'aide des fichiers de configuration (`localDefs`, `globalDefs` et `siteDefs`).

`xosautoconfig` configure la ressource et envoie une requête d'enregistrement à la grille. Cette requête d'enregistrement doit être approuvée par un administrateur de la grille depuis un nœud de service. Lorsque la demande d'enregistrement est confirmée, les certificats signés sont récupérés et installés sur la ressource. Après l'obtention et l'installation réussie des certificats, la ressource est intégrée à la grille et est prête à exécuter des tâches.

3.4.2.3 Utilisation des machines virtuelles fournies par une centrale numérique

Une fois qu'une machine virtuelle est enregistrée, a obtenu et installé ses certificats, elle fait partie intégrante de Grillade. Les tâches soumises peuvent s'exécuter sur cette nouvelle ressource. Les utilisateurs qui soumettent des tâches peuvent utiliser XtreemFS pour les entrées / sorties de leurs applications.

3.4.3 Validation

Nous avons mis en place un protocole expérimental afin de simuler l'utilisation d'une centrale numérique publique qui offre ses services à travers une passerelle accessible via Internet. Pour ce faire nous utilisons deux sites de Grid'5000 (Rennes et Sophia). Sur le site de Sophia nous déployons une grille XtreemOS composée d'un nœud de service et de quatre nœuds d'exécution. Sur le site de Rennes, nous déployons une centrale numérique Nimbus. Cette centrale Nimbus est constituée d'un portail d'accès. De plus, les adresses IP attribuées aux machines virtuelles de la centrale Nimbus sont accessibles depuis le site de Sophia (cette configuration est réaliste, cela correspond par exemple à Amazon EC2 qui fournit des adresses IP publiques à ses machines virtuelles). La figure 3.22 présente la configuration mise en œuvre.

Nous avons validé qualitativement notre prototype. Grillade est capable de demander l'exécution de machines virtuelles Nimbus, de les configurer automatiquement et d'y exécuter des applications.

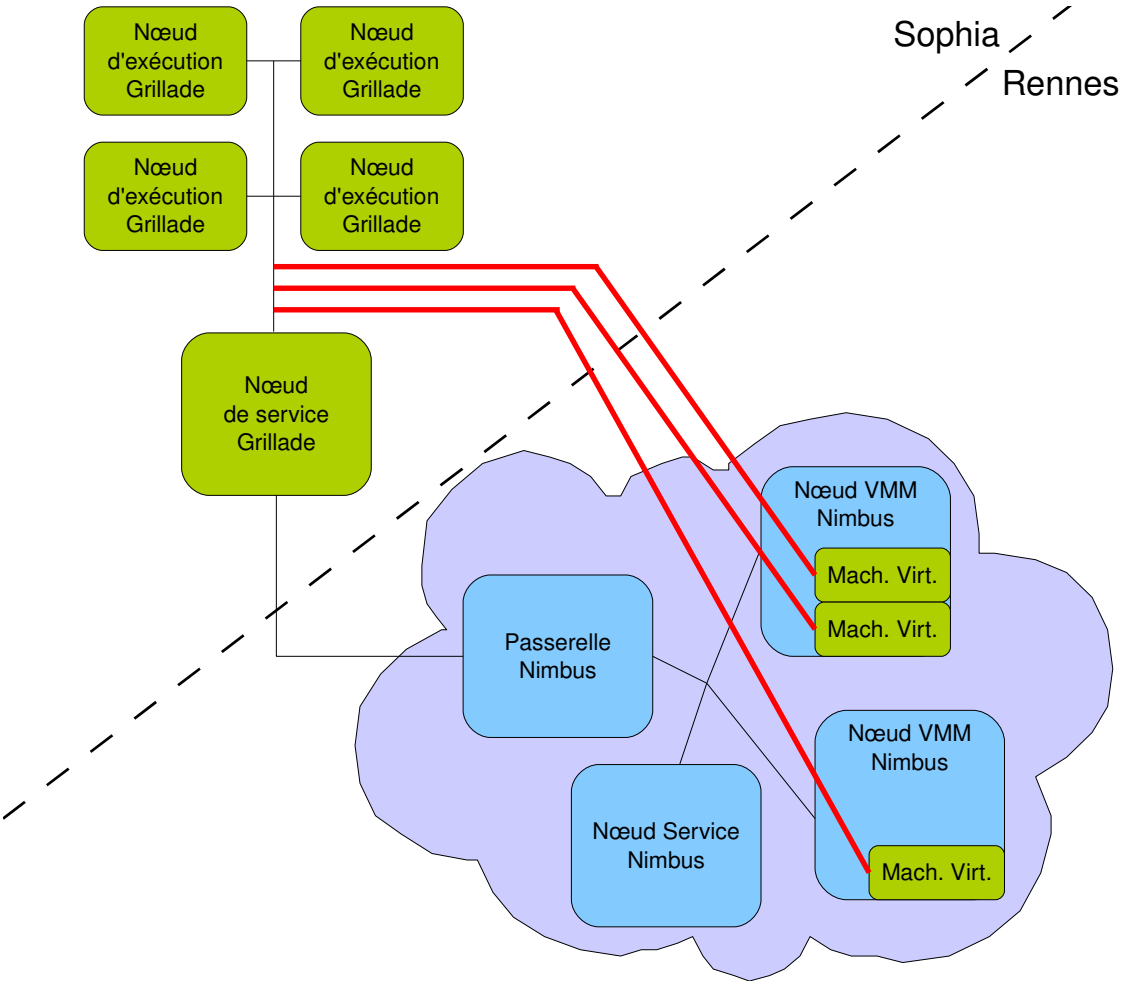


FIGURE 3.22 – Configuration expérimentale : Grillade à Sophia et nuage à Rennes

3.4.3.1 Temps de déploiement et de configuration de ressources fournies par Nimbus

Nous avons évalué notre prototype en mesurant le temps nécessaire au déploiement et à la configuration de ressources fournies par une centrale numérique dans Grillade. Pour ce faire, comme le décrit la figure 3.22 nous déployons Grillade sur le site de Sophia et nous mesurons le temps nécessaire pour l'obtention de machines virtuelles fournies par la centrale Nimbus exécutée sur le site de Rennes. Le graphique 3.23 présente le temps total avant utilisation des machines virtuelles dans Grillade. Ce temps total correspond au temps de déploiement des machines virtuelles dans Nimbus et le temps de configuration des ressources pour les intégrer à la grille XtreemOS.

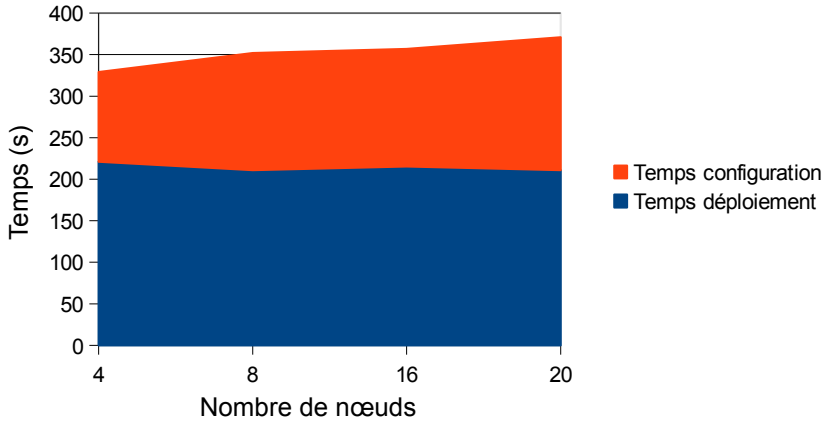


FIGURE 3.23 – Temps nécessaire au déploiement et à la configuration de ressources de Grillade dans la centrale Nimbus
Le temps total mesuré correspond au temps de déploiement des ressources dans Nimbus et au temps nécessaire à la configuration des machines virtuelles (intégration à Grillade).

On remarque que le temps de déploiement des machines virtuelles évolue peu quel que soit le nombre de machines virtuelles déployées. Cela est dû à l'architecture de Nimbus, qui dispose de mécanismes de copie efficace des machines virtuelles similaires à Saline.

De plus, on constate que le temps nécessaire à la configuration évolue de manière faible avec le nombre de machines virtuelles (de l'ordre de 3,5%). Cela est dû au fait que les opérations de configuration des différentes machines virtuelles se font en parallèle. Le goulot d'étranglement qui dans le cas présenté est négligeable se situe sur le nœud de service de Grillade-Ext qui est unique : toutes les machines virtuelles sont gérées par le nœud de service connecté à la centrale numérique et reçoivent de celui-ci leurs certificats signés d'appartenance à la grille.

Pour intégrer un grand nombre de machines virtuelles, l'utilisation de plusieurs nœuds de service pourrait être envisagée.

3.4.4 Conclusion

Dans cette seconde partie du chapitre, nous avons présenté Grillade-Ext à travers les concepts et la mise en œuvre de mécanismes permettant l'extension du système d'exploitation pour grille XtreemOS avec des machines virtuelles de centrales numériques. L'axe de flexibilité proposé par Grillade-Ext permet d'étendre une infrastructure informatique avec des ressources fournies par une centrale numérique.

Grillade-Ext utilise les services de découverte et réservation des ressources d'XtreemOS que nous avons modifiés pour prendre en compte les catalogues de centrales numériques. Une fois les ressources trouvées et créées, leur intégration dans XtreemOS utilise les mécanismes standard d'ajout de nœuds. Une fois les machines virtuelles intégrées dans la grille pour le compte d'une VO, elles sont gérées comme les autres ressources de la grille.

Les validations de notre prototype ont été faites avec des ressources d'une centrale Nimbus entre deux sites de Grid'5000. Après ajout de ces ressources à la grille, les tâches exécutées par XtreemOS peuvent en tirer avantage de manière transparente.

En perspectives, il serait intéressant d'étudier les politiques de décision de création et de terminaison des machines virtuelles pour rendre le système entièrement automatique. Il serait aussi intéressant d'étendre nos évaluations en mesurant l'impact de l'exécution d'une application exécutée sur la grille, sur la centrale numérique, et à cheval entre les ressources de la grille et de la centrale numérique.

3.5 Travaux apparentés

Cette section présente les travaux apparentés à Grillade que l'on a classés en quatre catégories :

- les travaux liés à l'interopérabilité entre les grilles et les centrales numériques,
- les travaux liés aux gestionnaires de multi-domaines d'administration et de partage des ressources,
- les travaux liés à la performance de création des machines virtuelles,
- les travaux liés au centrales numériques et aux fédérations de centrales numériques.

Interopérabilité entre grilles et centrales numériques L'interopérabilité entre les grilles et les centrales numériques est étudiée depuis peu par différents groupes de recherche. Une solution d'interopérabilité sur le plan applicatif entre les grilles et les centrales de ressources fondée sur une extension de SAGA [110] (standard appuyé par l'OGF) montre qu'il est possible d'isoler une application de l'infrastructure matérielle sous-jacente : une application peut tirer avantage de manière transparente, des ressources de la grille et des centrales numériques [126]. Cependant, cette interopérabilité au niveau applicatif a un coût : les applications doivent être conçues de manière à être compatibles avec l'interface SAGA.

D'autres travaux plutôt orientés sur la gestion des infrastructures de calcul, proposent d'étendre des systèmes de grilles avec des mécanismes de création et de gestion de grappes virtuelles. Ces grappes virtuelles peuvent être utilisées pour fournir un plus haut degré d'abstraction dans l'utilisation des grilles, à la manière des centrales numériques [50, 74, 152].

Domaines d'administration, partage de ressources Globus Toolkit™ [41, 76] est un intergiciel de grille qui permet grâce à différents outils (*Globus Toolkit*) d'accéder, d'utiliser et de gérer les ressources de la grille. Cependant, bien que des travaux ont été menés pour déployer des machines virtuelles sur Globus [153], ils ne permettent pas de gérer le cyle de vie complet des machines virtuelles sur la grille (migration, sauvegarde et restauration). Grillade-CN qui est fondé sur XtreemOS et sur les mécanismes de gestion des collections de machines virtuelles à l'échelle de la grille, offre un axe de flexibilité supplémentaire par rapport à Globus, en alliant à la fois, l'utilisation en « mode » grille qui permet l'exécution de tâches constituées de processus, et l'utilisation en « mode » centrale numérique, qui permet le déploiement

de collections de machines virtuelles disposant d'un environnement personnalisé.

Performances SnowFlock [115] est un système de création de machines virtuelles dont l'objectif est de pouvoir créer des machines virtuelles aussi facilement et rapidement qu'il est possible de créer des processus dans un système d'exploitation. Pour ce faire, Snowflock dispose de mécanismes de clonage des machines virtuelles. Cependant, Snowflock traite uniquement de la problématique de la création rapide des machines virtuelles. D'autres solutions doivent être utilisées pour mettre en place des collaborations et gérer des collections de machines virtuelles entre différents sites. Ce type de mécanisme pourrait être intégré à Grillade-CN pour la création des machines virtuelles.

Centrales numériques Nimbus [111] ou Eucalyptus [135] sont des logiciels libres qui permettent aux utilisateurs de mettre en œuvre une centrale numérique privée sur leurs ressources. Cependant, ce type de système ne permet pas de gérer la sauvegarde des machines virtuelles ainsi que leur migration à l'échelle de la grille sur des ressources appartenant à différents domaines d'administration. De plus, la granularité des tâches traitées est uniquement celle des machines virtuelles. Grillade permet de faire cohabiter des tâches sur des nœuds d'exécution et des machines virtuelles. Il offre également pour les machines virtuelles des fonctionnalités de partage de ressources entre différents domaines d'administration, de partage des données entre les nœuds de la grille.

OpenNebula [120] est une boite-à-outils permettant de « construire » aisément différents types de centrale numérique : privée, publique et mixte. OpenNebula est capable de gérer le stockage, la configuration du réseau, ainsi que les technologies de virtualisation afin de pouvoir dynamiquement placer les collections de machines virtuelles sur les infrastructures distribuées. OpenNebula configuré en centrale numérique privée est capable de tirer profits des ressources d'une centrale numérique publique. Cependant, ce système ne peut gérer des ressources matérielles appartenant à des domaines d'administration différents. Ce dernier point est à nuancer car au cours de l'année 2010 le projet BonFIRE, qui utilise entre autre le système OpenNebula, a pour objectif de pouvoir étendre des centrales numériques avec des ressources fournies par d'autres en envisageant des scénarios complexes de connexion réseau [4, 39].

Enfin, StratusLab [29] est un projet fondé par la commission Européenne dont le but est de créer un logiciel libre pour créer et gérer des centrales numériques privées. Ce logiciel libre intègrera le plus possible de solutions existantes (comme OpenNebula). Un axe de recherche du projet est d'offrir aux utilisateurs les outils pour la création et l'exécution de machines virtuelles disposant d'un environnement entièrement personnalisé tout en s'assurant que leurs exécutions ne compromettra pas les politiques définies par les administrateurs.

Fédération de centrales numériques Enfin, les travaux portants sur les fédérations de centrales numériques (*sky computing*), dont le but est de pouvoir bénéficier du potentiel de plusieurs centrales numériques, émergent. Ainsi, dans la logique des fédérations de centrales numériques, il devrait être possible à un utilisateur de bénéficier de ressources offertes par plusieurs fournisseurs de centrale numérique.

En ce sens, Grillade est un système de fédération de centrales numériques. En effet, Grillade est multi-domaines d'administration, à la manière du *sky-computing* [113], en offrant une grande flexibilité quant à la gestion de l'infrastructure physique grâce à des mécanismes de collaboration, de fédération de données et d'extension. Grillade

apporte une vision plus étendue de ce que pourrait être le *sky-computing* en y proposant une double granularité (tâches / ressources) ainsi que les autres mécanismes présentés qui combinent ceux des grilles et des centrales dans un seul système.

Enfin, à notre connaissance, une fonctionnalité unique à Grillade est sa capacité de déployer à la demande un système à image unique pour agréger des nœuds et déployer sur le nœud multi-cœur virtuel obtenu, des machines virtuelles. Cet axe de flexibilité dans la configuration de l'infrastructure matérielle permet aux utilisateurs de mieux dissocier le besoin de l'utilisateur (par exemple une machine virtuelle de 40 Go) de l'infrastructure matérielle (pour ce même exemple, 4 nœuds de 10 Go). Le tableau 3.3 présente les mécanismes de Globus, XtreemOS, Nimbus et OpenNebula ainsi que ceux de Grillade.

	Domaines d'administration, partage des ressources	Flexibilité des environnements d'exécution	Flexibilité de gestion de l'infrastructure	Granularité	Extension
Grille XtreemOS	**multi-domaines d'administration, partage des ressources, VO**	environnement préconfiguré	**agrégation de nœuds : système à image unique LinuxSSI**	tâches (processus)	non
Grille Globus	**multi-domaines d'administration, partage des ressources, VO**	environnement préconfiguré	non	tâches (processus)	non
Centrale Nimbus	partiel	**personnalisable**	machine virtuelle	machines virtuelles (OS)	des travaux sont en cours
Centrale OpenNebula	partiel	**personnalisable**	machine virtuelle	machine virtuelle (OS)	**oui**
Grillade	**multi-domaines d'administration, partage des ressources, VO**	**environnement préconfiguré et personnalisable**	**LinuxSSI + machine virtuelle**	**tâches (processus) + machines virtuelles (OS)**	**oui**

TABLE 3.3 – Mécanismes conçus et mis en œuvre dans Grillade présentés dans cette partie. Ce tableau est un raffinement du tableau 3.1. Les mécanismes concernés sont représentés en **gras**.

3.6 Conclusion

Dans ce chapitre, nous avons présenté Grillade, une extension du système d'exploitation pour grille XtreemOS permettant de gérer les collections de machines virtuelles et de s'étendre sur des ressources externes fournies par une centrale numérique. Le tableau 3.3 présente un résumé des fonctionnalités de Grillade comparé à d'autres systèmes de grille et centrale numérique.

En s'appuyant sur des mécanismes de grille, Grillade offre un nouveau niveau de flexibilité dans la gestion et l'utilisation des ressources. Les axes de flexibilité couverts par Grillade pour les administrateurs et les utilisateurs sont :

Multi-domaines d'administration et partage des ressources Grillade peut gérer une infrastructure informatique distribuée sur plusieurs domaines d'administration. Les administrateurs peuvent définir des classes de clients grâce aux VO.

Granularité Grillade est fondé sur le système de grille XtreemOS, il bénéficie donc des mécanismes standard d'XtreemOS pour la gestion des tâches (processus). De plus, avec les mécanismes de gestion des collections de machines virtuelles que nous avons conçus et mis en œuvre, Grillade peut gérer des collections de machines virtuelles. La granularité de gestion de grillade est double : les processus et les machines virtuelles. Les utilisateurs ont alors le choix d'utiliser l'environnement préconfiguré de la grille, ou de personnaliser leur environnement d'exécution en utilisant des machines virtuelles.

Flexibilité de l'environnement d'exécution Grillade gère des processus dans des environnements d'exécution préconfigurés par XtreemOS (cas standard d'utilisation d'XtreemOS). De plus, Grillade gère des environnements d'exécution personnalisés, grâce aux machines virtuelles qu'il gère sur la grille.

Flexibilité de gestion de l'infrastructure matérielle Grillade gère des collections de machines virtuelles sur la grille. Ces machines virtuelles apportent de la flexibilité par rapport à l'infrastructure matérielle (migration, sauvegarde et restauration des machines virtuelles). De plus, Grillade dispose de mécanismes d'agrégation des nœuds (LinuxSSI), lui permettant de « créer » des ressources disposant de capacité matérielle fondée sur l'agrégation de plusieurs nœuds.

Capacité d'extension Grillade propose des mécanismes d'extension avec des machines virtuelles de centrales numériques. Une fois intégrée à la grille, ces machines virtuelles sont utilisées comme des nœuds standard, et les applications des utilisateurs peuvent s'y exécuter de manière transparente.

Grillade propose aux administrateurs une palette de mécanismes permettant de configurer et reconfigurer leurs ressources en fonction des demandes des utilisateurs :

- mécanismes d'agrégation de nœuds pour y déployer des machines virtuelles dont les capacités sont celles de l'agrégation des nœuds,
- mécanismes de gestion des collections de machines virtuelles, à la manière d'une centrale numérique, sur une infrastructure de type grille,
- mécanismes d'extension de l'infrastructure existante avec des machines virtuelles fournies par une centrale numérique.

Nous avons conçu, mis en œuvre et évalué Grillade sur Grid'5000. Concernant Grillade-CN, nous avons déployé des machines virtuelles gérées par Grillade. Nous avons également validé les mécanismes d'agrégation des nœuds avec LinuxSSI. Concernant Grillade-Ext, nous l'avons évalué avec une centrale numérique Nimbus. Pour ce faire, nous avons déployé

Grillade et une centrale numérique Nimbus sur Grid'5000. Nous avons validé que Grillade-Ext était capable de configurer et d'ajouter des machines virtuelles fournies par une centrale numérique de manière transparente.

Les perspectives ouvertes par Grillade sont principalement liées à la gestion des politiques de création et de gestion des collections de machines virtuelles (dans le cas de Grillade-CN) et d'extension automatisée (dans le cas de Grillade-Ext). De plus, il serait intéressant d'étudier les politiques de gestion de partage des nœuds entre les tâches de type processus et les tâches de type machine virtuelle.

En conclusion, Grillade apporte une grande flexibilité par rapport à la gestion et l'utilisation de l'infrastructure matérielle. De plus, Grillade permet de dissocier l'infrastructure matérielle de la vision qu'a l'utilisateur des ressources qu'il utilise. Cette notion d'affranchissement des ressources sous-jacentes est, selon nous, une notion clé de l'informatique distribuée qui est de plus en plus performante, mais également de plus en plus complexe à maîtriser. C'est dans cette vision de l'informatique distribuée que nous avons poursuivi nos travaux : offrir à l'utilisateur la vision des ressources dont il a besoin et non celle imposée par l'infrastructure.

Chapitre 4

Tropicbird : vers un méta-système de gestion des ressources pour plus de flexibilité dans les infrastructures informatiques

Dans les chapitres précédents, nous avons étudié quatre axes de flexibilité pour les infrastructures informatiques réparties sur plusieurs sites. Le premier axe de flexibilité est celui de la gestion de collections de machines virtuelles sur une infrastructure distribuée de type grille pour les tâches interruptibles. Cette première contribution, qui a donné lieu à la conception et à la mise en œuvre de Saline, permet d'encapsuler des tâches interruptibles dans des machines virtuelles et les exécuter sur une grille en s'affranchissant de la localisation et de l'hétérogénéité des ressources matérielles.

Dans la continuité de ces travaux nous avons étudié et rapproché les systèmes de gestion de grappe, grille et centrale numérique sous l'angle de la flexibilité qu'ils offrent et nous avons proposé le système Grillade qui combine avantageusement les mécanismes de flexibilité de ces trois types de système. Trois axes de flexibilité ont retenu notre attention. Le premier axe que nous avons conçu et mis en œuvre dans le sous-système Grillade-Ext permet d'étendre une infrastructure informatique avec des ressources fournies par une centrale numérique, de manière transparente pour les applications. La flexibilité offerte par Grillade-Ext est liée à la taille de l'infrastructure matérielle qui peut être agrandit. Le deuxième axe est lié à la technologie des systèmes à image unique qui permettent d'offrir l'illusion d'un nœud multicœur en agrégeant plusieurs nœuds d'une grappe. Le troisième axe, mis en œuvre dans Grillade-CN, permet de construire un service de type IaaS qui exploite les ressources de plusieurs domaines d'administration. Dans ce cas, la flexibilité est orientée dans la gestion de ressources disponibles sur plusieurs sites.

Ces travaux nous ont montré qualitativement qu'il était possible de combiner des gestionnaires de ressources pour obtenir une vision des ressources différente de l'infrastructure matérielle existante. L'exemple le plus représentatif est celui de la combinaison des systèmes à image unique avec les systèmes de virtualisation : il est possible d'agréger plusieurs nœuds et de créer une machine virtuelle bénéficiant des capacités de ces nœuds agrégés.

