Performances des fonctions et architectures de supervision de réseaux et de services

Abdelkader Lahmadi

NANCY 2 UNIVERSITE

UFR mathématiques et informatique **École doctorale IAEM Lorraine**

Département de formation doctorale en informatique

Performances des fonctions et architectures de supervision de réseaux et de services

THÈSE

présentée et soutenue publiquement le

pour l'obtention du

Doctorat de l'université Nancy 2

(spécialité informatique)

par

Abdelkader Lahmadi

Composition du jury

Rapporteurs :	Omar Cherkaoui, Professeur des Universités	Université de Québec, Montréal
	Guy Pujolle, Professeur des Universités	Université Paris 6, Paris
Examinateurs :	Laurent Andrey, Maître de Conférences	Université Nancy 2, Nancy
	Olivier Festor, Directeur de Recherche	INRIA Lorraine, Nancy
	Jean François Mari, Professeur des Universités	Université Nancy 2, Nancy
	Aiko Pras, Professeur des universités Associé	Université de Twente, Pays-Bas

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Remerciements

Je tiens à remercier en premier lieu les rapporteurs et les examinateurs de cette thèse pour l'intérêt qu'ils ont porté à mes travaux de recherche et pour avoir accepté de faire partie de mon jury de thèse.

Je remercie très sincèrement Olivier Festor, mon directeur de thèse, pour toute l'attention qu'il m'a apporté. J'ai pris beaucoup de plaisir à travailler avec lui en tant que ingénieur au sein de l'équipe MADYNES durant trois ans, et en tant que thésard durant les trois dernières années. Son dynamisme, des conseils de tout ordre et sa passion communicative pour la recherche furent très appréciable tout au long de ce travail.

Un très grand merci à mon encadrant de thèse, Laurent Andrey, pour la qualité de sa collaboration, sa disponibilité, ses précieux conseils, son aide constante et pour la façon efficace et amicale avec laquelle il a suivi ce travail. Pendant les nombreuses heures passées ensemble, j'ai beaucoup appris à son contact.

Mes remerciements vont également à tous les membres de l'équipe MADYNES où l'ambiance qui règne a été un facteur important du bon déroulement de ce travail.

Plus personnellement, je tiens à remercier mon épouse Anne Catherine pour sa patience et sa disponibilité qui m'ont été précieuse tout au long de l'écriture de ce manuscrit.

A toutes et à tous, merci

À mes parents
À ma famille

Table des matières

Table des figures

Liste des tableaux

Introduction générale

1 Contexte de recherche et motivation

Réseaux et services nouvelle génération

L'infrastructure actuelle de l'Internet a évolué de simples systèmes monolithiques vers des systèmes complexes impliquant plusieurs tiers. Les exemples les plus significatifs sont probablement l'engouement pour les architectures orientées services (SOA) pour transformer les applications monolithiques sur mainframe ou en client-serveur en autant de services que l'on désire. Les applications web, par exemple, ne délivrent plus simplement du contenu pré-écrit disponible sur un serveur monolithique mais elles génèrent de façon dynamique, et à la volée du contenu en se basant sur des architectures multi-tiers distribuées. Ces applications s'appuient de plus en plus sur des réseaux hétérogènes, dynamiques, fédérés par IP et offrent des capacités différentes. Ces réseaux de nouvelle génération sont dus à l'extension de l'Internet aux réseaux domestiques (xDSL) ainsi qu'aux environnements sans fil connectant des terminaux mobiles. Parallèlement, un vaste éventail de services se déploie sur ces réseaux, parmi eux on peut citer la ToIP[1], les services de téléphonie hybride *GSM/Wi-Fi*, l'*IPTV* sur *ADSL*, *UMTS*, la vidéo à la demande sur *ADSL* et des offres de services comme celles de *Google* ou *Amazon*. Pour exploiter la flexibilité et la dynamicité du nouvel Internet, de nouvelles technologies ne cessent d'apparaître comme par exemple la virtualisation qui permet de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, séparément les uns des autres, comme si elles fonctionnaient sur des machines physiques distinctes.

Pour nombre des acteurs des ces infrastructures de communication, le maître-mot est la qualité de service (QoS). Ce mot signifie une haute disponibilité, performance et sécurité à moindre coût afin de maximiser les bénéfices économiques de l'opérateur ou du fournisseur de service. Ainsi, la qualité d'un service de communication est le reflet de la qualité de l'infrastructure technologique sur laquelle elle s'appuie. Cela signifie que la disponibilité de service est cruciale. En effet, une perte de la disponibilité du service entraîne des pertes économiques énormes pour ces acteurs [92]. Par exemple, une rupture d'un service de communication bancaire entraîne une perte moyenne de l'ordre de 828000 euros ce qui est énorme par rapport à une perte de moyenne de l'ordre de 19000 euros pour un secteur plus classique[2].

Les systèmes de gestion de services et réseaux sont confrontés à cet ensemble de défis technologiques et économiques afin de garantir la qualité des services de communication au sein de l'Internet.

[1] Téléphonie sur IP.
[2] Source HP analysis of 158 companies accross EMEA, summer 2005.

Défis de la gestion du nouvel Internet

L'introduction de ces nouveaux environnements technologiques de l'information ne va pas sans poser de problèmes. Les solutions classiques de gestion sont devenues invalides dans ces nouveaux environnements [75]. Nous avons identifié trois caractéristiques de ces environnements qui représentent des défis majeurs à relever par la gestion :

- **Dimension**
  Les approches traditionnelles de gestion s'appuient sur une relation de type 1 : 1 entre un équipement et une application, un client et un serveur, un réseau et un service, pour retrouver les données de gestion dans ces environnements. Par exemple, dans une approche de gestion classique les données de gestion d'un service web sont collectées d'un serveur web qui est connecté à un réseau avec un lien physique. En revanche actuellement, un service web est la composition des interactions entre plusieurs composants autonomes, reliés via un intergiciel utilisant le protocole SOAP[3] pour transporter les requêtes. Ainsi, les réseaux de communication actuels transportent plusieurs services avec un fort couplage entre eux et s'appuient sur plusieurs serveurs virtuels où cohabitent plusieurs applications. Par conséquent, les infrastructures futures vont s'appuyer sur un modèle de type 1 : n qui prendra la place du modèle 1 : 1. Cette mutation va augmenter la complexité des infrastructures de communication ainsi que le nombre de systèmes à gérer.
- **Intégration**
  Les approches actuelles de gestion, notamment l'approche JMX [97, 99] dans le monde Java, sont généralement couplées au plan fonctionnel de l'application à superviser et l'agent de supervision est implanté sous la forme d'un composant dans l'application. L'incidence de cette intégration des deux plans supervision et fonctionnel sur les performances semble non négligeable dans des environnements contraints, où les ressources sont allouées avant tout pour servir les requêtes utilisateurs. Généralement, les équipementiers, les opérateurs ou fournisseurs de services restreignent les ressources allouées aux activités de gestion. Selon leur point de vue, les ressources réseaux et matérielles doivent être dédiées aux activités utilisateurs à valeur ajoutée plutôt qu'aux activités de gestion et de supervision [49]. Par exemple, les opérateurs exigent que les activités de gestion n'utilisent pas plus de 1 à 5% de la bande passante du réseau [52] ou des ressources matérielles (CPU, mémoire) des systèmes à gérer. Ainsi, une activité de gestion non maîtrisée peut facilement dégrader très sérieusement la performance d'un serveur applicatif en terme de nombre de transactions cliente servies par seconde. L'augmentation de l'intensité de demande de supervision afin de collecter plus souvent des données de gestion peut monopoliser les ressources et affamer les requêtes utilisateurs du plan fonctionnel. Cela se traduit par une dégradation de la qualité de service qui est supposée être maintenue par les activités de gestion.
- **Précision**
  Un défi qui découle des deux précédents auxquels sont confrontées les approches de gestion, est celui de la précision de leurs accomplissements. Cette précision est très importante pour accomplir des tâches de supervision, pour garantir la qualité de services et pour réagir à temps en cas de problèmes qui peuvent surgir sur les systèmes gérés. La qualité des estimations produites par une fonction de supervision repose sur deux propriétés fondamentales. La première est la précision des données collectées et leurs rapprochements des valeurs réelles des objets gérés. La deuxième représente les délais que subissent les opérations portant sur ces données de gestion pour altérer ou collecter leurs valeurs. Les approches actuelles de gestion supposent qualitativement que cette précision est stable. En

[3] Simple Object Access Protocol.

revanche, à cause des défis de passage à l'échelle et de l'intégration de la gestion dans le plan fonctionnel des services, la précision devient un défi à laquelle les approches de gestion doivent faire face pour la garantir. En effet, une mauvaise précision des données estimées pour accomplir des fonctions de gestion introduit un risque important de perte de contrôle du système géré.

Problématique

La supervision regroupe un ensemble d'activités qui permet de surveiller et de contrôler les réseaux et leurs services. Comme nous l'avons évoqué précédemment, elle est aujourd'hui confrontée à des environnements à gérer avec des dimensions grandissantes où ses activités sont entrées en concurrence directe avec les requêtes utilisateurs du plan fonctionnel. La sûreté et le bon fonctionnement d'un système géré dans ces environnements dépendent étroitement de la performance de son système de gestion. La performance d'une approche de gestion est délicate à définir et à mesurer d'une façon claire et déterministe, et il n'existe pas de consensus au niveau des acteurs de la communauté de gestion de réseaux et de services pour sa définition et sa mesure. En effet, les études actuelles qui tentent de quantifier la performance, le coût et la qualité des approches de gestion présentent de nombreuses limites au niveau des métriques et des méthodes employées. Généralement, les métriques utilisées sont restreintes pour qu'elles puissent comparer la performance d'une approche de gestion à une autre. Les méthodes employées par ces études sont souvent figées dans un contexte bien spécifique reposant sur des facteurs de performances avec un espace de valeurs limité. Par conséquent, la majorité des études existantes ne sont ni comparables, ni reproductibles, ni représentatives.

C'est pourquoi, il est indispensable de déterminer un ensemble de métriques pour mesurer la performance d'une approche de gestion d'une manière claire, précise et déterministe. Sur la base de cette définition, une méthodologie doit être élaborée pour mesurer ces métriques sous des facteurs d'échelle et d'intégration auxquels est confrontée la gestion.

2 Contributions

Dans cette thèse nous adressons la définition et la mise en œuvre des méthodes permettant l'analyse de performances des applications et des protocoles de gestion de réseaux et de services. Nous nous focalisons essentiellement sur les métriques, les méthodes et les outils en vue de l'étude de la performance des technologies de gestion couramment utilisées. Notamment, le protocole JMX (*Java management eXtension*) [73] pour la gestion de services et d'applications dans le monde Java. Le choix de ce protocole repose sur sa forte intégration dans les plans fonctionnels des applications gérées. Il n'existe à notre connaissance aucun travail autre que celui-ci faisant une synthèse des modèles, savoir faire, et outils nécessaires pour conduire des analyses de performances des applications et architectures de gestion. Ce travail comporte une part importante d'analyse et d'élaboration très fine de campagnes de mesures de performances dont les résultats sont utiles pour illustrer les métriques proposées. Une seconde contribution porte sur l'analyse d'algorithmes de supervision en fonction de leurs modèles de performances élaborés dans la première phase. Cette analyse a abouti à la simulation des comportements de ces algorithmes dans des conditions à large échelle, où nous avons notamment identifié l'effet des délais sur la vue qu'ils perçoivent des systèmes gérés.

Les travaux de recherche présentés dans ce manuscrit ont été réalisés dans le cadre de ma thèse au sein de l'équipe MADYNES dirigée par Olivier Festor au LORIA/INRIA Lorraine à Nancy. Celle-ci s'est déroulée sous la direction d'Olivier Festor et Laurent Andrey et s'inscrit dans

la thématique de notre équipe portant sur la gestion de réseaux et de services dynamiques. Elle a été partiellement effectuée dans le cadre du réseau d'excellence (NoE) européen EMANICS[4].

3 Organisation de la thèse

Le manuscrit est composé de huit chapitres groupés en trois parties principales. Une annexe complète le document. Les trois parties correspondent respectivement à (1) l'état de l'art de notre domaine de recherche, (2) la présentation de nos métriques de performances et de leur méthodologie de mesure (3) la description des expérimentations que nous avons menées sur l'évaluation de performances du framework JMX sous différents scénarios. La figure 1 présente le plan de ce manuscrit ainsi que les liens entre les différents chapitres et parties.

3.1 Partie I : État de l'art

La première partie comprend 3 chapitres. Le premier chapitre constitue une introduction à l'évaluation de performances dans laquelle sont présentées les principales briques de base de cette discipline ainsi que des notions élémentaires d'analyse statistique que nous avons utilisées au cours de notre travail de recherche. Le deuxième chapitre est une introduction à la gestion de réseaux et de services dans laquelle sont dépeintes les principales approches de gestion ainsi que leurs caractéristiques communes. Nous présentons aussi l'approche JMX dédiée à la gestion des applications reposants sur la technologie Java. Cette approche JMX illustre les défis de l'intégration de la gestion dans le plan fonctionnel, ainsi que les différentes combinaisons possibles qui sont offertes pour gérer une application.

Le troisième chapitre présente les travaux existants portant sur l'évaluation de performances de la gestion. Elle relève les lacunes de ces approches et nous sert de support pour positionner nos travaux de recherche.

3.2 Partie II : Contributions

La seconde partie présente nos travaux portant sur une approche d'évaluation par mesure de la gestion. Celle-ci se traduit par la définition, dans le chapitre 4, d'un ensemble de métriques primaires pour évaluer une approche de gestion en terme de rapidité, de coût et de qualité. Les métriques proposées répondent aux exigences du standard IPPM [161] concernant leur clairté et leur caractère déterministe. À partir de ces métriques, nous avons défini une métrique d'efficacité d'une approche de gestion qui met en relation ces différentes métriques primaires. Dans le chapitre 5, nous présentons une méthodologie de mesure pour ces métriques primaires. Cette méthodologie repose sur la conception et le développement d'un banc de mesure appelé *MAGON* (*Management trAffic GeneratOr and Benchmark*), dédié au *benchmarking* et à la génération de charge ou de trafic de gestion d'une manière synthétique ou à partir d'un ensemble de paramètres qui caractérisent des traces réelles de gestion. Ce banc de mesure répond à nos exigences en terme de représentativité, de reproductibilité et de comparabilité des tests de mesure. Nous avons mis en œuvre une instance de ce banc pour la mesure de performances du framework JMX.

3.3 Partie III : Expérimentations

La troisième partie est dédiée aux expérimentations d'évaluation des pratiques de gestion basées sur le framework JMX. Ces expérimentations reposent sur l'instance JMX du banc de

[4] http ://www.emanics.org

mesure MAGON. Dans le chapitre 6 nous avons analysé le passage à l'échelle d'une approche JMX en variant le nombre d'agents attachés à un gestionnaire. Cette analyse s'appuie sur une métrique unique qui capture le degré de passage à l'échelle en fonction de la dégradation de l'efficacité d'une approche de gestion sous un facteur d'échelle. Cette métrique est inspirée de celle proposée par Woodside [23] dans le cadre des systèmes distribués. Nous avons mené des expérimentations sur une grappe de machines sur laquelle nous avons déployé notre banc de mesure pour illustrer cette métrique.

Dans le chapitre 7, nous avons défini une métrique qui capture l'incidence de la gestion sur une application gérée. Dans un premier temps, nous avons analysé l'impact d'une activité de surveillance sur le serveur applicatif *J2EE JBoss* [63]. Ce serveur repose sur un modèle à forte intégration de l'agent JMX qui représente son noyau fonctionnel. Dans la suite, nous avons étendu nos expérimentations pour identifier l'impact de différents modèles d'intégration d'un agent JMX sur la performance d'un serveur web. Les deux chapitres 6 et 7 nous ont montré l'importance des délais de la supervision dans ces différents scénarii. Dans le chapitre 8, nous avons mené des expérimentations dans des environnements de type grappe et grille multi-sites (*Grid5000*) afin de caractériser les délais de gestion par le biais d'une analyse statistique. La caractérisation de délais nous a permis d'identifier la distribution de Weibull comme distribution statistique sous jacente de ces délais. Cette caractérisation est inédite dans notre domaine de recherche. Grâce à cette caractérisation, nous avons pu simuler le comportement d'un algorithme classique de supervision en présence de délais, notamment la distorsion temporelle d'une fonction d'agrégation des valeurs d'une variable trouvée depuis un groupe d'agents.

Enfin, la conclusion donne un résumé des contributions de cette thèse et expose un ensemble de perspectives de recherche.

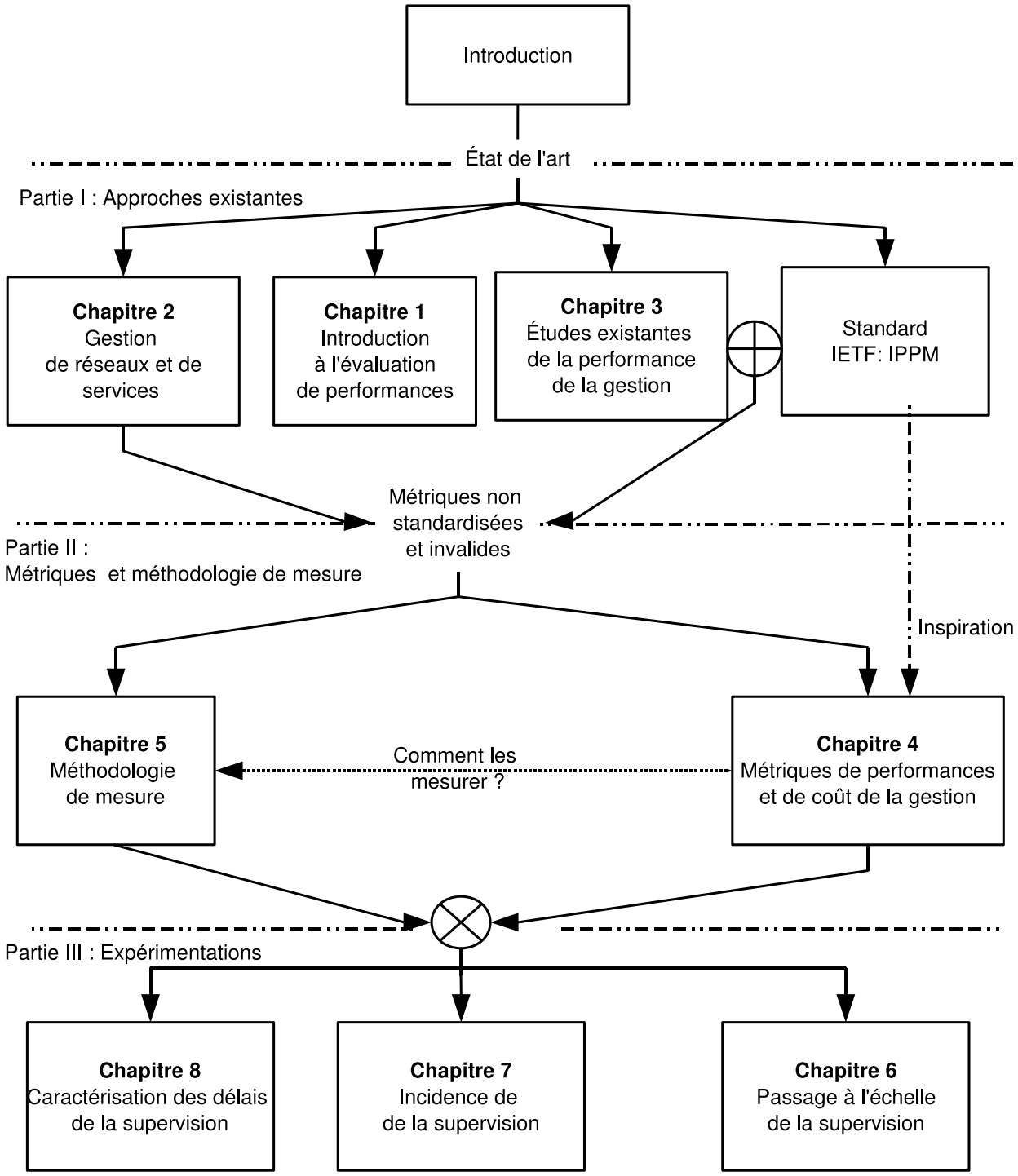


FIG. 1 – Organisation de la thèse

Première partie

État de l'art

Chapitre 1

Principes de l'évaluation de performances et de l'analyse statistique

Sommaire

1.1 Introduction

L'évaluation de performances est une étape importante dans la compréhension d'un système informatique lors de sa conception, son implantation, son déploiement, son utilisation et son évolution. Dans la première partie de ce chapitre nous introduisons les briques de base de l'évaluation de performances d'un système informatique. Notre objectif est d'introduire ici la terminologie, les techniques d'évaluation de performances utilisées dans cette thèse. Nous ne visons pas à fournir un support complet sur l'évaluation de performances. Pour plus de détails, le lecteur peut se référer à [21], qui nous a servi de référence au cours de notre travail. Dans la deuxième partie, nous faisons un bref rappel des méthodes standards d'analyse statistique qui nous ont servi pour l'analyse des mesures des métriques de performances d'un système de supervision.

1.2 Une approche systématique d'évaluation de performances

Le terme *performance* désigne la mesure chiffrée du fonctionnement d'un système ou de l'un de ses composants lors de la réalisation d'une tâche qui lui a été confiée. En revanche, la majorité des problèmes d'évaluation de performances d'un système sont uniques : les métriques, la charge (*workload*) et la technique ne peuvent pas souvent être réutilisés pour un autre problème. Néanmoins, certaines étapes sont communes entre les différents projets d'évaluation de performances. Ces étapes sont les suivantes :

Identifier l'objectif de l'étude et définir les limites du système. Dans un même environnement matériel et logiciel, la définition du système dépend de l'objectif de l'évaluation. La définition des limites du système affecte les métriques de performances ainsi que la charge utilisée pour comparer les systèmes. Par exemple, soit une étude ayant pour objectif la comparaison de l'effet de deux agents différents sur le temps de réponse des requêtes de gestion d'un algorithme de supervision. Dans ce cas notre système est un système de gestion et les résultats vont dépendre d'autres composants extérieurs aux agents.

Liste des services du système et résultats possibles. Chaque système dispose d'un ensemble de services avec des résultats possibles qui ne sont pas forcément désirables. Par exemple, un agent de supervision peut répondre à une requête correctement, incorrectement ou il ne renvoi aucune réponse. La liste des services et des résultats possibles est utile pour la sélection des métriques et de la charge à exercer.

Sélection des métriques de performances. Les critères pour comparer la performance sont appelés métriques. Généralement ces métriques sont relatives à la rapidité, la précision et la disponibilité des services. Par exemple, la performance d'un réseau est souvent mesurée en terme de débit, de délais (rapidité) et de son taux d'erreur (précision).

Liste des paramètres du système et de la charge. Les paramètres sont des variables qui affectent la performance d'un système. Ces paramètres sont groupés en paramètres systèmes et ceux relatifs à la charge. Les paramètres systèmes caractérisent l'aspect logiciel et matériel du système. Ils ne varient pas sur un ensemble d'installations du système. En revanche, les paramètres de la charge caractérisent les requêtes utilisateurs qui varient d'une installation à une autre du système.

Sélection des facteurs et de leurs valeurs. Les paramètres à varier lors d'une étude de performances sont nommés facteurs auxquels on associe des plages de valeurs. Les facteurs sont déduits de la liste des paramètres qui affectent considérablement la performance du système. Il faut noter qu'il est important de limiter le nombre de facteurs ainsi que les plages de leurs valeurs pour que l'étude soit faisable.

Sélection de la technique d'évaluation. Le choix est à faire parmi les techniques : (1)analytiques, (2)de simulation ou (3)de mesure. Dans la section 1.3, nous discutons des critères de choix pour celles-ci.

Sélection de la charge. La charge représente la liste de l'ensemble des requêtes de services exercées sur le système au cours du processus d'évaluation. Il est important que la charge soit représentative d'une utilisation réelle du système. Pour produire une charge réelle, nous aurons besoin de mesurer et de caractériser des systèmes existants. Par exemple, le travail dans [116] présente une caractérisation de la charge d'une supervision basée sur le protocole SNMP [53].

Conception des expérimentations. Il est important de définir une séquence d'expérimentations qui fournit le maximum d'informations avec un effort minimal.

Déroulement des expérimentations. Des techniques existent pour définir la durée et le nombre d'expérimentation nécessaire pour juger le degré de confiance aux résultats obtenus [21][pages 216,423].

Analyse et interprétation des résultats. Les résultats de la mesure et de la simulation sont des quantités aléatoires puisqu'ils peuvent être différents d'une expérimentaion à une autre. Des techniques statistiques sont indispensables pour leurs analyses d'une manière précise. Dans notre contexte, ces techniques sont présentés dans la section 1.5. Une interprétation correcte des résultats est très importante puisqu'elle dirige les décisions à prendre.

Présentation des résultats. Il est aussi important de présenter les résultats du processus d'évaluation pour qu'ils soient faciles à comprendre et à interpréter. Généralement, la représentation graphique est toujours privilégiée.

1.3 Les techniques candidates pour l'évaluation de performances

Dans la littérature, il existe trois approches d'évaluation de performances des systèmes : les modèles analytiques, les simulations et les mesures. Le choix d'une approche ou d'une autre pour évaluer les performances d'un système dépend de plusieurs critères [21]. Nous pouvons classifier ces techniques en deux grandes catégories (voir figure 1.1) : les techniques basées sur la mesure et les techniques basées sur la modélisation. La première catégorie permet de quantifier les critères de performances en les mesurant directement sur un système réel. Dans cette catégorie, nous distinguons deux variantes d'approches de mesure : l'instrumentation du code source de l'application en insérant des codes de mesure (*in-side*), ou l'utilisation de moniteurs externes à l'application pour quantifier les mesures au cours de cycles d'exécution de l'application (*out-side*). La deuxième catégorie d'approches basées sur la modélisation, permet de spécifier le système à évaluer afin de prédire ses performances. Les modèles élaborés par ces approches seront utilisés pour effectuer des simulations ou une étude analytique du système. La différence entre ces deux approches est que les simulations mettent en œuvre des modèles conceptuels du système sous test qui nécessitent leurs développements dans un outil de simulation. En revanche, la deuxième méthode (l'étude analytique) met en œuvre des modèles approximatifs (files d'attentes, réseaux de Petri, ...) parfois calculables, d'une façon approchée, sous la forme de formules mathématiques ou tout autre formalisme.

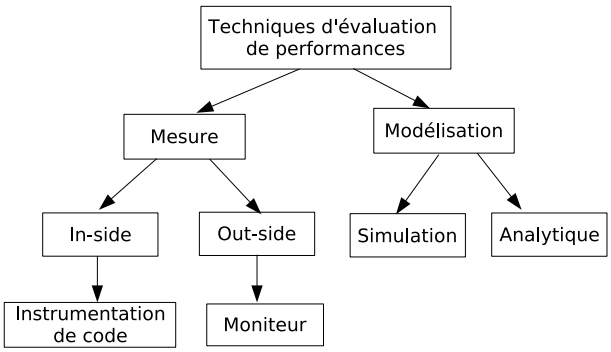


Fig. 1.1 – Classification des techniques d'évaluation de performances

1.3.1 La technique de mesure

La technique de mesure permet d'évaluer un système de gestion existant ou au moins un prototype de ce système. Elle se base sur des outils d'instrumentation. L'instrumentation d'un système est une méthode indispensable pour mesurer ses performances au cours de son exécution, en insérant des codes de mesure dans son code source, ou en utilisant des outils d'observations extérieurs collectant des mesures. L'un des principaux avantages de cette technique est la réalité de ses résultats, puisque les données sont prises sur un système réel. Ce réalisme des mesures est l'un des arguments majeurs en faveur de cette technique pour évaluer les performances d'un système. Cependant, ces résultats, malgré leur réalisme, sont relatifs et variables. Ils dépendent de plusieurs paramètres, comme la configuration du système, la charge et le temps de mesure, qui sont généralement uniques et relatifs à chaque test d'évaluation de performances. Ces résultats, ainsi, ne sont pas généraux mais spécifiques à un test particulier. Un autre inconvénient majeur de cette technique est le coût nécessaire pour sa réalisation. En effet, cette technique d'évaluation introduit un coût important dans un projet puisqu'elle nécessite des équipements réels, l'instrumentation des applications et un temps important de développement et de collecte de mesures. Un troisième inconvénient de cette technique est qu'elle n'est pas toujours réalisable. Dans certains environnements, il est impossible de procéder par cette technique pour évaluer les performances d'un système, par exemple un système embarqué sur un satellite où la contrainte d'énergie rend impossible de procéder par cette technique pour évaluer les performances du système.

La technique de mesure se base sur deux méthodes : l'instrumentation de code du système sous test ou le monitorage du système au cours de son exécution. La première méthode consiste à placer des capteurs dans le code source du système pour mesurer ses performances sous une charge réelle. Cette facilité offerte par cette technique parvient à combler les lacunes d'autres techniques qui n'arrivent pas généralement à exécuter une telle charge réelle et complexe. Des bibliothèques d'instrumentation (par exemple ARM : Application Response Measurement [18]) existent et elles s'intègrent facilement dans le code source de l'application sous test en offrant un cadre efficace et transparent pour le représentation de données d'instrumentation. D'autres outils ne nécessitent pas de modification ou de compilation du code source de l'application mais instrumentent l'application à l'exécution. Parmi ces outils[5], citons le *profiler* fourni par *Netbeans* [29] ou *Gprof* (Gnu profiling Tool, [32]). La seconde méthode se base sur des moniteurs qui se positionnent comme des processus indépendants de l'application sous test. Ces moniteurs observent les activités de l'application au cours de son cycle d'exécution, collectent les mesures et les affichent ou les placent dans un fichier.

Il faut noter qu'il existe deux type de mesures : la mesure passive et la mesure active. La mesure passive ne perturbe pas le système sous test (idéalement). En revanche, la mesure active induit, généralement, des perturbations sur le système. Il existe deux approches de la mesure passive. Une approche de *vraie mesure passive*, où l'outil de mesure n'induit aucune perturbation sur le système sous test. À titre d'exemple un analyseur des paquets dans un réseau LAN sans fil (WLAN) ne perturbe pas le fonctionnement des autres équipements dans le réseau. Une approche de *faible mesure passive*, où le système sous test doit être modifié (par exemple instrumenté) mais sans changer son état de fonctionnement. À titre d'exemple, certains commutateurs *Ethernet* fournissent des ports spécifiques vers lesquels tous les paquets seront copiés sans perturber leurs transmissions. Un analyseur réseau peut s'attacher à ce type de port. L'utilisation de la mesure active ne nécessite pas seulement la modification du système sous test mais impacte aussi sa

[5] Ces outils sont généralement spécifique aux applications Java et utilisent les facilités offertes par l'instrumentation de la machine virtuelle Java.

charge et son état interne de fonctionnement. À titre d'exemple la technique *packet-pair* utilisée pour mesurer les bandes passantes dans l'Internet [25] perturbe la charge et l'état du réseau sous test. Dans des cas extrêmes, l'utilisation de cette approche de mesure active peut induire le phénomène de *Heisenbugs* [27] en faisant apparaître des bugs dans le système sous test.

Généralement, ces méthodes d'instrumentation et ces approches de mesure perturbent le système sous test et engendrent un coût (*overhead*) sur les performances réelles du système sous test. Ce coût est dû aux instructions supplémentaires à exécuter par l'application en utilisant la première méthode (instrumentation) ou aux ressources consommées par un moniteur externe à l'application dans le cas de la deuxième méthode.

1.3.2 L'évaluation par simulation

La simulation semble la technique la plus utilisée pour évaluer les performances des systèmes informatiques. Elle représente un moyen utile pour prédire les performances d'un système et les comparer sous plusieurs de ses configurations. Un atout majeur de cette technique est sa flexibilité puisqu'elle permet d'évaluer le système sous plusieurs conditions et configurations. Même, si le système est déjà implanté la technique de simulation reste favorable puisqu'elle offre une flexibilité, difficile à réaliser avec la technique de mesure. En revanche, la technique de simulation (ainsi que l'approche par mesure) nécessite une excellente maîtrise des techniques statistiques pour une analyse pertinente des résultats, ainsi qu'une maîtrise de la programmation pour développer le modèle à simuler sous un langage approprié. La confiance ou une validation préalable des modèles utilisés par cette technique est indispensable (exemple, les limites d'un modèle de mobilité de type Random Way Point [16]). L'inconvénient majeur de cette technique, est qu'elle nécessite, un temps potentiellement important et des ressources de calcul conséquentes.

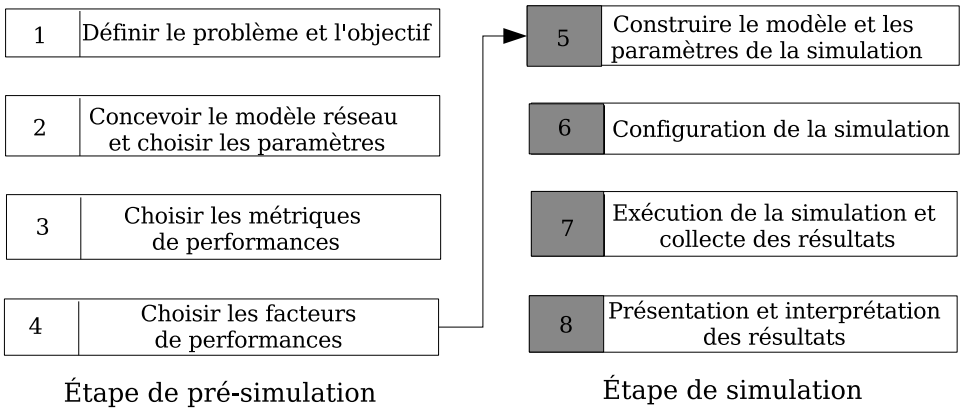


FIG. 1.2 – Les étapes de réalisation d'une simulation.

La simulation d'un système de supervision nécessite la réalisation des étapes suivantes (voir figure 1.2) :

1. Définir le problème et l'objectif de la simulation : dans cette étape il faut essentiellement spécifier le système de supervision à évaluer et ses différents niveaux de détails. Dans notre contexte, l'objectif est d'évaluer les performances du système de supervision sous test.

2. Concevoir le modèle du système de supervision et définir l'ensemble de ses paramètres de performances. Notre objectif est d'évaluer les performances, ainsi, nous n'avons pas besoin de modéliser le fonctionnement exact du système. Par exemple, la modélisation d'un agent de supervision ne nécessite pas forcément la représentation détaillée de la récupération des informations de gestion du système d'exploitation, mais nous considérons juste les valeurs du temps nécessaire pour récupérer ces informations ou sa distribution mathématique (exponentielle, géométrique,...).
3. Choisir l'ensemble des métriques de performances du système de supervision.
4. Choisir l'ensemble des facteurs de performances du système de supervision.
5. Développer le modèle du système de supervision dans l'outil de simulation et fixer l'ensemble des paramètres de performances.
6. Configurer l'application développée dans l'outil de simulation afin de produire les résultats de performances.
7. Exécuter la simulation et collecter les données relatives aux performances du système de supervision.
8. Représenter les données collectées et procéder à leur interprétation.

Pour simuler un système, plusieurs types de modèles existent [21] [page 399]. On trouve notamment les modèles temporels où les états du système sont continus ou discrets en fonction du temps. On trouve aussi les modèles à base d'évènements où les variables d'état du système sont continues ou discrètes. Ces derniers modèles peuvent utiliser les premiers pour spécifier la nature des états du système (discrets ou continus). Plusieurs approches de simulation existent dans la littérature [21] reposant sur ces modèles pour simuler un système. En effet, on trouve notamment les approches suivantes : l'émulation [6], la simulation dirigée par trace et la simulation à évènements discrets.

L'outil de simulation est un choix important lors de l'utilisation de cette technique. Deux principaux outils de simulation existent dans nos domaines réseaux. Le premier est le Network Simulator (NS) [14] qui est libre source et le deuxième est OPNET [35] qui est un produit commercial [7]. Les différences majeures entre ces deux outils sont essentiellement l'interface graphique conviviale de OPNET qui manque dans NS2 et la documentation conséquente de OPNET. Généralement, il est recommandé d'utiliser OPNET plutôt que NS2 pour effectuer des simulations, mais ces deux outils représentent les outils standards pour toute étude basée sur la simulation dans notre domaine. Gilberto Flores et al [28] ont fait une étude comparative portant sur le degré de précision de ces deux outils dans le cadre de la simulation d'un réseau au niveau paquet. Dans leur étude, ils ont comparé la simulation avec ces deux outils d'un réseau avec un trafic de type CBR[8] et des sessions FTP. Ils ont trouvé que d'un point de vue activités de recherche, les résultats de NS-2 et OPNET sont similaires. Mais, NS-2 reste plus attractive car il est libre source et gratuit. D'un point de vue d'un opérateur réseau, OPNET est plus attractif puisque il fournit plus de modules que NS-2. Ils ont aussi montré qu'une simulation précise d'un comportement réseau nécessite l'ajustement d'un ensemble de paramètres du simulateur. Un autre aspect à ne pas négliger est la crédibilité des résultats obtenus par la technique de simulation.

[6] L'émulation représente une des approches de la simulation. En revanche, elle se présente sous la forme d'un coeur de simulation avec des interfaces d'applications standard. Dans ce cas les applications ne sont pas des modèles.

[7] Une version gratuite de OPNET est disponible pour les chercheurs et les académiques.

[8] CBR : Constant Bit Rate. Il s'agit d'un trafic à débit binaire constant comme celui des applications de vidéo conférence ou de téléphonie.

L'étude de Pawlikowski et al [31] porte sur cet aspect. Les auteurs ont montré que deux conditions sont indispensables pour qualifier une simulation de crédible. La première condition est l'utilisation d'un pseudo générateur aléatoire de nombres indépendants et uniformément distribués. La deuxième condition est l'analyse statistique appropriée, en tenant notamment compte des erreurs introduites au niveau des données résultantes de la simulation.

1.3.3 La technique analytique

La technique analytique est l'un des moyens le plus rapide, comparée aux deux autres, pour évaluer au moindre coût les performances d'un système. Elle se base sur la modélisation des systèmes sous forme de paramètres, de variables et un ensemble de formules mathématiques qui régissent leurs relations. Nous pouvons, ainsi, modéliser et évaluer d'une manière flexible les comportements d'un système. En revanche, la technique analytique nécessite beaucoup de simplifications et d'hypothèses pour arriver à un modèle cohérent du système. Ces simplifications se présentent comme l'une de ses limites majeures et mettent en cause le degré de précision des résultats obtenus avec cette technique.

Cette technique se base sur plusieurs approches (Réseaux de files d'attentes, Réseaux de Pétri, chaînes de Markov,...) pour modéliser un système et prédire ces performances. La théorie de réseaux de files d'attentes [24] est la plus utilisée dans l'évaluation de performances des systèmes informatiques. Elle représente et analyse, généralement, des systèmes à ressources partagées. Un modèle de réseau de files d'attente se présente sous la forme d'une collection de serveurs qui interagissent entre eux, représentant les ressources du système et d'un ensemble des clients qui représentent les utilisateurs partageant ces ressources. Ce modèle se représente formellement comme un graphe orienté direct avec des nœuds qui représentent ces serveurs et les liens entre eux représentent le comportement des sollicitations de clients à ces serveurs. Ces modèles sont analysés par plusieurs algorithmes de façon efficace pour obtenir des valeurs moyennes des indicateurs de performances du système modélisé. À titre d'exemple, citons l'algorithme de convolution ou l'algorithme d'analyse par valeurs moyennes (MVA[9]) [10]. La construction de ces modèles nécessite la réalisation des étapes suivantes :

1. La définition des centres de service (serveurs). Cette définition inclut leur nombre, leur classe, leurs clients et leur topologie.
2. La définition des paramètres du modèle : processus d'arrivée des clients, taux de service et le nombre de clients.
3. L'évaluation pour obtenir une description quantitative du système modélisé en calculant la valeur de ses indicateurs de performances (utilisation de ressource, débit du système et temps de réponse). Ces indicateurs peuvent être locaux pour un serveur spécifique ou globaux pour tout le système.

La figure 1.3 montre un réseau de files d'attentes simplifié d'une architecture de supervision. Nous identifions trois centres de service[10] : le gestionnaire, le réseau et l'agent. Ce réseau de files d'attentes représente les routes que les requêtes, provenant du superviseur, doivent suivre afin d'accomplir leurs tâches de supervision. Les paramètres du modèle sont les taux de services $(t_i, i \in \{1..3\})$ des différents centres de services.

Des extensions des réseaux de files d'attentes sont apparues afin de représenter des nouveaux aspects des systèmes réels, comme par exemple la synchronisation, les contraintes liées à la concurrence, le multi-threading, et la possession simultanée des ressources. Le réseau de files

[9] En anglais MVA : Mean Value Analysis.
[10] Il s'agit bien d'un modèle de supervision de type gestionnaire-agent.

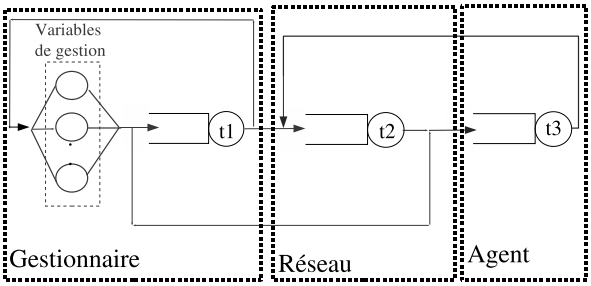


FIG. 1.3 – Un réseau de files d'attentes d'un système de supervision.

d'attentes en couches (LQN : Layered Queueing Networks) [38] est l'une de ces extensions. Elle permet de modéliser des architectures client/serveur distribuées avec des interactions concurrentes. Dans ce type de files d'attentes, un serveur devient un client pour d'autre serveurs tout en servant les requêtes de ses propres clients. Cet aspect représente la différence majeure entre les réseaux de files d'attentes classiques et les LQN. Différents outils existent pour l'évaluation de performances de ces modèles LQN. Parmi eux, citons l'outil LQNS (Layered Queueing Networks Solver, [15]). La figure 1.4 montre un modèle LQN simplifié d'une architecture de gestion de type gestionnaire-agent.

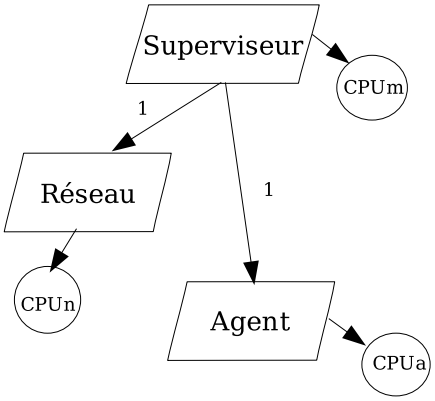


FIG. 1.4 – Un modèle LQN d'un système de supervision.

Un modèle LQN se présente sous la forme d'un graphe acyclique dont les nœuds (les tâches) représentent les entités logicielles et matérielles du système et les arcs représentent les requêtes de service. Les tâches sont représentées par des parallélogrammes et les composants matériels (CPU) sous forme des cercles.

1.3.4 Comparaison des techniques

Le tableau 1.1 représente une comparaison qualitative des trois techniques d'évaluation de performances présentées précédemment. L'utilisation d'une seule technique pour évaluer les performances d'un système n'est pas recommandée. Elle sera insuffisante pour évaluer de façon pertinente et crédible les performances d'une architecture de gestion. Il faut envisager d'utiliser au moins deux techniques d'une façon séquentielle, pour que l'une puisse valider les résultats de l'autre. Idéalement, la technique utilisée, devrait répondre aux critères suivants :

Critère	**Technique analytique**	**Technique de simulation**	**Technique de mesure**
Étape	N'importe	N'importe	Nécessite l'existence d'un système réel ou un prototype
Temps requis	Faible	Moyen	Variable
Outils	Des analystes	Langages de programmation, simulateurs	Instrumentation, moniteurs
Précision	Faible	Moyen	Variable
Compromis de l'évaluation	Facile	Moyen	Difficile
Coût	Faible	Moyen	Élevé

TAB. 1.1 – Comparaison des techniques d'évaluation de performances [21].