L'idée défendue dans ce chapitre est celle qu'on puisse obtenir plus de flexibilité dans la gestion et l'utilisation des nœuds d'une infrastructure informatique, en ayant un découplage complet entre ces derniers et la vue qu'a l'utilisateur de l'environnement d'exécution de

son application, et ceci même si cela implique de faire des concessions sur les performances d'exécution de ladite application [90].

Pour étayer notre propos, nous nous plaçons du point de vue d'un utilisateur non spécialiste en informatique. Nous pouvons considérer qu'il existe deux sortes d'utilisateurs : ceux qui ont besoin de performances maximales, on pense là à ceux qui veulent de la haute performance (HPC, *High Performance Computing*) et ceux dont la performance n'est pas l'objectif premier. Ce chapitre cible plus particulièrement les utilisateurs non-spécialistes en informatique qui ne visent pas la haute performance. Nous illustrons cela en prenant l'exemple d'un biologiste qui veut exécuter une application distribuée de calcul phylogénique. Ce type d'application peut prendre plusieurs jours avant de converger et de fournir un résultat. Ainsi, pour ce type d'utilisateur, la haute performance n'est pas absolument requise, car sur une durée de plusieurs jours d'exécution, quelques heures de plus ou de moins peuvent paraître négligeables s'il peut exécuter son application de manière simple, robuste et efficace sans perdre ces mêmes heures dans la configuration du système, configuration qui de surcroît n'apporte pas de valeur ajoutée au travail de calcul phylogénique.

Ainsi, le système que l'on propose dispose des mécanismes adéquats pour affranchir les utilisateurs de la complexité de la distribution des ressources. Nous proposons un formalisme permettant de classer les différents systèmes de gestion des ressources. En effet, ces différents systèmes, seuls ou en les combinant, permettent de configurer l'infrastructure matérielle afin que l'utilisateur obtienne une vue logique des ressources qui n'est pas forcément celle de l'infrastructure matérielle disponible. La figure 4.1 présente un exemple de découplage [92, 169]. Le système proposé a pour but de configurer automatiquement et de manière transparente le niveau « flexibilité » représenté sur cette figure. Cela se fait en configurant l'infrastructure physique ainsi que l'environnement d'exécution des applications en combinant les gestionnaires de ressources avec des outils de déploiement adéquats.

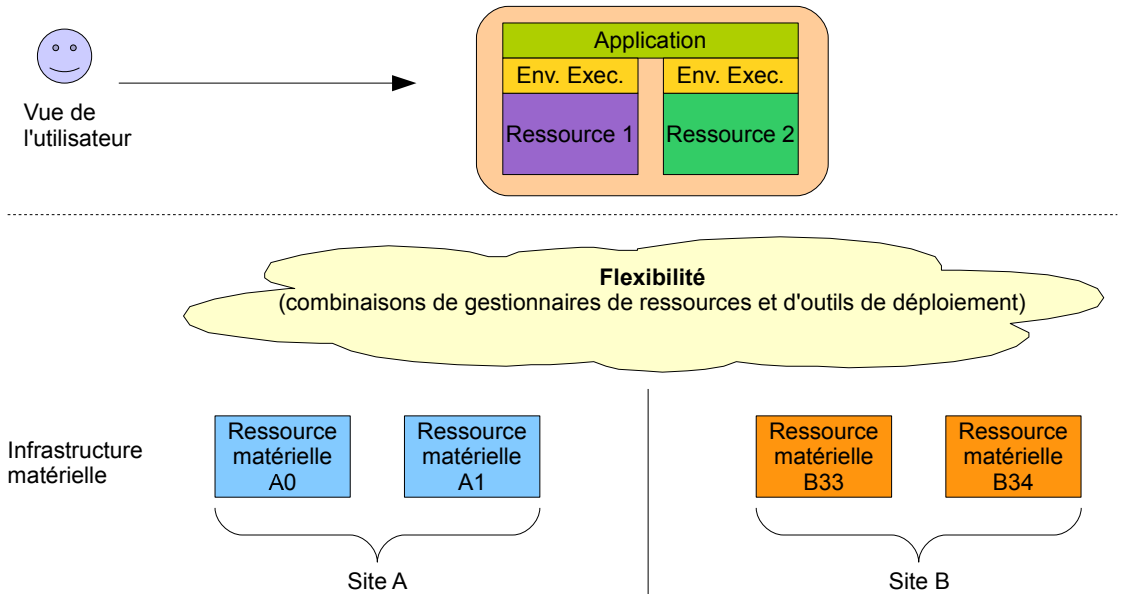


FIGURE 4.1 – Représentation du découplage entre l'infrastructure matérielle et la vue logique des ressources qu'a l'utilisateur

Les travaux présentés dans ce chapitre sont le fruit de collaborations avec Adrien Lèbre,

chargé de recherche à l'Ecole des Mines de Nantes, Geoffroy Vallée, associé de recherche à l'ORNL (Oak Ridge National Laboratory), Thomas Naugthon, associé de recherche à l'ORNL (Oak Ridge National Laboratory). Ces travaux se sont déroulés dans le contexte de l'équipe associée SER-OS [1] de l'INRIA et ont fait l'objet de plusieurs publications [88, 89, 90, 91, 92, 169].

4.1 Différencier la composition de l'infrastructure matérielle de la vue qu'en a l'utilisateur, une histoire qui commence dans les années 1960

David Wheeler, scientifique de renom dans le domaine des sciences informatiques et connu en particulier pour avoir été distingué en 1951 premier docteur en informatique au monde, est co-inventeur des fonctions (*subroutine*) permettant de diviser un « gros » programme en différentes « petites » fonctions. C'est en lien direct avec ces travaux qu'il dit : « *Any problem in computer science can be solved by another level of indirection.* », ce qui signifie que tous les problèmes qu'on peut rencontrer dans les sciences informatiques peuvent être résolus en ajoutant un niveau d'abstraction au problème à résoudre [73]. Placer cette citation dans le contexte de notre étude revient à dire que, pour une ressource matérielle, une ressource virtuelle est un niveau *d'indirection* entre le nœud et l'application à exécuter dans la ressource virtuelle. De la même manière, une centrale numérique est un niveau *d'indirection* entre l'infrastructure matérielle et les applications exécutées dans les machines virtuelles.

Cette notion *d'indirection* a été étudiée précisément et l'introduction du concept de machine à états abstraits (ASM, *Abstract State Machine*) a permis de donner une définition formelle de la notion d'abstraction [55]. Les ASM permettent de concevoir et analyser des systèmes informatiques complets en les modélisant par des ensembles et des fonctions entre ces ensembles. Ces travaux ont ensuite été étendus au domaine de la grille [101, 136], et deux principales abstractions sont à noter dans les systèmes de grille : l'abstraction des ressources et l'abstraction des utilisateurs. L'abstraction des ressources masque la complexité de gestion de la distribution des ressources. L'abstraction des utilisateurs assure la correspondance entre les comptes utilisateurs et leurs droits d'accès aux ressources de la grille, de manière transparente. Ces deux abstractions de très haut niveau sont ensuite mises en œuvre de manière plus précise par différents autres niveaux de moins en moins abstraits avant d'arriver au niveau d'abstraction le plus bas, celui des ressources matérielles.

4.1.1 Années 1970 : apparition des concepts de virtualisation et d'émulation

Goldberg présente dans les années 1970 une description formelle des systèmes de virtualisation qui selon lui n'a que peu (voire pas du tout) de rapport avec les systèmes d'émulation. La définition qu'il propose est la suivante : « *A system [...] which [...] is hardware-software duplicate of a real existing machine, in which a non-trivial subset of the virtual machine's instructions execute directly on the host machine[...]* » [96, 108, 122, 167]. À cette époque les systèmes d'exploitation des ordinateurs sont mono-tâche, les systèmes de virtualisation ont comme intérêt de pouvoir transformer le système mono-tâche en n

1. http://www.irisa.fr/myriads/ser-os/

sous-systèmes virtuels afin d'avoir, du point de vue de l'utilisateur, un système multi-tâches. La figure 4.2 présente ces différents systèmes.

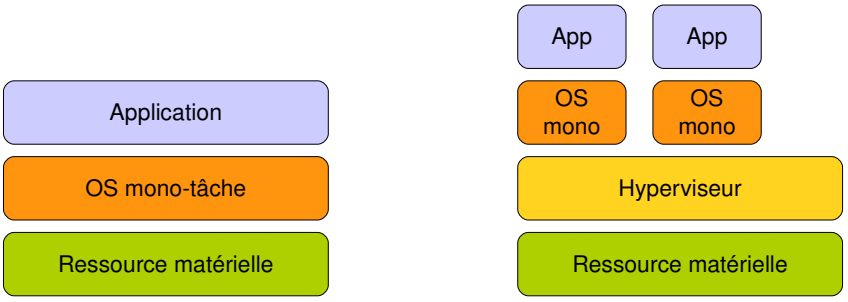


FIGURE 4.2 – La virtualisation à l'origine a pour but de rendre multi-tâches des systèmes d'exploitation mono-tâche avec le minimum d'impact sur les performances
Du point de vue de l'utilisateur, le dessin de gauche présente le système mono-tâche et celui de droite le système multi-tâches.

À cette même période, une autre communauté scientifique se concentre plus spécifiquement sur les systèmes d'émulation et la définition suivante d'un système d'émulation peut être notée : « *[...] a process whereby one computer is set up to permit the execution of programs written for another computer. This is done with [...] hardware features [...] and software [...]* » [118, 122].

Présentés ainsi, les concepts d'émulation et de virtualisation sont très différents : l'émulation est axée sur le fait qu'il est possible d'exécuter un programme écrit pour un type d'architecture matérielle donné (jeu d'instructions A – *ISA, Instruction Set Architecture*) sur un ordinateur hôte de type différent (jeu d'instructions C), alors que la virtualisation est axée sur l'exécution directe des programmes sur l'architecture matérielle hôte en « dupliquant » cette ressource matérielle (programme prévu pour le jeu d'instructions C sur un ordinateur hôte disposant du jeu d'instructions C).

Cependant, d'un point de vue concret, ces deux types de système peuvent être mis en relation et un émulateur peut être défini comme une machine virtuelle, au sens de Goldberg, dont le jeu d'instructions est différent de celui de l'ordinateur hôte [122]. La figure 4.3 schématise la différence entre émulation et virtualisation.

Dans ce document nous considérons que les systèmes d'émulation et de virtualisation bien que différents dans leur mise en œuvre et leur finalité apportent un même résultat : l'obtention d'une vue logique des ressources matérielles.

4.1.2 Années 2000 : regain d'intérêt pour les systèmes de virtualisation et d'émulation

De ces fondamentaux, présentés au cours des années 1970 sur la formalisation et la mise en œuvre de systèmes de virtualisation, sont apparues des déclinaisons que l'on connait sous le nom de machine langage ou de simulateur. C'est au cours des années 2000 que les machines virtuelles au sens de Goldberg ont connu un regain d'intérêt. Cela est lié à l'évolution des technologies permettant l'utilisation des systèmes de virtualisation et d'émulation, et plus précisément en 2004-2005 à l'arrivée sur le marché d'ordinateurs équipés de processeurs Intel (technologie Intel-VT) ou AMD (technologie AMD-Pacifica) capables de gérer nativement la virtualisation. C'est ainsi qu'en 2005 Smith et Nair ont

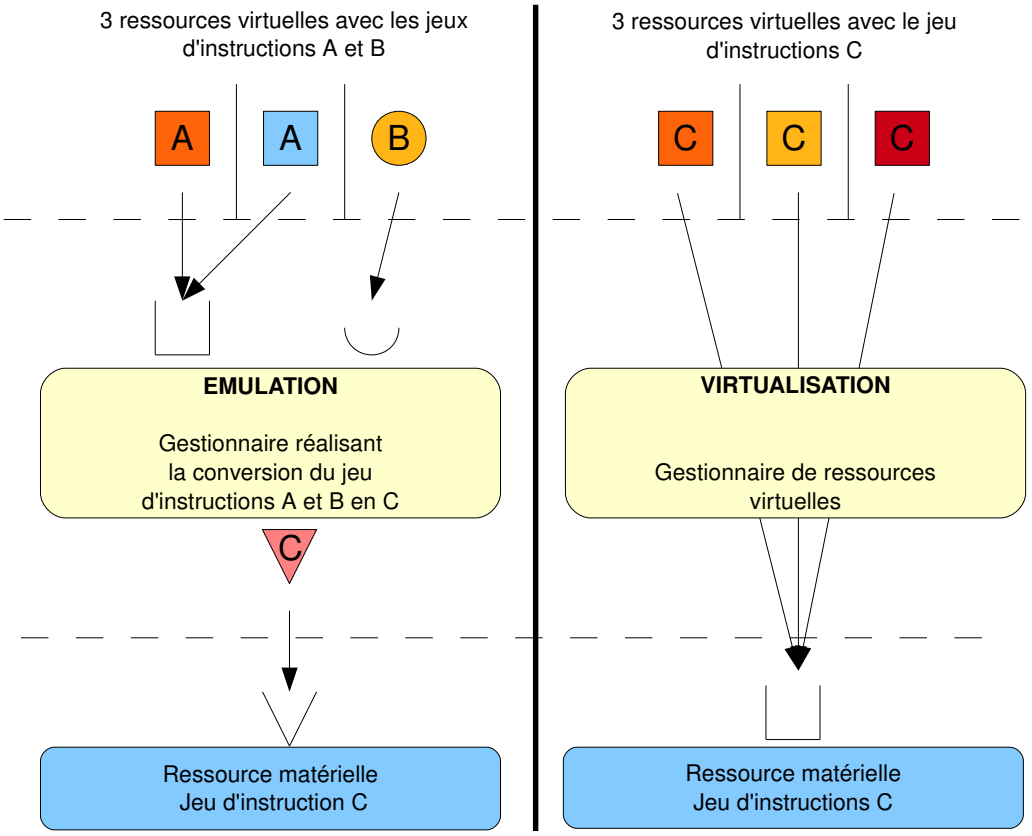


FIGURE 4.3 – Différence entre émulation et virtualisation

proposé une taxonomie permettant de classifier les différents systèmes de virtualisation et d'émulation [159] (voir figure 4.4).

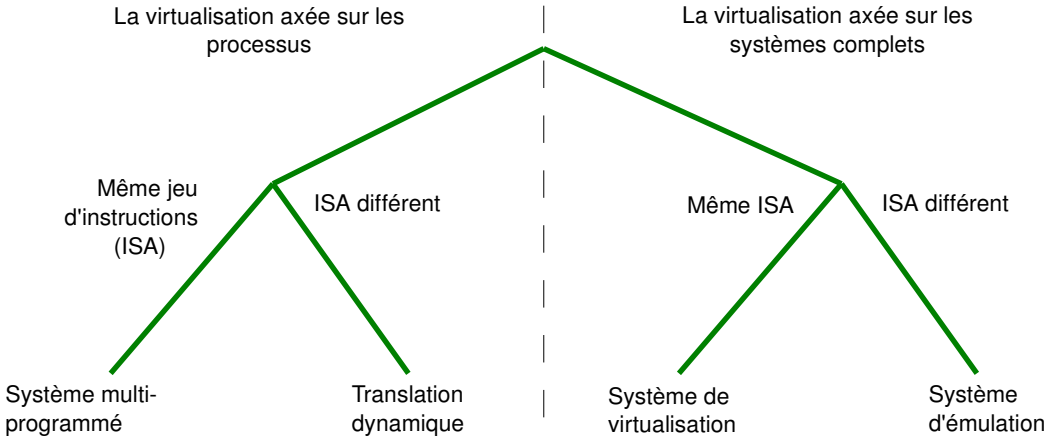


FIGURE 4.4 – Taxonomie des systèmes de virtualisation proposée par Smith et Nair Cette figure est extraite de [159].

4.2 Objectifs

Dans ce chapitre, nous défendons l'idée que la vision des ressources du point de vue de l'utilisateur (et de son application) doit être découplée de l'infrastructure matérielle réellement disponible. Pour y parvenir, nous proposons un formalisme permettant de décrire les différentes « transformations » qui doivent être effectuées pour projeter la vue logique qu'a l'utilisateur des ressources sur les ressources matérielles. Ainsi, nous proposons d'étendre la théorie de Goldberg qui est axée sur les systèmes de virtualisation à une plus large gamme de systèmes informatiques [43, 95, 98, 145] (voir figure 4.1). Cette extension permet de classer les différents systèmes de gestion des ressources informatiques comme des « outils » qu'il est possible de combiner. La combinaison de ces outils offre une vue logique des ressources découplée de l'infrastructure matérielle que nous appelons : plate-forme virtuelle. Nous proposons la conception et une mise en œuvre préliminaire de Tropicbird, un système de gestion des infrastructures informatiques distribuées fondé sur cette extension de la théorie de Goldberg pour la construction de plates-formes virtuelles sur des infrastructures matérielles.

4.3 Formalisation des systèmes de virtualisation

Afin de proposer un formalisme qui peut être utilisé pour découpler la vue des ressources qu'a l'utilisateur de l'infrastructure matérielle, nous proposons un raffinement de la théorie de Goldberg en étendant son application de la virtualisation à une large gamme de systèmes informatiques.

En 1973, Goldberg propose une formalisation du concept de la virtualisation. Son modèle repose sur deux fonctions ϕ et f [97, 145]. La fonction ϕ fait la correspondance entre un processus et la machine virtuelle dans laquelle il s'exécute. La fonction f fait la correspondance entre les ressources allouées pour une machine virtuelle et les ressources

matérielles. Les fonctions ϕ et f sont totalement indépendantes, ϕ est liée à des processus, f est liée à des ressources.

Les paragraphes suivants rappellent les points importants de la théorie de Goldberg.

4.3.1 Définition de la fonction f

Soit :
- R_0, l'ensemble des éléments constituant un nœud (niveau $n = 0$) tel que $R_0 = \{r_{0.0}, r_{0.1}, ..., r_{0.n}\}$.
- R_1, l'ensemble des éléments constituant une machine virtuelle s'exécutant sur un nœud (niveau $n = 1$) tel que $R_1 = \{r_{1.0}, r_{1.1}, ..., r_{1.m}\}$.

Goldberg définit $f : R_1 \rightarrow R_0$ tel que pour $y \in R_1$ et $z \in R_0$ alors, $f(y) = z$ si z est la ressource matérielle pour la machine virtuelle y. La fonction f fait le lien entre deux niveaux de ressources adjacents.

De plus Goldberg définit que pour $t \notin R_0$, si $f(y) = t$ alors cela cause une faute qui doit être traitée par l'hyperviseur. Il appelle ce type de faute les fautes de machine virtuelle (*VM-fault*).

4.3.1.1 Récursivité

La récursivité, c'est-à-dire, l'utilisation de machines virtuelles dans des machines virtuelles peut être faite en interprétant R_1 et R_0 comme deux niveaux adjacents de ressources virtuelles. Ainsi la ressource matérielle est au niveau 0 et les machines virtuelles sont au niveau n, avec $n > 0$. Dans le cas de la récursivité, f réalise la correspondance entre un niveau n et $n + 1$.

Exemple de récursivité Si $f_1 : R_1 \rightarrow R_0$, $f_2 : R_2 \rightarrow R_1$, alors une machine virtuelle y au niveau 2 peut être définie ainsi : $f_1(f2(y))$ ou encore $f_1 o f_2(y)$.

Goldberg généralise la récursivité au niveau n ainsi : $f_1 o f_2 o ... o f_n(y)$.

4.3.2 Définition de la fonction ϕ

Soit :
- $P = \{p_0, p_1, ..., p_j\}$ l'ensemble des processus.

Goldberg définit $\phi : P \rightarrow R_n$ tel que si $x \in P$, $y \in R_n$, alors $\phi(x) = y$ si y est une ressource pour le processus x. La fonction ϕ fait le lien entre les processus exécutés dans les machines virtuelles et la couche du niveau sous-jacent (le matériel : si $n = 0$, machine virtuelle sinon).

De plus, il définit $e \notin R_n$, tel que si $\phi(x) = e$ alors une exception est levée et celle-ci doit être traitée par le système d'exploitation de la machine virtualisée. Les exceptions sont donc différentes des fautes de machine virtuelle.

4.3.3 Exécution d'une machine virtuelle : composition de f et ϕ

Si l'ensemble des processus $P = \{p_0, p_1, ..., p_j\}$ est exécuté sur des machines virtuelles $R_1 = \{r_{1.0}, r_{1.1}, ..., r_{1.m}\}$, alors on peut écrire $\phi : P \rightarrow R_1$. De plus, les machines virtuelles sont elles aussi exécutées sur d'autres ressources telles que : $f : R_1 \rightarrow R_0$. Finalement on obtient l'ensemble complet d'exécution : $f o \phi : P \rightarrow R_0$, sachant qu'en cas d'exceptions ou de fautes de machine virtuelle, celles-ci sont traitées par l'hyperviseur.

À partir des définitions données précédemment on peut présenter le fonctionnement d'un ordinateur virtuel au niveau n par : $f_1 o f_2 o ... o f_n o \phi$.

4.3.4 Lien avec les systèmes de virtualisation de Type-I et de Type-II définis par Goldberg

Avec l'introduction des fonctions f et ϕ de Goldberg, il nous est maintenant possible de donner une définition formelle des systèmes de virtualisation de Type-I et de Type-II décrits dans la section 1.2.1.1. Pour rappel, Goldberg a classifié les systèmes de virtualisation en deux types :

Type-I Les systèmes de virtualisation de Type-I, sont des systèmes dont l'hyperviseur s'exécute directement sur le nœud hôte.

Type-II Les systèmes de virtualisation de Type-II, sont des systèmes dont l'hyperviseur s'exécute sur le système d'exploitation du nœud hôte.

Les systèmes de virtualisation de Type-I sont ceux qui dérivent directement de la définition de la fonction f de Goldberg où : $f : R_{n+1} \rightarrow R_n$. Dans le cas des systèmes de virtualisation de Type-II, l'ensemble d'arrivée de la fonction f n'est pas l'ensemble des ressources du niveau n, mais c'est l'ensemble des processus s'exécutant au niveau n tel que : $f : R_{n+1} \rightarrow P_n$. En d'autres termes, un système de virtualisation de Type-II est l'intermédiaire entre les ressources au niveau $n+1$ et un processus qui s'exécute sur des ressources au niveau n.

4.3.5 Pourquoi et comment proposer un raffinement de la théorie de Goldberg ?

Les contributions précédentes ainsi que la thèse défendue dans ce document montrent que la flexibilité dans les systèmes informatiques peut être atteinte grâce à la combinaison d'outils (gestionnaire de ressources et d'environnement) sur une infrastructure matérielle. Si l'on considère un utilisateur, un ensemble d'outils, et une infrastructure matérielle, la question que l'on se pose est : comment combiner les outils que l'on a à disposition sur l'infrastructure matérielle afin de répondre au besoin de l'utilisateur ?

La théorie proposée par Goldberg donne les fondements des systèmes de virtualisation axés sur les systèmes complets. Nous voulons raffiner cette théorie en l'étendant à d'autres systèmes informatiques. Le but recherché est de pouvoir décomposer une large majorité de systèmes informatiques pour les classer dans une sous-catégorie des systèmes de virtualisation ou d'émulation. Cette classification permettrait d'établir des règles de combinaison possible entre les outils, par exemple : définir que l'outil *système d'exploitation* ne s'installe que sur un *nœud de tel type* ou une *machine virtuelle de tel type*. Ces combinaisons peuvent ensuite être mises en œuvre au-dessus des infrastructures distribuées afin de répondre aux besoins des utilisateurs.

4.4 Proposition d'un formalisme de description des ressources informatiques

Cette section présente notre proposition d'extension de la théorie de Goldberg. Pour ce faire, nous décomposons les systèmes informatiques en différents niveaux (voir figure 4.5) :

(0) L'infrastructure matérielle distribuée : ce niveau correspond aux ressources matérielles. Les ressources peuvent être regroupées sur un site (grappe) ou bien distribuées sur différents sites (grille).