- Le coût de la technique : ce coût ne doit pas être considérable en terme de temps de réalisation des tests et de leur coût financier. La technique de mesure est la plus coûteuse puisqu'elle nécessite l'existence d'un prototype de l'architecture de gestion à évaluer. La technique analytique est la moins coûteuse en terme de coût financier puisqu'elle ne nécessite pas le développement ou le déploiement d'un prototype du système de supervision. La technique de simulation possède un coût variable. Son coût dépend du type de simulation utilisée (*trace driven, execution driven, event driven, complete ou components system simulation*) [21], de l'existence ou non d'un modèle de l'architecture de gestion à simuler dans l'outil de simulation utilisé.
- La faisabilité de la technique : le choix de la technique dépend de l'environnement du système de supervision sous test. Certains environnements peuvent défavoriser l'utilisation d'une technique par rapport à une autre. L'évaluation de performances d'un système de supervision embarqué dans un satellite défavorise l'utilisation de la technique de mesure. Les mesures vont altérer et dégrader le fonctionnement du système (consommation excessive des batteries). Dans ce cas, les techniques de simulation ou analytique sont plus adéquates. En revanche, la non disponibilité des modèles communs pour les architectures de gestion favorise l'utilisation de la technique de mesure, étant donné l'existence de prototypes de ces architectures.
- La flexibilité de la technique : le modèle du système sous test utilisé par une technique d'évaluation doit être facile à modifier et à étendre. Les architectures de gestion et leurs environnements sont constamment en évolution et de nouvelles approches ne cessent d'apparaître. Ainsi, la technique d'évaluation utilisée, doit offrir la possibilité d'une intégration facile de ces évolutions dans les modèles élaborés.
- La précision de la technique : elle présente la précision des résultats obtenus par une technique d'évaluation de performances par rapport à ceux obtenus d'une utilisation réelle du système sous test. La techniques analytique est généralement la moins précise. Elle nécessite des simplifications du système sous test afin de construire un modèle calculable sous forme des formules mathématiques pour donner une estimation des métriques de performances. Les modèles élaborés par la technique de simulation incluent plus de détails du système sous test et impliquent moins de simplifications. Ainsi les résultats seront potentiellement plus précis. Ces modèles sont plus proches du système réel de supervision. La précision de

la technique de mesure est variable. Le degré de précision des résultats obtenus peut être fort comme faible. Cela est dû au choix de l'ensemble des paramètres, des facteurs et de la charge de performances à appliquer lors de la réalisation d'une expérimentation par cette technique.
- L'impact de la technique : la technique choisie ne doit pas altérer les performances du système de supervision sous test et le système supervisé.

Nous constatons que ces critères peuvent être en conflit lors du choix d'une technique adéquate pour évaluer les performances d'un système de supervision. Par exemple, la technique analytique est la moins coûteuse mais elle est la moins précise. Systématiquement, on peut se baser sur les deux critères suivants pour choisir une technique parmi les trois :
- Si le système sous test existe et qu'il est accessible avec un effort raisonnable, nous pouvons utiliser la technique de mesure.
- Si le système sous test n'est pas implanté ou s'il est trop compliqué d'accès, alors un modèle de performances doit être développé en utilisant la technique analytique ou en construisant un modèle de simulation pour analyser ses performances.

1.4 Les métriques de performances

Dans cette section, nous allons introduire quelques généralités sur les métriques de performances d'un système. Une métrique de performances est un critère de mesure choisi pour quantifier les performances d'un système. Dans cette section, nous allons donner la définition de certaines métriques génériques de performances qui existent dans la littérature [21]. Deux catégories de métriques de performances : primaires et secondaires existent [21].

1.4.1 Métriques primaires

Cette première catégorie de métriques est en relation avec les trois caractéristiques intrinsèques d'un service d'un système qui sont le temps, le débit et la ressource. Le temps est celui mis par le système pour réaliser un service. Le débit est celui avec lequel le service est réalisé. Les ressources sont celles consommées au cours de la réalisation du service. Cette catégorie contient les trois familles de métriques primaires suivantes [21] :

1. Les métriques de réponse : ces métriques caractérisent la rapidité d'un système. Elles mesurent le temps écoulé entre l'invocation et la fin d'une opération.
2. Les métriques de production : ces métriques mesurent la productivité (débit) d'un système. Elles représentent la quantité du travail accomplie par le système par unité de temps.
3. Les métriques d'utilisation : ces métriques mesurent les ressources consommées sur un système au cours de la réalisation d'un service. Nous identifions les métriques d'utilisation suivantes : l'utilisation CPU, l'utilisation mémoire et l'utilisation réseau.

Cette première catégorie propose un ensemble de métriques considérées comme des critères de performances *primaires* puisqu'elles sont généralement relatives à des opérations unitaires ou à des transactions.

1.4.2 Métriques dérivées

La deuxième catégorie de métriques quantifie le comportement global du système sous test. Dans cette catégorie, nous identifions les métriques suivantes : la fiabilité, la disponibilité et le passage à l'échelle.

1. La métrique de fiabilité est définie par la probabilité qu'une erreur se produise lors de la réalisation d'un service, ou comme étant le temps moyen entre l'apparition des erreurs dans le système.
2. La métrique de disponibilité est définie comme étant la fraction de temps durant laquelle le système est disponible pour répondre aux requêtes des utilisateurs.
3. La métrique de passage à l'échelle : dans la littérature, nous trouvons plusieurs définitions [23, 11, 22, 26] du passage à l'échelle d'un système. Woodside et al [23] proposent qu'un *système passe à l'échelle s'il peut être déployé d'une façon efficace et économique sur des plages convenablement définies et de tailles différentes.* Dans le cas des systèmes de gestion de réseaux, Vilà et al [37] proposent la définition suivante : *un système passe à l'échelle si ses performances ne se dégradent pas considérablement dès l'augmentation des dimensions du système.*

Cette seconde catégorie propose des critères de performances considérés comme *dérivés*. Ils sont plutôt relatifs au fonctionnement global d'un système et elles sont dérivées de l'ensemble de métriques primaires. En effet, la première catégorie de métriques est directement mesurable et permet d'élaborer les métriques de la deuxième catégorie.

1.5 Analyse statistique

Dans cette section nous présentons les principes d'analyse des échantillons de mesure collectées sur le banc de mesure que nous avons développé pour l'évaluation de performances des approches de gestion. Pour analyser ces données, nous avons dû faire face aux points suivants :

- Traiter de gros volumes de données : nos tests de mesure ont généré un volume de données de l'ordre de 50 Go (Giga octets) par expérimentation d'un facteur de mesure.
- Trouver les distributions statistiques qui puissent modéliser correctement les données qui nous intéressent.

En se basant sur ces points, nous avons élaboré une méthodologie d'analyse de ces données. Cette méthodologie repose sur les méthodes de statistiques que nous présenterons dans les sous sections qui suivent (1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.5.5 et 1.5.6).

1.5.1 Recours aux statistiques robustes

Au cours de l'analyse de données de mesure nous avons été confronté au problème de bruit introduit au cours d'une expérimentation. Ce bruit est dû au caractère variable de la précision des données collectées par la technique de mesure (voir le tableau 1.1). En effet, il arrive souvent d'un petit nombre de mesures ait un impact très important sur l'ensemble d'une étude (expérience, mesure). À cause de ces valeurs extrêmes (ou *outliers*), la moyenne arithmétique et la variance des valeurs de l'échantillon sont biaisées. Cela nous donne une sur(sous)-estimation de la métrique mesurée. Pour résoudre ce problème, les staticiens ont développé des statistiques robustes [20]. Ces estimateurs statistiques résistent à ces valeurs extrêmes. Dans notre contexte, nous avons eu recours à la médiane et à l'écart interquartile (IQR[11]) comme estimateurs robustes. Au contraire de la moyenne, la médiane n'est pas affectée par des valeurs extrêmes de l'échantillon de mesures. Le calcul de la médiane d'un échantillon $\{x_i\}$ de N valeurs observées s'effectue de la façon suivante :
Notons $\{x_{(i)}\}$ la série de ces N valeurs rangées dans l'ordre croissant au sens large.

[11] Interquartile range ou IQR.

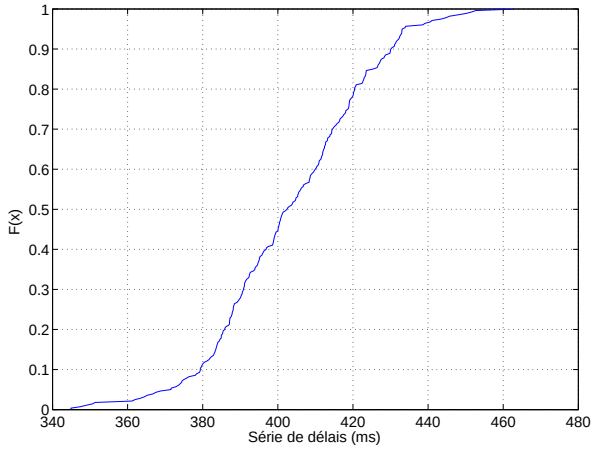


FIG. 1.5 – Fonction de répartition empirique de délais de collecte d'une variable de gestion depuis un agent JMX.

- Si N est impair, médiane= $x_{(\frac{N+1}{2})}$.
- Si N est pair, toute valeur de l'intervalle médian $[x_{(\frac{N}{2})}, x_{(\frac{N}{2}+1)}]$ convient. Le plus souvent, on choisit médiane= $\frac{1}{2}(x_{(\frac{N}{2})} + x_{(\frac{N}{2}+1)})$

Pour estimer la variation des valeurs observées, on utilise l'écart interquartile. L'écart interquartile est la différence entre le troisième quartile Q_{75} et le premier quartile Q_{25}[12]. Un quantile d'ordre $\alpha\%$, noté Q_α, est un réel x tel que $\alpha\%$ des valeurs observées soient inférieures ou égales à x.

1.5.2 Fonction de répartition empirique

La moyenne, l'écart type, la médiane et l'écart interquantile sont des estimateurs qui résument au mieux un ensemble d'observations. En revanche, on a besoin de connaître la proportion de valeurs, parmi l'échantillon des valeurs observées, pour lesquelles la métrique étudiée prend une valeur inférieure ou égale à l'une des valeur observées. Ainsi, la fonction de répartition empirique définie à partir de l'échantillon $\{x_i\}$ est la fonction de $\mathbb{R}$ dans $[0;1]$ qui à tout réel x associe le réel : $F(x) = \frac{card\{x_i \leq x\}}{N}$ [13]. Si x est inférieur à la valeur minimale observée alors $F(x) = 0$, et si x est supérieur au égale à la valeur maximale observée alors $F(x) = 1$. La figure 1.5 montre un exemple d'une fonction de répartition empirique de délais que subit une paire requête/réponse pour collecter les valeurs d'une variable depuis un agent.

1.5.3 Q-Q Plot d'une série

La représentation Quantile-Quantile (Q-Q) d'une série est une technique graphique pour déterminer si deux ensembles de données viennent de populations possédant une distribution commune. Le Q-Q Plot est la représentation de la fonction quantile d'une première série de données en fonction de la fonction quantile d'une deuxième série de données. La première bissectrice est aussi tracée dans une représentation Q-Q Plot. Il s'agit d'une ligne de référence. Si les

[12] Un quartile est l'un des 3 valeurs de quantile qui divisent un échantillon en 4 parts égales.
[13] La fonction de répartition est définie par la probabilité que $X \leq x$, où X est une variable aléatoire.

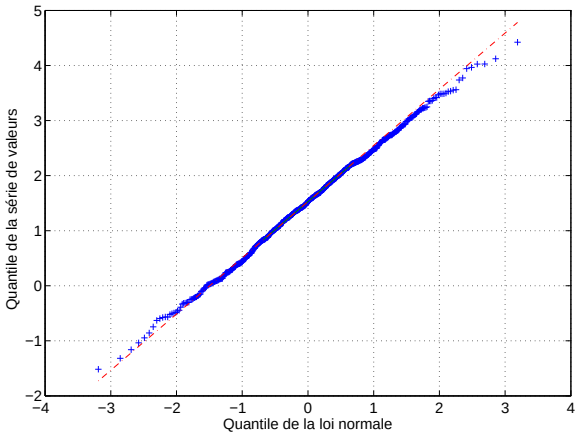


FIG. 1.6 – Série de points dont la distribution suit une loi normale

deux ensembles de données ont la même distribution, les points sont répartis approximativement le long de cette ligne de référence. Au contraire, plus la représentation s'éloigne de la ligne de référence, plus les deux ensembles de données comportent des distributions différentes.

La figure 1.6 représente la comparaison de la fonction quantile d'une série de points au quantile d'une série de points d'une loi normale. Sur l'axe horizontal on marque N points régulièrement espacés qui matérialisent les quantiles de la distribution de référence (la loi normale). Sur l'axe vertical on marque N points qui matérialisent les quantiles de l'échantillon. Pour construire le Q-Q plot on range tout d'abord les observations par ordre de valeurs croissantes. L'observation de rang i sera représentée sur le diagramme par un point d'ordonnée $\frac{i}{N+1}$, c'est à dire le quantile de l'observation. L'abscisse de ce point est le quantile de la distribution de référence, défini par la valeur de l'observation. Dans ce cas de figure, le nuage de points suit la première bissectrice, ainsi la distribution peut être approchée par une loi normale. Au contraire, si le nuage de points s'éloigne fortement de la première bissectrice, la distribution ne peut être approchée par une loi normale.

1.5.4 Distributions statistiques

Dans notre contexte de travail lors de l'analyse de valeurs observées d'une métrique de performances spécifique, nous avons essayé de trouver des distributions classiques bien connues pour modéliser correctement la distribution de valeurs d'une métrique spécifique. Parmi les distributions trouvées pour certaines métriques observées on trouve la distribution de *Weibull* [14]. La distribution de *Weibull* est une distribution à décroissance lente comparée à des distributions exponentielles ou normales. Elle est déterminée par deux paramètres qui sont le paramètre d'échelle α et le paramètre de forme β. La densité de probabilité de cette distribution est est définie comme suit :

$$f(x|\alpha,\beta) = \frac{\beta x^{\beta-1}}{\alpha^{beta}} e^{-(x/\alpha)^{\beta}} \tag{1.1}$$

Sa fonction de répartition est définie par :

$$F(x|\alpha,\beta) = 1 - e^{-(x/\alpha)^{\beta}} \tag{1.2}$$

[14] La distribution de Weibull est nommée d'après *Waloddi Weibull* reconnu pour ses travaux sur la fatigue des matériaux.

Les deux paramètres forme et échelle identifient la structure et les estimateurs statistiques de la distribution. Le paramètre α est relatif au pique de la distribution ; alors que que le paramètre β concerne l'allure de sa queue. Ces deux paramètres déterminent le premier et le deuxième moment de la distribution, comme indiqué dans le tableau 1.2. Le calcul de la moyenne et de l'écart type d'une distribution de Weibull repose sur l'utilisation de la fonction Gamma d'Euler, noté Γ.

Médiane : $\bar{x} = \alpha(ln2)^{1/\beta}$
Moyenne : $\bar{x} = \alpha\Gamma(\frac{\beta+1}{\beta})$
Écart type : $\sigma_x = \alpha\sqrt{\Gamma(\frac{\beta+2}{\beta}) - \Gamma(\frac{\beta+1}{\beta})^2}$

TAB. 1.2 – Premier et deuxième moments de la distribution de Weibull

1.5.5 Estimations de paramètres et test d'adéquation

La technique de Q-Q plot présentée dans la sous section 1.5.3, pour utile qu'elle soit, ne constitue pas une réponse mathématique au problème de l'adéquation. En effet, il est possible de quantifier l'éloignement de la distribution empirique par rapport à une loi théorique en utilisant des distances entre leurs lois de probabilités respectives. Notre objectif est de trouver une distribution qui s'ajuste bien à un échantillon de mesures d'une métrique ou une caractéristique du système. Pour cela nous avons besoin d'estimer les paramètres des distributions à partir de données et de tester l'hypothèse statistique : « est-ce que les données suivent la distribution Y ? » ; pour trouver la distribution qui se rapproche le mieux des données. Il existe plusieurs techniques permettant de définir les paramètres d'une distribution à partir d'un échantillon [34]. Dans ce travail, nous avons eu recours à l'estimateur du maximum de vraisemblance (MLE[15]). Cette méthode consiste à prendre comme estimateur d'un ensemble de paramètres θ, les valeurs qui maximisent la fonction de vraisemblance L. Dans le cas d'une seule observation X, cette fonction de vraisemblance est égale à la fonction de densité, noté $f(X, \theta)$ de la loi théorique. En revanche, dans le cas de n observations $X = (X_1, X_2, ..., X_n)$, la fonction de vraisemblance devient : $L(X, \theta) = \prod_{i=1}^{n} f(X_i, \theta)$. En pratique on prend comme estimation de θ la solution de l'équation :

$$\frac{\partial lnL(X, \theta)}{\partial \theta_j} = 0, \forall j = 1..p \tag{1.3}$$

où p est la dimension de θ (i.e, le nombre de paramètres à estimer) et X est une loi probabiliste, ou un échantillon de mesure dans notre contexte.
Pour vérifier l'adéquation d'une distribution à un échantillon de mesure, nous avons utilisé le test d'Anderson-Darling (A^2). Ce test statistique est un mécanisme qui permet de trancher entre 2 hypothèses, les données d'un échantillon $\{x_i\}$. Soient H_0 et H_1 ces 2 hypothèses dont une seule est vraie. Dans notre cas H_0 est l'hypothèse statistique à tester. H_0 est défini comme suit : l'échantillon $\{x_i\}$ suit une loi Y, c'est à dire $F_x = F_y$, avec F_x et F_y les fonction de répartitions respectives de X et de Y. Pour tester cette hypothèse, on utilise un test statistique qui suit une certaine loi si H_0 est vraie. Le test statistique d'Anderson-Darling suit une loi uniforme sur l'intervalle [0,1] si la loi empirique X et la loi théorique Y coïncident.

Afin de trouver la distribution qui approche le mieux les données observée, nous avons utilisé la métrique de type : λ^2 définie dans [161]. Cette métrique est basée sur la répartition des données en plusieurs classes (intervalles disjoints). En effet, on discrétise l'échantillon en m classes de

[15] En anglais : Maximum Likehood Estimator (MLE)

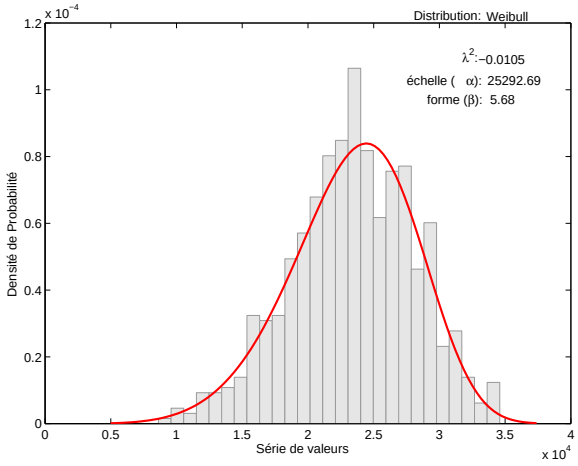


FIG. 1.7 – Ajustement d'une série de valeurs à une distribution de Weibull.

probabilités théorique $p_1, p_2, ..., p_m$. Ensuite, dans chaque classe C_i (bin) on compte le nombre Y_i de valeurs de l'échantillon présents. On a ainsi, $\sum_{i=1}^{m} Y_i = n$. Sur la figure 1.7, ces classes sont représentés par les rectangles de l'histogramme. La plage de l'histogramme correspondant à cette classe vaudra alors : $\frac{Y_i}{n}$. Le principe de la méthode de λ^2 est de définir une *distance* afin de comparer le nombre de données observées dans cet intervalle avec le nombre théorique donné par la distribution. Le test du λ^2 utilise la statistique suivante :

$$\lambda^2 = \frac{X^2 - K - df}{n - 1} \tag{1.4}$$

où $X^2 = \sum_{i=1}^{m} \frac{D_i^2}{E_i}$. Avec $E_i = n \times p_i$ le nombre théorique donné par la distribution, $D_i = Y_i - E_i$ est la distance entre les Y_i par rapport à E_i, $K = \sum_{i=1}^{m} \frac{D_i}{E_i}$ et $df = m - 1 - p$, où p est le nombre de paramètres estimés à partir de l'échantillon.

En effet la distribution qui approche au mieux un échantillon observé est celle qui minimise l'indicateur λ^2. La figure 1.7 donne un exemple d'un ajustement d'une série de valeurs à une loi de Weibull en se basant sur l'estimateur du maximum de vraisemblance pour estimer les paramètres de la distribution et l'indicateur λ^2 pour identifier la meilleure loi qui le minimise.

1.5.6 Outils statistiques

Nous avons utilisé le logiciel Matlab[16] pour toutes les études statistiques. Celui-ci, inclut une boîte d'outils dédiée à l'analyse statistique et l'ajustement des distributions. Cependant, il n'inclut pas l'indicateur λ^2, et le test d'Anderson-Darling. Pour le calcul de ces indicateurs, nous avons utilisé des scripts *Matlab* développés par l'université de Napoli[17]. Néanmoins, nous ne nous sommes pas restreint au seul outil *Matlab* mais nous avons vérifié nos tests d'ajustement à l'aide de l'outil *Dataplot* développé par le NIST[18] qui offre plus de tests statistiques ainsi qu'un nombre plus conséquent de lois de distributions.

[16] http ://www.mathworks.com
[17] http ://www.grid.unina.it/Traffic/Tools/statools.php
[18] http ://www.itl.nist.gov/div898/software/dataplot/

1.6 Synthèse

Ce chapitre était consacré à des notions de base qui nous ont servi pour l'évaluation de performances des approches de gestion et de l'analyse statistique des échantillons de données issues de leurs processus d'évaluation. Dans notre contexte, il s'agit d'un niveau de confiance de l'usage de ces méthodes pour répondre à notre objectif d'évaluation de performances des systèmes de gestion. Ce processus d'évaluation reposera sur l'une des techniques candidates qui sont la technique analytique, la simulation ou la mesure. Dans ce travail, nous avons uniquement utilisé la technique de mesure. Ce choix est justifié par notre objectif de définir des métriques déterministes pour la mesure de la performance d'une approche de gestion. En effet, seule la technique de mesure nous permettra de valider ce type de métriques déterministes. Cette validation reposera sur le développement d'un banc de mesure dédié à la performance des approches de gestion qui met en œuvre une méthodologie de mesure de ces métriques. Cette méthodologie couplée à l'analyse statistiques de résultats collectés nous permet une meilleure compréhension de performances des activités de gestion. Une utilisation future de techniques de simulation ou de méthodes analytiques peuvent tirer profit de ces mesures pour la validation de leurs résultats.