(1) Les gestionnaires de ressources : ce niveau correspond au système de gestion des ressources (par exemple, un système d'exploitation, un système d'exécution par lots,

un hyperviseur). Les gestionnaires de ressources peuvent être combinés. Par exemple, un système à exécution par lots s'installe au-dessus d'un système d'exploitation.

(2) L'environnement de configuration : ce niveau correspond à la configuration de l'environnement d'exécution pour une application donnée.

(3) L'application : ce niveau correspond à l'ensemble des processus constituant l'application à exécuter.

Sur la figure 4.5 pour des raisons de clarté, nous représentons ces différentes couches de manière empilée. Cependant, mis à part le niveau 0 qui correspond aux ressources matérielles (le niveau le plus bas), et au niveau 3 qui correspond à l'application (le niveau le plus haut), les niveaux 1 et 2 tels qu'on les décrit peuvent se combiner. Par exemple, un hyperviseur de Type-II (gestionnaire de ressource qui dans notre description correspond au niveau 1) que l'on exécute sur un système d'exploitation hôte (niveau 1 également) s'exécutera dans un environnement d'exécution (niveau 2) fournit par le système d'exploitation hôte.

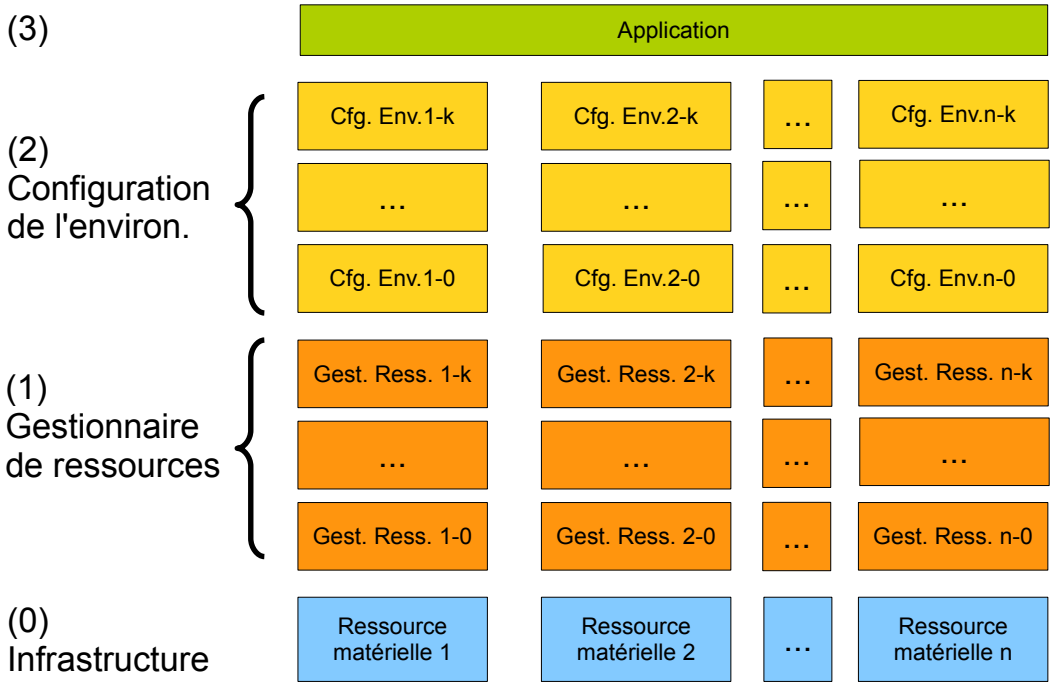


FIGURE 4.5 – Configuration des systèmes informatiques
Le niveau 0 représente l'infrastructure matérielle distribuée. Le niveau 1 représente l'ensemble des gestionnaires de ressources manipulant l'infrastructure matérielle distribuée. Le niveau 2 représente l'ensemble des librairies et programmes constituant l'environnement de configuration. Le niveau 3 correspond à l'application à exécuter.

On propose d'ajouter de la flexibilité dans les infrastructures matérielles en se concentrant sur les niveaux (0) et (1) (voir section 4.4.1). Ainsi, on propose un raffinement de la théorie de Goldberg afin d'étendre les concepts de virtualisation et d'émulation en introduisant ceux d'abstraction, d'agrégation et d'identité. Ces concepts servent à décrire les ressources matérielles ainsi que leurs gestionnaires comme des briques de Lego® qu'il est possible de combiner afin d'obtenir encore plus de fonctionnalités : le but est d'obtenir des fonctionnalités au niveau n non disponibles au niveau 0 quelles que soient les ressources

constituant le niveau 0, à condition d'avoir les gestionnaires de ressources adéquats pour leur mise en œuvre [87, 88, 89, 91] sur l'infrastructure matérielle.

Les niveaux (2) et (3) permettent la configuration de l'environnement d'exécution de l'application. Nous étudions ces niveaux dans la section 4.4.2.

Afin de mettre en valeur notre argumentation selon laquelle la vision des ressources qu'a l'utilisateur doit être découplée de l'infrastructure matérielle, nous n'utilisons plus le terme nœud pour désigner la machine capable de faire des calculs, mais nous utilisons l'expression *ressource matérielle* qui peut être un ordinateur, un super-calculateur, un périphérique mobile... Ces ressources peuvent être distribuées à l'échelle d'un site (une grappe) ou de plusieurs sites (une grille). De même, nous n'utilisons plus le terme machine virtuelle au profit de l'expression *ressource virtuelle*, qui est plus générique et peut désigner une machine virtuelle comme tout autre système s'appuyant sur un niveau $n > 0$ pour offrir des fonctionnalités à un niveau $n + 1$.

4.4.1 Raffinement de la fonction f de Goldberg

Nous pensons que la définition de Goldberg liée à la virtualisation peut être raffinée et étendue, afin d'introduire l'émulation ainsi que des sous-ensembles de l'émulation et de la virtualisation, pour l'étude et la description d'une plus large gamme de systèmes informatiques, et pas seulement les systèmes de virtualisation.

Cette section se concentrant sur la gestion des ressources, nous proposons ainsi un raffinement de la fonction f décrite par Goldberg entre les niveaux $n+1$ et n. De plus, et pour le reste de ce chapitre, les ensembles présentés suivent la théorie Zermelo-Fraenkel, avec l'axiome du choix (ZFC) [170].

Définition 4.1 (Un système) *On définit un système (S) par une ressource (R) (qui peut être elle-même décomposée en plusieurs ressources) à laquelle on applique la fonction f de Goldberg et que l'on étudie entre les niveaux n et $n+1$. Dans la suite de ce chapitre, on prend comme convention d'écriture : $S = (f, R_{n+1}, R_n)$.*

Par exemple, un système (S) dont la ressource est un « *ordinateur* » à laquelle on applique la fonction « *système d'exploitation* » entre les niveaux « *n+1* » et « *n* » s'écrit : $S = (système\_exploitation, ordinateur_{n+1}, ordinateur_n)$.

Définition 4.2 (Granularité) *On définit la granularité de l'étude d'un système par le fait que le système peut être étudié dans son ensemble, ou bien qu'il peut être décomposé en sous-ensembles pouvant eux-même être étudiés.*

Par exemple, on peut étudier le système (S1) : $S1 = (système\_exploitation, ordinateur_1, ordinateur_0)$, mais on peut également étudier le sous-système (S2) : $S2 = (système\_exploitation, mémoire_1, mémoire_0)$.

Définition 4.3 (Les attributs de capacité, de fonctionnalité et d'état) *On définit trois types d'ensemble d'attributs associés à une ressource :*

- *les attributs de capacité (attributC) sont liés à la notion d'espace dont la ressource dispose. On note $C_{R_n} = \{attributC_{1_n}, attributC_{2_n}, ..., attributC_{k_n}\}$ l'ensemble qui contient les attributs de capacité de la ressource (R) au niveau n. $C_{R_n} \subset C$, avec C l'ensemble des attributs de capacité qu'une ressource peut avoir.*
  Par exemple, pour un ordinateur (O) au niveau n, $C_{O_n} = \{attributC_{MEM_n}, attributC_{CPU_n}, attributC_{HD_n}\}$ ou bien encore, pour une mémoire (M) au niveau n, $C_{M_n} = \{2\ GB\}$.

- *les attributs de fonctionnalité (attributQ) sont liés à la notion de fonction de la ressource. On note* $Q_{R_n} = \{attributQ_{1_n}, attributQ_{2_n}, ..., attributQ_{k_n}\}$ *l'ensemble qui contient les attributs de fonctionnalité de la ressource (R) au niveau n.* $Q_{R_n} \subset Q$*, avec Q l'ensemble des attributs de fonctionnalité qu'une ressource peut avoir.*
  Par exemple, pour un ordinateur (O) au niveau n, $Q_{O_n} = \{attributQ_{MEM_n}, attributQ_{CPU_n}, attributQ_{HD_n}\}$ ou bien encore, pour une mémoire (M) au niveau n, $Q_{M_n} = \{read, write\}$.
- *les attributs d'état (attributE) sont liés à la notion d'état de la ressource. On note* $E_{R_n} = \{attributE_{1_n}, attributE_{2_n}, ..., attributE_{k_n}\}$ *l'ensemble qui contient les attributs d'état de la ressource (R) au niveau n.* $E_{R_n} \subset E$*, avec E l'ensemble des attributs d'état qu'une ressource peut avoir.*
  Par exemple, pour un ordinateur (O) au niveau n, $E_{O_n} = \{attributE_{MEM_n}, attributE_{CPU_n}, attributE_{HD_n}\}$ ou bien encore, pour un disque dur (DD) au niveau n, $Q_{DD_n} = \{ext2\}$.

La figure 4.6 donne un exemple de représentation d'une ressource ayant une mémoire (MEM), un processeur (CPU) et un disque dur (DD).

À partir des ensembles précédemment définis, nous proposons trois fonctions permettant de les manipuler.

Définition 4.4 (Fonctions de capacité, fonctionnalité et d'état) *La fonction f définie par Goldberg caractérise la transformation d'une ressource entre un niveau n et* $n + 1$.

Nous proposons de raffiner cette fonction en définissant trois nouvelles fonctions :

- *c, une fonction allant d'un ensemble d'attributs de capacité au niveau* $n + 1$ *vers un ensemble d'attributs de capacité au niveau n pour une ressource (R), c'est-à-dire,* $c_{R_{n+1}} : C_{R_{n+1}} \rightarrow C_{R_n}$.
  Par exemple, pour une mémoire (MEM) qui dispose d'un attribut de capacité au niveau n tel que $AttributC_{MEM_n}$ = "$2Go$" et au niveau $n + 1$ tel que $AttributC_{MEM_{n+1}}$ = "$1Go$", on peut écrire : $c_{MEM_{n+1}}$("$1Go$") = "$2Go$".
- *q, une fonction allant d'un ensemble d'attributs de fonctionnalité au niveau* $n + 1$ *vers un ensemble d'attributs de fonctionnalité au niveau n pour une ressource (R), c'est-à-dire,* $q_{R_{n+1}} : Q_{R_{n+1}} \rightarrow Q_{R_n}$.
  De plus, on définit q_action, une fonction d'arité a (qu'on note arité$(q\_action) = a$*) correspondant à une action capable d'être exécutée par le niveau n allant d'un ensemble d'attributs de fonctionnalité du niveau n à un ensemble d'attributs de fonctionnalité du niveau n tel que pour une ressource R,* $q\_action : Q^a_{R_n} \rightarrow Q_{R_n}$.
  Par exemple, pour un processeur (CPU) qui dispose d'attributs de fonctionnalité au niveau n tel que $AttributQ_{1_{CPU_n}}$ = "$move$" et $AttributQ_{2_{CPU_n}}$ = "add" et au niveau $n + 1$ tel que $AttributQ_{CPU_{n+1}}$ = " + ", on peut écrire : $q_{CPU_{n+1}}$(" + ") = $q\_action(move, add)$ avec q_action d'arité 2 correspondant à l'exécution coordonnée des primitives *move* et *add* fournies par le CPU.
- *e, une fonction allant d'un ensemble d'attributs d'état au niveau* $n + 1$ *vers un ensemble d'attributs d'état au niveau n pour une ressource (R), c'est-à-dire,* $e_{R_{n+1}} : E_{R_{n+1}} \rightarrow E_{R_n}$.
  De plus, on définit e_action, une fonction d'arité a (qu'on note arité$(e\_action) = a$*) correspondant à une action capable d'être exécutée par le niveau n allant d'un ensemble d'attributs d'état du niveau n à un ensemble d'attributs d'état du niveau n tel que pour une ressource R,* $e\_action : E^a_{R_n} \rightarrow E_{R_n}$.
  Par exemple, pour un disque dur (DD) qui dispose d'un attribut d'état au niveau n tel

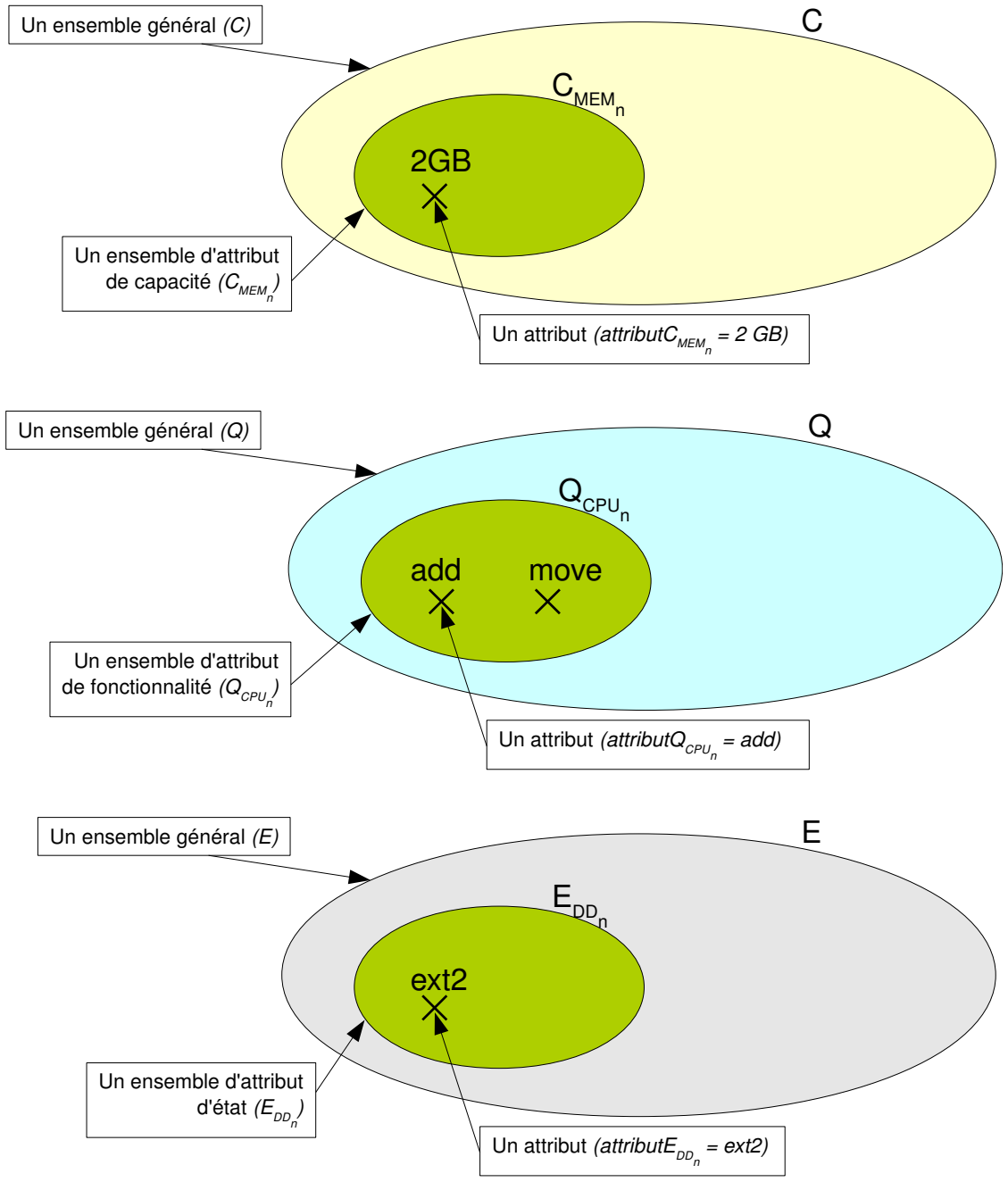


FIGURE 4.6 – Représentation d'un ensemble de capacité, fonctionnalité et état pour une ressource ordinateur

que $AttributE_{DD_n}$ = “$ext2$” et au niveau $n+1$ tel que $AttributE_{DD_{n+1}}$ = “$ntfs$”, on peut écrire : $e_{DD_{n+1}}(\text{“}ntfs\text{”}) = e\_action(ext3)$ avec e_action d'arité 1 correspondant à la transformation des actions $ntfs$ en $ext2$.

La figure 4.7 schématise deux ensembles d'attributs de capacité aux niveaux n et $n+1$ pour une ressource (M).

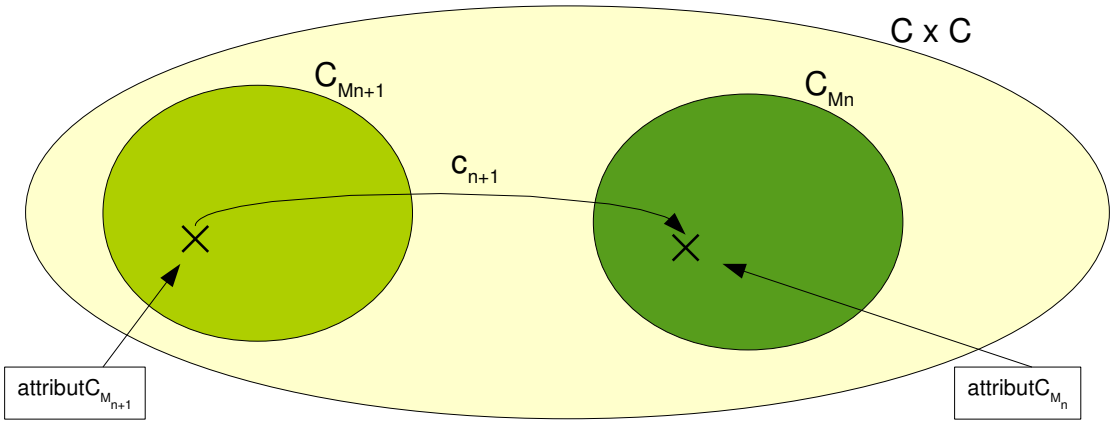


FIGURE 4.7 – Représentation de la fonction d'attributs de capacité
Soit une ressource (M), (M) dispose d'un ensemble d'attributs de capacité C_{M_n} au niveau n et $C_{M_{n+1}}$ au niveau $n+1$. Dans cet exemple, $c_{n+1}(attributC_{M_{n+1}}) = attributC_{M_n}$.

Considérant le système S tel que $S = (f, R_{n+1}, R_n)$, nous définissons dans les paragraphes suivants notre raffinement de la définition de la virtualisation ainsi que de l'émulation avec les concepts de : virtualisation-identité (identité), virtualisation-partitionnement (partitionnement), virtualisation-agrégation (agrégation), émulation-simple (émulation), émulation-abstraction (abstraction).

4.4.1.1 La virtualisation

Cette section présente notre raffinement de la définition de la virtualisation et de ses trois sous-catégories : la virtualisation-identité (ou identité), la virtualisation-partitionnement (ou partitionnement) et la virtualisation-agrégation (ou agrégation).

Définition 4.5 (Virtualisation)
$Virtualisation \Leftrightarrow (Q_{R_{n+1}} = Q_{R_n}) \wedge (E_{R_{n+1}} = E_{R_n})$

Cette définition est dérivée de la définition de Goldberg qui dit que dans la virtualisation, la majorité du code non-privilégié doit s'exécuter directement sur le matériel, c'est ce que l'on définit ici en indiquant que les ensembles d'attributs de fonction et d'état ne varient pas lors d'une opération de virtualisation.

Dans le cas de la virtualisation, il n'y a pas de microcode exécutant le code virtuel sur la ressource physique (le code est exécuté directement sur la ressource physique). Dans ce cas $arité(q\_action) = arité(e\_action) = 1$, et $q\_action = e\_action = id$ (avec id correspondant à la fonction mathématique identité).

Définition 4.6 (Virtualisation-Identité ou Identité)
$Identité \Leftrightarrow (Virtualisation) \wedge (C_{R_{n+1}} = C_{R_n})$

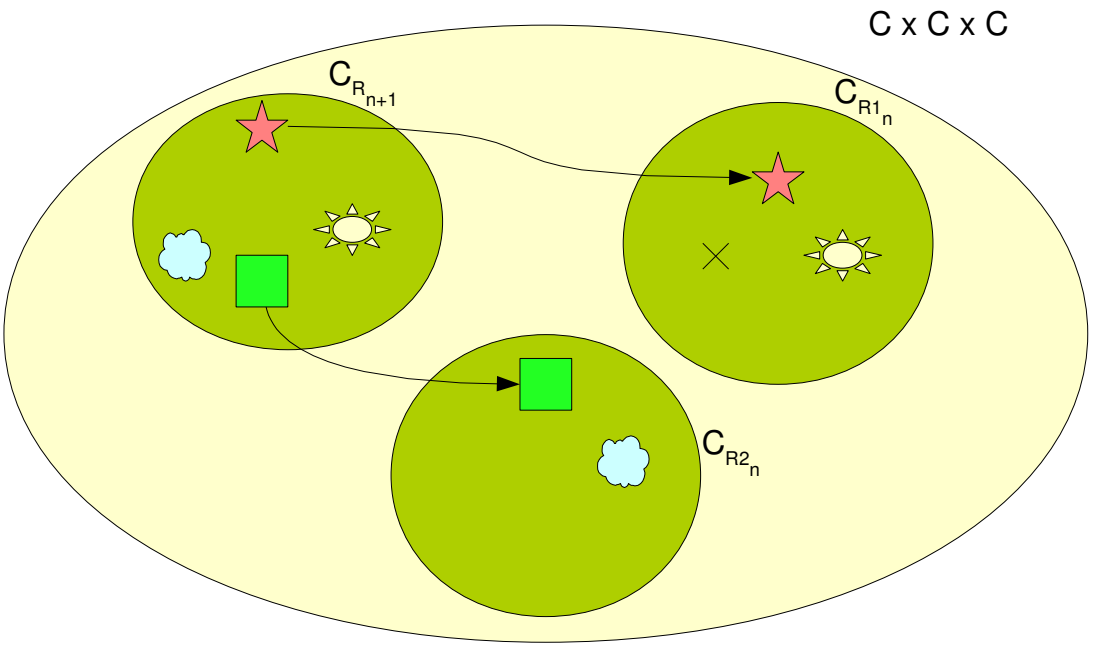


FIGURE 4.8 – L'agrégation

$$Agrégation \Leftrightarrow (Virtualisation) \wedge (C_{R_{n+1}} \subset \uplus_{i\geq 2} C_{Ri_n}) \wedge (\exists\,(x,y) \in C_{R_{n+1}},\ c_{n+1}(x) \in C_{Ri_n} \wedge c_{n+1}(y) \in C_{Rj_n}, i \neq j)$$

Dans le cas de l'identité, tous les ensembles d'attributs (C, Q et E) sont identiques d'un niveau n à $n+1$.

Définition 4.7 (Virtualisation-Partitionnement ou Partitionnement)
$Partitionnement \Leftrightarrow (Virtualisation) \wedge (C_{R_{n+1}} \subset C_{R_n})$

Dans le cas du partitionnement, l'ensemble des attributs de capacité du niveau $n+1$ est inclus dans celui du niveau n.

Définition 4.8 (Virtualisation-Agrégation ou Agrégation)

$$Agrégation \Leftrightarrow (Virtualisation) \wedge (C_{R_{n+1}} \subset \uplus_{i\geq 2} C_{Ri_n}) \wedge (\exists\,(x,y) \in C_{R_{n+1}},\ c_{n+1}(x) \in C_{Ri_n} \wedge c_{n+1}(y) \in C_{Rj_n}, i \neq j)$$

Dans le cas de l'agrégation, il y a au moins 2 attributs de capacité du niveau $n+1$ qui proviennent d'au moins deux ensembles d'attributs de capacité de ressources distinctes au niveau n. La figure 4.8 schématise l'agrégation.

4.4.1.2 L'émulation

Concernant l'émulation, on peut noter l'émulation-simple (ou émulation) ainsi que l'émulation-abstraction (ou abstraction).

Définition 4.9 (Emulation-Simple ou Emulation) $Emulation \Leftrightarrow \neg(Virtualisation)$

Cette définition indique qu'il y a une variation des attributs de fonctionnalité et/ou d'état d'un niveau $n+1$ à n. Conformément à la définition de l'émulation présentée en début de chapitre, cela signifie qu'un microcode est en charge de traduire les instructions du niveau $n+1$ au niveau n.

Définition 4.10 (Emulation-Abstraction ou Abstraction) *On note arité($fnct$), l'arité de la fonction $fnct$.*
Soit fnct *une fonctionnalité présente au niveau $n+1$ telle que : $fnct \in Q_{n+1}$. De même on définit* statut *un état présent au niveau $n+1$ tel que : statut $\in E_{n+1}$. On est dans le cas de l'abstraction si : $Abstraction \Leftrightarrow (Emulation) \wedge ((arité(q\_action) > 1) \vee (arité(e\_action) > 1))$*

L'abstraction est une simplification logique au niveau $n+1$ des fonctionnalités offertes au niveau n.

4.4.1.3 Synthèse

Nous proposons un raffinement de la théorie de Goldberg qui étend la virtualisation à une large gamme de systèmes informatiques :

L'émulation permet de fournir à un niveau $n+1$ des caractéristiques non disponibles au niveau n. Lorsque les caractéristiques fournies au niveau $n+1$ sont plus « simples » que celles du niveau n, on est dans le cas de l'abstraction.

La virtualisation permet d'accéder directement du niveau $n+1$ au niveau n. La virtualisation peut être raffinée en identité (toute la ressource au niveau n est disponible au niveau $n+1$), en partitionnement (le niveau $n+1$ dispose d'une vue partielle du niveau n) et enfin l'agrégation (au moins deux ressources distinctes au niveau n sont fournies au niveau $n+1$).

La figure 4.9 présente ces différents concepts.

4.4.2 Raffinement de la fonction ϕ de Goldberg

Cette section présente le raffinement de la fonction ϕ de Goldberg, raffinement qui est lié au niveau (2) de la figure 4.5.

Le but de ce raffinement est de pouvoir décrire des environnements d'exécution indépendants de la plate-forme ciblée. Pour ce faire, nous fondons notre travail sur le concept de collections de paquets (*package set*). Dans la suite de cette section nous présentons un résumé de ces travaux [168].

Définition 4.11 (Un paquet) *Un paquet est une abstraction qui permet de facilement installer, configurer, et supprimer un logiciel d'un système d'exploitation. Différentes « opérations » sont réalisables sur un paquet. D'un point de vue pratique, seulement un sous-ensemble d'« opérations » est important : il est possible de combiner des collections de paquets ou encore de faire des intersections entre différentes collections de paquets. Ces opérations permettent de configurer et gérer les environnements d'exécution de manière très fine (un environnement d'exécution étant composé d'un ou plusieurs paquets).*

4.4.2.1 Les opérations sur les paquets

Diverses opérations peuvent être appliquées sur les collections de paquets :

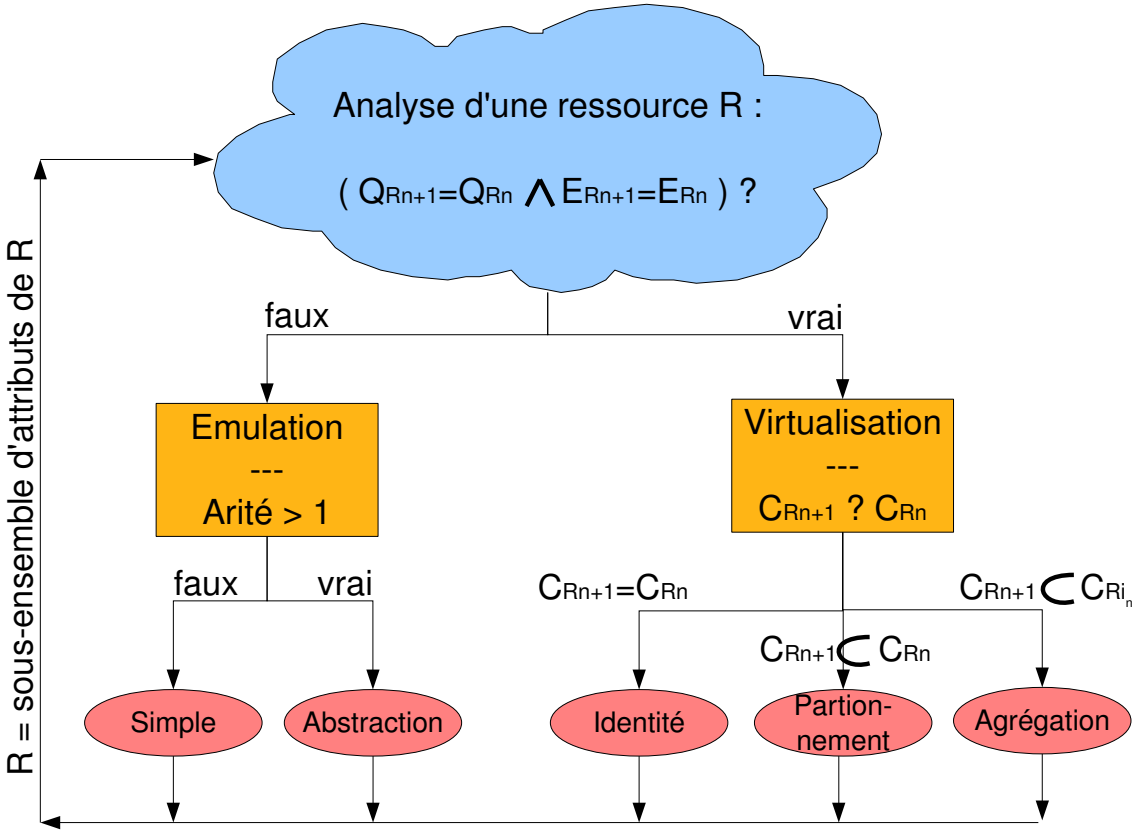


FIGURE 4.9 – Représentation simplifiée du raffinement de la théorie de Goldberg

La combinaison de paquets Il est possible de combiner des paquets ensemble : $CollPaquets_A \cup CollPaquets_B$. Par exemple, si un administrateur a pour politique d'intégrer un système de surveillance des ressources spécifiques dans tous les environnements d'exécution, il peut le spécifier dans une collection de paquets administrateur ($CollPaquets_{Administrateur}$) qui sera ensuite combinée avec la collection de paquets de l'utilisateur ($CollPaquets_{Utilisateur}$).

L'intersection entre les collections de paquets Il est possible de définir l'intersection entre des collections de paquets : $CollPaquets_A \cap CollPaquets_B$. Cette opération est utilisée pour identifier des paquets communs à des environnements d'exécution différents.

La validation de collection de paquets Il est possible de s'assurer que les paquets peuvent se combiner correctement. Cette opération s'appuie sur des mécanismes de gestion des versions (*versioning*) et des dépendances entre les paquets.

La gestion des versions Il est possible de spécifier la version d'un paquet spécifique dans une collection de paquets. Des opérateurs standard sont fournis afin de gérer les numéros de version : *égal*, *supérieur à*, *inférieur à*, *supérieur ou égal à* et *inférieur ou égal à*.

4.4.2.2 L'utilisation des collections de paquets

L'utilisation des paquets sert à créer une image de référence qui est spécifique aux ressources matérielles ciblées. Une fois la création de l'image de référence terminée, elle peut être déployée sur lesdites ressources.

4.5 Tropicbird : proposition d'architecture d'un système fondé sur le formalisme proposé

Dans la section précédente, nous avons proposé une extension de la théorie de Goldberg afin de présenter un formalisme permettant de décrire une large gamme de systèmes informatiques qui ne se limite pas aux systèmes de virtualisation. Ceci nous permet de décrire de manière découplée l'infrastructure matérielle de la vue logique des ressources. Cette vue logique des ressources, dont nous donnons plus de détails dans cette section (cf section 4.5.1.1), est appelée plate-forme virtuelle. Une plate-forme virtuelle est la réponse à un besoin formulé par un utilisateur. Elle est construite sur l'infrastructure matérielle à l'aide des gestionnaires de ressources et de configuration d'environnement. Dans cette section, nous décrivons plus précisément l'architecture de Tropicbird, un système s'appuyant sur ce formalisme. L'architecture proposée est composée de cinq modules principaux (voir figure 4.10).

A. La description L'utilisateur décrit ses contraintes (besoins) et l'administrateur décrit ses politiques. Les contraintes et les politiques sont définies en terme de caractéristiques d'environnement d'exécution et de plate-forme virtuelle. Les contraintes de l'utilisateur sont à définir pour chaque nouvelle requête alors que les politiques des administrateurs sont définies à l'avance (et non pas à chaque nouvelle requête d'un utilisateur).

B. Le module de conciliation ce module vérifie la compatibilité entre les contraintes de l'utilisateur et les politiques définies par l'administrateur.

C. Le module de planification en fonction des informations reçues du module de conciliation, le module de planification prépare les différentes étapes nécessaires pour satisfaire la requête de l'utilisateur en suivant les politiques de l'administrateur.

D. Le module de découverte, d'allocation et de configuration ce module reçoit le plan du module de planification et l'exécute selon les ressources disponibles.

E. La boîte à outils ce module contient la liste ainsi que l'ensemble des outils (gestionnaire de ressources et d'environnement) permettant de configurer et reconfigurer les ressources matérielles et les environnements d'exécution.

4.5.1 Le module de description : les concepts de plate-forme virtuelle et d'environnement d'exécution virtuel

Ce chapitre étudie les moyens de découpler la vue logique des ressources qu'a l'utilisateur de l'infrastructure matérielle. Nous appelons cette vue logique des ressources une plate-forme virtuelle. Cette section présente le concept de plate-forme virtuelle (VP, *Virtual Platform*)[90] pour la configuration de l'infrastructure matérielle et rappelle celui d'environnement d'exécution virtuel (VSE, *Virtual System Environment*) [71, 168] pour la configuration des environnements d'exécution. Les plates-formes virtuelles ainsi que les environnements d'exécution virtuels permettent à l'utilisateur et à l'administrateur d'exprimer leurs besoins en terme de ressources et d'environnements d'exécution sans faire d'hypothèse sur les ressources matérielles sous-jacentes. Par exemple, un utilisateur peut demander un ensemble de ressources distribuées avec une latence maximale entre elles. Un administrateur peut définir une politique dans laquelle tous les utilisateurs appartenant à un groupe spécifique ne peuvent accéder aux ressources d'une grappe d'un site.

La séparation des contraintes en terme de plate-forme virtuelle (contraintes sur les ressources) et d'environnement d'exécution virtuel (contraintes sur les logiciels) permet d'obtenir une plus grande flexibilité par rapport aux attentes des utilisateurs. Un utilisateur ayant des besoins extrêmement précis pourra définir à la fois la plate-forme virtuelle qu'il désire et l'environnement d'exécution. Au contraire, un utilisateur voulant s'affranchir au maximum de la configuration de l'infrastructure ne définira que l'environnement d'exécution qu'il désire. Tropicbird est dans ce cas en charge de définir et si besoin, de construire lui-même la plate-forme virtuelle adéquate.

4.5.1.1 Les plates-formes virtuelles

Le système proposé a pour objectif de gérer différents types d'infrastructures distribuées. La distribution ainsi que l'hétérogénéité de ces infrastructures rendent complexes leur gestion et leur utilisation. Cependant, cette complexité peut être masquée à l'utilisateur qui, dans le cas que nous traitons, n'ambitionne pas d'avoir des performances extrêmes ou de savoir exactement quelle est la configuration de l'infrastructure matérielle. C'est pour cela que nous définissons une abstraction entre l'infrastructure matérielle et la vue logique des ressources que nous appelons *plate-forme virtuelle*.

Une plate-forme virtuelle est une vue logique des ressources qu'a l'utilisateur pour l'exécution de son application et qui peut être différente de l'infrastructure matérielle disponible. La description d'une plate-forme virtuelle peut être faite sans aucune hypothèse sur l'infrastructure matérielle sous-jacente. Le système proposé dans ce chapitre met en œuvre la plate-forme virtuelle en combinant différents outils classés par catégorie (comme décrit dans la section 4.4, à savoir des outils pour faire de l'émulation, du partitionnement, de l'agrégation, ...) afin de construire une plate-forme virtuelle personnalisée.

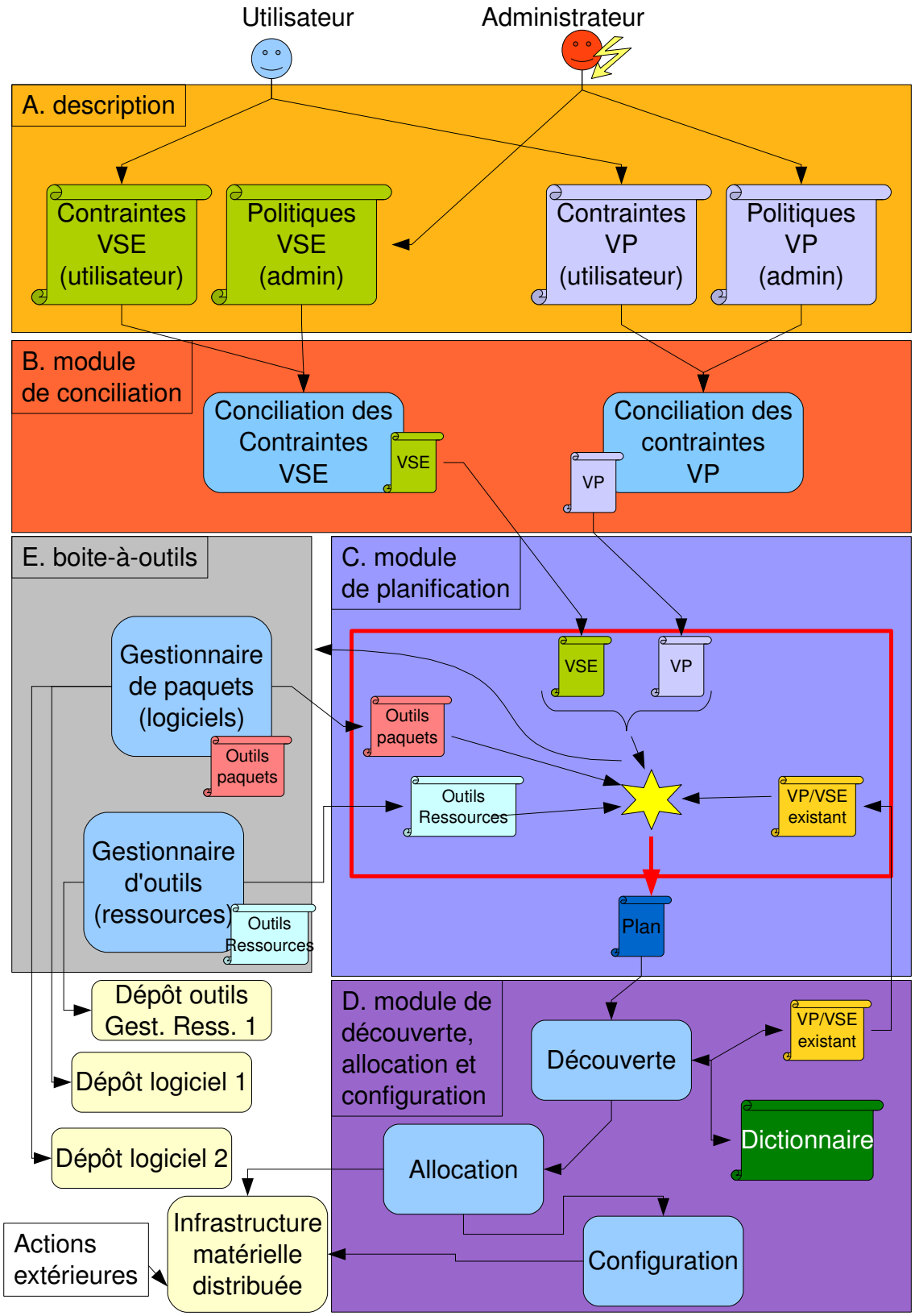


FIGURE 4.10 – Architecture générale de Tropicbird

Par exemple, considérons une infrastructure matérielle constituée de deux sites contenant chacun 15 nœuds. Un utilisateur peut avoir besoin de 20 ressources sans besoin particulier en terme de performance. L'administrateur peut définir une politique autorisant la reconfiguration des ressources des différents sites. Ainsi, le système va selon les ressources disponibles réserver et configurer avec des outils adéquats (gestionnaire de ressources et d'environnements) 10 ressources dans le premier site et 10 autres dans le second, en les mettant dans un même réseau local (utilisation d'outils de type VPN par exemple). Du point de vue de l'utilisateur, sa plate-forme virtuelle est constituée de 20 ressources appartenant à un même réseau local. Du point de vue de l'infrastructure matérielle, ces ressources sont distribuées sur deux sites.

4.5.1.2 Les environnements d'exécution virtuels

Un environnement d'exécution virtuel permet de décrire une application en terme de besoins logiciels. Ce type de besoins concerne le système d'exploitation et les bibliothèques (*libraries*). Par exemple, une application MPI peut être faite pour être exécutée sur un système d'exploitation *Red Hat Entreprise Linux 4.0* avec les bibliothèques *LAM/MPI 7.1.3*. Si l'une de ces contraintes change, par exemple, un changement du système d'exploitation en *Microsoft Windows 7*, l'application nécessitera d'être modifiée afin de pouvoir être exécutée dans ce nouvel environnement.

Cependant, l'exécution de l'application de l'utilisateur ne devrait pas être impactée par un changement de configuration de l'infrastructure matérielle. En d'autres termes, l'utilisateur ne doit pas avoir à être confronté à des modifications non-souhaitées de son environnement de travail. Cela entraînerait une adaptation (par exemple une reconfiguration ou une recompilation) de son application, opération qui peut être coûteuse en terme de temps.

Un environnement d'exécution virtuel est une méta-description de l'environnement d'exécution souhaité par l'utilisateur. Cet environnement d'exécution est mis en œuvre sur la plate-forme virtuelle. En se référant à la gestion des paquets décrite dans la section 4.4.2, un utilisateur peut décrire l'environnement d'exécution dont il a besoin sans faire d'hypothèse sur la plate-forme virtuelle utilisée. De même, un administrateur peut définir des contraintes qu'il impose aux environnements d'exécution virtuels des utilisateurs (par exemple, imposer l'installation d'un système de surveillance). Finalement, en reprenant l'exemple précédemment cité sur un changement de système d'exploitation pour une mise à jour avec le système *Windows 7*, l'avantage des environnements d'exécution virtuels est que ceux-ci se reconfigureront automatiquement, de manière transparente pour les utilisateurs afin de fournir un environnement d'exécution adéquat (à condition que les paquets adéquats soient disponibles et puissent être installés).

4.5.2 Le module de conciliation : conciliation entre les contraintes de l'utilisateur et les politiques de l'administrateur

Le module de conciliation est celui qui vérifie que les contraintes des utilisateurs respectent les politiques définies par les administrateurs. Un utilisateur fait une requête en terme d'environnement et/ou de plate-forme virtuelle. Les administrateurs peuvent définir des politiques liant les utilisateurs, les environnements d'exécution (quels logiciels peuvent être installés), les plates-formes virtuelles et l'infrastructure matérielle (quel type de reconfiguration de l'infrastructure est autorisé). Ce module fusionne les besoins de l'utilisateur avec ceux de l'administrateur : du point de vue de l'environnement d'exécution le système doit construire une description complète prenant en compte les besoins de l'utilisateur et

les politiques de l'administrateur, et du point de vue des ressources, le système doit configurer la plate-forme automatiquement selon les besoins de l'utilisateur et les politiques de l'administrateur afin de déployer l'environnement d'exécution. Finalement, les requêtes conciliées de l'utilisateur et de l'administrateur sont transmises au module de planification.

4.5.3 Le module de planification : mise en œuvre des plates-formes virtuelles et des environnements d'exécution virtuels

Le module de planification a la charge de créer un plan de construction afin de pouvoir mettre en œuvre la plate-forme virtuelle ainsi que l'environnement d'exécution virtuel sur les ressources matérielles disponibles. Trois étapes peuvent être notées :

- le module d'exécution vérifie si l'environnement d'exécution virtuel demandé est déjà disponible sur l'infrastructure matérielle,
- si oui, le module de planification fait une requête d'allocation de l'environnement trouvé,
- sinon, le module de planification crée un plan de mise en œuvre de la plate-forme virtuelle sur l'infrastructure matérielle et de l'environnement d'exécution virtuel sur la plate-forme virtuelle précédemment créée,
- le module de planification vérifie qu'il n'existe pas une partie du plan qui est déjà mise en œuvre sur l'infrastructure matérielle. Dans le pire des cas, aucune partie du plan n'est trouvée, la plate-forme virtuelle ainsi que l'environnement d'exécution virtuel sont construits sur l'infrastructure matérielle (depuis le niveau $n = 0$).

La figure 4.11 présente la mise en œuvre des plates-formes virtuelles ainsi que des environnements d'exécution virtuels sur une infrastructure matérielle distribuée pour un exemple de requête d'un utilisateur.

4.5.3.1 La disponibilité de l'environnement

Le module de planification a la charge de mettre en œuvre la plate-forme virtuelle ainsi que l'environnement virtuel selon la description fournie par le module de conciliation. Tout d'abord, le module de planification vérifie qu'il n'existe pas déjà un environnement répondant aux critères recherchés (si un environnement répondant aux critères existait déjà, ça ne serait pas la peine de construire une plate-forme virtuelle). Cette recherche est faite avec le module de découverte.

- Si un environnement d'exécution correspondant à la requête existe, est disponible et que son utilisation n'entre pas en conflit avec des politiques définies par l'administrateur, Tropicbird envoie une demande de réservation au module d'allocation des ressources.
- Si la recherche précédente se révèle infructueuse, le module de planification va mettre en œuvre la plate-forme virtuelle et l'environnement d'exécution virtuel (voir section suivante 4.5.3.2).

4.5.3.2 Le plan : la mise en œuvre des plates-formes virtuelles et des environnements d'exécution virtuels

Le module de planification planifie les actions à réaliser afin de configurer les logiciels sur les ressources. Pour ce faire, le module de planification interroge la boîte à outils afin d'obtenir la liste des outils disponibles pour construire la plate-forme virtuelle ainsi que l'environnement d'exécution virtuel. Le module de planification vérifie que ces listes n'enfreignent pas les contraintes de l'utilisateur et les politiques des administrateurs.