Comme indiqué dans ce chapitre, l'étape cruciale dans un processus d'évaluation de performances est l'identification du système sous test, de ses différents composants et ses différentes fonctions. Dans notre contexte de travail, le système sous test est un système de supervision de réseaux et de services. Dans le chapitre suivant, nous présenterons les différentes caractéristiques des approches de gestion ainsi que les sources de variation de leurs performances.

Chapitre 2

Introduction à la gestion de réseaux et de services

Sommaire

2.1 Introduction

La gestion[19] de réseaux et de services de l'Internet est une activité en plein essor dans le domaine des réseaux IP et des services associés. Les opérateurs réseaux et les fournisseurs de services utilisent des systèmes de gestion pour assurer le bon fonctionnement et l'évolution de ces réseaux et services. Dans ce chapitre nous allons présenter les caractéristiques et les fonctionnalités liées aux systèmes de gestion. Nous introduisons aussi les complexités et les défis auxquels sont confrontés les systèmes de gestion, défis qui influent leurs performances et leurs coûts. Pour bien assimiler les généralités de la gestion de réseaux et de services, nous présentons dans la deuxième partie de ce chapitre, la technologie JMX dédiée à la gestion de services et de applications basées sur Java et le protocole SNMP dédié à la gestion de réseaux IP. À travers la présentation de ces approches de gestion nous comprendrons mieux les défis et les lacunes auxquelles sont confrontées ces approches en terme de performances et d'efficacité.

2.2 Quelques éléments sur la gestion de réseaux et de services

Dans la littérature plusieurs définitions de la gestion de réseaux et de services existent [93]. La plupart de ces définitions sont issues des organismes de standardisations (OSI, IETF, DMTF) qui utilisent certaines terminologies spécifiques à leurs domaines d'applications. Ces définitions sont très larges et couvrent la totalité de l'activité de gestion. Elles restent cependant trop générales pour permettre la compréhension de la performance de la gestion. La définition ci-dessous permet d'affiner le concept de gestion, tel que nous le comprenons et sur lequel nous travaillons.

[19] Dans ce manuscrit, nous employons indifféremment les termes gestion ou supervision.

Définition 1 *Gérer un système, c'est l'initialiser, le surveiller et le contrôler afin qu'il satisfasse les demandes des utilisateurs et les contraintes des propriétaires.*

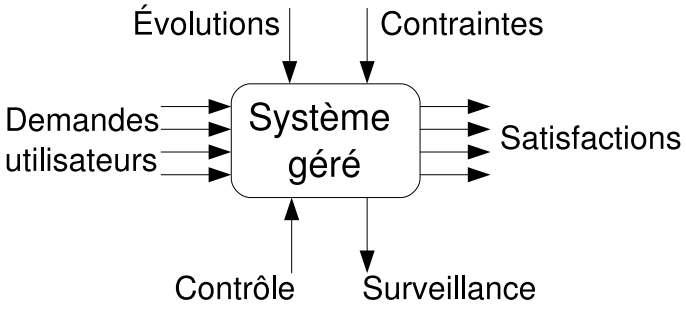


FIG. 2.1 – Abstraction de fonctionnement d'un système géré.

Cette définition issue de [61, 86] fait apparaître comme l'indique le tableau 2.1 trois processus d'activités de base : **l'initialisation**, **la surveillance** et **le contrôle**. Un système géré, comme l'indique la figure 2.1, est tout composant réseau, (équipement, un réseau, le trafic sur un réseau) ou logiciels (système, application, service) qui doit répondre aux besoins des utilisateurs. Pour chaque composant, les fonctions de gestion diffèrent mais les objectifs restent les même : initialiser, surveiller et contrôler. En effet, cela implique l'initialisation de ces différents composants (gestion de la configuration). En cas d'absence d'erreurs, ces composants se mettent en service et la phase opérationnelle commence. Au cours de cette phase, la gestion surveille ces composants pour s'assurer de l'absence de problèmes ou erreurs qui empêchent leurs bon fonctionnement. Dans le cas de détection d'un problème, le composant introduisant ce problème est identifié, isolé et réparé (gestion de fautes). Si ce composant est irréparable, il doit être remplacé par un nouveau qui aura besoin d'être initialisé. Pour améliorer la performance d'un service et mieux répondre aux usagers, de nouveaux composants peuvent être déployés. Ces nouveaux composants ont souvent besoin d'être reconfigurés. La surveillance de tout composant réseau ou logiciel est cruciale pour détecter les changements au niveau de la charge qui y est exercée (que se soit au niveau du demande de trafic ou de calcul). À la détection d'un tel changement, les paramètres des composants impliqués dans le service fourni sont ajustés pour optimiser les performances (Gestion de performance). Pour résumer, il est intéressant de remarquer que les fonctions classiques de gestion définies par le modèle OSI [43] qui sont, fautes, configuration, *accounting*, performance et sécurité sont regroupées sous la forme des trois processus d'activité que nous avons introduits précédemment : Initialisation, Surveillance et Contrôle. D'après ces définitions des activités de

Initialisation	Surveillance	Contrôle
Configurer un composant avant sa mise en opération.	Observer le fonctionnement d'un composant au cours de sa phase opérationnelle.	Altérer le comportement d'un composant sans interruption de ses opérations.

TAB. 2.1 – Processus des activités de base de la gestion de réseaux et de services.

gestion, on s'aperçoit que ses activités sont de grande importance pour le maintient d'un réseau ou d'un service afin de garantir les bénéfices financiers du fournisseur de services ou d'un opérateur Internet. Cet objectif se traduit par la garantie de la qualité de ces services en les

surveillants, et par la rapidité de leurs déploiements (initialisation) avec un moindre coût. Pour assurer les activités qui leurs sont confiées et répondre à leurs objectifs plusieurs approches et modèles de gestion ont été proposés.

2.2.1 Entités de gestion

Dans le modèle que nous proposons, nous faisons abstraction du type de l'entité de gestion proposée dans les approches traditionnelles (gestionnaire, agent). En revanche, nous proposons une définition d'une entité de gestion indépendamment de son rôle classique dans un système de gestion :

Définition 2 : *Une entité de gestion est toute entité capable de produire ou de consommer des variables de gestion. La production ou la consommation d'une variable se fait suite à l'appel à une méthode de gestion.*

Par exemple, un gestionnaire est une entité consommatrice de variables de gestion suite à l'appel d'une méthode de lecture (*get*) vers un agent qui représente une entité de gestion productrice. Nous rencontrons aussi dans la littérature un autre type d'entité de gestion que sont les passerelles de gestion et les sous-gestionnaires qui font la liaison entre deux protocoles de gestion différents, voir les mêmes. Par exemple, une passerelle SNMP/XML assure la liaison entre un protocole de gestion basé sur SNMP et un protocole de gestion à base de XML [57]. Il s'agit d'un type d'entité de gestion qui est à la fois producteur et consommateur et agit comme un relais entre différentes entités. Nous aboutissons ainsi à trois types d'entités de gestion définies selon la façon avec laquelle elles manipulent les variables de gestion : source, puits et relais. Il faut noter qu'avec notre terminologie, un objet géré est aussi une entité de gestion, puisqu'il produit des variables de gestion pour une autre entité de gestion qui est l'agent dans le cas d'un processus d'activité de surveillance où les consomment dans le cas d'une activité de contrôle ou d'initialisation. La communication entre l'agent et l'objet géré est régie par des protocoles de communication souvent non standardisés.

2.2.2 Modèle gestionnaire-agent

Dans ce travail, nous nous sommes focalisé sur des systèmes de gestion basés sur le modèle gestionnaire-agent. Ce choix est dû au fait que ce modèle est la pierre angulaire de nombreuses approches de gestion. En effet, le modèle gestionnaire-agent est le modèle normatif dans de multiples approches (normes OSI, normes de l'UIT-T, standards de l'IETF, documents du DMTF) pour réaliser les tâches de gestion des réseaux et services. Ce modèle comme son nom l'indique s'appuie sur deux entités à la manière qu'une architecture client-serveur. L'agent est intégré au niveau du système géré et fournit les fonctionnalités permettant au gestionnaire d'initialiser, surveiller et contrôler le comportement du système géré. Pour illustrer ce modèle, on utilise souvent un parallèle avec le monde de l'automatique qui propose de voir le gestionnaire, agent, système géré comme un système à boucle fermée [68]. La figure 2.2 illustre ce parallèle en montrant que la gestion consiste à introduire une boucle de régulation autour d'un système qui est initialement libre et dont l'état n'est donc pas contrôlable. L'entrée d'un tel système asservi est une consigne, qui dans le cadre des réseaux, peut être un objectif à atteindre (dans le cadre d'une entreprise, une administration ou une université), ou un contrat de service (dans le cadre où le service délivré est payant). Le système géré représente aussi bien une infrastructure matérielle, comme des machines, des routeurs ou des serveurs que des services, tels que la messagerie électronique, un

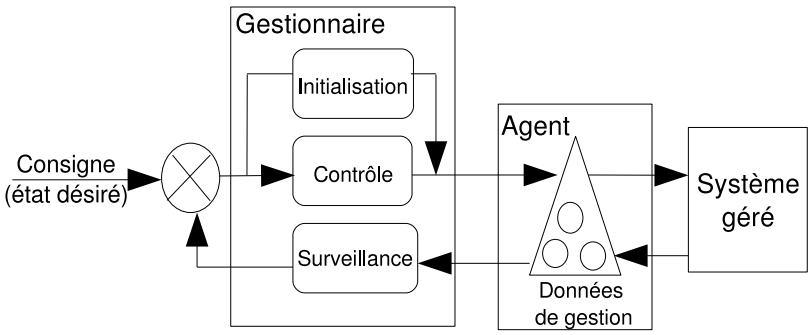


FIG. 2.2 – Analogie entre le modèle gestionnaire-agent et le modèle en boucle fermé en automatique.

réseau virtuel privé (VPN) ou un pare-feu. La sortie du système se caractérise par son état, qui inclut par exemple sa performance, sa longévité ou son coût.

L'intégration d'une infrastructure de gestion basée sur ce modèle à un tel système va consister en l'ajout des trois processus d'activité présentés précédemment.

Le premier processus, qui concerne la surveillance, représente la capacité d'observer et d'obtenir une vue abstraite du système supervisé. Différents moyens permettent d'accomplir cette tâche. Les deux principaux sont (1) le monitorage qui consiste à surveiller le système par le biais d'une scrutation régulière[20] et (2) la mise en place d'alarmes[21] qui informent de l'apparition de problèmes sur le système ou sa dérive hors des limites de fonctionnement fixées. Le traitement des alarmes pose un défi majeur qui est la notification rapide de problèmes au gestionnaire. Celui-ci doit réagir aussi rapidement que possible afin d'éviter la dégradation des services et les pertes financières liées. En effet, un aspect temps réel ou quasi-temps réel des actions est requis pour atteindre cet objectif.

Le second processus d'activité de la gestion concerne le contrôle. Elle représente la capacité à agir sur le système pour le maintenir dans des limites de fonctionnement établies dans les objectifs. L'action principale du contrôle est le réglage[22] du système supervisé au cours de son fonctionnement.

Le troisième processus est l'initialisation. De nombreuses actions sont comprises dans l'initialisation comme : l'installation, la mise en œuvre, la maintenance et la mise à jour du système supervisé. L'initialisation peut être considérée comme un cas de contrôle mais à la différence qu'elle est effectuée hors de la phase opérationnelle du système.

On s'aperçoit que ces processus d'activités mis en œuvre par le gestionnaire et l'agent, introduisent une charge de calcul et de communications qui sont souvent non négligeables. Pour réaliser leurs tâches, ces processus doivent être efficaces en terme de transport de requêtes, de traitement de réponses et de l'ordonnancement des tâches. En effet, la réalisation de chacune de ces activités nécessite l'échange de messages entre le gestionnaire et un agent. Cet échange introduit un coût de communication sur le réseau et sa logique applicative (l'algorithme de gestion) introduit une charge de calcul sur les différentes entités impliquées.

Au cours du cycle de vie du réseau géré, différents équipements sont ajoutés venant de différents vendeurs. Au même moment, de nouveaux utilisateurs sont abonnés demandant une

[20] appelé *polling*.
[21] appelé *reporting* pour SNMP ou *notification* pour JMX.
[22] appelé *tuning*

variété de services communicants. Ainsi, les dimensions des réseaux grandissent et la nécessité de garder son fonctionnement sous contrôle s'accroît. En revanche, cet objectif devient difficile à atteindre si les processus d'activités et les entités de gestion deviennent inefficaces. Cette éventuelle inefficacité est due non seulement aux caractéristiques des systèmes gérés mais aussi aux caractéristiques intrinsèques des systèmes de gestion qui rendent l'efficacité plus difficile à atteindre. Dans la suite, nous allons présenter ces caractéristiques intrinsèques qui affectent l'efficacité des systèmes de gestion.

2.2.3 Caractéristiques

Actuellement, on recense plusieurs approches pour la gestion des réseaux et services [77, 71]. Ces approches qualifiées de traditionnelles[23] s'appuient sur le modèle gestionnaire-agent présenté dans la section précédente. En effet, d'un point de vue performance et efficacité ces approches malgré leurs diversités ont été développées pour faire face à des caractéristiques communes qui sont les facteurs d'échelle et la dynamique. Ces facteurs ont notamment introduit au niveau du système de gestion le besoin de collecter de larges volumes de données, puis de les analyser et réagir en cas de détection d'anomalies.

L'élément commun est la variable de gestion

Malgré la diversité des approches, des architectures, des technologies pour la gestion de réseaux et de services, elles ont toutes en commun élément qui est la variable de gestion. Une variable est décrite selon un modèle d'information standard (SMI[24] pour SNMP, par exemple) qui fait abstraction d'un objet réel d'un système géré. À un instant donné, à chaque variable est associée une valeur qui représente une valeur d'un objet du système géré à initialiser, à surveiller ou à contrôler.

Il est évident que cet élément est commun puisque l'objectif d'un système de gestion quelque soit l'approche sur laquelle il repose est de transférer des variables d'états du système géré vers un point de décision ou vice-versa. Un paramètre important d'une variable est sa fréquence de changement. Le changement de la valeur d'une variable de gestion se présente sous différents angles de vue selon son point d'observation, comme l'indique la figure 2.3.

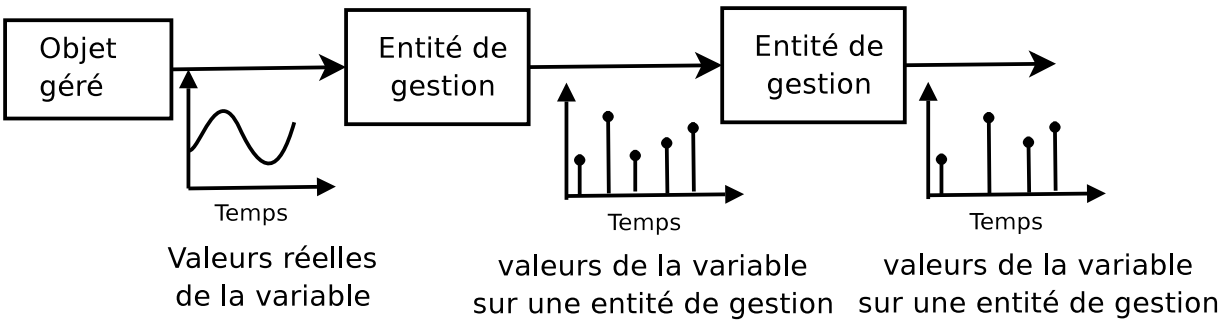


FIG. 2.3 – Points d'observations des valeurs d'une variable de gestion.

[23] *A priori* à cause de l'apparition d'une nouvelle vague d'approches ces dernières années, comme la gestion autonome basée sur la théorie du contrôle qui se présente comme le prochain standard pour la gestion des réseaux et de services.

[24] Structure of Management Information.

Sur l'objet géré, la fréquence réelle de changement de la variable est celle de la ressource réelle qu'elle représente. L'observation de cette variable par une entité de gestion possède éventuellement une fréquence d'observation de changement différente. Dans le cas où des perturbations se présentent au niveau de ces différentes entités de gestion, la vue obtenue de cette variable peut être biaisée par rapport à la vue réelle de l'objet qu'elle représente [129]. Ces perturbations sont notamment dues aux délais importants lors du transfert des valeurs de la variable d'une entité à une autre. Cela se traduit par la déviation de la valeur obervée par rapport à sa valeur réelle sur le système géré. Ainsi, la performance de la supervision joue un rôle important sur le cycle de vie du système géré. Une vue observée biaisée du système géré entraîne une perte du contrôle.

Nombre de gestionnaires et d'agents

Le nombre de gestionnaires d'une infrastructure de gestion, noté m représente son degré de décentralisation. Par exemple, si le trafic généré entre un gestionnaire et l'ensemble des agents qui y sont attachés dépasse la limite de bande passante allouée à la gestion (généralement cette limite est de 5% de la bande passante minimale disponible du réseau [105]), une approche plus décentralisée deviendra plus appropriée. Une des raisons de l'augmentation du volume du trafic de gestion est une haute intensité de demande de scrutation ou un nombre considérable d'agents rattachés à un seul gestionnaire. Par conséquent, le nombre de gestionnaires m est proportionnel au nombre n des agents afin de distribuer les charges de calcul et réseau entre les différents gestionnaires et faire face aux nombres grandissants des équipements réseaux, des applications et des services. Comme le montre la figure 2.4, l'évolution de la vitesse de processeurs suivant la

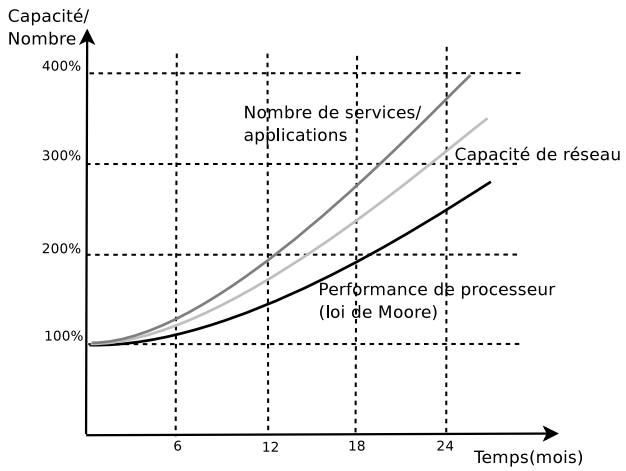


FIG. 2.4 – Évolution du facteur d'échelle des systèmes et des services.

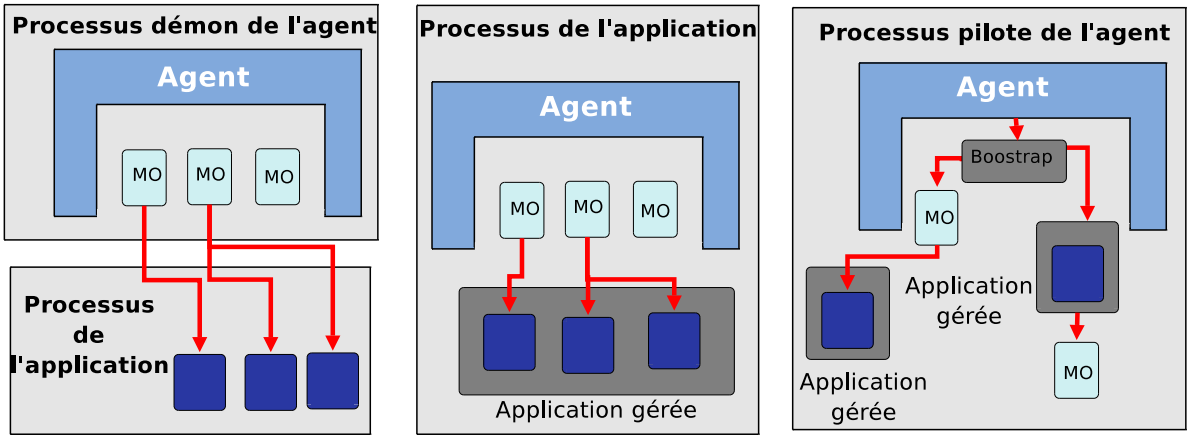
loi de Moore a engendré une augmentation de la capacité des réseaux en terme de débit. Cela a engendré l'augmentation du nombre des applications, des services et de leurs utilisateurs. Tous ces facteurs ont privilégié une gestion plus distribuée. Mais l'engouement pour des approches plus distribuées a conduit à un nombre considérable d'agents et de gestionnaires. En effet, la taille moyenne d'un système de gestion est de l'ordre de $m \times n \times r$, avec r le nombre de systèmes gérés par agent.

Connectivité entre les différentes entités

Un autre paramètre important d'une approche de gestion est le nombre de relations de communications inter-gestionnaires, noté c. Ce paramètre mis en relation avec le nombre de gestionnaires m nous indique le degré de connectivité d'une approche de gestion, noté par C, défini par : $C = \frac{c}{m}$. Ce degré de connectivité identifie la redondance de l'approche et évidement sa robustesse ainsi que le niveau de coopération pour réaliser une tâche de gestion. En effet, plus ce degré est élévé plus l'approche de gestion est robuste et distribuée. En revanche, l'augmentation de la la connectivité entre les gestionnaires introduit une forte complexité de leur coordination. Il faut définir pour chacun de gestionnaires ses tâches respectives pour qu'il puisse converger vers l'objectif global du maintient des systèmes gérés dans des états désirés. Cela nécessite entre autre une transparence au niveau de la distribution des fonctions et au niveau de l'adressage de ces entités. Ce dernier aspect est un nouveau défi auquel sont confrontés les systèmes de gestion notamment en cas d'une forte connectivité entre ses entités dans des environnements dynamiques. En effet, la distribution des fonctions de gestion nécessite l'augmentation de la connectivité entre les entités de gestion, mais en cas d'une dynamique spatiale ou temporelle de ces entités, le plan de gestion doit être capable de les localiser d'une façon transparente sans perturber son fonctionnement [110].