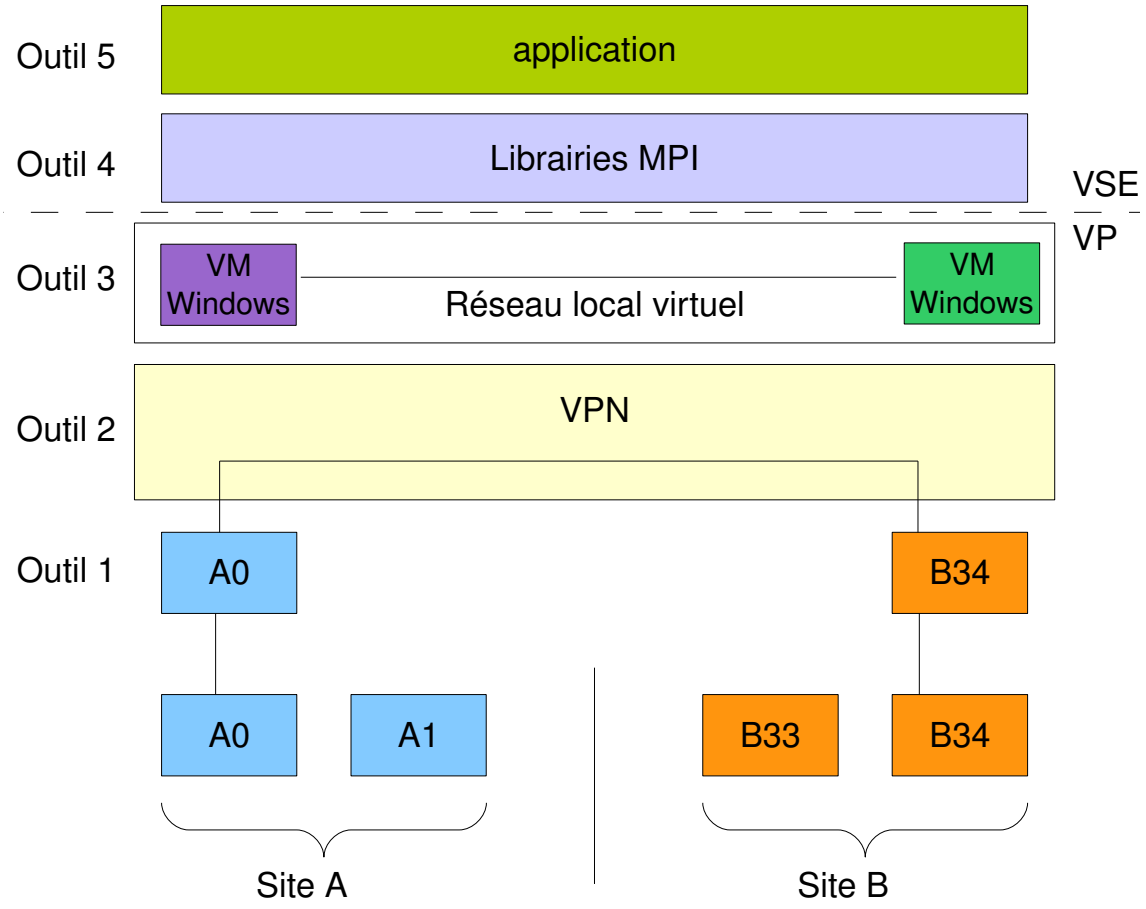


FIGURE 4.11 – Plate-forme virtuelle et environnement d'exécution virtuel sur un même schéma

On considère que les ressources matérielles sont déployables à la manière des nœuds dans Grid'5000, c'est-à-dire que les nœuds disposent d'une console de service leur permettant d'être redémarrés et reconfigurés par le réseau.

Exemple de plan pour la requête utilisateur suivante « application nécessitant MPI sur Windows distribuée sur 2 ressources » et la politique administrateur « aucune restriction » :

1. Allouer les ressources,
2. Mettre en œuvre la plate-forme virtuelle :
   a) utiliser « *outil1* » pour le déploiement du « système d'exploitation »,
   b) utiliser « *outil2* » pour mettre en œuvre un réseau privé (*VPN*) entre les différentes ressources,
   c) utiliser « *outil3* » pour l'installation d'un « système de virtualisation » avec un système d'exploitation Windows,
3. Mettre en œuvre un environnement d'exécution virtuel :
   a) utiliser « *outil4* » pour l'installation des « librairies »,
   b) utiliser « *outil5* » pour l'installation de l'application, ...

Si aucune contrainte et politique ne sont enfreintes, le module de planification demande au module de découverte de chercher si une partie de la mise en œuvre du plan n'est pas déjà disponible sur l'infrastructure matérielle cible. Par exemple, si le plan consiste à déployer une application MPI sur des ressources Windows que l'on note (ressources+Windows+MPI+application) et qu'aucune ressource disponible ne contient d'environnement Windows, MPI et l'application, alors Tropicbird va assouplir le plan en recherchant un environnement Windows MPI uniquement (ressources+Windows+MPI) dans le but d'installer l'application par lui-même. Si aucun environnement n'est trouvé, le plan est assoupli de nouveau. Enfin, dans le pire des cas, le plan consiste à rechercher des ressources matérielles « nues » (niveau $n = 0$) et à les configurer totalement. Lorsque les ressources sont trouvées, le plan est envoyé au module d'allocation et de configuration des ressources pour exécution (cependant, si aucune ressource n'est trouvée ou que les ressources trouvées ne peuvent être configurées – par exemple, dans notre cas, pas de possibilité d'installer Windows –, la plate-forme virtuelle ne peut alors pas être construite).

4.5.4 Découverte, allocation et configuration de l'infrastructure matérielle

Le plan est donné au module de découverte, d'allocation et de configuration en charge de mettre en œuvre le plan, c'est-à-dire de mettre en œuvre la plate-forme virtuelle et l'environnement d'exécution virtuel. En cas d'erreur, par exemple si aucune ressource n'est disponible au moment de l'allocation, le module de planification est chargé de modifier le plan et de faire une nouvelle soumission de recherche de ressources (selon les cas, il est possible que plusieurs combinaisons d'outils soient possibles pour construire la plate-forme virtuelle, le module de planification les essayera toutes).

La figure représente l'action du module de configuration. Le module de configuration prend en entrée un environnement présent sur l'infrastructure (niveau n) et lui applique un outil de gestion. En sortie, le module de configuration produit un nouvel environnement (niveau n+1)

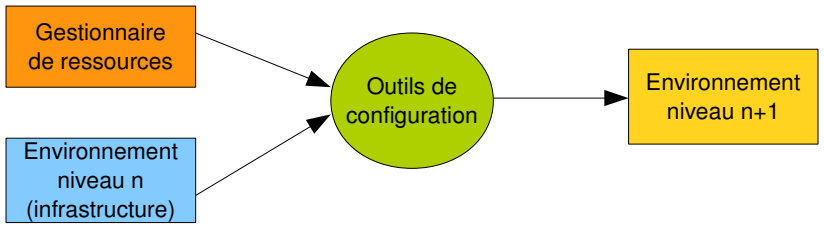


FIGURE 4.12 – Configuration des gestionnaires de ressources

4.5.5 Boîte à outils

La boîte à outils gère la collection d'outils capables de mettre en œuvre et de configurer les plates-formes virtuelles et les environnements d'exécution virtuels. Deux types d'outils peuvent être distingués : les outils de gestion des logiciels (configuration des environnements d'exécution virtuels), et les outils de gestion des ressources (configuration des plates-formes virtuelles).

Un troisième type d'outils est à noter. Ce sont les outils de déploiement des outils de gestion des ressources. En effet, notre architecture se repose sur le module de configuration

(voir figure 4.12) pour construire les plates-formes virtuelles. Cependant, comme nous le verrons dans les perspectives de travaux, le module de configuration pourrait lui-même se construire de manière dynamique pour se doter des outils dont il a besoin pouvoir ensuite déployer les gestionnaires de ressources pour construire la plate-forme virtuelle désirée.

4.5.5.1 Outils pour les environnements d'exécution virtuels

Afin de fournir un système efficace dans la gestion des paquets, notre solution se fonde sur l'utilisation du concept de catalogue de paquets (*repository*). Les paquets sont accessibles depuis un endroit défini et peuvent être téléchargés puis installés et configurés sur les ressources matérielles en utilisant les outils adéquats.

Par exemple, avec les systèmes d'exploitation *Debian*, il existe des catalogues de paquets Debian qui permettent à l'utilisateur d'installer et de configurer des logiciels en provenance de ces catalogues avec l'outil *apt-get* ou *aptitude* [5].

Pour une application spécifique, la boîte à outils est capable de fournir au module de planification une liste de toutes les dépendances des paquets nécessaires à son installation et sa configuration pour les architectures matérielles ciblées.

4.5.5.2 Outils pour les plates-formes virtuelles

De la même manière que précédemment, nous utilisons le concept de catalogue. Ici, le catalogue contient une liste des outils disponibles pour faire de l'émulation, du partitionnement, de l'identité, de l'agrégation. Par exemple, comme présenté dans le chapitre 3 traitant de Grillade, le système d'exploitation Kerrighed permet de faire de l'agrégation des ressources. Un autre exemple est Saline, qui est capable de gérer des collections de machines virtuelles à l'échelle de la grille en faisant ainsi du partitionnement et de l'abstraction de l'infrastructure matérielle.

4.6 Éléments de mise en œuvre

Tropicbird est un prototype mettant en œuvre des plates-formes virtuelles sur une infrastructure matérielle et des environnements d'exécution virtuels sur les plates-formes virtuelles. Il s'appuie sur des outils existants (OSCAR, TakTuk) ainsi que sur les contributions présentées dans ce document (Grillade et Saline). Actuellement Tropicbird n'est pas intégralement opérationnel. Nous présentons dans cette section les points clés de la mise en œuvre effectuée.

4.6.1 La description et le module de conciliation

Notre prototype permet une description des plates-formes virtuelles en se concentrant sur le nombre de nœuds, de processeurs et la taille mémoire souhaitée par nœud. Cette configuration se fait en passant des arguments en ligne de commande. Nous menons des travaux pour améliorer cette manière de décrire les ressources en utilisant un fichier de type JSDL.

Les environnements d'exécution virtuels sont également décrits à travers un fichier de configuration. Actuellement, deux méthodes ayant chacune un degré d'abstraction différent sont proposées (nous travaillons à l'intégration de ces deux méthodes pour n'offrir à l'utilisateur qu'une seule méthode générale). Dans le premier cas, l'utilisateur peut définir l'ensemble des paquets désirés par rapport au système d'exploitation demandé. Cette méthode est requise pour les utilisateurs soucieux de définir par eux-même précisément

l'environnement d'exécution qu'ils veulent (par exemple, la version des paquets et leurs dépendances). Dans le second cas, il peut décrire l'ensemble des paquets de manière générique dans un fichier XML et le donner en argument de l'outil OSCAR qui est en charge de trouver les paquets et de les configurer automatiquement en fonction de la distribution ciblée. Cette seconde méthode est destinée aux utilisateurs qui veulent un degré d'automatisation élevé et qui ne cherchent pas à préciser explicitement que tel ou tel paquet doit être installé dans l'environnement d'exécution.

Concernant la gestion des politiques, nous utilisons le service de gestion des organisations virtuelles proposé dans XtreemOS. Les administrateurs définissent des organisations virtuelles dans lesquelles sont attribuées des nœuds et des utilisateurs. Les politiques sont ensuite définies à l'aide de fichier XML au standard XACML [37]. Dans l'état actuel de notre prototype nous avons validé l'utilisation des organisations virtuelles en définissant des politiques XACML permettant à un utilisateur appartenant à une organisation virtuelle d'avoir accès sans restrictions à toutes les ressources de cette organisation virtuelle (politique par défaut d'XtreemOS).

4.6.2 Le module de planification

Le module de planification, encore en cours d'élaboration, est l'élément central pour la construction d'un environnement complet (plate-forme virtuelle et environnement d'exécution virtuel). Son rôle est, selon les plates-formes virtuelles et les environnements d'exécution virtuels déjà configurés sur l'infrastructure matérielle (parchemin *VP/VSE existant* de la figure 4.10), d'établir un plan pour la configuration et/ou la reconfiguration des ressources et des environnements d'exécution pour répondre aux requêtes des utilisateurs et des administrateurs (parchemins *VP* et *VSE* de la figure) grâce aux outils présents dans la boîte à outils (parchemin *Outils paquets* et *Outils ressources* de la figure).

La version actuelle de notre prototype, se concentre sur les caractéristiques suivantes : le type d'environnement (système d'exploitation, *env*), le type de processeur (*cpu*) et la taille de la mémoire (*mem*).

4.6.2.1 La requête

La requête reçue du module de conciliation (parchemins *VP* et *VSE* de la figure 4.10) contient les informations nécessaires pour la configuration et la reconfiguration de la plate-forme virtuelle et l'environnement d'exécution virtuel. Le tableau 4.1 présente deux exemples de requête dont les identifiants sont *requête[0]* et *requête[1]*. Une requête se caractérise par deux besoins : les ressources matérielles d'une part et la plate-forme virtuelle et environnement d'exécution virtuel d'autre part. Le besoin correspondant à la plate-forme et l'environnement d'exécution virtuels correspond à la plate-forme virtuelle et l'environnement d'exécution virtuel souhaités. Par exemple, la *requête[0]* précise qu'elle attend un environnement Solaris avec un processeur de type *SPARC* équipé de *32Go* de mémoire. Le besoin en ressources matérielles correspond aux besoins de l'utilisateur en terme de ressources matérielles. Lorsqu'un champ d'une contrainte matérielle est renseigné par « - » cela signifie que c'est au système de le compléter.

Par exemple, la *requête[0]* qui souhaite un environnement Solaris avec un processeur de type *SPARC* équipé de *32Go* de mémoire, ne précise rien concernant les ressources matérielles du niveau $n = 0$. C'est au système de trouver les ressources matérielles et la bonne combinaison d'outils pour donner une réponse à la requête. La *requête[1]* par opposition précise que l'utilisateur veut accéder à des ressources matérielles directement (la plateforme virtuelle et l'environnement d'exécution virtuel souhaités sont les mêmes

que ceux souhaités sur les ressources matérielles au niveau $n = 0$). De cette manière il est possible à l'utilisateur de guider la construction du plan : par exemple, dans le cas de la *requête[1]* aucun outil d'émulation ne pourra être utilisé alors que dans celui de la *requête[0]* toutes les combinaisons sont autorisées.

Identifiant	Ressources matérielles ($n = 0$)	Plate-forme et environnement d'exécution virtuels souhaités
requête[0]	env=-, cpu=-, mem=-	env=Solaris, cpu=SPARC, mem=32Go
requête[1]	env=Windows, cpu=i386, mem=2Go	env=Windows, cpu=i386, mem=2Go

TABLE 4.1 – Structure de données utilisée pour la formulation des requêtes (représentation des parchemins *VP* et *VSE* de la figure 4.10)

4.6.2.2 La découverte des plates-formes et des environnements d'exécution virtuels déjà présents sur l'infrastructure matérielle

Avant de commencer la planification de la construction de la plate-forme virtuelle et de l'environnement d'exécution souhaité par la requête, le module de planification recherche sur l'infrastructure matérielle l'ensemble des plates-formes virtuelles et des environnements d'exécution disponibles et déjà configurés. Pour cela, le module de planification fait une requête au module de découverte des resssources et obtient la liste des plates-formes et des environnements présents sur l'infrastructure matérielle (parchemin *VP/VSE existant* de la figure 4.10).

Nous proposons de décrire cette liste avec la structure de données représentée dans le tableau 4.2. Les plates-formes virtuelles et les environnements d'exécution présents sur l'infrastructure matérielle sont représentés par un identifiant du type *vp-vse[X]*. Un *vp-vse[X]* est caractérisé par une entrée et une sortie.

La sortie correspond aux caractéristiques de la plate-forme virtuelle et de l'environnement d'exécution considéré. Par exemple, il existe un environnement d'exécution identifié par *vp-vse[0]* qui offre les fonctionnalités suivantes : un système d'exploitation Kubuntu exécuté sur un processeur de type *x64* avec 5 Go de mémoire.

L'entrée correspond aux caractéristiques sur lesquelles l'environnement est exécuté. Lorsqu'un champ est renseigné par « - » cela signifie que l'environnement d'exécution ou la plate-forme virtuelle concerné n'est pas reconfigurable. Par exemple, l'environnement *vp-vse[0]* n'est pas reconfigurable, alors que l'environnement *vp-vse[3]* peut être reconfiguré pour fournir un environnement *deployable* équipé de 2 processeurs de type *i386* et de 16 Go de mémoire.

On considère que le mot clé *deployable* est un mot du système qui signifie qu'un nœud ne dispose pas d'environnement fixe et qu'il peut être reconfiguré selon les requêtes des utilisateurs. Par exemple, l'environnement *vp-vse[0]* offre en sortie un environnement *Kubuntu* qui ne peut pas être modifié alors que celui *vp-vse[1]* offre en sortie un environnement déployable qu'il est possible de reconfigurer.

Identifiant	Entrée	Sortie
vp-vse[0]	env=-, cpu=-, mem=-	env=Kubuntu, cpu=1(x64), mem=5(Go)
vp-vse[1]	env=-, cpu=-, mem=-	env=*déployable*, cpu=1(x64), mem=5(Go)
vp-vse[2]	env=-, cpu=-, mem=-	env=*déployable*, cpu=1(x64), mem=16(Go)
vp-vse[3]	env=*déployable*, cpu=2(i386), mem=16(Go)	env=Fedora, cpu=2(i386), mem=16(Go)
vp-vse[4]	env=-, cpu=-, mem=-	env=*déployable*, cpu=1(x64), mem=16(Go)

TABLE 4.2 – Structure de données utilisée pour la description des plates-formes virtuelles et des environnements d'exécution (représentation du parchemin *VP/VSE existant* de la figure 4.10)

4.6.2.3 Liste des outils disponibles

Si aucune plate-forme virtuelle ou environnement d'exécution n'est immédiatement disponible sur l'infrastructure matérielle, le module de planification fait une requête auprès du gestionnaire d'outils afin d'obtenir une liste d'outils (gestionnaire de ressources et d'environnement) pouvant répondre à la requête (parchemin *Outils paquets* et *Outils ressources* de la figure 4.10).

Les outils de cette liste sont décrits selon le raffinement de la théorie de Goldberg que nous proposons dans la section 4.4. La structure de données que nous utilisons est similaire à celle utilisée pour la structure de données des plates-formes virtuelles et des environnements d'exécution : un outil est caractérisé par des éléments d'entrée et de sortie.

Parmi les trois caractéristiques que nous étudions (*env*, *cpu* et *mem*), nous qualifions l'attribut d'environnement *env* de type état, le nombre de processeurs est un attribut de capacité, le type de processeur est un attribut de fonctionnalité et la taille de la mémoire est un attribut de capacité.

Le tableau 4.3 présente un exemple de liste d'outils. Par exemple, *outil[0]* est un outil faisant de l'identité au sens que nous décrivons dans la section 4.4, c'est-à-dire, qu'il assure que d'un niveau n (entrée) à un niveau $n+1$ (sortie) les ensembles de capacités, de fonctionnalités et d'états ne varient pas. Ainsi, cet outil permet à partir d'une ressource *déployable* disposant de z processeurs de type $x64$ avec y *Go* de mémoire de fournir un environnement *Debian* avec z processeurs de type $x64$ et y *Go* de mémoire.

Un autre exemple est de prendre l'*outil[1]* qui est un outil de partitionnement. Par exemple, si une ressource matérielle propose 5 Go de mémoire et qu'en sortie, la requête veut 2,5 Go de mémoire, le système va ajuster la variable interne n tel que $n = 1/2$.

4.6.2.4 Algorithme de planification

L'algorithme de planification est en charge, en fonction de la requête, de la liste des plates-formes virtuelles et environnements d'exécution disponibles sur l'infrastructure matérielle et de la liste des outils disponibles, de construire le plan. Pour ce faire, l'algorithme que nous concevons cherche toutes les combinaisons valides qu'il est possible de faire entre la requête, les outils et l'infrastructure matérielle. Une combinaison valide

Identifiant	Entrée	Sortie	Type
outil[0]	env=déployable, cpu=z(x64), mem=y(Go)	env=Debian, cpu=z(x64), mem=y(Go)	Identité (Debian)
outil[1]	env=Debian, cpu=n*z(x64), mem=n*y(Go)	env=m*Debian, cpu=z(x64), mem=y(Go)	Partitionnement, Debian-container lxc ($n < 1$)
outil[2]	env=m*déployable, cpu=n*z(x64), mem=n*y(Go)	env=Debian, cpu=z(x64), mem=y(Go)	Agrégation, Kerrighed ($n > 1$)
outil[3]	env=Debian, cpu=i386, mem=n*y(Go)	env=déployable, cpu=x64, mem=y(Go)	Emulation, QEMU avec q_action(x64)=i386 et $n > 0$
outil[4]	env=déployable, cpu=z(SPARC), mem=y(Go)	env=Solaris, cpu=z(SPARC), mem=y(Go)	Identité
outil[5]	env=Debian, cpu=x64, mem=n*y(Go)	env=déployable, cpu=SPARC, mem=y(Go)	Emulation, QEMU avec q_action(SPARC)=x64 et $n > 0$

TABLE 4.3 – Exemple de catalogue d'outils de type gestionnaire de ressources (parchemins *Outils paquets* et *Outils ressources* de la figure 4.10)

est une combinaison adéquate des *entrées* et des *sorties* des *outils* sur l'infrastructure matérielle et qui satisfait la requête initiale.

4.6.3 Découverte, allocation et configuration des ressources

Le service de découverte, allocation et configuration des ressources utilise principalement les contributions mises en œuvre dans Grillade et Saline :

Service de découverte des ressources Le service de découverte des ressources utilise celui d'XtreemOS présenté dans la section 3.2.4.

Allocation et configuration des ressources de l'infrastructure matérielle L'allocation et la configuration des plates-formes virtuelles sont réalisées grâce aux mécanismes de Grillade et de Saline (agrégation des ressources avec LinuxSSI, partitionnement et émulation avec Saline/QEMU, extension avec des ressources d'une centrale numérique, XtreemFS).

Configuration des environnements d'exécution virtuels La configuration des environnements d'exécution virtuels est pour l'instant effectuée en utilisant spécifiquement les paquets Mandriva (la version d'XtreemOS que nous utilisons est fondée sur Mandriva). Nous travaillons à utiliser des mécanismes plus génériques, comme par exemple OSCAR, permettant de configurer de manière transparente différents types d'environnement d'exécution (par exemple Debian, RedHat).

4.6.4 La boîte à outils

La boîte à outils est constituée d'outils pour la gestion des plates-formes virtuelles et des environnements d'exécution :

Outils pour la gestion des plates-formes virtuelles Pour la gestion des plates-formes virtuelles, Tropicbird dispose des fonctionnalités de Grillade et Saline qui exploitent des ressources virtuelles de type QEMU/KVM à la manière des centrales numériques sur une infrastructure distribuée sur plusieurs sites. De plus, Tropicbird en s'appuyant sur Grillade dispose de fonctionnalités d'extension d'une infrastructure matérielle avec des ressources d'une centrale numérique. Qualitativement, on comprend bien que, plus les outils sont nombreux, plus la flexibilité qu'il est possible d'avoir sur l'infrastructure matérielle est grande.

Outils pour la gestion des environnements d'exécution virtuels Concernant la gestion des environnements d'exécution virtuels, pour l'instant notre prototype n'utilise que des catalogues de paquets de type Mandriva. Nous comptons lever cette limitation en travaillant sur la configuration dynamique des environnements d'exécution en utilisant OSCAR. En effet, OSCAR, par l'intermédiaire de catalogue de paquets gère différentes distributions comme CentOS, Fedora, Debian. A partir des besoins en environnement d'exécution fournis par le module de conciliation, OSCAR crée une image de référence qui est mise en œuvre sur la plate-forme virtuelle selon la distribution choisie et contenant tous les paquets nécessaires ainsi que leurs dépendances.

4.7 Cas d'utilisation

Afin d'illustrer les capacités de la méthode que l'on propose, nous présentons un cas d'utilisation comme exemple.

Un utilisateur veut exécuter une application (*MonApplication*) avec une bibliothèque spécifique (*Mabibliothèque*) qui n'est disponible que sur des systèmes de type Solaris avec un minimum de 32 Go de mémoire et une architecture processeur de type SPARC. L'utilisateur a l'habitude d'exécuter son application sur un ordinateur dédié disposant de *Solaris* avec 32 Go de mémoire et un processeur SPARC.