Modèles d'intégration de l'agent

Le modèle d'intégration d'un agent de supervision au niveau du système géré présente la façon par laquelle l'agent accède à ce dernier. Ces modèles sont clairement définis dans une approche de gestion reposant sur la technologie JMX. Ils se présentent sous la forme de trois modèles qui sont le démon, le composant et le pilote (*driver*). En revanche ces modèles sont génériques et sont applicables à d'autres approches de gestion. Dans une approche SNMP basée sur le protocole AgentX [58], ces modèles sont valables pour représenter l'intégration de sous agents au niveau d'un agent maître. La figure 2.5 présente les trois modèles d'intégration d'un agent au



(a) (b) (b)

FIG. 2.5 – Modèles d'intégration d'un agent par rapport au système géré. (a) Démon. (b) Composant. (c) Pilote.

sein d'une application gérée. Dans le modèle démon 2.5 (a) le processus de l'agent est séparé de celui de l'application gérée. Un mécanisme de communication est indispensable entre les deux processus pour accéder aux données de gestion. Ce modèle est couramment utilisé pour gérer

des applications déjà existantes reposant sur des technologies autre que Java où le coût de son portage est important. Dans le modèle composant 2.5 (b), l'application gérée et l'agent partagent le même processus. Dans ce modèle c'est l'application qui est responsable de l'instanciation de l'agent. Les deux composantes partagent les mêmes ressources physiques (réseaux, cpu, mémoire). Le conteneur de servlet *Tomcat*[25] est l'un des exemples des applications qui repose sur ce modèle.

Dans le modèle pilote (*driver*) 2.5 (c), l'agent est le noyau du système géré. L'initialisation de l'application gérée s'effectue à travers l'agent. Ce modèle nécessite l'intégration de l'agent et sa conception dès le développement de l'application à gérer. Le serveur applicatif JBoss[26] repose sur ce modèle d'intégration.

Concurrence des opérations

La communication entre les gestionnaires et les agents se présente souvent sous deux formes : séquentielle ou concurrente. Dans la première les requêtes sont envoyées séquentiellement, c'est à dire une nouvelle requête n'est envoyée qu'après une éventuelle réception de la réponse de la requête en cours. Dans la deuxième forme les requêtes sont parallélisées, envoyées d'une façon concurrente et les réponses sont reçues au fur et à mesure. Ces deux formes influent différemment la performance de la gestion notamment en terme de délais comme l'indique la figure 2.6. Pour

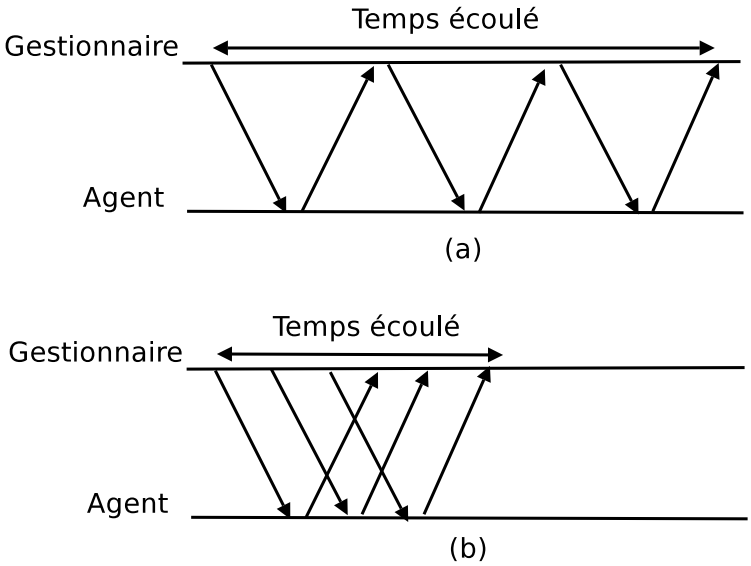


FIG. 2.6 – Modèles de concurrence des opérations de gestion. (a) opérations séquentielles. (b) opérations concurrentes

un système de gestion nous définissons le degré de concurrence, noté Co_M, au niveau d'une entité de gestion à un instant t comme étant [116] : $Co_M = \sum_{i=1..n} Co_o(o_i, t)$, avec o_i une instance d'une opération et $Co_o(o_i, t)$ désigne le nombre d'opérations en concurrence à un instant t. En effet, deux opérations o_1 et o_2 sont en concurrence à un instant t si : $s(o_1) < t < e(o_1)$ et $s(o_2) < t < e(o_2)$, avec $s(o_i)$ le temps de début de l'opération et $e(o_i)$ le temps d'achèvement de l'opération.

[25] http ://tomcat.apache.org
[26] http ://www.jboss.org

Granularité des opérations

La granularité d'une opération de gestion se traduit par la taille des données sur lesquelles elle opère. Nous distinguons deux types d'opérations selon leur niveau de granularité. Le premier type est unitaire ; l'opération porte sur une ou plusieurs variables simples de gestion. Dans cette catégorie on rencontre essentiellement l'opération *Get* de SNMP ou *getAttribute* pour JMX. Dans ce cas, nous définissons la multiplicité de l'opération comme étant le nombre de variables sur lequel opére l'opération. Le deuxième type est logique où l'opération porte sur une variable de gestion composée de plusieurs variables unitaires. Dans cette catégorie, nous identifions par exemple les opérations de type *getTable* définies dans la cadre d'une gestion basée sur XML. D'un point de vue performance, comme le montre la figure 2.7 la granularité de l'opération influe notamment sur ses délais et le nombre de messages introduits sur le réseau [142, 138, 123].

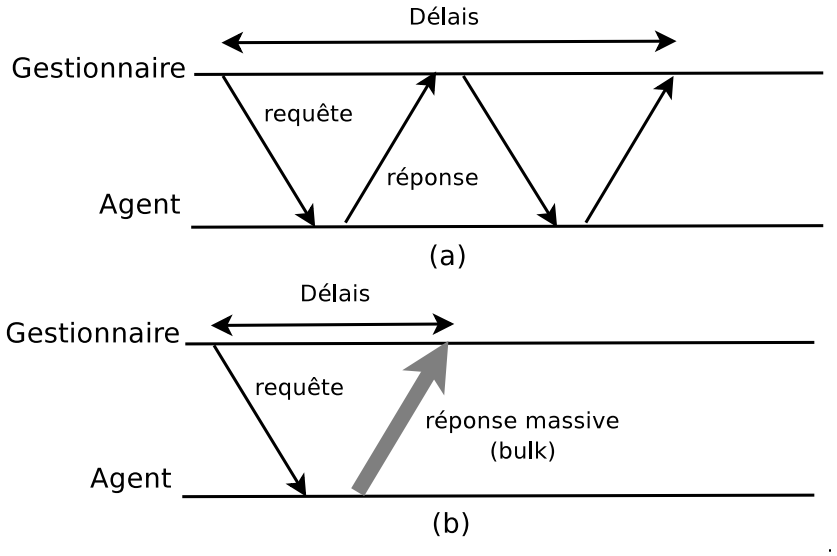


FIG. 2.7 – Granularité des opérations de gestion. (a) opérations unitaires. (b) opération logique.

Hétérogénéité de la gestion

Une infrastructure de gestion déployée chez un opérateur ou un fournisseur de services est rarement homogène. Cette hétérogénéité est liée à l'hétérogénéité intrinsèque des systèmes gérés. Une telle infrastructure opérationnelle repose souvent sur des supports de communications différents (WiFi, satellites, Boucles Locales Radio), des usages différents (mobilité, ubiquité) et des terminaux variés (ordinateurs portables, PDA, téléphones 3G). Ceci génère une forte diversité au niveau de l'infrastructure de gestion. De ce fait, il n'est pas surprenant de devoir gérer des connexions, des équipements et des services traversant des domaines de réseaux aux comportements très différents. En effet, ces systèmes à gérer possèdent des caractéristiques intrinsèquement différentes, comme la capacité de communications (bande passante, débits générés) et de traitements ou la présence des perturbations réseaux (pertes de paquets, déconnexions temporaires). En revanche, la gestion est toujours vue comme homogène. Afin de qualifier un système de gestion d'homogène, nous avons identifié certains critères auxquels il doit y répondre :

1. Tous les domaines de gestion sont homogènes si leurs architectures de gestion respectives sont identiques.

2. Tous les sites sont homogènes si leurs activités de gestion respectives sont identiques.
3. Tous les agents de gestion sont homogènes si leurs fournisseurs d'implantations et leurs plates-formes physiques sont identiques.
4. Toutes les charges de gestion sont homogènes si leurs services et objets de gestion sont identiques.
5. Tous les services de gestion sont homogènes si leurs opérations de gestion sont identiques.
6. Tous les objets de gestion sont homogènes si :
   - Le nombre d'objets de gestion à collecter depuis les agents est le même.
   - Les types des objets de gestion à collecter depuis les agents sont les mêmes.

Cette hypothèse d'homogénéité est généralement implicite dans la majorité des cas d'étude de la performance de la gestion [108, 139]. Nous la considérons trop forte d'un point de vue performance parce qu'elle ne reflète pas une réelle plate-forme de supervision. Toutefois, quelques études [140, 142] ont varié les implantations des agents de gestion (SNMP, services web) d'une plate-forme de gestion. Elles ont montré une réelle différence de performances entre ces différents agents. Afin d'avoir un critère d'homogénéité d'une plate-forme de gestion, nous définissons son degré d'homogénéité comme étant le produit des degrés d'homogénéité de ses groupes de composants. Le degré d'homogénéité, noté $GrHom$ d'un groupe r, est défini par :

$$GrHom(r) = \frac{1}{\text{nombre de classes d'équivalences différentes dans r}} \tag{2.1}$$

Ainsi, le degré d'homogénéité d'un système S, noté $SysHom$ est :

$$SysHom(S) = \prod_{1 \leq i \leq n} GrHom(r_i) \tag{2.2}$$

Nous qualifions un système d'homogène si son degré d'homogénéité est égal à 1. Donnons un exemple d'une infrastructure de gestion composée de 5 agents SNMP situés dans un seul site, soumis à une charge identique et 2 agents parmi les 5 sont des agents fournis par Cisco et le reste sont des agents fournis par NET-SNMP [27]. Le degré d'homogénéité du groupe d'agents de gestion est de $\frac{1}{2}$ (nous avons deux types d'agents différents), le degré d'homogénéité des sites de gestion est 1 (un seul site), le degré d'homogénéité des charges est 1, ce qui nous ramène à un degré d'homogénéité de l'infrastructure de gestion égal à $\frac{1}{2}$.

2.2.4 Classification des approches de la gestion

[95] présente un modèle de classification des approches de gestion selon leur degré de connectivité et le nombre de gestionnaires et d'agents sur lesquelles elles s'appuient. Une approche de gestion peut être déployée de différentes manières qui dépendent de son modèle de distribution des fonctions de gestion caractérisé par les deux critères présentés précédemment. On trouve par exemple, dans un contexte d'entreprise une infrastructure de gestion déployée sur le réseau local pour gérer une dizaine voire une centaine de systèmes. C'est le cas par exemple d'une université ou d'une institution gouvernementale. Pour ce type d'environnement, il semble envisageable d'utiliser une infrastructure de gestion centralisée pour assurer sa supervision. D'un autre côté, on trouve des infrastructures de gestion déployées à l'échelle de l'Internet avec des milliers de systèmes à gérer (par exemple, des *data centers* hébergeants des services comme *Google*), qui sont répartis sur des domaines administratifs différents. La classification des approches tradi-

[27] http ://net-snmp.sourceforge.net/

Nombre de gestionnaires	Connectivité	Approche
$1 = m$	$C = 0$	Gestion centralisée
$1 < m \ll n$	$0 < C < 1$	Gestion à faible distribution
$1 \ll m < n$	$1 \leq C \ll m$	Gestion à forte distribution
$m \approx n$	$1 \ll C \leq 1$	Gestion coopérative

TAB. 2.2 – Classification des approches de gestion selon le nombre de gestionnaires m, le nombre d'agents n et le degré de connectivité C.

tionnelles de la gestion est illustrée par le tableau 2.2 qui les recense selon les critères décrits précédemment. La figure 2.8 montre le modèle organisationnel de ces différentes approches. On peut constater que ces approches sont construites autour du modèle gestionnaire-agent avec le degré de connectivité et un nombre de gestionnaires différents.

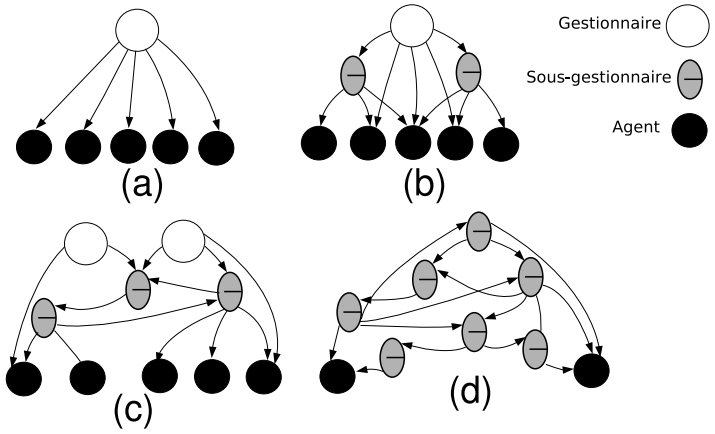


FIG. 2.8 – Extrait de [95]. Exemple d'approches de gestion présentant différents niveaux de distribution. (a) centralisée. (b) faible distribution. (c) forte distribution. (d) coopérative.

2.3 JMX pour la gestion de services basés sur Java

Ces dernières années, les applications basées sur la technologie Java sont devenues de plus en plus répandues pour le développement des solutions et des services d'E-commerce, mobiles, embarquées ou autres [59]. La caractéristique majeure de ces solutions basées sur la technologie Java est la dynamicité. Cette dynamicité se traduit par le démarrage et l'arrêt à chaud des composants, la migration d'une application d'un environnement à un autre, etc.

Plusieurs industriels[28] ont développé le framework JMX [97, 99], pour aboutir à une approche standard pour la supervision et la gestion des applications basées sur la technologie Java.

[28] Alcatel-Luscent, Sun, IBM, Bull, TIBCO, Motorola, Borland, IONA, Apache Software Foundation, Lutris, Schmid Telecom, BEA Systems, MGE UPS Systems.

2.3.1 Les bases de JMX

L'approche JMX [97] fournit une architecture et une API[29] permettant la supervision des ressources pouvant s'interfacer à une JVM (Java Virtual Machine). Ces ressources incluent des applications logicielles ainsi que des périphériques physiques. Tout comme pour la majorité des systèmes de supervision, JMX est basé sur une architecture (voir figure 2.9) en trois couches : le niveau instrumentation, le niveau agent et le niveau superviseur. Le concept de base de l'archi-

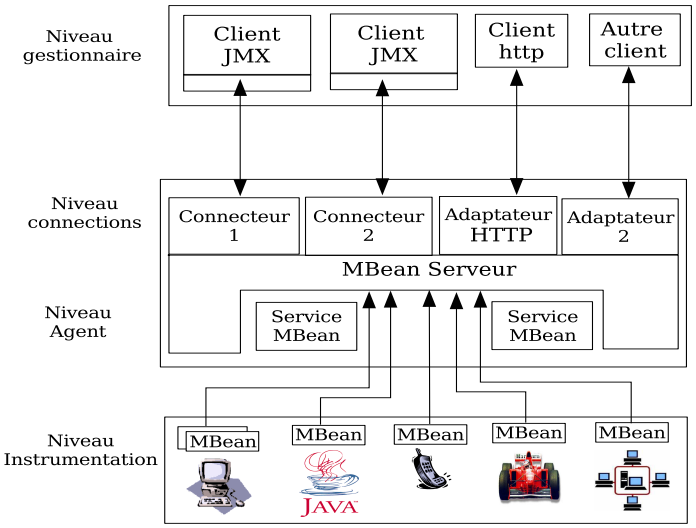


FIG. 2.9 – L'architecture générale de JMX.

tecture JMX est le MBean. Un MBean est un objet Java qui respecte un nommage et un patron de programmation. Il est utilisé pour instrumenter les ressources à gérée. Il forme la base du niveau instrumentation. Tout MBean est accédé à travers un conteneur appelé *MBeanServer*. Il offre à l'agent l'ensemble des méthodes pour créer, détruire un MBean, lire des attributs, modifier des attributs et invoquer des méthodes sur les MBeans. Afin de permettre aux applications de supervision d'accéder aux MBeans, l'architecture JMX propose deux types d'accès à distance : les connecteurs et les adaptateurs de protocoles. Le connecteur est présent du côté serveur et client. Il permet à un client de faire des appels de méthodes à distance sur le serveur des MBeans. Typiquement un connecteur peut être bâti au dessus de RMI (Remote Method Invocation) [66]. Les adaptateurs de protocoles sont présents uniquement côté serveur. Ils assurent la liaison entre des protocoles spécifiques (par exemple SNMP ou HTTP) et les services locaux d'un serveur de MBeans. Le niveau superviseur comporte l'ensemble des applications de supervision et outils d'accès aux informations fournies par les agents. Dans le monde JMX, 5 types de MBean adaptés chacun à un profil d'instrumentation donné sont définis. Ces MBeans sont :

– Le MBean standard : il s'agit du plus simple des MBeans. Il doit implémenter sa propre interface de supervision qui définit les signatures des méthodes d'accès aux attributs et opérations disponibles depuis un agent.

[29] JMX offre une interface normalisée permettant à un développeur d'intégrer facilement de code d'instrumentation dans une application Java.

- Le MBean dynamique : La dénomination dynamique pour ces MBeans vient du fait que l'interface de supervision de ces MBeans n'est pas figée à la compilation mais fournie par les MBeans eux-mêmes à l'exécution.
- Le MBean modèle : Il s'agit de MBeans dynamiques qui fournissent un cadre d'utilisation générique. En effet, ils permettent la création des MBeans sans codage de classes et ceci à partir d'un MBean générique via un service d'initialisation.
- Le MBean ouvert : Ce sont des MBeans dynamiques spécifiques. En effet, ils possèdent les mêmes caractéristiques que ces derniers mais restreignent les types de données pour leurs attributs, paramètres de constructeurs et opérations à un sous ensemble défini de classes java sérialisables dans un format binaire pour être transporter sur le réseau.
- Le MXBean : Ce sont des MBeans qui permettent d'encapsuler des données de gestion de différents types sans que le client soit forcément configuré pour supporter ces types. Ils se basent sur les types restreints des MBean ouverts pour masquer les types utilisateurs complexes définis au niveau de MXBean. Ces MBeans sont prévus pour remplacer les MBean standards dans la prochaine spécification de JMX[30].

L'accès aux valeurs des attributs de ces MBeans s'effectue selon deux modes : la scrutation ou la notification. Le mode de scrutation consiste à appeler les méthodes exposées par le MBean serveur pour récupérer ou altérer la valeur d'un ou plusieurs attributs d'un MBean. Au niveau de JMX, ces méthodes sont les suivantes : *getAttribute*, *setAttribute*, *getAttributes*, *setAttributes* et *invoke*. Le deuxième mode est la notification. Le principe de ce mode est similaire à celui des évènements Java. Une notification est soumise au gestionnaire à l'apparition d'un évènement comme par exemple, le dépassement de la valeur d'un attribut d'un seuil fixé. Le mécanisme de notification dans JMX suit un modèle de type émetteur/souscripteur[31]. Dans ce modèle un objet joue le rôle d'un émetteur (*publisher*). Les autres objets qui dépendent des changements de cet émetteur sont les souscripteurs (*listeners*). L'émetteur maintient une liste de ces souscripteurs. Lorsqu'un composant veut devenir un abonné, il utilise l'interface de souscription fournie par l'émetteur. Au niveau de JMX le MBean serveur expose les méthodes *add/removeNotificationListener* pour que des objets peuvent s'abonner ou se désabonner pour la réception des notifications depuis un MBean. Le mécanisme de notification au niveau d'un MBean peut être implanter selon deux manières différentes : l'utilisation du service de *monitoring* fourni par l'agent ou l'implantation directe des notifications au niveau du MBean. Le service de *monitoring* permet de surveiller des attributs d'autres MBeans localement et notifier les souscripteurs en cas d'un évènement. Dans JMX, le mécanisme de transfert des notifications aux souscripteurs repose sur l'utilisateur d'un buffer au niveau de l'agent. En effet, les notifications émises par les différents MBean sont stockées dans un buffer. Ensuite, c'est à la charge de connecteur de solliciter ce buffer et de transférer ces notifications aux souscripteurs.

2.3.2 Sources de variation de performances de JMX

L'analyse du modèle du framework JMX révèle les sources majeures de variabilité affectant sa performance. Nous allons rapidement les passer en revue.

[30] La prochaine spécification de JMX est le JSR 255. Ce JSR n'est pas encore disponible au moment de l'écriture de ce manuscrit.

[31] En anglais : publisher/subscriber.

La couche d'instrumentation

L'instrumentation est le processus d'exposition d'une ressource gérée à travers un MBean. Les interfaces d'un MBean exposent :

- les constructeurs comme tout objet Java pour son instanciation ;
- les attributs qui peuvent être lus ou modifiés ;
- les méthodes permettant de modifier le comportement d'un MBean (par exemple une opération de ré-initialisation de l'application) ;
- les méthodes envoyant des notifications correspondant à des évènements particuliers.

Cette couche est sensible aux nombreux facteurs qui peuvent influencer la performance d'une application de gestion basée sur JMX. Les principaux facteurs sont : le type d'instrumentation (interne ou externe), le type des MBeans, leur nombres, le nombre d'attributs qu'ils exposent, le modèle employé pour la collecte de données depuis la ressource gérée (*pushing* ou *pulling*). Nous identifions deux facteurs important dans cette couche : le type de l'instrumentation et la granularité des MBeans. Le type d'instrumentation dépend essentiellement de l'objectif des activités de gestion. Une instrumentation externe interagit avec la ressource gérée en dehors de son processus d'exécution. Elle se base sur des fichiers[32], des outils et des programmes pour accéder aux données et contrôler la ressource gérée. Des activités de gestion comme le déploiement, l'installation, la configuration et la surveillance sont supportées par ce type d'instrumentation. Le deuxième type est l'instrumentation interne. Cette instrumentation est plus intrusive, puisque elle nécessite l'introduction de code au niveau de la ressource gérée pour répondre aux besoins des activités de gestion. L'instrumentation JMX est de type interne, mais cela ne l'empêche pas de fournir une instrumentation externe pour certaines applications de gestion. La granularité du MBean est un facteur important lors de la conception de la couche instrumentation. Elle définit le nombre de MBeans, ainsi que le nombre d'attributs par MBean. Cela se traduit par la définition de plusieurs MBeans avec un faible nombre d'attributs ou peu de MBeans avec un nombre considérable d'attributs. Ce choix influence considérablement la performance de l'application de gestion comme nous allons le détailler dans le chapitre 5.

La couche agent

La composante principale de cette couche est le serveur de MBeans. Le serveur de MBeans est un objet Java responsable de l'exposition des MBeans aux applications de gestion. Il maintient une liste de tous les MBeans qu'il contrôle. Toutes les opérations de supervision appliquées sur un MBean sont effectuées à partir de l'interface MBeanServer du serveur de MBeans. Cette interface définit les méthodes permettant d'instancier et d'enregistrer un nouveau MBean dans l'agent, d'accéder aux attributs du MBean et d'appeler les opérations qu'il implante. D'autres services sont fournis par l'agent pour créer des relations entre les MBeans, la surveillance des attributs des MBeans par le biais de jauges et de compteurs de supervision et un service de temporisation pour l'émission des notifications ou l'exécution de certaines méthodes à des instants spécifique ou à des intervalles réguliers. Ces services sont implantés sous la forme de MBeans et enregistrés au niveau du serveur de MBeans.

Le modèle d'intégration de l'agent au niveau de la ressource gérée est un facteur important lors de la conception d'une plate-forme de gestion. Il a une influence non négligeable sur les performances. Nous avons détaillé ces trois modèles d'intégration de la gestion dans la section 2.2.3.

[32] Des fichiers journaux (logfile) entre autres.

La couche communication

Ce niveau spécifie des composants particuliers, appelés connecteurs ou adaptateurs de protocoles, permettant aux applications de supervision d'accéder à distance aux agents JMX en respectant divers protocoles de communication. L'un des avantages de JMX est sa capacité à utiliser n'importe quel connecteur (RMI, SOAP) ou adaptateur (HTTP, SNMP) pour assurer la connectivité entre les gestionnaires et les agents. En effet, dans la spécification de la couche de communication de JMX [99], un connecteur est un objet java, déployé comme un MBean au sein du MBean serveur. Ce concept générique permet à JMX d'être indépendant d'un connecteur particulier et lui offre une flexibilité de communication importante et un vaste choix d'intégration dans des plates-formes de gestion (pair-à-pair, Java Messaging System, services web, etc). Par exemple, récemment dans [102], un connecteur web-service est spécifié pour JMX pour assurer une connectivité avec des clients web-services non développés en java, et l'interopérabilité avec le protocole Ws-Management spécifié par le DMTF. Comme indiqué sur la figure 2.10, chaque

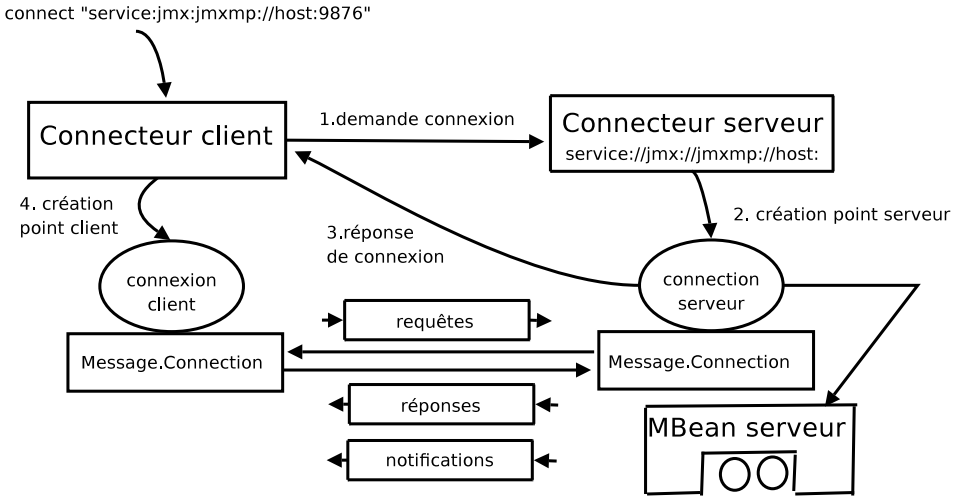


FIG. 2.10 – Extrait de [99]. Établissement d'une connexion entre les parties cliente et serveur d'un connecteur JMX.

connecteur JMX se base sur deux composantes : une partie cliente attachée à l'application cliente de gestion (un gestionnaire par exemple) et une partie serveur attachée au MBean serveur. D'un point de vue de performance, le type de connecteur choisi à une grande influence sur la performance d'un système de gestion basé sur JMX.

La couche gestionnaire

Cette couche n'est pas standardisée, étant donné que les spécifications JMX [97, 99] définissent essentiellement les couches présentées précédemment. En revanche, nous avons identifié certains facteurs qui influent sa performance. Le mode de supervision (scrutation, notification) utilisé par le gestionnaire affecte différemment la performance de l'application de gestion. Le mode scrutation introduit comme paramètres : la fréquence de scrutation, la nature de la scrutation qui est soit statique où ses fréquences sont fixes, soit adaptative où ses fréquences sont variables, le type de l'opération (unitaire ou logique), le niveau de concurrence (voir section 2.2.3). Le mode notification introduit aussi un ensemble de paramètres relatif à son fonctionnement. Parmi eux on peut citer : la fréquence de notification au niveau des agents et la nature du seuil de

franchissement (statique ou dynamique).

2.3.3 JMX au dessus de RMI

Comme spécifié dans [99], le connecteur RMI [66] est obligatoire dans toute implantation du protocole JMX. Ce connecteur se base sur l'infrastructure RMI pour assurer la communication entre un gestionnaire et un agent. RMI définit deux modes de transport, le premier est JRMP (Java Remote Method Protocol) et le deuxième est IIOP (Inter-ORB Protocol). Par défaut, c'est le mode JRMP qui est activé. Le mode IIOP a été défini pour le protocole CORBA, et son utilisation au dessus de RMI assure l'interopérabilité avec d'autres langages de programmation. Le connecteur client RMI possède deux moyens possibles pour contacter un connecteur serveur RMI. Le premier moyen est d'utiliser une URL[33] de service JMX sous forme de l'objet *stub* du serveur encodé dans un format binaire de *Base64*. Le deuxième moyen est d'utiliser un serveur de nomage et un chemin JNDI[34] spécifiant l'emplacement du stub RMI. Par exemple l'URL *service :jmx :rmi :///jndi/rmi ://agentJMX.loria.fr :1099/server* spécifie qu'un serveur de nomage écoutant le port 1099 se trouve sur la machine *agentJMX.loria.fr* et que le stub est enregistré sous le chemin */server*. Le protocole RMI [66] utilise la notion de flux pour toute communication entre le client et un agent. Le flux sortant *out* est envoyé du client vers le serveur et le flux entrant *in* est envoyé de l'agent vers le client. Chaque type de flux possède plusieurs types de messages. Au cours d'une connexion RMI, plusieurs types de messages sont échangés entre le client et le serveur. En revanche dans le contexte de JMX, deux messages RMI sont importants, ils représentent les requêtes et les réponses de supervision. Le message *Call* envoyé par le client vers l'agent pour invoquer une méthode (par exemple la méthode *getAttribute*) fournit par le MBean serveur pour récupérer la valeur d'un attribut depuis un MBean et le message *ReturnData* qui correspond à la réponse de l'agent. La figure 2.11 représente le format des messages *Call* et *ReturnData* du protocole JRMP/RMI. En revanche le protocole RMI utilise d'autres messages

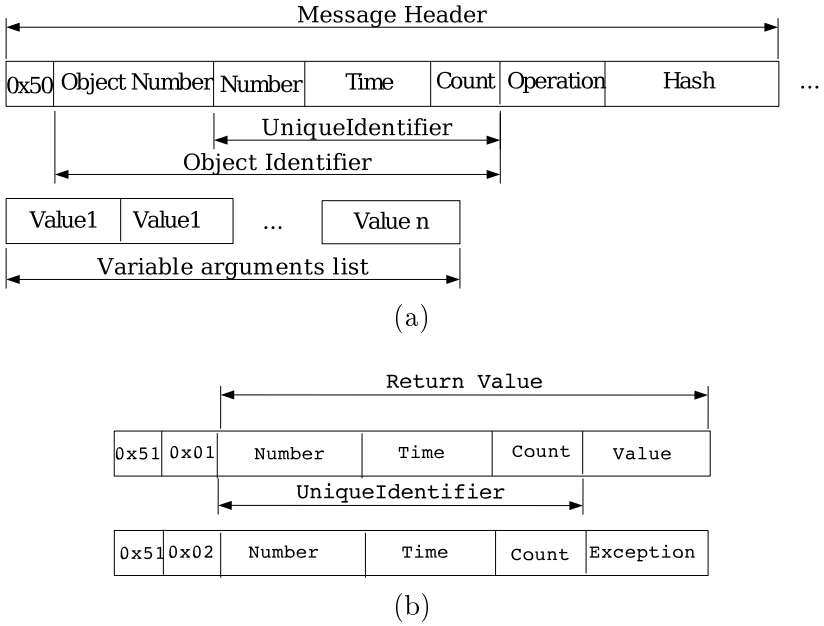
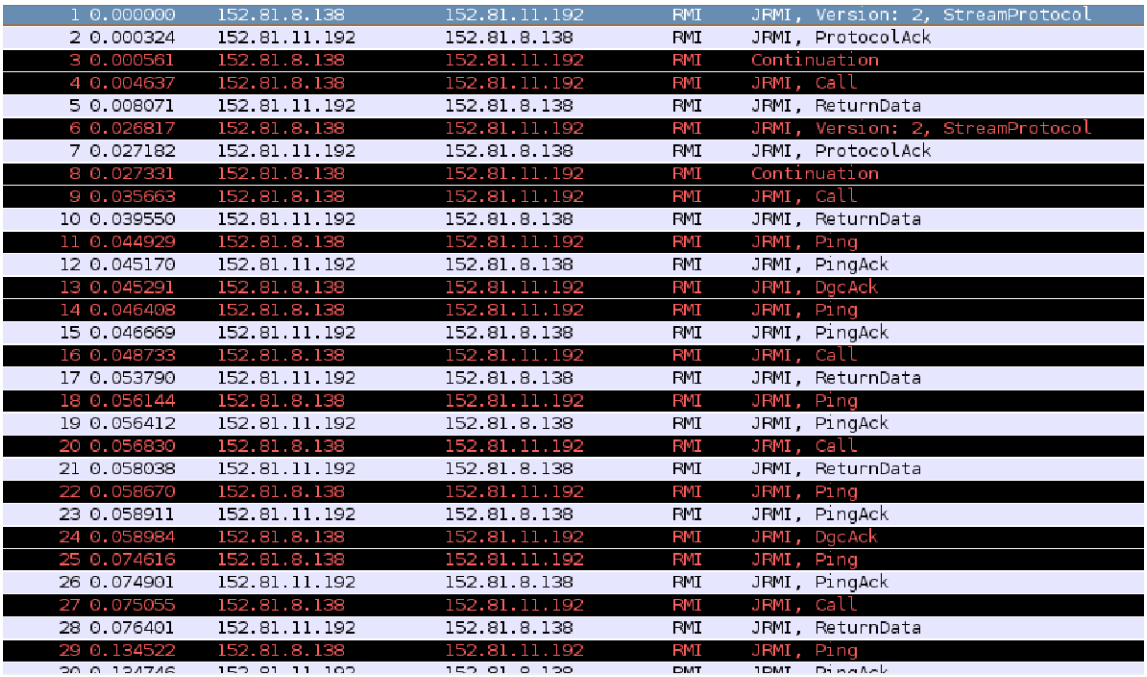


FIG. 2.11 – Format des messages RMI/JRMP. (a) le message *Call*. (b) le message *ReturnData*.

[33] URL : Universal Remote Location
[34] JNDI : Java Naming and Directory Interface

comme : *ping*, *pingAck* et *DgcAck* pour gérer la connexion entre le client et le serveur. La figure 2.12 montre un exemple d'échange de messages RMI entre un gestionnaire et un agent JMX, capturé avec l'outil Wireshark[35]. Le protocole RMI utilise le protocole de sérialisation des objets



1 0.000000	152.81.8.138	152.81.11.192	RMI	JRMI, Version: 2, StreamProtocol
2 0.000324	152.81.11.192	152.81.8.138	RMI	JRMI, ProtocolAck
3 0.000561	152.81.8.138	152.81.11.192	RMI	Continuation
4 0.004637	152.81.8.138	152.81.11.192	RMI	JRMI, Call
5 0.008071	152.81.11.192	152.81.8.138	RMI	JRMI, ReturnData
6 0.026817	152.81.8.138	152.81.11.192	RMI	JRMI, Version: 2, StreamProtocol
7 0.027182	152.81.11.192	152.81.8.138	RMI	JRMI, ProtocolAck
8 0.027331	152.81.8.138	152.81.11.192	RMI	Continuation
9 0.035663	152.81.8.138	152.81.11.192	RMI	JRMI, Call
10 0.039550	152.81.11.192	152.81.8.138	RMI	JRMI, ReturnData
11 0.044929	152.81.8.138	152.81.11.192	RMI	JRMI, Ping
12 0.045170	152.81.11.192	152.81.8.138	RMI	JRMI, PingAck
13 0.045291	152.81.8.138	152.81.11.192	RMI	JRMI, DgcAck
14 0.046408	152.81.8.138	152.81.11.192	RMI	JRMI, Ping
15 0.046669	152.81.11.192	152.81.8.138	RMI	JRMI, PingAck
16 0.048733	152.81.8.138	152.81.11.192	RMI	JRMI, Call
17 0.053790	152.81.11.192	152.81.8.138	RMI	JRMI, ReturnData
18 0.056144	152.81.8.138	152.81.11.192	RMI	JRMI, Ping
19 0.056412	152.81.11.192	152.81.8.138	RMI	JRMI, PingAck
20 0.056830	152.81.8.138	152.81.11.192	RMI	JRMI, Call
21 0.058038	152.81.11.192	152.81.8.138	RMI	JRMI, ReturnData
22 0.058670	152.81.8.138	152.81.11.192	RMI	JRMI, Ping
23 0.058911	152.81.11.192	152.81.8.138	RMI	JRMI, PingAck
24 0.058984	152.81.8.138	152.81.11.192	RMI	JRMI, DgcAck
25 0.074616	152.81.8.138	152.81.11.192	RMI	JRMI, Ping
26 0.074901	152.81.11.192	152.81.8.138	RMI	JRMI, PingAck
27 0.075055	152.81.8.138	152.81.11.192	RMI	JRMI, Call
28 0.076401	152.81.11.192	152.81.8.138	RMI	JRMI, ReturnData
29 0.134522	152.81.8.138	152.81.11.192	RMI	JRMI, Ping

FIG. 2.12 – Exemple d'échange de messages RMI entre un agent et un gestionnaire JMX, capturé avec l'outil *Wireshark*.

java [66] pour pouvoir donner une représentation binaire d'un objet avant de le faire transporter sur le réseau. Les types primitifs (*int, byte, long, float, boolean,...*) sont encodés dans un format en *ByteCode* Java défini par la spécification standard du langage Java. Les chaînes de caractères sont codées suivant l'encodage *UTF-8* qui supporte des caractères codés sur 1 octet, 2 octets ou 3 octets. Les objets utilisateurs nécessitent la définition de leurs propres méthodes de sérialisation comme indiqué dans [66]. Dans [69], les auteurs ont analysé le coût de la sérialisation Java sur laquelle est basée le protocole RMI en le comparant avec la sérialisation XML en terme de temps de sérialisation et déserialisation et l'occupation mémoire. Leurs analyses ont montré que la sérialisation binaire Java est plus efficace que celle de XML en terme de rapidité (5 à 9 fois plus rapide) et taille de données générées : la taille d'un stream sérialisé XML est 4 fois plus important qu'un stream binaire java. Plusieurs travaux [70, 45] ont analysé les performances de RMI ainsi que son efficacité comme un middleware de communication. Dans [70], les auteurs ont comparé les performances de RMI, CORBA et RMI-IIOP sous un et plusieurs clients et différents type de données (primitives et composite), en terme des temps de réponse (*round trip times*), de débit (*throughput*) et de dégradation de performances (passage à l'échelle). Le protocole RMI Java présente de meilleurs résultats que les autres. Les problèmes de performances majeurs dont souffrent ces protocoles se situent au niveau de la gestion de *threads*, des algorithmes de multiplexage de données et des primitives de bas niveau de communications (niveau sockets).

[35] http ://www.wireshark.org

2.3.4 Applications de JMX

La technologie JMX est adéquate pour instrumenter et gérer n'importe quelle application. Ces applications varient d'un téléphone portable, une suite applicative, ou un composant d'un serveur d'application. En effet, la JVM à partir de sa version 1.55 est instrumentée par JMX [100] et fournit un outil nommé *jconsole* pour sa surveillance et l'exposition de certaines statistiques (cpu, mémoire, thread, etc). Dans le monde des applications destinées aux entreprises et du E-commerce, les plates-formes java sont devenues quasiment un standard à travers l'utilisation des serveurs J2EE[36] applicatifs. Actuellement la majorité de serveurs applicatifs[37] J2EE exposent une interface de gestion JMX au moins pour leurs propres supervision [101]. Dans le monde des services web et SOA[38] la gestion de ces services par le biais de JMX commence à prendre de l'intérêt et certaines plates-formes[39] l'utilise pour la gestion des annuaires de services (UDDI : Universal Description, Discovery, and Integration). Il faut aussi noter que JMX est utilisé par JBoss[40] comme modèle interne pour ses différents composants. En effet, le serveur applicatif est construit autour du MBean serveur suivant le modèle pilote présenté au 2.3.2. Récemment, le département Australien de planification et des infrastructures a eu recours à la technologie JMX pour la gestion de son système d'information nommé *TRELIS* [41]. Ce système d'information repose sur un million de lignes de code Java et sur 5000 composants Java. Il est donc clair que l'utilisation de JMX pour la gestion des applications dans le monde Java est indispensable.

2.4 SNMP pour la gestion de réseaux

SNMP (Simple Network Protocol) [53] est un protocole de supervision de réseau proposé par l'IETF qui se veut simple dans son utilisation. Afin de pouvoir l'utiliser il est nécessaire que les équipements du réseau soient équipés d'un agent SNMP. Comme le protocole JMX, ce protocole est basé sur un modèle gestionnaire-agent (figure 2.13). Le gestionnaire émet un paquet UDP représentant une requête SNMP à destination de l'équipement du réseau. Cette requête doit contenir le nom de la communauté SNMP de lecture (respectivement le nom de la communauté de lecture/écriture) pour accéder en lecture aux variables de la MIB SNMP (respectivement en lecture/écriture). Lorsque l'agent SNMP est en mesure de répondre, il retourne, dans un paquet UDP vers l'émetteur, les valeurs de variables SNMP demandées. Il faut noter que plusieurs versions de SNMP existent. La première version du protocole est SNMPv1 qui est définie par le RFC 1157 [53]. La deuxième est SNMPv2 qui a apporté plusieurs améliorations par rapport à la première en terme de sécurité et d'opérations du protocole. Cette version a été initialement définie dans le RFC 1901 [54]. La troisième version est SNMPv3. Elle notamment apporté deux modules de sécurité : l'un pour l'authentification, l'intégrité et le chiffrement et l'autre pour le contrôle d'accès. Cette version est normalisée par le RFC [67].

[36] Java 2 Entreprise Edition : distribution Java destinée aux applications Java des entreprises.

[37] Une liste de ses serveurs est disponible dans [73].

[38] *SOA : Service Oriented Architecture* (pour architecture orientée services) définie un modèle d'interaction applicative mettant en oeuvre des connexions en couplage lâche entre divers composants logiciels.

[39] Le système d'information du ministère des finances est une architecture SOA, basée sur JMX pour la gestion de web-services associés. source *Atos Origin : http ://www.atosorigin.com.*

[40] JBoss est un serveur applicatif J2EE : http ://www.jboss.com

[41] TRELIS est developé par ADI Limited : http ://www.adi-limited.com, une filiale de THALES.

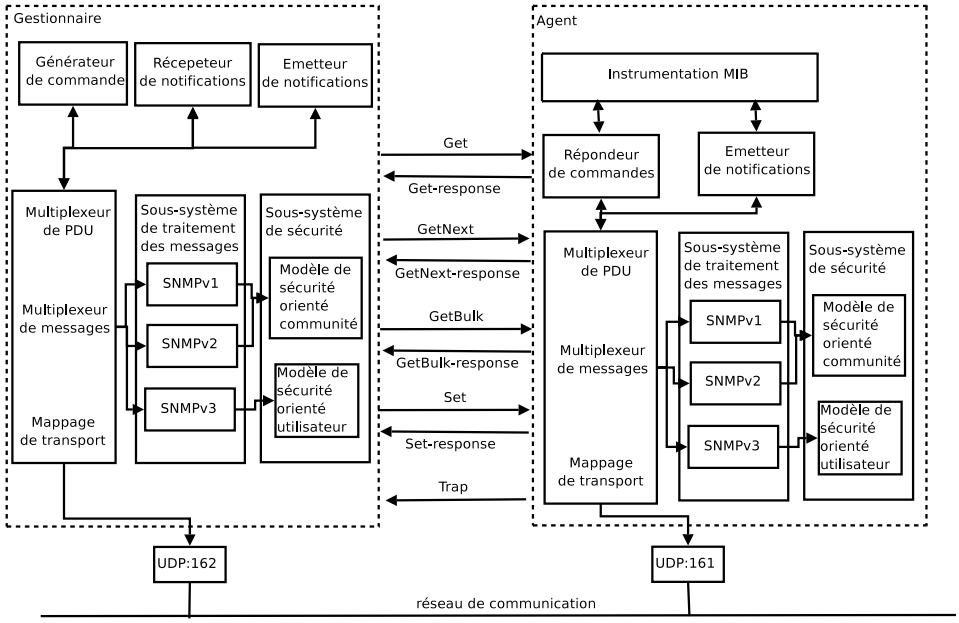


FIG. 2.13 – Architecture gestionnaire-agent sur laquelle se base le protocole SNMPv1,v2,v3.