Cependant, si cette ressource est indisponible et que les seules ressources disponibles sont deux ordinateurs équipés d'un processeur *x64* et munis de 16 Go de mémoire chacun, alors l'utilisateur est confronté à trois choix : (1) attendre qu'un ordinateur Solaris avec les caractéristiques adéquates soit de nouveau disponible, (2) modifier l'application ainsi que les bibliothèques afin de l'adapter au nouvel environnement, (3) utiliser un système de configuration des ressources intelligent, comme Tropicbird, afin de tirer profit des quatre ordinateurs présents dans l'infrastructure sans avoir à modifier l'application. Dans notre exemple, on considère que l'utilisateur prend la décision (3).

L'utilisateur formule la requête *requête[0]* présentée dans le tableau 4.1. On considère que les ressources et les outils disponibles sont ceux décrits dans les tableaux 4.2 et 4.3. Après action du module de planification, on obtient l'élaboration du plan suivant qui est représenté sur la figure 4.13 :

1. utilisation des ressources *vp-vse[2]* et *vp-vse[4]*,
2. utilisation de l'*outil[2]* pour l'installation et la configuration du système d'agrégation Kerrighed,
3. utilisation de l'*outil[5]* pour l'installation et la configuration du système d'émulation QEMU,
4. utilisation de l'*outil[4]* pour déployer l'environnement Solaris.

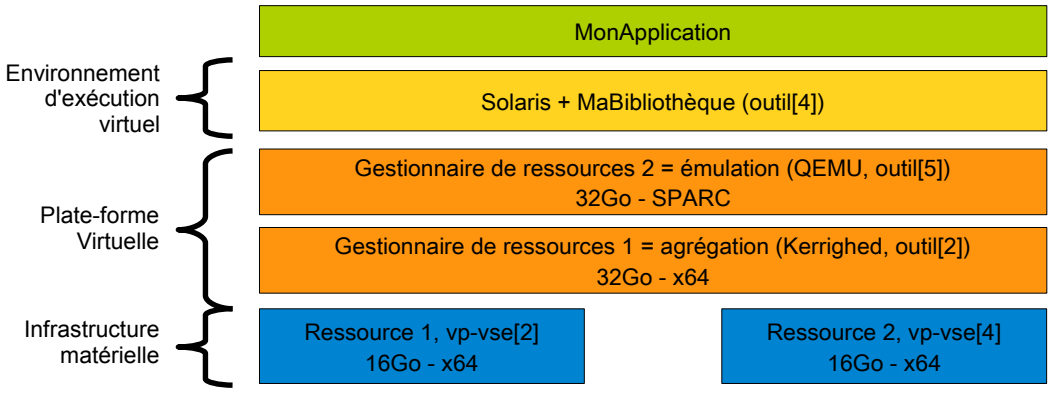


FIGURE 4.13 – Cas d'utilisation

4.8 Travaux apparentés

Comme nous l'avons présenté dans la section 1.3, les gestionnaires de ressources orientent l'utilisation de l'infrastructure qu'ils gèrent selon des axes de flexibilité.

Les systèmes à exécution par lots sont généralement utilisés pour gérer des grappes. Ces systèmes attribuent les tâches sur les nœuds pour une durée déterminée [123]. Torque [31], SGE [28], OAR [56] sont des exemples de système à exécutions par lots. Ils offrent un faible degré de personnalisation des environnements d'exécution et sont orientés vers l'exécution de tâches non-interactives.

L'alliance Globus a introduit le concept de *workspace service* [112] similaire au concept de plate-forme virtuelle que l'on présente dans ce chapitre. Cependant, le concept *workspace service* ne prend pas en compte la gestion des systèmes informatiques depuis les ressources matérielles jusqu'à l'application de l'utilisateur.

Plus récemment, des projets libres de gestion des centrales numériques sont apparus. Nimbus [111] et Eucalyptus [135] en sont des exemples. Ces systèmes permettent la personnalisation des environnements d'exécution en s'affranchissant de l'infrastructure matérielle. Cependant, par rapport au sens général que nous donnons aux plates-formes virtuelles, les centrales numériques ne gèrent que des machines virtuelles ce qui borne la plate-forme virtuelle à être une machine ou une collection de machines virtuelles.

Du point de vue de l'infrastructure matérielle, des institutions se sont fixées pour objectif de fournir des environnements entièrement configurables en laissant à l'utilisateur la possibilité de déployer la pile logicielle qu'il désire sur les nœuds. ALADDIN-G5K est un exemple d'infrastructure de ce type [53]. Cette infrastructure est destinée aux chercheurs qui peuvent contrôler et définir toute la pile logicielle de leur environnement d'exécution. Des outils de la suite *Kadeploy* permettent de gérer ces nœuds de manière flexible en spécifiant par exemple le système d'exploitation à déployer. Cependant, il n'est pas encore possible de définir une description de type plate-forme virtuelle et environnement d'exécution. Les utilisateurs sont contraints de suivre l'évolution des outils de la plate-forme et de l'infrastructure matérielle pour mettre à jour leurs scripts ou leurs applications.

Plusieurs langages de description des applications en terme de logiciel et de matériel ont été proposés. Par exemple, JSDL (*Job Description Language (JSDL)* [44]) et son extension pour les applications parallèles [151] permettent de décrire des applications non-interactives. Cette description est utilisée par le gestionnaire de ressources au moment du déploiement de l'application.

Architecture Description Language (ADL) [93, 121] permet de décrire une application indépendamment de son environnement d'exécution. Pour cela, leurs travaux s'appuient sur une forte séparation entre la plate-forme d'exécution et l'environnement d'exécution. Ces travaux guident nos propositions de description des plates-formes virtuelles et d'environnement d'exécution virtuels.

Enfin, avec l'émergence des centrales numériques, différents standard ont vu le jour. Ces standards sont orientés vers la gestion des infrastructures et des cycles de vie des logiciels. *Configuration Description, Deployment, and Life-cycle Management Specification (CDDLM)* [165] et *Solution Deployment Descriptor (SDD)* [137] en sont des exemples et nos travaux doivent le plus possible s'appuyer sur ces standards.

4.9 Conclusion

Dans ce chapitre nous avons proposé une approche pour obtenir plus de flexibilité dans la gestion et l'utilisation des infrastructures informatiques en découplant la vue des ressources logiques de l'infrastructure matérielle. Nous proposons de réaliser ce découplage en introduisant la notion de plate-forme virtuelle et d'environnement d'exécution virtuel. Une plate-forme virtuelle correspond à la vue des ressources pour l'utilisateur. La plate-forme virtuelle est mise en œuvre grâce à des outils (gestionnaire de ressources) sur une infrastructure matérielle. De plus, nous utilisons le concept d'environnement d'exécution virtuel. Un environnement d'exécution virtuel définit les paquets et les dépendances nécessaires pour l'exécution de l'application de l'utilisateur.

Pour « construire » une plate-forme virtuelle sur une infrastructure matérielle, il est nécessaire de combiner les outils offrant de la flexibilité selon certaines règles. C'est ainsi

que nous proposons un formalisme de description de ces outils. Ce formalisme est fondé sur la théorie de Goldberg traitant des systèmes de virtualisation et l'étend à une large gamme de systèmes informatiques.

Cette approche prend en compte les contraintes des utilisateurs et les politiques des administrateurs et les met en œuvre sur les ressources disponibles de l'infrastructure matérielle (grappe, grille et/ou centrale numérique). Nous proposons une utilisation de la *flexibilité*, dans laquelle les possibilités de configuration et reconfiguration d'un environnement (matériel ou virtuel) sont un élément clé permettant de gérer plus simplement les systèmes distribués, non seulement pour les environnements d'exécution comme c'est généralement le cas, mais aussi pour le matériel. Dans ce sens, nous argumentons en faveur de nouveaux concepts liés à l'émulation et la virtualisation (abstraction, identité, partitionnement, agrégation) parmi lesquels chaque élément composant un environnement informatique (matériel ou virtuel) peut être défini et mis en œuvre sans faire d'hypothèse sur les ressources matérielles et les ressources virtuelles existant sur l'infrastructure informatique sous-jacente.

Notre approche se fonde sur la conciliation des besoins des utilisateurs avec les politiques définies par les administrateurs. Cela permet aux utilisateurs de définir des environnements matériels et logiciels correspondant aux besoins de leurs applications sans compromettre les politiques fixées par les administrateurs.

Nous avons présenté la conception et des éléments de mise en œuvre de notre prototype, Tropicbird, qui est fondé sur les deux premières contributions présentées dans ce document : Saline et Grillade. Nous avons mis en œuvre une première version du module de planification et l'avons validé en formulant une requête de plate-forme et d'environnement d'exécution virtuels souhaités, simple (ne demandant pas d'agrégation de ressources matérielles). Le module de planification gère les caractéristiques d'environnement, de processeurs et de mémoire (*env*, *cpu* et *mem*) de la plate-forme virtuelle et de l'environnement d'exécution virtuel.

L'objectif de Tropicbird est de pouvoir configurer et reconfigurer de manière transparente les environnements d'exécution ainsi que les infrastructures matérielles. L'architecture de notre prototype est conçue de manière modulaire en permettant ainsi l'interaction avec d'autres gestionnaires de ressources comme par exemple des centrales numériques lors de l'extension de l'infrastructure matérielle.

Nos travaux futurs continueront d'explorer et d'approfondir notre raffinement de la théorie de Goldberg tout en finalisant l'intégration de notre prototype. De plus, il serait intéressant d'étudier de quelle manière le module de configuration des ressources pourrait être lui-même configuré par des outils fournis par la boîte à outils de gestionnaire de ressources. En effet, dans la vision actuelle de Tropicbird, le module de configuration contient un ensemble d'outils préinstallés et prêt à l'emploi sur l'infrastructure matérielle ciblée. On pourrait avoir un modèle dans lequel, ce module de configuration s'autoconfigurerait en fonction des gestionnaires de ressources qu'il aurait à installer pour la construction des plates-formes virtuelles.

Tropicbird est par construction évolutif. Il gère les outils (gestionnaires de ressources) et les déploie pour construire les plates-formes virtuelles désirées. Cela signifie d'une part, que plus le nombre d'outils est grand, plus la flexibilité et les possibilités de construction des plates-formes virtuelles seront grandes. D'autre part, Tropicbird peut évoluer « naturellement » et utiliser de nouveaux outils disponible sur le marché.

Pour conclure, nous pensons que les concepts de flexibilité architecturant Tropicbird sont très important pour l'émergence du EaaS (*Everything as a Service*). L'EaaS est un domaine dans lequel tous les constituants d'un système informatique sont considérés comme

des services et peuvent être configurés et reconfigurés à la demande. Ces configurations et reconfiguration se font selon les politiques définies par les administrateurs. Le système est en charge, en accord avec les politiques, de construire la plate-forme virtuelle désirée par l'utilisateur sur l'infrastructure matérielle à l'aide des outils de configuration des ressources et des environnements d'exécution.

Conclusion

Dans cette thèse, nous nous sommes intéressés à la flexibilité dans la gestion et l'utilisation des infrastructures informatiques distribuées de type grappe, grille et centrale numérique (*cloud*). Les différents systèmes de gestion des ressources informatiques comme les systèmes d'exploitation, les hyperviseurs, les systèmes à image unique apportent différents axes de flexibilité dans la gestion des ressources matérielles. Par exemple un hyperviseur permet d'exécuter des machines virtuelles qui sont dissociées des ressources matérielles. Un système à image unique offre l'illusion d'une machine multi-processeur au-dessus d'une grappe. Nos travaux se sont portés sur l'exploitation flexible des infrastructures informatiques distribuées pour les administrateurs et les utilisateurs. Les administrateurs mettent en place des systèmes de gestion des ressources et des politiques d'usage des infrastructures. Les utilisateurs bénéficient de la latitude offerte par les axes de flexibilité des gestionnaires de ressources de l'infrastructure informatique dans le respect des politiques définies par les administrateurs pour déployer leurs applications.

1 Contributions

Nos travaux traitant de la flexibilité pour la gestion et l'utilisation d'infrastructures informatiques distribuées s'articulent autour de trois contributions principales. Tout d'abord nous nous sommes concentrés sur la conception, la mise en œuvre et l'évaluation d'un système de gestion flexible de collections de machines virtuelles sur des infrastructures distribuées de type grille. Puis, nous avons conçu, mis en œuvre et évalué un système de gestion d'infrastructure distribuée exploitant les axes de flexibilité offerts dans les grilles et centrales numériques. Enfin nous avons conçu un système permettant la dissociation complète entre les ressources fournies à l'utilisateur et l'infrastructure matérielle gérée par les administrateurs.

1.1 Saline : gestion transparente des collections de machines virtuelles à l'échelle de la grille

Tout d'abord nous nous sommes intéressés à la flexibilité offerte par les systèmes de virtualisation sur une infrastructure informatique de type grille. Nous avons conçu un système de gestion des collections de machines virtuelles pour l'exécution des tâches interruptibles dans une grille. Ce système est conçu pour s'intégrer à l'infrastructure informatique de manière transparente, c'est-à-dire sans modification de la configuration des gestionnaires de ressources. Grâce aux mécanismes de virtualisation, ce système permet à une très large gamme d'applications distribuées d'être exécutée en mode interruptible. Pour les administrateurs, les tâches interruptibles permettent d'augmenter le taux d'utilisation des ressources.

Nous avons proposé la mise en œuvre d'un prototype nommé Saline [82, 84, 85, 86].

Saline gère le cycle de vie des collections de machines virtuelles (création, exécution, surveillance, déplacement d'un site à un autre, sauvegarde et restauration). Lors de l'initialisation des collections de machines virtuelles, Saline configure leurs adresses MAC et IP de manière à éviter les conflits avec les autres collections de machines virtuelles en cours d'exécution et avec les ressources de l'infrastructure matérielle. Pour la sauvegarde des images des machines virtuelles, Saline propose un algorithme de gestion efficace permettant la copie efficace de ces images, des nœuds d'exécution vers un nœud maître. Cet algorithme permet de réguler le flux de copies vers le nœud maître et de libérer une majorité de nœuds d'exécution le plus rapidement possible. De plus, l'architecture modulaire de Saline lui permet de s'interfacer facilement avec des gestionnaires de placement des machines virtuelles comme Entropy, pour faire de la consolidation de serveur.

Nous avons expérimenté Saline sur Grid'5000 en l'interfaçant avec le gestionnaire de ressources OAR. Nous avons validé Saline avec et sans gestionnaire de placement intelligent des machines virtuelles. Sans gestionnaire de placement intelligent des machines virtuelles, Saline applique une politique de placement des machines virtuelles sur les nœuds en *round-robin*. Avec le gestionnaire de placement intelligent Entropy, Saline déploie les machines virtuelles et laisse Entropy les placer selon des critères de charge processeur et de taille mémoire sur les nœuds. Les évaluations que nous avons menées montrent que les objectifs initiaux sont atteints : Saline gère des collections de machines virtuelles sur une infrastructure de type grille sans modification du gestionnaire de ressources. De plus, les expérimentations effectuées ont permis de valider les choix de mise en œuvre de Saline : la technique de la copie sur écriture réduit considérablement la taille des images des machines virtuelles dans une large majorité de cas, l'utilitaire TakTuk permet un déploiement efficace des machines virtuelles, l'algorithme mis en œuvre dans Saline pour rapatrier les machines virtuelles sauvegardées sur un nœud de stockage est efficace en évitant un goulot d'étranglement sur ce dernier.

1.2 Grillade : combiner grilles et centrales numériques pour une flexibilité accrue

Notre seconde contribution est fondée sur l'identification que nous avons faite des axes de flexibilité offerts par les systèmes de gestion des grappes, grilles et centrales numériques [63, 83, 128, 130, 164]. Trois axes de flexibilité ont particulièrement retenu notre attention : l'extension d'une infrastructure matérielle avec des ressources fournies par une centrale numérique, la construction d'une centrale numérique utilisant les ressources matérielles fournies par plusieurs domaines d'administration et l'utilisation d'un système à image unique pour l'agrégation de nœuds (création d'une ressource virtuelle dont les capacités sont celles de nœuds agrégés). Cette étude a donné lieu à la conception et la mise en œuvre de Grillade, un système fondé sur le système de grille XtreemOS, intégrant chacun des axes de flexibilité précédemment cités. Grillade peut être décomposé en deux sous-systèmes : Grillade-Ext permet l'extension d'une infrastructure de grille avec des ressources fournies par une centrale numérique et Grillade-CN fournit un service de type IaaS sur la grille en combinant l'utilisation d'un système à image unique pour accroître la flexibilité de l'infrastructure matérielle et le catalogue de machines virtuelles proposé aux clients.

Extension d'une grille avec des machines virtuelles fournies par une centrale numérique Les mécanismes d'extension de Grillade-Ext permettent d'étendre une infrastructure de grille avec des machines virtuelles fournies par une centrale numérique, par exemple, lorsqu'il n'y a plus suffisamment de ressources disponibles.

Ces mécanismes offrent plus de flexibilité pour l'infrastructure matérielle en permettant de faire varier sa taille. Une fois l'extension réalisée, l'exécution des tâches sur les machines virtuelles se fait de manière transparente.

Grillade-Ext a été évalué avec des ressources fournies par une centrale numérique gérée par le système Nimbus. Nos validations expérimentales ont été réalisées sur deux sites de Grid'5000. En déployant une centrale numérique Nimbus sur le site de Rennes et le système Grillade sur le site de Sophia, nous avons validé la création et l'intégration de machines virtuelles dans Grillade. De plus, grâce aux commandes de déploiement et de configuration des machines virtuelles qui s'effectuent en parallèle, le temps total requis pour l'intégration des machines virtuelles reste constant quel que soit le nombre de machines virtuelles.

Centrale numérique utilisant les ressources réparties sur différents domaines d'administration Grillade-CN exploite les fonctionnalités de haut niveau offertes par XtreemOS pour construire une centrale numérique utilisant les ressources de plusieurs domaines d'administration.

- Le système de fichiers distribué XtreemFS qui est utilisé pour stocker les images des machines virtuelles permet de les rendre accessibles depuis tous les sites. Cela facilite la conception d'une centrale numérique répartie sur plusieurs domaines d'administration.
- Les organisations virtuelles sont utilisées pour définir des classes d'utilisateurs et de ressources. Les administrateurs de la centrale numérique utilisent les mécanismes d'organisations virtuelles pour gérer le partage des ressources des différents domaines d'administration entre les classes de clients : ils assignent les ressources et les utilisateurs dans les organisations virtuelles.

Grillade-CN bénéficie à la fois des fonctionnalités offertes par XtreemOS et des fonctionnalités que nous avons conçues et mises en œuvre pour la création de centrales numériques. La gestion des machines virtuelles s'appuie sur les mécanismes mis en œuvre dans Saline. Grillade-CN offre aux utilisateurs une double flexibilité. Les utilisateurs peuvent déployer leurs applications dans les environnements préconfigurés d'XtreemOS ou définir leurs environnements d'exécution personnalisés qui seront déployés dans des machines virtuelles adaptées à leurs besoins. Cette axe de flexibilité ouvre la voix à la notion de cohabitation entre les utilisateurs de grilles et ceux de centrales numériques sur une même infrastructure informatique distribuée.

Grillade-CN a été déployé sur Grid'5000. Nous avons défini des classes d'utilisateurs. Ces classes d'utilisateurs correspondent à des organisations virtuelles dans lesquelles des ressources réparties sur les différents domaines d'administration ont été assignées. Les machines virtuelles d'un utilisateur appartenant à une classe sont créées sur les nœuds appartenant à cette classe.

Agrégation des nœuds pour plus de flexibilité dans la gestion des ressources matérielles En agrégeant des nœuds à la demande, il est possible d'offrir aux utilisateurs une plus large gamme de ressources matérielles (en terme de taille de la mémoire et de nombre de processeurs) que celles disponibles dans l'infrastructure distribuée. Un système à image unique est une technologie qui permet d'agréger des nœuds.

La technologie de système à image unique, LinuxSSI, est utilisée par Grillade-CN pour créer des nœuds multi-cœurs à la demande. Cette technologie est utile lorsque l'utilisateur fait la requête d'une machine virtuelle dont les capacités sont plus grandes que celles du « meilleur » nœud disponible de la centrale numérique. Il est alors possible d'agréger des nœuds afin de créer un nœud multi-cœur virtuel dont les

capacités sont celles de l'agrégation des nœuds le constituant. Une fois l'agrégation effectuée, la création d'une machine virtuelle se fait de manière transparente sur ce nœud multi-cœur virtuel.

Nous avons évalué et validé le comportement de ce mécanisme de création automatique de nœuds multi-cœurs sur Grid'5000. Les résultats montrent que, grâce aux actions effectuées en parallèle, le temps d'agrégation ou de désagrégation des nœuds est constant quel que soit le nombre de nœuds. De plus, l'impact des performances de l'agrégation des nœuds n'entraîne pas de baisse de performance significative quant à l'utilisation de la mémoire (expérience réalisée avec une application utilisant la mémoire de manière intensive).

1.3 Tropicbird : vers plus de flexibilité dans les systèmes informatiques

Notre troisième contribution s'intéresse à la différenciation complète entre la vue qu'a l'utilisateur des ressources nécessaires à l'exécution de son application et l'infrastructure matérielle et logicielle disponible, en introduisant le concept de plate-forme virtuelle [87, 88, 89, 90, 91, 92, 169]. Une plate-forme virtuelle est une vue des ressources correspondant aux besoins d'un utilisateur et qui est construite sur des ressources matérielles par le système Tropicbird.

Pour décrire ce concept nous avons repris la formalisation de la virtualisation proposée par Goldberg dans les années 1970 et étendu cette théorie à une large gamme de systèmes informatiques (système d'exploitation, système à image unique, émulateur, etc.). De cette manière il est possible de classer les systèmes de gestion des ressources en cinq catégories : émulation, émulation-abstraction, virtualisation-identité, virtualisation-partitionnement, virtualisation-agrégation. Cette classification permet de décrire les gestionnaires de ressources d'une manière uniforme et permet de décrire des règles de combinaison entre ces différents gestionnaires de ressources. La combinaison de ces systèmes de gestion permet de mettre en œuvre les plate-formes virtuelles. Le prototype Tropicbird, fondé sur Grillade, est en cours de réalisation. Il fait suite à nos deux premières contributions : la flexibilité ultime qui offre une complète indépendance entre les ressources et la plate-forme virtuelle proposée à l'utilisateur. Ainsi, si l'infrastructure ne dispose pas des environnements d'exécution adéquats pour une application donnée, le système va construire automatiquement l'environnement requis en utilisant les gestionnaires de ressources appropriés pour créer la plate-forme virtuelle, et les gestionnaires de paquets pour configurer l'environnement d'exécution.

Enfin, par conception Tropicbird est évolutif car il utilise les gestionnaires de ressources et de paquets existants pour construire les plates-formes virtuelles et les environnements d'exécution. Cela signifie qu'avec une description adéquate, Tropicbird peut intégrer dans sa base de gestionnaires de ressources et de paquets, les nouveaux outils du marché.

Le prototype Tropicbird est en cours de développement. Saline et Grillade en constituent les fondations. Nous avons mis en œuvre une première version du module de planification que nous avons validé à partir de l'exemple décrit dans le chapitre 4.

2 Perspectives

Les travaux que nous avons menés tout au long de cette thèse ouvrent plusieurs perspectives qu'il serait intéressant de traiter à court et à long terme.