2.4.1 La MIB

Chaque agent SNMP met à la disposition du gestionnaire des objets (appelés aussi variables) qui sont structurés dans une MIB [93] définie par la RFC 1213 [78]. Ces objets sont réunis dans des groupes (le groupe *system* décrit l'équipement, et par exemple la valeur de la variable *system.sysDescr.0* peut être la chaîne de caractères suivante : *HP ETHERNET MULTI-ENVIRONMENT,ROM R.22.01,JETDIRECT,JD95,EEPROM R.25.09,CIDATE 07/24/2003*). Chaque objet est identifié dans la MIB SNMP par un OID (Object IDentifier) qui définit sa position au sein de la structure arborescente de la MIB. En suivant l'arbre depuis la racine jusqu'à une feuille, on forme l'OID. Cet OID est constitué d'une suite d'étiquettes numériques séparées par des points. Par exemple, l'OID *1.3.6.1.2.1.1.1.0* correspond à la variable *systemDescr* du groupe *system* de la MIB. La valeur 0 à la fin de l'OID correspond au fait que la variable recherchée est un objet simple (*system.systemDescr.0*). En revanche si la variable est une table on trouvera OID.<valeur index 1>....<valeur index n> (par exemple *interfaces.ifTable.ifEntry.1* jusqu'à la valeur *interfaces.ifTable.ifEntry.n*).

2.4.2 Fonctionnement du protocole

Le protocole SNMP définit 5 types de messages (Protocol Data Unit) (voir la table 2.3). Les messages de type Get-Request, Get-Next-Request sont des messages utilisés pour de la collecte d'informations de supervision sur un agent SNMP. Le message de type Get-Request est émis depuis un outil de supervision vers un agent SNMP. L'agent SNMP répond par un message de type Get-Response contenant la valeur des variables demandées. Le message particulier Get-Next-Request est un message qui demande à l'agent SNMP la variable suivante dans la MIB SNMP. Par exemple, un message Get- Next-Request *1.3.6.1.2.1.1.1.0* (*system.systemDescr.0*) vers un agent SNMP sera suivi d'un message de type Get-Response contenant la valeur de l'objet d'OID *1.3.6.1.2.1.1.2.0* correspondant à l'objet system.sysObjectID. Par un échange de messages de type Get-Next-Request et Get-Response entre l'outil de supervision et un agent SNMP, il est

possible d'obtenir toute la MIB de l'agent SNMP. Un message de type GetBulk-Request permet de rechercher un ensemble de variables regroupées. Ce type de requête est une généralisation de l'opération Get-Next-Request où l'agent procède à une série interne de cette dernière. Un message de type Set-Request émis par l'outil de supervision à destination de l'agent SNMP permet la mise-à-jour d'une ou plusieurs variables. La réponse en retour est soit la liste des variables mises à jour, soit un message d'erreur rapportant une incohérence dans la mise à jour.

Un message de type Trap peut être émis depuis l'équipement actif vers la station d'administration, message non sollicité, afin de prévenir la station d'administration d'un événement. Par exemple, un trap SNMP peut être émise lors du changement d'état d'une interface réseau (passage de l'état *linkUp* à l'état *linkDown*). L'administrateur configure l'agent SNMP pour que celui-ci émette une alerte, en spécifiant le type d'alerte et l'adresse IP du gestionnaire de réseau.

Émetteur	Message	Fonctionnalité
Gestionnaire	Get-Request	obtenir une variable (v1, v2, v3)
Gestionnaire	GetNext-Request	obtenir variable suivante (v1, v2, v3)
Gestionnaire	GetBulk-Request	obtenir un ensemble de variables regroupées (v2, v3)
Gestionnaire	Set-Request	modifier la valeur d'une variable (v1, v2, v3)
Agent	Get-Response	la réponse (Get, Get-Next, Set, GetBulk) (v1, v2, v3)
Agent	Trap	message d'alerte (v1, v2, v3)

TAB. 2.3 – Opérations définies par les différentes versions (v1, v2 et v3) du protocole SNMP.

2.4.3 Comparaison entre SNMP et JMX

Dans cette section, nous présentons une comparaison qualitative entre les deux protocoles SNMP et JMX. Les informations de gestion au niveau du protocole SNMP sont stockées dans une MIB. Une unité de données de la MIB est une variable simple ou une structure tabulaire. Les modules de MIB sont structurés sous forme d'arbre. Chaque nœud de l'arbre possède un identifiant unique OID qui présente une séquence d'entiers de la racine jusqu'au nœud courant. Pour des raisons historiques l'OID de la plupart des objets de gestion commence par 1.3.6.1 (iso.org.dod.internet). Au niveau de JMX chaque nom d'attribut de gestion est une chaîne de caractères, définie par le développeur de l'application pour identifier l'attribut au niveau d'un MBean. Un MBean est identifié du côté de l'agent et du gestionnaire par un nom unique de type *ObjectName*. Ce nom est exposé par le serveur de MBeans. Un ObjectName peut aussi représenter un motif de filtrage de MBeans dans une requête portant sur plusieurs MBeans. L'ObjectName est composé de deux parties : le domaine et une liste de couples clés-valeurs. Par exemple, *com.sun.someapp :type=whatsit,name=25* est un ObjectName d'un MBean. Ainsi, si SNMP possède un modèle d'information bien standardisé avec un langage de description spécifique (SMI), JMX ne repose sur aucun modèle d'information. En effet, JMX fourni seulement un type de nomage prédéfini (ObjectName) pour identifier les MBeans et laisse le nomage des leurs attributs à la volonté des développeurs.

Le protocole SNMP repose sur un petit ensemble d'opérations : Get, Get-Next, Set et Trap. À partir de sa version 2, une nouvelle opération GetBulk a été introduite pour améliorer l'efficacité de la collecte de donnée tabulaires. Au niveau de JMX, le serveur de MBeans fournit des méthodes pour accéder ou altérer les valeurs des attributs. Ces méthodes sont les suivantes :

- getMBeanInfo : renvoie l'ensemble des informations d'un MBean comme ses attributs, ses opérations, ses classes et ses notifications.
- getAttribute : renvoie la valeur d'un seul attribut d'un MBean.
- getAttributes : renvoie les valeurs d'un ensemble d'attributs d'un seul MBean.
- setAttribute : modifie la valeur d'un attribut d'un seul MBean.
- setAttributes : modifie les valeurs d'un ensemble d'attributs d'un seul MBean.
- invoke : invoque une opération définie au niveau d'un MBean

On s'aperçoit que les opérations de JMX diffèrent de celles de SNMP d'un point de vue granularité. Les opérations JMX opèrent sur un ou plusieurs attributs d'un seul nom de MBean à la fois. En revanche les opérations SNMP peuvent manipuler différents OID de différents groupes.

Généralement, le protocole SNMP repose sur le protocole UDP pour le transport de ses messages entre le gestionnaire et l'agent. Malgré l'existence d'un RFC 3430 [88] introduisant TCP comme protocole de transport pour SNMP, son utilisation reste très limitée. En revanche, le protocole JMX repose essentiellement sur le protocole TCP. La table 2.4 résume les différences majeures entre le protocole SNMP et JMX.

	SNMP	JMX
Structure des objets de gestion	arbres de MIB	MBeans
Langage de description des objets	SMI	Java
Identification des objets	OID	ObjectName, nom de l'attribut
Couche de présentation	ASN.1	sérialisation Java
Protocole de transport	UDP	RMI/TCP, HTTP, SOAP, etc

TAB. 2.4 – Comparaison qualitative entre SNMP et JMX.

2.4.4 Lacunes de performances du protocole SNMP

Le protocole SNMP est relativement simple dans son fonctionnement (5 types d'opération) et ses données de gestion (scalaires et tableaux de type ligne-colonne). Toutefois, cette simplicité est aussi bien sa puissance que sa faiblesse. En effet, plusieurs études ont montré les lacunes dont souffre le protocole SNMP lorsqu'il est confronté à des données de taille importante ou un nombre considérable d'objets gérés. Il s'avère d'après ces études que le protocole SNMP possède une performance acceptable dans certains cas d'utilisation précis où le nombre d'objets gérés est faible de l'ordre de dizaine. Les travaux [147, 83] ont montré qu'une supervision reposant sur le protocole SNMP avec des intervalles de l'ordre de 5 minutes est inefficace pour reporter certains compteurs de trafic réseaux essentiellement les micros congestions[42] où l'utilisation des liens dépasse les seuils tolérés par les opérateurs. Le choix de l'intervalle de 5 minutes est dû au fait que les administrateurs choisissent un intervalle de supervision important pour que le gestionnaire puisse scruter l'ensemble des variables depuis les routeurs durant cette intervalle. Généralement, les opérateurs exigent que le niveau d'utilisation d'un réseau grande échelle ne dépasse pas les 50%[43]. Ces compteurs portent sur l'utilisation de liens de réseau sont de grande importance pour le dimensionnement du réseau et la garantie de la qualité de délais. En effet, l'utilisation de SNMP avec de faibles intervalles de supervision de l'ordre de la milliseconde

[42] Une micro congestion est une épisode de congestion qui se manifeste sous un laps de temps de l'ordre de quelques millisecondes.

[43] Au delà de cette valeur les délais d'attente au niveau des routeurs deviennent importants.

introduit des problèmes de performances au niveau du gestionnaire et de l'agent qui deviennent surchargés ainsi qu'un coût important au niveau de la bande passante utilisée pour la gestion. Plusieurs études ont aussi montré l'inefficacité de SNMP dans le cas de la collecte de larges données de gestion, comme de tables avec un nombre considérable de lignes. Dans [130], les auteurs ont présenté une comparaison entre SNMP et une application CORBA pour la collecte des données d'une table BGP[44] avec une taille variant entre 1 et 2000 lignes. Ils ont montré que la bande passante consommée et les délais d'une gestion basée sur CORBA sont moins importante que ceux d'une gestion à base de SNMP avec les opérations *Get-Next* et *Get-Bulk*, lorsque la taille de données à collecter est supérieure à 200 lignes.

Dans des environnements avec une forte perte commes les réseaux sans fil ou sous une mauvaise qualité des liens, les auteurs de l'article [131] ont montré que le temps de réponse de l'opération Get-Next du protocole SNMP devient important variant entre plusieurs secondes et quelques minutes. Ceci entraîne une perte de la vue réelle du système géré et la fonction de supervision est inutilisable. Ainsi, cette étude a mis l'accent sur la performance de SNMP et son inefficacité à superviser l'état de systèmes dans ces environnements (réseaux sans fil, réseaux satellite, etc). Dans des environnements avec un niveau de congestion important, le protocole SNMP a aussi montré son inefficacité. Les auteurs de [121], ont montré que la perte de paquets SNMP est de l'ordre de 8% sous une congestion moyenne et atteint une valeur de 16% sous une forte congestion du réseau où le trafic utilisateur est important.

2.5 Synthèse

La première étape vers l'évaluation de performances des systèmes de supervision est de détailler leurs caractéristiques intrinsèques qui influencent ses performances. Durant cette étape, nous avons recensé les différentes caractéristiques communes aux approches de gestion traditionnelles reposant sur le modèle gestionnaire-agent. Les dimensions de la gestion, la distribution de ses fonctions, son intégration et son hétérogénéité au niveau de ses environnements, les modes d'interaction entre ses différentes entités, la concurrence et la granularité de ses opérations sont des caractéristiques qui affectent considérablement sa performance et son efficacité. Comme illustration, nous avons choisi l'approche JMX dédiée à la gestion des applications Java et le protocole SNMP pour la gestion de réseaux. Nous avons présenté les caractéristiques majeures de ces deux protocoles afin d'une part de familiariser le lecteur avec ces approches et d'autre part de motiver la recherche présentée dans la seconde partie de ce document. En effet, la flexibilité de l'approche JMX vue son absence de modèle d'information standard et sa nature intrusive à cause de son intégration facile dans le plan fonctionnel des systèmes gérés où il peut même être le noyau d'un système, le place comme meilleur candidat à représenter les problèmes de performances auxquels est actuellement confrontée la gestion de réseaux et de services. Si l'approche JMX n'a pas encore eu d'attention pour mesurer sa performance, plusieurs études se sont focalisées sur d'autres approches de gestion.

[44] BGP : Border Gateway Protocol.

Chapitre 3

Évaluation par mesure de la performance de la gestion

Sommaire

3.1 Introduction

La supervision des réseaux et de ses équipements repose sur le protocole SNMP à cause de la disponibilité de celui-ci dans la majorité des équipements réseaux et leurs applications. Cependant, dans le monde Java et ses applications, JMX est le standard pour leurs gestion. Dans le chapitre 2 , nous avons présenté quelques caractéristiques de ces cadres de supervision et identifié les problèmes de performances auxquels ils sont soumis. Plusieurs études relatives à l'évaluation de performances de systèmes de supervision ont été conduites. En revanche, très rares ou même inexistantes sont les études de performances portant sur l'approche JMX. Dans ce chapitre, nous ferons l'inventaire des études portant sur l'évaluation de performances du protocole SNMP ainsi que d'autres (gestion basée sur les services web, XML, CORBA, etc). Nous avons constaté que la majorité de ces différents travaux, fondés sur des modèles analytiques ou des mesures, ne fournissent qu'une évaluation figée et restreinte de cas précis des approches de gestion. Dans un objectif d'élaboration d'un ensemble de métriques bien définies pour l'évaluation de performances de la gestion, nous passerons en revue, dans la seconde partie de chapitre, les standards de mesure de performances de réseaux IP. Notamment, ceux du groupe de travail IPPM de l'IETF.

3.2 Approches existantes pour l'évaluation de performances de la gestion

On compte actuellement plusieurs travaux qui portent sur l'évaluation de performances des approches de gestion et des technologies sur lesquelles elles reposent. Ceux-ci sont souvent motivés par la comparaison de performances de deux approches différentes pour convaincre que l'une est meilleure que l'autre. On distingue deux grandes classes d'approches d'évaluation de performances de la gestion. La première classe d'études est rétroactive. Elles reposent sur un objectif comparatif, consacrée à l'évaluation de performances des approches existantes de gestion. Ces approches ne sont pas l'œuvre des auteurs de ses études. En effet, ces études portent généralement sur la comparaison d'une nouvelle technologie ou approche de gestion à une approche SNMP. Récemment, plusieurs études [142, 138, 123, 148, 137] ont évalué d'une façon plus pertinente des approches de gestion basées sur les service web en les comparant avec l'approche SNMP.

La seconde classe comporte des études où les auteurs proposent plutôt leurs propres nouvelles approches ou technologies de gestion. Ils comparent d'une façon plus restreinte la performance de ces nouvelles approches avec SNMP ou autre cadre. Ces études sont plus anciennes[45] que celles de la première classe. Cela peut s'expliquer par le besoin de la communauté de gestion de réseaux et de services de développer de nouvelles approches et adapter de nouvelles technologies pour répondre au besoin de la gestion. Néanmoins, nous avons repéré certaines études complètement consacrées à l'analyse de performances de certains aspects de SNMP. Pattinson, dans son étude [139] analyse par simulation le comportement d'un agent SNMP soumis séparément à une activité de collecte périodique de donnée de gestion et une activité intensive de détection d'une faute au niveau des tables de routage. Plusieurs études [125, 122, 136] ont analysé pertinemment le coût introduit par des mécanismes de sécurité (SSH[46], modules de sécurité SNMPv3, TLS[47]) au niveau du protocole SNMP.

3.2.1 Méthodologies pour la mesure de performances de la gestion

Nous allons analyser un ensemble d'études portant sur l'évaluation de performances des approches de gestion pour dégager les éléments de bases des méthodologies d'évaluation sur lesquels elles reposent. Ces éléments de base incluent, les techniques d'évaluation utilisées, l'architecture de gestion sous test, le facteur d'hétérogénéité des agents de gestion (nous avons examiné, essentiellement, l'hétérogénéité des implantations des agents sous test) et les métriques de performance choisies. Nous avons aussi examiné l'ensemble des facteurs et des paramètres de performances utilisés par ces études. Cet ensemble inclut les caractéristiques des sites de gestion (nombre d'agents et de managers), les paramètres de la charge (nombre d'objets de gestion, nombre d'objets accédés par requête, fréquence de scrutation), le modèle de communication (caractéristiques du réseau) et la fonction de gestion sous test (surveillance ou résolution de problèmes) et son mode d'interaction : scrutation ou notification.

La majorité des études que nous avons examinées reposent sur la technique de mesure plus que les deux autres techniques (simulation et analytique). Ceci est dû au fait que des architectures de gestion comme SNMP, agents mobiles ont été déjà implantés ou prototypés, ce qui favorise la mesure de leurs performance sur des plates-formes ou des prototypes réels. En revanche, certaines études ont analysé d'une manière empirique certaines métriques de performance, essentiellement le traffic introduit par les activité de gestion pour proposer des formulations quantitatives de ces

[45] Nous situons ces études entre les années 96 au 2001.
[46] SSH : Secure SHell est à la fois un programme et un protocole de communication sécurisé.
[47] TLS : Transport Layer Security est un protocole de sécurisation des échanges sur Internet.

métriques. Ces études ont essentiellement porté sur la gestion à base d'agents mobiles [145, 120, 127, 134]. Parmi ces études analytiques, on trouve le travail de Chen et al [108], où les auteurs ont proposé une formulation analytique de l'utilisation CPU, le délais de détection de changement d'une variable de gestion et le trafic généré par une entité de gestion dans différentes approches. Ils ont analysé les approches suivantes : client-serveur, hiérarchique, faible mobilité et forte mobilité. Rares sont les études [139, 96, 131] qui reposent sur la technique de simulation et la modélisation analytique[48]. L'une de ces rares étude est la thèse de Rajeh Subramanyan [96] qui s'est penché sur le développement d'une plate-forme hiérarchique basée sur SNMP pour la supervision d'un système hétérogène de calcul à large échelle. Une partie de sa thèse a été consacrée à l'évaluation par des modèles à base de files d'attentes et la simulation de son approche.

Les métriques de performances les plus utilisée sont les délais que subissent les opérations de gestion et le trafic généré par ces opérations. Comme indiqué dans [130], les délais des opérations de gestion présentent deux composantes : le délai introduit par la couche de communication et le délai de traitement. Cela permet d'isoler et d'identifier les problèmes de performances de la gestion. En revanche, nous avons constaté dans la majorité de ces études un problème de terminologie, de sémantique et d'unités de mesure des métriques de délais analysées. Certaines études utilisent le terme temps de réponse, d'autre complexité de temps ou encore temps d'exécution, etc. D'autre part, la majorité des études ne spécifient pas le point de mesure de ces délais (agent, gestionnaire) ainsi que le niveau (applicatif, noyau ou réseau). L'instruction la plus utilisée pour mesurer ces délais au niveau applicatif est *gettimeofday*[49]. Toutefois, dans [122], les auteurs ont utilisé l'instruction RDTSC[50] pour récupérer le temps au niveau du processeur.

Si nous considérons les architectures fondamentales de gestion, nous pouvons identifier quatre scénarios possibles à évaluer. Comme l'indique la figure 3.1, ces quatre possibilités sont : un gestionnaire vers un agent, un gestionnaire vers plusieurs agents, plusieurs gestionnaires vers un agent et un gestionnaire vers un intermédiaire (un sous-gestionnaires ou une passerelle de gestion) vers un agent. Les études existantes sont focalisées sur une topologie centralisée avec un gestionnaire et un agent reposant sur des technologies de gestion sous-jacente différentes (CORBA, services web ou SNMP). D'autres études ont considéré des topologies composites : gestionnaire-intermédiaire-agent. L'entité intermédiaire se présente sous la forme d'un sous-gestionnaire dans le cas d'une approche distribuée ou un relai (une passerelle de gestion) qui assure la communication entre deux technologies de gestion différentes (service web et SNMP, par exemple). Dans cette classe de topologies, on trouve les travaux de [138, 123]. Leur topologie est composée d'un gestionnaire employant le protocole SOAP pour accéder aux données de gestion d'un agent implantant SNMP. Entre le gestionnaire et l'agent une passerelle fonctionne comme relai pour assurer la conversion des requêtes du gestionnaire en requêtes SNMP vers l'agent.

Les facteurs d'évaluation employés par ces études sont généralement restreints. Cette restriction est due à la difficulté de couvrir tous les facteurs possibles ainsi que leurs plages de valeurs. En effet, comme indiqué dans le chapitre 2 la performance d'une approche de gestion dépend de plusieurs facteurs qui sont les fréquences de supervision, la taille de données, les types d'opérations, le nombre d'agents, les conditions réseaux, etc. Dans les études portant sur SNMP, une majorité des études [142, 143, 128] utilisent la table des interfaces de la MIB-II [78] comme

[48] Par modélisation analytique, on désigne l'utilisation des outils ou des méthodes d'analyse comme indiqué dans la section 1.3.3.

[49] La fonction *gettimeofday* fournit le nombre de secondes et microsecondes écoulées depuis l'Epoch (le 1 er janvier 1970 à 0h).

[50] L'instruction RDTSC (Real Time Stamp Counter) une instruction pour les processeurs Intel x86. L'instruction retourne dans le couple de registre EDX :EAX le nombre de ticks (unité de mesure du temps dans le noyau Unix) écoulés depuis la dernière remise à zéro du processeur (Reset).