2.1 Indépendance par rapport à l'architecture d'une grille

Dans sa mise en œuvre actuelle, Saline utilisé sur Grid'5000 s'interface avec le gestionnaire de ressources pour grille OAR et la suite de configuration des nœuds Kadeploy. Il serait intéressant d'utiliser Saline avec d'autres outils de configuration des grappes. Par exemple, la combinaison de Saline et d'OSCAR donnerait la possibilité de manipuler la pile logicielle complète d'une grappe. En effet, OSCAR propose des mécanismes de configuration des grappes. Deux axes de flexibilité peuvent être distingués :

- OSCAR gère pratiquement toutes les distributions de type Linux et peut installer et configurer les environnements de virtualisation nécessaires à Saline pour l'exécution des machines virtuelles sur la grappe. OSCAR permettrait à Saline d'être indépendant par rapport à l'architecture des nœuds.
- Dans sa mise en œuvre actuelle, Saline déploie des images de machines virtuelles déjà pré-configurées. OSCAR pourrait être utilisé pour configurer les machines virtuelles déployées par Saline et ainsi offrir une plus grande flexibilité par rapport à la personnalisation des environnements d'exécution.

Il est aussi important de réfléchir aux mécanismes nécessaires pour s'affranchir de la limitation actuelle de Saline, selon laquelle toutes les machines virtuelles d'une même collection doivent être exécutées sur un même site. Cela permettrait d'exécuter une collection de machines virtuelles s'étendant sur des ressources matérielles de plusieurs sites. Une solution pourrait être d'utiliser des techniques de virtualisation du réseau comme avec VPN, Vine ou IPOP. Ces travaux et ces pistes de réflexion ouvrent la voie au projet Européen Contrail [8] faisant suite à XtreemOS. À travers le concept d'architecture virtuelle, Contrail étend la gestion d'une collection de machines virtuelles distribuées sur un site (grappe virtuelle) à celle de collection de machines virtuelles distribuées sur différents sites (grille virtuelle).

2.2 Centrale numérique répartie sur différents sites

Les travaux relatifs au système Grillade sont naturellement liés à ceux d'XtreemOS. Il serait intéressant de mieux intégrer Grillade à XtreemOS en mettant en œuvre la solution « idéale » proposée dans le chapitre 3. Une solution intégrée permettrait de rendre les mécanismes conçus plus robustes et réactifs dans leurs prises de décision (cela éviterait par exemple qu'une action prise par un module spécifique à Grillade soit contredite par XtreemOS). Leur maintenance en serait également facilitée.

Dans cette thèse, nous ne nous sommes pas intéressés à la facturation de l'usage des ressources, comme cela est fait dans les centrales numériques commerciales. Il serait intéressant d'étendre le module de comptabilité du taux d'utilisation des ressources d'XtreemOS aux collections de machines virtuelles. Cela permettrait de facturer l'utilisation des ressources aux clients, à la manière des centrales numériques.

De plus, les expériences que nous avons réalisées doivent être consolidées à plus grande échelle. Nous projetons également de valider les mécanismes d'extension avec des centrales numériques publiques comme Amazon EC2 ou privées comme OpenNebula ou Nimbus. Le but est de pouvoir déployer une application distribuée sur différents sites gérés par Grillade et sur des ressources fournies par une ou plusieurs centrales numériques en utilisant les interfaces standard d'accès aux centrales numériques.

Dans nos travaux, nous avons mis l'accent sur les mécanismes de gestion des infrastructures distribuées pour y apporter plus de flexibilité. Il serait intéressant de travailler sur la définition de politiques de gestion et d'utilisation des ressources matérielles.

Par exemple, Grillade-Ext permet d'étendre une infrastructure de type grille avec des

ressources fournies par des centrales numériques. Il serait intéressant d'étudier des politiques de choix des centrales numériques. Ces politiques pourraient, par exemple, être fondées sur la tarification proposée par les centrales numériques.

Concernant Grillade-CN, il serait intéressant d'étudier les politiques d'allocations des ressources matérielles à l'échelle des sites. Par exemple, un administrateur pourrait désirer mettre en place des politiques d'économie d'énergie sur son site.

Enfin, un autre axe de politique qu'il serait intéressant d'étudier est celui de la définition de politique à l'échelle de l'ensemble des sites. Prenons l'exemple d'une infrastructure distribuée sur plusieurs sites répartis sur la planète et considérons qu'un site dépense moins d'énergie de nuit, parce que les températures extérieures sont plus fraîches et que les climatiseurs sont moins sollicités. On pourrait songer à une politique d'économie d'énergie entre les différents sites dans lesquels les tâches à exécuter seraient placées en priorité sur les sites où il fait nuit. Selon la charge de travail des différents sites, les tâches pourraient être déplacées lors de la levée du jour.

Enfin, les travaux menés dans Grillade ont servi à alimenter la réflexion menée lors du montage du projet Contrail. Contrail a pour objectif de briser les frontières qu'il peut y avoir entre les centrales numériques aussi bien pour les administrateurs que pour les utilisateurs. Pour les administrateurs, Contrail devrait permettre à une centrale numérique de s'étendre sur des ressources fournies par d'autres centrales numériques. Pour les utilisateurs, l'objectif est de les rendre indépendants par rapport aux centrales numériques en permettant la migration de leurs applications d'une centrale numérique à une autre, par exemple.

2.3 La reconfiguration d'une infrastructure matérielle distribuée

Nos travaux liés à l'extension de la théorie de Goldberg nous ont permis de décrire un cadre formel permettant de classifier une large gamme de systèmes informatiques. Ces systèmes informatiques peuvent être combinés et installés sur les infrastructures matérielles pour créer des plates-formes virtuelles.

A court terme, une première étape sera de finaliser la mise en œuvre du système Tropicbird : finaliser le module de planification et l'interconnecter avec les autres modules. A plus long terme, il faudrait l'évaluer et valider son comportement en réalisant une campagne d'expérimentation. Ces expérimentations se concentreront sur l'aspect qualitatif des configurations et les performances. En effet, dans les travaux que nous présentons, nous prenons le parti de faire des concessions sur les performances d'exécution d'applications pour obtenir plus de flexibilité. Il sera intéressant d'étudier la limite à partir de laquelle une dégradation des performances serait trop forte, limite à partir de laquelle, la construction de la plate-forme virtuelle ne serait pas « viable ».

2.4 Plus de flexibilité pour les plates-formes virtuelles

Dans ce document nous avons décrit la notion de plate-forme virtuelle : une vue des ressources qu'a l'utilisateur et qui est indépendante de l'infrastructure sous-jacente. Dans la vision actuelle, les plates-formes virtuelles sont statiques dans le sens où, une fois créées, elles ne peuvent varier : par exemple, dans le cas d'une défaillance matérielle, la plate-forme virtuelle pourrait subir des dysfonctionnements voire être détruite complètement. Trois axes pourraient être étudiés : les plates-formes virtuelles hautement disponibles, les plates-formes virtuelles dynamiques et enfin les plates-formes virtuelles autonomes.

Plate-forme virtuelle hautement disponible Un autre axe de recherche serait d'étudier les mécanismes nécessaires pour rendre les plates-formes virtuelles haute-

ment disponibles. Une plate-forme virtuelle hautement disponible permettrait d'assurer la disponibilité de la plate-forme même en cas de défaillance des ressources matérielles. Pour assurer cette haute disponibilité, des techniques de tolérance aux fautes, de contrôle de l'état de la plate-forme virtuelle et de l'infrastructure matérielle sont à mettre en place.

Plate-forme virtuelle dynamique Une plate-forme virtuelle dynamique est une plate-forme virtuelle dont les caractéristiques peuvent varier au cours du temps (extension ou réduction du nombre de ressources de la plate-forme). Ces techniques d'extension et de réduction pourraient être fondées par exemple, sur les mécanismes d'extension que nous présentons dans Grillade.

Dans le cadre de la plate-forme virtuelle dynamique, il serait intéressant d'étudier les mécanismes de migration nécessaires pour déplacer tout ou partie des éléments constituant la plate-forme virtuelle. Une possibilité de migration des plates-formes virtuelles serait utile pour offrir plus de flexibilité aux administrateurs dans la gestion de leur infrastructure matérielle. Par exemple, un administrateur pourrait, avant de faire une maintenance sur des nœuds, forcer une action de migration des plates-formes virtuelles en cours d'exécution sur ces nœuds. Cette migration doit se faire de manière transparente pour l'utilisateur et pour les applications exécutées sur les plates-formes virtuelles impactées.

Plate-forme virtuelle autonome Notre vision ultime de la flexibilité serait celle d'une plate-forme virtuelle autonome, regroupant les mécanismes des plates-formes virtuelles hautement disponibles et dynamiques. Une fois créée, une plate-forme virtuelle autonome pourrait être déplacée, arrêtée, suspendue, restaurée, répartie sur plusieurs sites. Elle s'adapterait de manière transparente aux ressources disponibles en fonction des contraintes de qualité de service imposées par l'utilisateur et des politiques d'utilisation des ressources définies par les administrateurs.

2.5 Le concept du « tout est service », EaaS

Les contributions proposées dans ce document présentent des mécanismes de gestion des infrastructures pour rendre leur gestion et leur utilisation plus flexible. L'orientation de nos contributions place la flexibilité au centre de trois ensembles : les infrastructures informatiques, les outils (gestionnaire de ressources et d'environnements) et les politiques des administrateurs. Notre vision de la gestion de ces infrastructures va dans le sens du concept émergent du « tout est service » (EaaS) [7, 10].

Tropicbird pose les fondements d'un système de type EaaS : nos travaux proposent un cadre générique permettant de lier et combiner des outils sur une infrastructure matérielle distribuée. Cette combinaison d'outils peut se faire selon des règles définies par le raffinement que nous proposons à la théorie de Goldberg. Bien sûr, plus les outils à disposition du système sont nombreux, et plus les possibilités de combinaison et de création de plate-forme virtuelle le sont également.

Notre vision du EaaS peut se comparer, à une autre échelle, à celle des systèmes électroniques. Dans cette vision, on compare un nœud à une puce FPGA : de la même manière qu'une puce FPGA peut être configurée et reconfigurée à la demande pour offrir des fonctionnalités particulières, un ensemble de nœuds doit pouvoir l'être également de manière transparente à l'échelle d'une infrastructure distribuée sur plusieurs sites. Dans le cas des nœuds, cette reconfiguration se fait avec des outils permettant de construire et gérer le cycle de vie des plates-formes virtuelles autonomes dans le respect des politiques d'utilisation des ressources définies par les administrateurs.

Bibliographie

[1] ALADDIN-G5K : ensuring the development of Grid'5000. http://www.grid5000.fr. 2.2.1.1

[2] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/. 1.2.3.3

[3] AppNexus. http://www.appnexus.com/. 1.2.3.3

[4] BonFIRE project. http://www.bonfire-project.eu/. 3.5

[5] Chapter 2. Debian package management. http://www.debian.org/doc/manuals/debian-reference/ch02.en.html. 4.5.5.1

[6] Chroot. http://www.gnu.org/software/coreutils/manual/coreutils.html#chroot-invocation. 1.2.1.1

[7] Cloud 3.0 : Everything is a Service. http://www.bio-itworld.com/2010/issues/jul-aug/HP.html. 4.9

[8] Contrail : Open Computing Infrastructures for Elastic Services. http://contrail-project.eu/. 4.9

[9] Ethernet Technologies. http://http://docwiki.cisco.com/wiki/Ethernet_Technologies. 1.1.2.1

[10] Everything as a Service : HP's view on the cloud. http://www.hp.com/hpinfo/initiatives/eaas/index.html. 4.9

[11] gLite - Lightweight Middleware for Grid Computing. http://glite.cern.ch/. 1.1.4.2

[12] IANA : Internet Assigned Numbers Authority. http://www.iana.org/. 2.3.3.4

[13] IEEE : Institute of Electrical and Electronics Engineers. http://www.ieee.org/. 2.3.3.4

[14] Infiniband. http://www.infinibandta.org. 1.1.2.1

[15] JeOS Ubuntu Documentation. https://help.ubuntu.com/8.04/serverguide/C/jeos.html. 2.3.3.2

[16] Legion. http://legion.virginia.edu/index.html. 1.1.4.2

[17] Libvirt Virtualization API. http://libvirt.org/. 2.3.4.1, 2.6.3

[18] Moab Cluster Suite. http://www.clusterresources.com/products/moab-cluster-suite.php. 2.1.2

[19] Myrinet. http://www.myri.com/myrinet/overview. 1.1.2.1

[20] NetApp : Infrastructure IT Flexible. http://www.netapp.com/fr/company/leadership/flexible-it/feature-story-future-with-flexible-it-fr.html. (document)

[21] OCC : Open Cloud Consortium. http://opencloudconsortium.org/. 1.2.3.3

[22] OCCI : Open Cloud Computing Interface. http://www.occi-wg.org/doku.php. 1.2.3.3

[23] OpenSolaris, JeOS : Just enough OS Project. http://hub.opensolaris.org/bin/view/Project+jeos/. 2.3.3.2

[24] OpenVZ. http://wiki.openvz.org/Main_Page. 1.2.1.1, 2.1.2

[25] Oracle Enterprise Linux JeOS. http://www.oracle.com/technology/software/products/virtualization/vm_jeos.html. 2.3.3.2

[26] Oracle VM Template. http://www.oracle.com/technology/products/vm/templates/ovmtb.html. 2.3.3.2

[27] Orange : Flexible Computing. http://www.orange-business.com/fr/entreprise/real-times/solutions-it/infrastructures/flexible-computing/. (document)

[28] SGE : Sun Grid Engine. http://www.sun.com/software/sge. 4.8

[29] Stratuslab. http://stratuslab.eu/doku.php/start. 3.5

[30] Tivoli Workload Scheduler LoadLeveler. http://www-03.ibm.com/systems/software/loadleveler/index.html. 1.1.3.2

[31] Torque Resource Manager. http://www.clusterresources.com/products/torque-resource-manager.php. 1.1.3.2, 4.8

[32] Unicore. http://www.unicore.eu. 1.1.4.2

[33] VMBuilder Ubuntu Documentation. https://help.ubuntu.com/8.04/serverguide/C/ubuntu-vm-builder.html. 2.3.3.2

[34] VMware. http://www.vmware.com. 1.2.1.1

[35] VMware : Cloud Computing. http://www.vmware.com/fr/solutions/cloud-computing/. (document)

[36] Vocabulaire de l'informatique et de l'internet, JORF n°0129 du 6 juin 2010 page 10453 texte n°42 (*Cloud Computing*). http://www.legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000022309303. (document)

[37] XACML : eXtensible Access Control Markup Language (OASIS). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#overview. 4.6.1

[38] XtreemOS : A Linux-based Operating System to support Virtual Organizations for next generation Grids. http://www.xtreemos.eu. 3.2

[39] OpenNebula in EU Initiative to Build a Multi-Site Cloud. http://blog.opennebula.org/?p=793, 2010. 3.5

[40] Marcim Adamski, Gracjan Jankowski, Norbert Meyer, Alvaro Arenas, Brian Matthews, Michael Wilson, Angelos Bilas, Paraskevi Fragopoulou, Vasil Georgiev, Alejandro Hevia et Jorg Platte. Trust and Security in Grids : A State of the Art, May 2008. CoreGRID White Paper WHP-0001. 1.1.4.1

[41] The Globus Alliance. http://www.globus.org. 1.1.4.2, 3.5

[42] Mark Allman, Vern Paxson et William Stevens. RFC : 5681 – TCP Congestion Control. http://tools.ietf.org/html/rfc5681, septembre 2009. 2.3.1

[43] G. M. Amdahl, G. A. Blaauw et Jr F.P. Brooks. Architecture of the IBM System/360. *IBM Journal of Research and Development*, 44(1/2), 1964. 4.2

[44] Ali Anjomshoaa, Fred Brisard, Michel Dresher, Donal Fellows, An Ly, Stephen McGough, Darren Pulsipher et Andreas Savva. Job Submission Description Language (JSDL) Specification, V. 1.0. Rapport technique GFD-R.136, Open Grid Forum, juillet 2008. 1.1.3.2, 4.8

[45] W. W. Rouse Ball. *A short account of the history of mathematics*. Dover Publications, New York, 1960. 1

[46] Amnon Barak et Oren La'adan. The MOSIX multicomputer operating system for high performance cluster computing. *Future Generation Computer Systems*, 13(4-5):361–372, 1998. 1.1.3.2

[47] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt et Andrew Warfield. Xen and the Art of Virtualization. Bolton Landing, New York, USA, October 2003. SOSP'03. 1.2.1.1

[48] Wolfgang Barth. *Nagios : System and Network Monitoring*. No Starch Press, San Francisco, CA, USA, 2nd édition, 2008. 2.3.3.5

[49] Fabrice Bellard. QEMU, a Fast and Portable Dynamic Translator. Anaheim, CA, USA, 2005. USENIX 2005 Annual Technical Conference. 1.2.1.1

[50] Mario Leandro Bertogna, Eduardo Grosclaude, Marcelo Naiouf, Armando Giusti et Emilio Luque. Dynamic on demand virtual clusters in grid. *Dans Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 2008)*, pages 13–22, Berlin, Heidelberg, 2008. Springer-Verlag. 3.5

[51] Havard K. F. Bjerke, Dimitar Shiyachki, Andreas Unterkircher et Irfan Habib. Tools and Techniques for Managing Virtual Machine Images. *Dans Proceedings of the 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC)*, 2008. 2.3.3.2

[52] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic et Wen-King Su. Myrinet : A gigabit-per-second local area network. *IEEE Micro*, 15:29–36, 1995. 1.1.2.1

[53] Raphaël Bolze, Franck Cappello, Eddy Caron, Michel Daydé, Frédéric Desprez, Emmanuel Jeannot, Yvon Jégou, Stephane Lanteri, Julien Leduc, Noredine Melab, Guillaume Mornet, Raymond Namyst, Pascale Primet, Benjamin Quetier, Olivier Richard, El-Ghazali Talbi et Iréa Touche. Grid'5000 : A Large Scale And Highly Reconfigurable Experimental Grid Testbed. *International Journal of High Performance Computing Applications*, 20(4):481–494, 2006. (document), 2.2.1.1, 2.3.3.1, 4.8

[54] Arthur W. Burks. The invention of the universal electronic computer : how the electronic computer revolution began. *Future Gener. Comput. Syst.*, 18:871–892, August 2002. (document)

[55] Egon Börger. High Level System Design and Analysis Using Abstract State Machines. *Dans Applied Formal Methods — FM-Trends 98*, pages 1–43, 1999. 4.1

[56] Nicolas Capit, Georges Da Costa, Yiannis Georgiou, Guillaume Huard, Cyrille Martin, Gregory Mounié, Pierre Neyron et Olivier Richard. A Batch Scheduler With High Level Components. *Dans Cluster computing and Grid 2005 (CCGrid05)*, pages 776–783, Washington, DC, USA, 2005. IEEE Computer Society. 1.1.3.2, 2.2.1.1, 4.8

[57] Eddy Caron et Frederic Desprez. Diet : A scalable toolbox to build network enabled servers on the grid. *International Journal of High Performance Computing Applications*, 20(3):335–352, 2006. 1.1.4.2

[58] G. Chen, K. Malkowski, M. Kandemir et P. Raghavan. Reducing Power with Performance Constraints for Parallel Sparse Applications. *Dans Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Washington, DC, USA, 2005. 1.2.3.2

[59] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt et Andrew Warfield. Live migration of virtual machines. *Dans Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, volume 2 de *NSDI'05*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association. 1.2.3.1

[60] Benoit Claudel, Guillaume Huard et Olivier Richard. TakTuk, adaptive deployment of remote executions. *Dans Proceedings of the 18th ACM international symposium on High performance distributed computing (HPDC)*, pages 91–100, New York, NY, USA, 2009. 2.3.4.1, 2.5.3

[61] XtreemOS Consortium. Design and Specification of a Prototype Service/Resource Discovery System (D3.2.4), décembre 2007. 3.2.4

[62] XtreemOS Consortium. Design and Implementation of Advanced Node-level VO Support Mechanisms (D2.1.5), décembre 2008. 1.2.2.2

[63] XtreemOS consortium. Evaluation of Linux Native Isolation Mechanisms for XtreemOS Flavours (D2.1.6), janvier 2009. (document), 3, 3.3, 4.9

[64] XtreemOS Consortium. Fourth Specification, Design and Architecture of the Security and VO Management Services (D3.5.13), décembre 2009. 3.2.2.4

[65] XtreemOS Consortium. Revised System Architecture (D3.1.7), janvier 2009. 3.2, 3.2.5

[66] XtreemOS Consortium. Final release of highly available and scalable grid services (D.3.2.16), avril 2010. 3.2.4

[67] XtreemOS Consortium. Installing XtreemOS on a Virtual Machine. Rapport technique 6, octobre 2010. 3.1.2, 3.4.2

[68] Massimo Coppola, Yvon Jégou, Brian Matthews, Christine Morin, Luis Pablo Prieto, Óscar David Sánchez, Erica Y. Yang et Haiyan Yu. Virtual Organization Support within a Grid-Wide Operating System. *IEEE Internet Computing*, 12(2):20–28, 2008. 1.1.4.2, 3.2

[69] DMTF. Open Virtualization Format Specification. http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf, janvier 2010. 2.3.3.2

[70] Wesley Emeneker et Dan Stanzione. Increasing Reliability through Dynamic Virtual Clustering. *Dans High Availability and Performance Computing Workshop*, 2006. 1.2.3.2, 2.3.1

[71] Christian Engelmann, Stephen L. Scott, Hong Ong, Geoffroy Vallée et Thomas Naughton. Configurable virtualized system environments for high performance computing. *Dans In Proceedings of the 1st Workshop on System-level Virtualization for High Performance Computing (HPCVirt) 2007, in conjunction with the 2nd ACM SIGOPS European Conference on Computer Systems (EuroSys) 2007*, 2007. 4.5.1

[72] Niels Fallenbeck, Hans-Joachim Picht, Matthew Smith et Bernd Freisleben. Xen and the Art of Cluster Scheduling. *Dans VTDC'06 : Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, page 4, Washington, DC, USA, 2006. IEEE Computer Society. 2.1.2

[73] Renato Figueiredo, Peter A. Dinda et José Fortes. Guest editors' introduction : Resource virtualization renaissance. *Computer*, 38:28–31, 2005. 4.1

[74] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayer et X. Zhang. Virtual Clusters for Grid Communities. *Dans CCGRID '06 : Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 513–520, Washington, DC, USA, 2006. IEEE Computer Society. 3.5

[75] Ian Foster. What is the Grid ? - a three point checklist. *GRIDtoday*, 1(6), juillet 2002. 1.1.4.1

[76] Ian Foster. Globus Toolkit Version 4 : Software for Service-Oriented Systems. *Journal of Computer Science and Technology*, 21(4), 2006. 1.1.4.2, 3.5

[77] Ian Foster et Carl Kesselman. *The Grid : Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1999. 1.1.4.1

[78] Ian Foster, Carl Kesselman, Jeffrey M. Nick et Steven Tuecke. The physiology of the grid : An open grid services architecture for distributed systems integration. 2002. 1.1.4.1

[79] Ian Foster, Carl Kesselman et Steven Tuecke. The Anatomy of the Grid : Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001. 1.1.4.1

[80] Ian Foster, Yong Zhao, Ioan Raicu et Shiyong Lu. Cloud computing and grid computing 360-degree compared. *Dans Grid Computing Environments Workshop, 2008. GCE '08*, novembre 2008. 1.2.3.3

[81] FranceTerme. Tous les termes publiés au *Journal Officiel* par la Commission générale de terminologie et de néologie – Ministère de la Culture et de la Communication. http://franceterme.culture.fr. Ordinateur : Arrêté du 30 novembre 1983. 1.1.1

[82] Jérôme Gallard. Kget+ : An efficient tool for managing VM image snapshots. *Dans Toulouse'2009 (RenPar'19 / SympA'13 / CFSE'7) – Poster Session*, Toulouse France, 2009. (document), 2, 2.3, 4.9

[83] Jérôme Gallard et Adrien Lèbre. Managing Virtual Resources : Fly through the Sky. *ERCIM News*, pages 36–37, octobre 2010. (document), 3, 3.3, 4.9

[84] Jérôme Gallard, Adrien Lebre, Oana Goga et Christine Morin. VMdeploy : Improving Best-Effort Job Management in Grid'5000. Research Report RR-6764, INRIA, décembre 2008. (document), 2, 4.9

[85] Jérôme Gallard, Adrien Lèbre et Christine Morin. Saline : Improving Best-Effort Job Management in Grids. Research Report RR-7055, INRIA, septembre 2009. (document), 2, 4.9

[86] Jérôme Gallard, Adrien Lèbre et Christine Morin. Saline : Improving Best-Effort Job Management in Grids. *Dans PDP 2010 : The 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing – Special Session : Virtualization*, Pisa Italie, 2010. (document), 2, 2.3, 4.9

[87] Jérôme Gallard, Adrien Lèbre, Geoffroy Vallée, Pascal Gallard, Stephen Scott et Christine Morin. Is Virtualization Killing Single System Image Research ? Research Report RR-6389, INRIA, 2007. (document), 3, 3.3.1.3, 4.4, 4.9

[88] Jérôme Gallard, Adrien Lèbre, Geoffroy Vallée, Christine Morin, Pascal Gallard et Stephen Scott. Refinement Proposal of the Goldberg's Theory. Research Report RR-6613, INRIA, juillet 2008. (document), 4, 4.4, 4.9

[89] Jérôme Gallard, Adrien Lèbre, Geoffroy Vallée, Christine Morin, Pascal Gallard et Stephen Scott. Refinement Proposal of the Goldberg's Theory. *Dans ICA3PP'09 : International Conference on Algorithms and Architectures for Parallel Processing*, pages 853–865, Tapei, Taiwan, 2009. (document), 4, 4.4, 4.9

[90] Jérôme Gallard, Christine Morin, Geoffroy Vallée, Thomas Naughton, Stephen Scott et Adrien Lèbre. Architecture for the Next Generation System Management Tools. *Dans First International Conference on Utility and Cloud Computing (UCC 2010)*, Chennai, India, décembre 2010. (document), 4, 4, 4.5.1, 4.9

[91] Jérôme Gallard, Geoffroy Vallée, Adrien Lèbre, Christine Morin, Pascal Gallard et Stephen Scott. Complementarity between Virtualization and Single System Image Technologies. *Dans Euro-Par 2008 Workshops - Parallel Processing : VHPC 2008, UNICORE 2008, HPPC 2008, SGS 2008, PROPER 2008, ROIA 2008, and DPA 2008*, Las Palmas de Gran Canaria Espagne, 2009. (document), 3, 3.3, 3.3.1.3, 4, 4.4, 4.9

[92] Jérôme Gallard, Geoffroy Vallée, Thomas Naughton, Adrien Lèbre, Stephen Scott et Christine Morin. Architecture for the Next Generation System Management Tools for Distributed Computing Platforms. Research Report RR-7325, INRIA, mai 2010. (document), 4, 4, 4.9

[93] David Garlan, Robert T. Monroe et David Wile. Foundations of component-based systems. chapitre Acme : architectural description of component-based systems, pages 47–67. Cambridge University Press, New York, NY, USA, 2000. 4.8

[94] Douglas P. Ghormley, David Petrou, Steven H. Rodrigues, Amin M. Vahdat et Thomas E. Anderson. GLUnix : A Global Layer Unix for a network of workstations. *Software Practice and Experience*, 28(9):929–961, 1998. 1.1.3.2

[95] Robert P. Goldberg. Hardware Requirements for Virtual Machine Systems. *Dans HICSS-4, Hawaii International Conference on System Sciences*, Honolulu, Hawaii, janvier 1971. 4.2

[96] Robert P. Goldberg. Virtual machines : semantics and examples. Proceedings IEEE International Computer Society, Conference Boston Massachusetts, 1971. 4.1.1

[97] Robert P. Goldberg. Architecture of virtual machines. AFIPS National Computer Conference, juillet 1973. 1.2, 4.3

[98] Robert P. Goldberg. Architecture of Virtual Machines. *Dans Proceedings of the Workshop on Virtual Computer Systems*, pages 74–112, Cambridge, MA, USA, mars 1973. 4.2

[99] GridOS. http://ppadala.net/research/gridos/. 1.1.4.2

[100] Andrew S. Grimshaw, Wm. A. Wulf et CORPORATE The Legion Team. The legion vision of a worldwide virtual computer. *Commun. ACM*, 40(1):39–45, 1997. 1.1.4.2

[101] Yuri Gurevich. Sequential abstract-state machines capture sequential algorithms. *ACM Trans. Comput. Logic*, 1:77–111, July 2000. 4.1

[102] Paul H. Hargrove et Jason C. Duell. Berkeley lab checkpoint/restart (BLCR) for Linux clusters. *Journal of Physics : Conference Series*, 46(1):494, 2006. 2.1.2

[103] Brian Hayes. Cloud Computing. *Communications of the ACM*, 51(7):9–11, 2008. 1.2.3.3

[104] Erik Hendriks. BProc : the Beowulf Distributed Process Space. *Dans ICS '02 : Proceedings of the 16th international conference on Supercomputing*, pages 129–136, New York, NY, USA, 2002. ACM Press. 1.1.3.2

[105] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller et Julia Lawall. Entropy : a Consolidation Manager for Clusters. *Dans Proceedings of the International Conference on Virtual Execution Environments (VEE)*, 2009. 1.2.3.2, 2.3.3.6

[106] Gerrit Huizenga. Cloud Computing : Coming out of the fog. *Dans Proceedings of the Linux Symposium*, Otawa, Ontario, Canada, juillet 2008. 1.2.3.3

[107] Felix Hupfeld, Toni Cortes, Björn Kolbeck, Jan Stender, Erich Focht, Matthias Hess, Jesus Malo, Jonathan Marti et Eugenio Cesario. XtreemFS - a case for object-based file systems in Grids. *Concurrency and Computation : Practice and Experience*, 20(8), 2008. 3.2.3

[108] R. P. Goldberg J. P. Buzen, P. P. Chen. Virtual machine techniques for improving system reliability. Processdings IEEE Symposium on Computer Software Reliability, New York, 1973. 4.1.1

[109] Shantenu Jha, Andre Merzky et Geoffrey Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurrency and Computation : Practice and Experience*, 21(8):1087–1108, 2009. 3.1

[110] Hartmut Kaiser, Andre Merzky, Stephan Hirmer et Gabrielle Allen. The SAGA C++ Reference Implementation : Lessons Learnt from Juggling with Seemingly Contradictory Goals. *Dans Second International Workshop on Library-Centric Software Design (LCSD'06)*, 2006. 3.5

[111] Katarzyna Keahey et Tim Freeman. Contextualization : Providing one-click virtual clusters. *Dans ESCIENCE '08 : Proceedings of the 2008 Fourth IEEE International Conference on eScience*, pages 301–308, Washington, DC, USA, 2008. IEEE Computer Society. 1.2.3.3, 3.5, 4.8

[112] Kate Keahey, Ian Foster, Tim Freeman, Xuehai Zhang et Daniel Galron. Virtual Workspaces in the Grid. *Dans 11th International Euro-Par Conference, Lisbon, Portugal*, 2005. 4.8

[113] Kate Keahey, Mauricio Tsugawa, Andrea Matsunaga et Jose Fortes. Sky computing. *IEEE Internet Computing*, 13(5):43–51, 2009. 1.2.3.4, 3.5

[114] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin et Anthony Liguori. KVM : the Linux Virtual Machine Monitor. *Dans Proceedings of the Linux Symposium*, 2007. 1.2.1.1

[115] H. Andres Lagar-Cavilla, Joseph Whitney, Adin Scannell, Philip Patchin, Stephen M. Rumble, Eyal de Lara, Michael Brudno et M. Satyanarayanan. SnowFlock : Rapid Virtual Machine Cloning for Cloud Computing. *Dans Proceedings of the 3rd European Conference on Computer Systems (Eurosys)*, Nuremberg, Germany, avril 2009. 3.5

[116] M. Lageman et S.C. Solutions. Solaris Containers, What They Are and How to Use Them. *Sun BluePrints OnLine*, pages 819–2679, 2005. 1.2.2.2

[117] Michael J. Lewis, Adam J. Ferrari, Marty A. Humphrey, John F. Karpovich, Mark M. Morgan, Anand Natrajan, Anh Nguyen-Tuong, Glenn S. Wasson et Andrew S. Grimshaw. Support for Extensibility and Site Autonomy in the Legion Grid System Object Model. *Journal of Parallel and Distributed Computing*, 63(5):525–538, 2003. 1.1.4.2

[118] H. A. Lichstein. When should you emulate ? *Dans Datamation 15*, pages 205–210, novembre 1969. 4.1.1

[119] Tim Lindholm et Frank Yellin. The Java™ Virtual Machine Specification. http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html. 1.2.1.1

[120] I. Llorente, R. Moreno-Vozmediano et R. Montero. Cloud Computing for on-Demand Grid Resource Provisioning. *Advances in Parallel Computing*, 18:177–191, 2009. 3.5

[121] Jeff Magee, Naranker Dulay et Jeff Kramer. Structuring Parallel and Distributed Programs. *Software Engineering Journal*, 8:73–82, 1993. 4.8

[122] Efrem G. Mallach. On the relationship between virtual machines and emulators. *Dans Proceedings of the workshop on virtual computer systems*, pages 117–126, New York, NY, USA, 1973. ACM. 4.1.1, 4.1.1

[123] Martin W. Margo, Kenneth Yoshimoto, Patricia Kovatch et Phil Andrews. Impact of reservations on production job scheduling. 13th Worshop on Job Scheduling Strategies for Parallel Processing, 2007. 4.8

[124] Matthew L. Massie, Brent N. Chun et David E. Culler. The ganglia distributed monitoring system : design, implementation, and experience. *Parallel Computing*, 30(7):817 – 840, 2004. 2.3.3.5

[125] Peter Mell et Tim Grance. The NIST Definition of Cloud Computing. http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc, 2009. 1.2.3.3

[126] Andre Merzky, Katerina Stamou et Shantenu Jha. Application Level Interoperability between Clouds and Grids. *Dans Workshops at the Grid and Pervasive Computing Conference*, pages 143–150, 2009. 3.5

[127] Christine Morin. XtreemOS : A Grid Operating System Making your Computer Ready for Participating in Virtual Organizations. *Dans ISORC'07 : Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 393–402, Washington, DC, USA, 2007. IEEE Computer Society. 3.2, 3.2.2

[128] Christine Morin, Jérôme Gallard, Yvon Jégou et Pierre Riteau. Clouds : a New Playground for the XtreemOS Grid Operating System. Research Report RR-6824, INRIA, février 2009. (document), 3, 3.1, 4.9

[129] Christine Morin, Pascal Gallard, Renaud Lottiaux et Geoffroy Vallée. Towards an Efficient Single System Image Cluster Operating System. *Future Generation Computer Systems*, 20(2), January 2004. 1.1.3.2, 3.2.1

[130] Christine Morin, Yvon Jégou, Jérôme Gallard et Pierre Riteau. Clouds : A New Playground for the XtreemOS Grid Operating System. *Parallel Processing Letters (PPL)*, 19:435–449, 2009. (document), 3, 3.1, 3.3, 4.9

[131] Christine Morin, Renaud Lottiaux, Geoffroy Vallée, Pascal Gallard, David Margery, Jean-Yves Berthou et Isaac Scherson. Kerrighed and Data Parallelism : Cluster Computing on Single System Image Operating Systems. *Dans Proc. of Cluster 2004*. IEEE, September 2004. 1.1.3.2, 2.1.2, 3.2.1

[132] John Mugler, Thomas Naughton, Stephen L. Scott, Brian Barrett, Andrew Lumsdaine, Jeffrey M. Squyres, Benoît des Ligneris, Francis Giraldeau et Chokchai Leangsuksun. OSCAR Clusters. *Dans Proceedings of the 5th Annual Ottawa Linux Symposium (OLS'03)*, Ottawa, Canada, juillet 23-26, 2003. 2.6.1

[133] Ripal Nathuji et Karsten Schwan. VirtualPower : coordinated power management in virtualized enterprise systems. *SIGOPS Oper. Syst. Rev.*, 41:265–278, octobre 2007. 1.2.3.2

[134] Bogdan Nicolae, Gabriel Antoniu, Luc Bougé, Diana Moise et Alexandra Carpen-Amarie. Blobseer : Next-generation data management for large scale infrastructures. *Journal of Parallel and Distributed Computing*, 71:169–184, février 2011. 2.3.3.3

[135] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff et Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. *Dans CCGRID '09 : Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131, Washington, DC, USA, 2009. IEEE Computer Society. 1.2.3.3, 3.5, 4.8

[136] Zsolt Németh et Vaidy Sunderam. Virtualization in Grids : A Semantical Approach. *Dans Grid Computing : Software Environments and Tools*, pages 1–18, 2006. 4.1

[137] OASIS. Solution Deployment Descriptor Specification 1.0, septembre 2008. 4.8

[138] Information Sciences Institute University of Southern California. RFC : 791 – Internet Protocol – Darpa Internet Program – Protocol Specification. 2.3.3.4

[139] Information Sciences Institute University of Southern California. RFC : 793 – Transmission Control Protocol – Darpa Internet Program – Protocol Specification. http://tools.ietf.org/html/rfc793, septembre 1981. 2.3.1

[140] Pradeep Padala et Joseph N Wilson. Gridos : Operating system services for grid architectures. *Dans In Proceedings of International Conference On High Performance Computing (HiPC'03)*, 2003. 1.1.4.2

[141] Philip M. Papadopoulos, Mason J. Katz et Greg Bruno. Npaci rocks : Tools and techniques for easily deploying manageable linux clusters. CLUSTER'01, 2001. 2.6.1

[142] Odysseas I. Pentakalos. An Introduction to the InfiniBand^{TM}Architecture. *Dans International CMG Conference*, pages 425–432, 2002. 1.1.2.1

[143] Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera et Taliver Heath. *Dynamic cluster reconfiguration for power and performance*, pages 75–93. Kluwer Academic Publishers, Norwell, MA, USA, 2003. 1.2.3.2

[144] James S. Plank, Micah Beck, Gerry Kingsley et Kai Li. Libckpt : Transparent Checkpointing Under Unix. *Dans Proceedings of the USENIX 1995 Technical Conference Proceedings on USENIX 1995 Technical Conference Proceedings (TCON'95)*, pages 213–223, Berkeley, CA, USA, 1995. USENIX Association. 2.1.2

[145] Gerald J. Popek et R. P. Goldberg. Formal requirements for virtulizable third generation architectures. Commun, juillet 1974. 1.2, 4.2, 4.3

[146] Yakov Rekhter, Robert G Moskowitz, Daniel Karrenberg, Geert Jan de Groot et Eliot Lear. RFC : 1918 – Address Allocation for Private Internets. http://tools.ietf.org/html/rfc1918, février 1996. 2.3.3.4

[147] Rolf Riesen, Ron Brightwell, Lee Ann Fisk, Tramm Hudson, Jim Otto et Arthur B. Maccabe. Cplant. *Dans Proceedings of the Second Extreme Linux workshop at the 1999 USENIX Annual Technical Conference*. 1.1.3.2

[148] Pierre Riteau, Mauricio Tsugawa, Andrea Matsunaga, Jose Fortes et Kate Keahey. Large-Scale Cloud Computing Research : Sky Computing on FutureGrid and Grid'5000. *ERCIM News*, pages 41–42, octobre 2010. 1.2.3.4

[149] Larry Roberts. The arpanet and computer networks. *Dans Proceedings of the ACM Conference on The history of personal workstations*, pages 51–58, New York, NY, USA, 1986. ACM. 1.1.2

[150] John Scott Robin et Cynthia E. Irvine. Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. 2000. 1.1.1.1

[151] Ivan Rodero, Francesc Guim, Julita Corbalán et Jesús Labarta. How the JSDL can Exploit the Parallelism ? *Dans Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, 2006. 4.8

[152] Manuel Rodríguez, Daniel Tapiador, Javier Fontán, Eduardo Huedo, Rubén S. Montero et Ignacio M. Llorente. Dynamic Provisioning of Virtual Clusters for Grid Computing. *Dans Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 2008)*, pages 23–32, Berlin, Heidelberg, 2008. Springer-Verlag. 3.5

[153] A.J. Rubio-Montero, E. Huedo, R.S. Montero et I.M. Llorente. Management of virtual machines on globus grids using gridway. *Parallel and Distributed Processing Symposium, International*, 0:358, 2007. 3.5

[154] Joseph F. Ruscio, Michael A. Heffner et Srinidhi Varadarajan. DejaVu : Transparent User-Level Checkpointing, Migration and Recovery for Distributed Systems. *Dans Proceedings of the 2006 ACM/IEEE conference on Supercomputing (SC '06)*, page 158, New York, NY, USA, 2006. ACM. 2.1.2

[155] R. Sandberg, D. Golgberg, S. Kleiman, D. Walsh et B. Lyon. Innovations in internetworking. chapitre Design and implementation of the Sun network filesystem, pages 379–390. Artech House, Inc., Norwood, MA, USA, 1988. 2.3.3.3

[156] Michael D. Schroeder et Jerome H. Saltzer. A hardware architecture for implementing protection rings. *Commun. ACM*, 15(3):157–170, 1972. 1.1.1.2

[157] Phil Schwan. Lustre : Building a file system for 1000 node clusters. *Dans Proceedings of the Linux Symposium*, pages 380–386, 2003. 2.3.3.3

[158] Edi Shmueli et Dror Feitelson. Backfilling with Lookahead to Optimize the Performance of Parallel Job Scheduling. *Dans* Dror Feitelson, Larry Rudolph et Uwe Schwiegelshohn, éditeurs. *Job Scheduling Strategies for Parallel Processing*, volume 2862 de *Lecture Notes in Computer Science*, pages 228–251. Springer Berlin / Heidelberg, 2003. 2.1.1

[159] James E. Smith et Ravi Nair. The Architecture of Virtual Machines. *Computer*, 38:32–38, 2005. 4.1.2, 4.4

[160] A. C. Sodan. Loosely coordinated coscheduling in the context of other approaches for dynamic job scheduling : a survey : Research articles. *Concurrency and Computation : Practice and Experience*, 17:1725–1781, December 2005. 2.1.1

[161] Borja Sotomayor, Ruben S. Montero, Ignacio M. Llorente et Ian Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13:14–22, 2009. 1.2.3.3

[162] Udo Steinberg et Bernhard Kauer. Nova : a microhypervisor-based secure virtualization architecture. *Dans Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 209–222, New York, NY, USA, 2010. ACM. 1.2.2.2

[163] Thomas Sterling, Donald J. Becker, Daniel Savarese, John E. Dorband, Udaya A. Ranawak et Charles V. Packer. BEOWULF : A Parallel Workstation For Scientific Computation. *Dans the 1995 International Conference on Parallel Processing*, 1995. 1.1.3.1

[164] Eliana-Dina Tîrşa, Pierre Riteau, Jérôme Gallard, Christine Morin et Yvon Jégou. Towards XtreemOS in the Clouds – Automatic Deployment of XtreemOS

Resources in a Nimbus Cloud. Technical Report RT-0395, INRIA, octobre 2010. (document), 3, 3.4, 4.9

[165] Peter Toft et Steve Lougrhan. Configuration Description, Deployment and Lifecycle Management Working Group (CDDLM-WG) Final Report, mars 2008. 4.8

[166] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936. 1.1.1.1

[167] R. P. Goldberg U. O. Gagliardi. Virtualizeable architectures. Proceedings ACM AICA International Computing Symposium Venice Italy, 1972. 4.1.1

[168] Geoffroy Vallée, Thomas Naughton, Hong Ong, Anand Tikotekar, Christian Engelmann, Wesley Bland, Ferrol Aderholdt et Stephen L. Scott. Virtual System Environments. *Dans Systems and Virtualization Management. Standards and New Technologies*, volume 18 de *Communications in Computer and Information Science*, pages 72–83. Springer Berlin Heidelberg, octobre 21-22, 2008. 4.4.2, 4.5.1

[169] Geoffroy Vallée, Thomas Naughton, Anand Tikotekar, Jérôme Gallard, Stephen Scott et Christine Morin. Architecture for the Next Generation System Management Tools for High Performance Computing Platforms. Research Report RR-7062, INRIA, octobre 2009. (document), 4, 4, 4.9

[170] Jean van Heijenoort. *From Frege to Gödel : A Source Book in Mathematical Logic, 1879-1931*. Harvard University Press, 2002. 4.4.1

[171] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres et Maik Lindner. A break in the clouds : towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39:50–55, décembre 2008. 3.1

[172] Anthony Velte et Toby Velte. *Microsoft Virtualization with Hyper-V*. McGraw-Hill, Inc., New York, NY, USA, 2010. 1.2.1.1

[173] Pascale Vicat-Blanc Primet, Jean-Patrick Gelas, Olivier Mornard, Dinil Mon Divakaran, Pierre Bozonnet, Mathieu Jan, Vincent Roca et Lionel Giraud. State of the Art of OS and Network virtualization solutions for Grids. Rapport technique, INRIA, septembre 2007. "Delivrable #1 : HIPCAL ANR-06-CIS-005". 1.2.2.2

[174] John von Neumann. First Draft of a Report on the EDVAC (transcription by Michael D. Godfrey). *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. 1.1.1.1

[175] Andrew Warfield, Russ Ross, Keir Fraser, Christian Limpach et Steven Hand. Parallax : managing storage for a million machines. *Dans Proceedings of the 10th conference on Hot Topics in Operating Systems - Volume 10*, pages 4–4, Berkeley, CA, USA, 2005. USENIX Association. 2.3.3.3

[176] Aaron Weiss. Computing in the Clouds. *netWorker*, 11(4):16–25, 2007. 1.2.3.3

[177] Brent Welch, Marc Unangst, Zainul Abbasi, Garth Gibson, Brian Mueller, Jason Small, Jim Zelenka et Bin Zhou. Scalable performance of the Panasas parallel file system. *Dans FAST'08 : Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pages 1–17, Berkeley, CA, USA, 2008. USENIX Association. 2.3.3.3

[178] Joachim J. Wlodarz. Virtualization : A double-edged sword. *ArXiv e-prints*, mai 2007. 1.2.2.2

Flexibilité dans la gestion des infrastructures informatiques distribuées

Résumé

Cette thèse s'intéresse à la flexibilité dans les infrastructures informatiques distribuées du point de vue de leurs utilisateurs et administrateurs. Pour les utilisateurs, il s'agit de trouver au moment où ils en ont besoin des ressources matérielles adaptées avec un environnement personnalisé à l'exécution de leur application. Pour les administrateurs, il s'agit de définir les politiques d'allocation des ressources (politiques d'usage et de sécurité) pour les applications des utilisateurs.

Nous avons étudié la problématique de la flexibilité dans le contexte des grilles et des centrales numériques. Tout d'abord, nous avons conçu et mis en oeuvre le système Saline qui s'appuie sur la virtualisation pour permettre l'exécution de tout type de tâche en mode interruptible dans les grilles. Nous avons également proposé le système Grillade qui combine les mécanismes de flexibilité offerts par les grilles et les centrales numériques pour d'une part, étendre dynamiquement une grille avec des ressources virtuelles fournies par des centrales numériques et d'autre part, construire des nuages de type IaaS fédérant les ressources de plusieurs sites. Grillade étend le système de grille XtreemOS. Il permet en outre grâce à la technologie de système à image unique de proposer aux utilisateurs des machines virtuelles exécutées sur une agrégation de nœuds. Enfin, nous proposons un formalisme permettant de classer les systèmes de gestion de ressources offrant de la flexibilité et de définir des règles pour les combiner. Le système Tropicbird qui s'appuie sur ce formalisme met en œuvre, à la demande, des plates-formes virtuelles spécifiées par les utilisateurs sur une infrastructure matérielle.

Mots clés

Flexibilité, virtualisation, infrastructure distribuée, informatique en nuage (centrale numérique, *cloud computing*), multi-centrales numériques (*sky computing*), grille, grappe, approvisionnement dynamique de ressources.