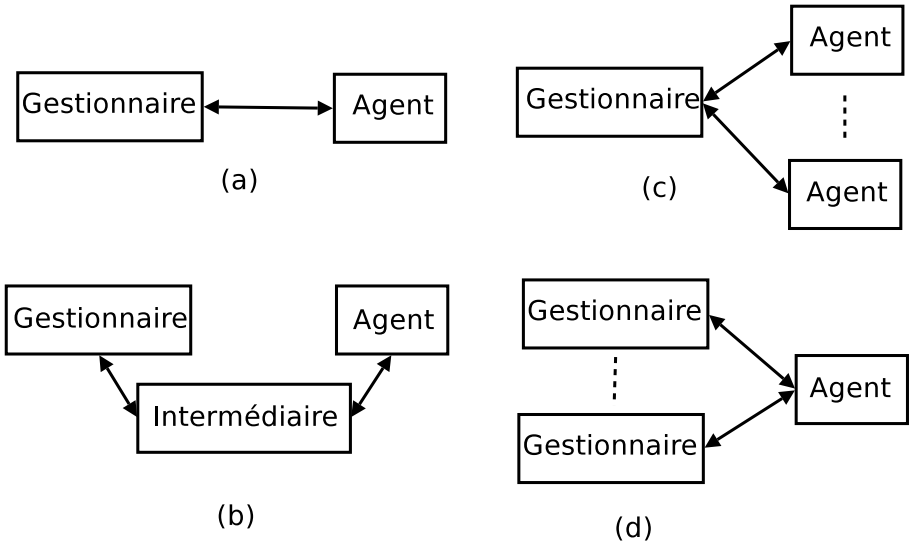


FIG. 3.1 – Scénarii fondamentaux des architectures de gestion. (a) un gestionnaire vers un agent. (b) un gestionnaire vers un intermédiaire vers un agent. (c) un gestionnaire vers plusieurs agents. (d) plusieurs gestionnaires vers un agent.

instance de données de gestion au niveau de l'agent sous test. Néanmoins, le choix de la nature de donnée de gestion (table ou une simple variable) est souvent lié au type de l'opération utilisée au cours de l'évaluation. En effet, des tables sont déployées au niveau de l'agent en cas d'évaluation des opérations multiples (GetBulk ou Get-next). Des variables simples sont collectées depuis une MIB spécifique en cas d'utilisation des opérations unitaires (get, par exemple) [150, 126]. Dans ce contexte, une étude intéressante est celle de Corrente et al [122], où les auteurs ont utilisé une MIB synthétique qui comporte des variables simples et tabulaires. Cela leur permet d'utiliser les différents types d'opérations unitaires ou multiples. Il faut noter que ce choix d'une MIB synthétique est judicieux puisqu'il permet à d'autres études de reproduire les expérimentations avec une instance de donnée de gestion commune. Les valeurs, ainsi que les types de données de cette instance de donnée peuvent être dérivés de l'étude de Schönwälder [115].

Les environnements réseaux sur lesquels se reposent les études que nous avons examinées, sont de réseaux IPv4, Ethernet de 10 à 100 Mbits de capacité. La plupart supposent que les réseaux sont uniformes, sans trafic fonctionnel et libre d'erreurs. Ces hypothèses sont trop fortes, notamment dans un contexte d'évaluation de performances représentative d'un cas d'utilisation réel d'une approche de gestion. Dans des réseaux à bande passante limitée, les temps de réponses sont importants (quelques minutes) [131], puisque le réseau devient un goulot d'étranglement, notamment dans des cas de collecte massive de données de gestion où plusieurs agents sont impliqués dans l'activité de gestion. Certaines études ont analysé les performances de SNMP dans ce type de réseaux. Dans [136], les auteurs ont analysé partiellement l'effet de perte de paquets sur les latences que subissent les requêtes SNMP. Dans leur cas d'étude, les auteurs ont montré que ces latences sont supérieurs à 10 secondes pour un niveau de perte de 0.5% de paquets. Dans un cas de 0% de perte, ces latences sont inférieurs à 4 secondes. Ces résultats sont valables pour l'utilisation des protocoles sous jacents TCP ou UDP au dessous du protocole SNMP. Dans des environnements sans fil, la seule étude que nous avons recensée portant sur la performance de SNMP dans de tels environnements est celle de [131]. Les auteurs affirment que

les temps de réponse des requêtes SNMP[51] deviennent importants avec une qualité médiocre de lien sans fil de l'ordre de 5 à 10 décibel[52] ou dans la condition d'un réseau chargé.

3.3 Inadaptation des méthodologies existantes

Dans la section précédente, nous avons analysé les alternatives d'évaluation de performances des approches de gestion sous plusieurs angles : (1) l'objectif de l'étude, (2) la technique d'analyse de performances considérée, (3) les paramètres de gestion sous test (modèle de distribution, donnée de gestion, le mode d'interaction et le modèle des opérations). Cette analyse a montré certaines lacunes dans ces études.

3.3.1 Observation I : Études non comparables, non reproductibles et non représentatives

La majorité des études comparent la performance de SNMP avec celle d'un autre protocole. En revanche, nous avons constaté que ces études sont incomparables entre elles. Par exemple, les deux études suivantes [140, 130] ont comparé la performance de SNMP et de CORBA avec un scénario 1 gestionnaire, 1 agent. En revanche, ces deux études, malgré leur objectif commun, se sont reposées sur des types des données de gestion différentes, table BGP et table TCP ainsi que deux opérations de nature différentes, *Get* pour la première et *GetBulk* pour la deuxième. Les études que nous avons recensées ne sont pas reproductibles. Cela veut dire que si l'on essaie de reproduire les mêmes expérimentations décrites dans ces études, la tâche sera très difficile car l'espace des options à prendre en compte est trop important. Plusieurs détails ont généralement été omis dans ces études. En effet, les fréquences de scrutation, la concurrence des opérations, l'état du réseau, le type des donnée de gestion et la méthodologie de mesure et d'analyse des métriques ne sont pas détaillées.

Une autre observation aussi importante que les deux précédentes est la nature non représentative de ces études. L'ensemble de ces études repose sur une charge synthétique de gestion, c'est à dire, une charge qui n'est pas réelle ou loin de représenter des cas réels d'utilisation d'une approche de gestion. La charge fonctionnelle est aussi absente dans plusieurs de ces études, bien que cette charge soit un facteur déterminant de la performance de la gestion. En effet, il existe un compromis entre les performances des deux plans à cause du facteur intégration (voir section 2.2.3) de la gestion dans le plan fonctionnel, c'est à dire la fusion de deux plans en un seul.

3.3.2 Observation II : Manque de métriques standards pour la mesure de la performance de la gestion

Malgré l'existence des spécifications et des implantations des différents protocoles de gestion (SNMP, JMX, Netconf, etc), leurs critères de mesure de performance sont différents au niveau de la terminologie et de la sémantique. Par exemple, pour identifier le coût réseau des activités de gestion, plusieurs études utilisent des termes différents : *bande passante utilisée, volume de trafic, trafic de gestion, utilisation réseau.* Cette diversité engendre une confusion et une mauvaise interprétation des résultats présentés dans les études portant sur la performance d'une approche de gestion. Deux études [142, 140] ont présenté une méthodologie commune pour la mesure de

[51] Il faut noter que dans cette analyse des études existantes de la performance de la gestion, nous avons gardé la terminologie employée par l'étude. Par exemple nous utilisons temps de réponse ou latence selon le terme utilisé par l'étude.

[52] Il s'agit de la valeur en décibel du rapport de bruit au signal (Signal to Noise Ratio) de lien.

la performance de SNMP comparée à celle d'une approche service web reposant sur le protocole SOAP. Comme l'indique le tableau 3.1, ces deux études reposent sur un scénario commun de type 1 gestionnaire, 1 agent. Malgré ce scénario commun on constate une différence au niveau des métriques employées et le choix de paramètres d'évaluation. Les deux études se sont basées sur deux types de donnée différents. La première utilise la table *iftable* de la MIB-II et la deuxième la MIB-TCP. Cette différence rend leurs résultats très difficiles à comparer.

Étude	Technique	Approche de gestion	Mode d'interaction	Hétérogénéité	Métriques
Pras et al [142]	mesure	SNMPv2c/v3 comparé au service web (SOAP), 1 gestionnaire - 1 agent	scrutation	agents hétérogènes	bande passante utilisée (octets), temps d'occupation CPU (ms), utilisation mémoire (Kilo octets) et délais d'aller-retour au niveau paquets (ms)
Pavlou et al [140]	mesure	SNMPv1/v2c comparé au service web (SOAP), 1 gestionnaire - 1 agent	scrutation	homogènes	trafic généré (octets), temps de réponse au niveau applicatif (ms)

TAB. 3.1 – Méthodologies de mesure de performance de SNMP comparée à celle d'une approche de gestion basée sur le service web dans un scénario 1 gestionnaire - 1 agent.

3.4 Autres sources d'inspiration pour la définition de métriques de performances pour la gestion

Comme indiqué précédemment, nous n'avons pas connaissance d'autres projets de standardisation cherchant à identifier des métriques de performances pour des systèmes de gestion, mais nous avons examiné d'autres travaux, notamment le travail en cours au sein du groupe IPPM[53] [160] de l'IETF[54]. En revanche récement (en 2007), l'IETF a lancé un nouveau groupe de travail nommé APM (Application Performance Metrics) dédié à l'élaboration des métriques de performances pour les couches protocolaires applicatives comme par exemple le protocole SIP (Session Initiation Protocol). Nous n'avons pas pu profiter des travaux de ce nouveau groupe puisqu'il est encore en phase intiale.

3.4.1 IPPM pour la mesure de réseaux IP

Le groupe de travail IPPM développe un ensemble de métriques standards qui fournissent des mesures quantitatives de la qualité, performance et fiabilité des services de livraisons de données de l'Internet. Ils vont fournir aux utilisateurs finaux, aux opérateurs réseau et aux fournisseurs des services de l'Internet une compréhension commune et une mesure précise de la performance et de la fiabilité des routes unidirectionnelles de bout en bout et des sous-réseaux IP. Ces indicateurs essayent d'apporter des mesures quantitatives, non biaisées des performances. On y trouve :

– la connectivité [157]

[53] IPPM : Internet Protocol Performance Metrics
[54] IETF : Internet Engineering Task Force

- le délai et le taux de perte dans un sens [152, 151]
- le délai et le taux de perte aller-retour [153]
- la variation de délai (gigue) [154]
- la distribution des pertes [156]
- le réordonnancement des paquets [159]
- la capacité de transport brute [158]
- métriques adaptées à la diffusion *multicast* [155]

Ce groupe de travail produit des documents permettant de mesurer séparément ces métriques et d'expliquer l'impact de ces métriques sur les différentes classes de services. Nous avons trouvé une vraie ressemblance entre les métriques que nous cherchons à définir pour la supervision et certaines définitions et méthodologies produites par le groupe IPPM. À titre d'exemple, dans [155] la terminologie des métriques multi-parties est valable pour mesurer des indicateurs de performance d'une plate-forme de supervision opérant en scrutation et impliquant un gestionnaire et plusieurs agents. Nous reviendrons plus en détails, dans le chapitre 4, sur les adaptations de métriques définies dans les travaux de l'IPPM pour mesurer la performance d'un système de gestion.

3.4.2 Caractéristiques des métriques IPPM

La métrique est une quantité spécifiée, liée à la performance et la fiabilité de l'Internet et précisée dans des unités de mesure standard (e.g. le bit pour l'information). S'inspirant du travail proposé dans le RFC de référence d'IPPM [160], nous définissons un ensemble des critères que des métriques d'évaluation de performances des systèmes de supervision doivent respecter pour qu'elles soient conformes à la vision d'IPPM :

- Les métriques doivent être concrètes et bien définies : il faut essentiellement éviter les métriques probabilistes et se focaliser sur des métriques déterministes. Par exemple, il est recommandé d'éviter des métriques du genre « *la probabilité de réponse d'un agent de supervision à des requête envoyées par un superviseur* ». Mais définir une métrique de la forme taux de réponse de l'agent en terme de nombre de requêtes correctes servies par seconde. En effet, une mesure selon la définition stochastique de cette métrique nous donne une valeur de la forme $0,80$ et la deuxième définition nous donne une valeur de la forme 80 sur 100 requêtes correctes servies par seconde. Les définitions dans des termes probabilistes peuvent cacher des hypothèses relatives au modèle stochastique du comportement mesuré (e.g. perte corrélée de paquets).
- La méthodologie de mesure d'une métrique doit avoir un caractère répétitif : si une méthodologie de mesure d'une métrique est utilisée plusieurs fois sous des conditions identiques, les résultats obtenus doivent être quasi identiques.
- Les métriques doivent être capables de refléter les performances de différentes opérations de supervision sous test. Par exemple, une comparaison de performance entre les opérations *GetAttribute* et *GetAttributes* fournies par l'approche JMX, nécessite l'utilisation d'une métrique de capacité mesurant le nombre d'attributs scrutés par seconde par le superviseur et non pas le nombre de requêtes par seconde. Le choix de cette métrique est dû au fait que ces deux opérations n'ont pas la même multiplicité en terme de nombre d'attributs accédés par une requête.
- Les métriques ne doivent pas être aberrantes pour des systèmes de gestion implantés par des approches identiques. Elles doivent être valables pour ces différents systèmes.
- Les métriques doivent fournir une compréhension claire des systèmes de gestion implantés par des approches différentes. Une comparaison de performance, par exemple, entre une

architecture basée sur SNMP et une autre basée sur JMX, nécessite l'utilisation d'une métrique de réponse mesurant le temps de réponse de la scrutation d'un même objet de gestion avec la même granularité de l'opération implanté avec ces deux technologies.
- Les métriques doivent être utiles aux opérateurs et aux équipementiers pour maîtriser les performances des architectures de gestion qu'ils utilisent.

Cet ensemble de critères est à appliquer sur l'ensemble des métriques que nous allons définir dans le chapitre 4. Ceci nous permettra de choisir parmi elles, les plus adéquates pour évaluer les performances des systèmes de supervision, ou de les comparer entre eux.

Type de métriques

Pour mieux comprendre les métriques empiriques proposées par le groupe IPPM, nous présentons dans ce paragraphe leurs différences par rapport aux métriques analytiques.
- Métriques analytiques : vu le riche cadre analytique existant dans le domaine de réseau IP, il est important de générer des caractérisations du réseau qui sont en accord avec ce cadre et les configurations pratiques à la fois. Ceci permet aux études non empiriques de corréler et de comprendre le comportement des réseaux réels en sélectionnant les propriétés des composants du réseau pertinentes pour une métrique analytique donnée (e.g. taux de transmission dans un routeur modélisé avec une file d'attente).
- Métriques empiriques : quand une métrique pertinente n'entre pas dans le cadre analytique à cause du fait qu'elle ne modélise pas convenablement le système réel, une métrique empirique peut être définie à sa place, accompagnée par une méthodologie de mesure de référence (e.g. le meilleur débit réalisable au long d'une route en utilisant un protocole de type TCP conforme au RFC 2001).

Composition de métriques

Il y a deux formes de composition des métriques, ce qui permet de réaliser des extrapolations :
- Composition spatiale : une métrique peut être appliquée à une route IP complète et/ou aux routes qui la composent (e.g. le délai total comparé aux délais sur certaines connexions).
- Composition temporelle : une métrique peut être appliquée au moment t_i et/ou aux moments t_j (e.g. le débit durant cinq minutes comparé au débit pendant cinq autres minutes).

Instance des métriques

Les métriques peuvent être classifiées en fonction du nombre d'instances utilisées, c'est à dire, le nombre de fois qu'une mesure est effectuée.

Métrique singleton : Une métrique singleton est une métrique pour laquelle une seule instance de la mesure a été produite.

Métrique échantillon : Une métrique échantillon implique un nombre d'instances distinctes de métriques singleton. Les échantillons donnent une indication sur les variations et la cohérence d'une métrique. Les échantillons peuvent être collectés au moins de trois façons : échantillonage périodique, aléatoire additif et poissonnien.

Métrique statistique : Une métrique statistique est dérivée d'une métrique d'échantillonnage par le calcul de statistiques sur les valeurs définies par la métrique singleton échantillonnée (e.g. la moyenne).

3.4.3 Méthodologie de mesure

Selon la terminologie IPPM, pour un ensemble de métriques bien définies, un ensemble de méthodologies de mesure est identifié. Nous citons à titre indicatif les méthodologies de mesure suivantes :

- La mesure directe d'une métrique en utilisant du trafic de test injecté dans le réseau ;
- La projection d'une métrique à base de mesures de niveaux plus bas ;
- L'estimation d'une métrique constituée par plusieurs mesures agrégées ;
- L'estimation d'une métrique au temps t' à base de métriques observées au temps t.

Les propriétés de toute méthodologie sont : (i) la répétabilité ; (ii) la continuité ; (iii) la cohérence avec elle-même ; (iv) la qualité de l'approximation (« goodness of fit » en anglais). En effet, une méthodologie doit présenter la propriété de répétition. Si la méthodologie est utilisée plusieurs fois dans des conditions identiques, elle doit donner des résultats cohérents. Une méthodologie pour une métrique donnée possède la propriété de continuité si pour une petite variation des conditions dans lesquelles la métrique est évaluée, l'évaluation finale de cette dernière subit aussi (et seulement) de petites variations. La qualité de l'approximation de données de mesure par rapport à une distribution statistique connue est définie par l'utilisation d'un estimateur comme celui de λ^2 que nous avons présenté dans la section 1.5.5.

3.4.4 Examen des incertitudes et des erreurs de mesure

La mesure d'une métrique selon une méthodologie, quelque soit sa précision, introduit des erreurs. Au cours de l'élaboration d'une campagne de mesure d'une métrique de performance, il a été suggéré de prendre en compte les trois recommandations suivantes :

- réduire les incertitude et erreurs.
- maîtriser et documenter les sources des incertitudes et erreurs.
- quantifier les incertitudes et erreurs.

Par exemple lors de la mise en œuvre d'une méthodologie de la mesure de délais entre une requête et sa réponse au niveau applicatif, nous devons tenir compte au mieux des erreurs introduites par les horloges (système et logiciel) et leurs méthodes de mesure.

3.5 Synthèse

La plupart des travaux de performances de la gestion se sont concentrés sur la comparaison de nouveaux protocoles au SNMP dans des cas précis et des environnements restreints. Comme nous l'avons indiqué dans ce chapitre, les métriques et les méthodologies de mesure de la performance de systèmes de gestion existantes souffrent du manque de définitions claires et de standardisation. Avant de définir des métriques de performance pour ces systèmes de gestion et étant donné que la plupart de ces systèmes opèrent au niveau IP, nous avons examiné dans ce chapitre en partie les travaux portant sur la mesure de performance de réseaux IP et la qualité de leurs services en suivant l'organisme qui régularise ce domaine qui est l'IETF. L'IETF et son groupe de travail IPPM ont une philosophie déterministe des métriques de performances d'un réseau IP, avec l'argument que les définitions en termes de probabilité peuvent cacher des suppositions seules relatives aux modèles stochastiques des comportements mesurés. Dans notre cas, nous allons nous inspirer de travaux de l'IPPM qui sont très complets et concrêts et pour lesquels les résultats sont déjà appliqués dans une série de projets et qui nous aideront pour identifier les méthodologies de mesure et les métriques pour la gestion.

Deuxième partie

Contributions

Chapitre 4

Maîtrise du coût et de la performance de la gestion

Sommaire

4.1 Introduction

L'objectif principal de ce chapitre est l'élaboration d'un ensemble de métriques pour mesurer la performance d'un système de gestion. Cette tâche semble difficile à cause du nombre et de la diversité des normes s'appliquant au domaine que ce soit au niveau organisationnel, au niveau de la technologie d'implantation, au niveau des modèles de l'information, au niveau de la couche de communication ou au niveau fonctionnel. Afin de mener à bien cette tâche, nous abordons le problème d'une manière pragmatique.

Nous présentons dans ce chapitre un ensemble de métriques qui permet de mesurer ce qui peut être produit et consommé par une approche de gestion. Ce chapitre vise un double objectif. D'une part, nous souhaitons identifier clairement la performance d'une approche de gestion. D'autre part, nous souhaitons quantifier cette performance dans un cadre générique en respectant le cadre standard fourni par IPPM. Nous avons étendu les métriques primaires IPPM par nos propres métriques portant sur le coût et la qualité d'une approche de gestion. Le chapitre sera organisé en conséquence de la manière suivante : nous présenterons tout d'abord la définition générique de la performance d'une approche de gestion. Nous décrivons ensuite les métriques primaires de performance que nous proposons. La troisième partie est consacrée à la définition d'une métrique dérivée de l'ensemble de métriques primaires proposées : « l'efficacité d'une approche de gestion ».

4.2 Définition de la performance de la gestion

Avant d'élaborer des métriques de performance de la gestion, une première interrogation s'impose : ***Qu'est ce que la performance d'une approche de gestion ?***.

Bien que les travaux de standardisation de la gestion datent des années 80, aucune définition de la performance de la gestion n'existe dans les documents produits par les organismes de standardisation (IETF, OSI, TMN, DMTF). Les quelques définitions existantes, bien qu'elles ne soient pas explicites, sont plutôt qualitatives [141, 103]. En outre, même si de nombreuses études quantitatives existent [94, 108, 142, 140, 139, 123, 96] (voir chapitre 3), leurs auteurs n'ont pas fourni une définition claire de la performance de la gestion selon leurs point de vue. Afin de combler cette lacune, nous proposons une définition de la performance quantitative de la gestion :

Définition 2 : *La performance de la gestion est sa* **capacité** *à* **maintenir** *et* **mettre à jour** *un système géré dans un* **état** *désiré avec le* **moindre coût**.

Cette définition met en évidence quatre termes clés : **capacité**, **maintenir**, **état** et **coût**. Nous allons nous appuyer sur ces différents termes pour définir des métriques capables de capturer la performance de la gestion.

4.2.1 État d'un système géré : données et informations de la gestion

L'état du système géré est exprimé par un système de gestion par le biais d'un ensemble de variables qui décrivent son état [62]. En effet, comme nous l'avons déjà indiqué dans la section 2.2.3, l'élément commun entre les différentes approches de gestion est cette notion de variable de gestion. Mais la question qui s'impose maintenant est : ***Qu'est ce qu'une variable de gestion ?*** Nous proposons la définition suivante pour décrire une variable de gestion d'une manière abstraite.

Définition 3 : *Une variable de gestion est une unité de données de gestion. Elle peut être manipulée par la biais des opérations de lecture (get), d'écriture (set) ou de notification (notify) où l'opération de lecture retourne sa valeur, l'opération d'écriture lui attribue une nouvelle valeur et l'opération de notification retourne un évènement portant sur sa valeur.*

Une variable occupe un espace de stockage au niveau des entités de gestion et ses valeurs doivent être transférés, en subissant des délais, entre plusieurs entités de gestion (par exemple, entre un gestionnaire et un agent). Le transfert de ces variables introduit un coût de communication entre ces entités. En revanche, les informations réellement utilisées par la gestion forment un sous ensemble des variables disponibles. Elles sont utiles pour déclencher des activités de gestion (mise à jour du système, détection d'un problème, envoi d'une *alarm*, etc). Nous désignons par données de la gestion ce sous ensemble. L'extraction de ces données de gestion introduit un coût de traitement au niveau des entités de gestion. Ainsi, nous proposons de nous appuyer sur cette notion de variable pour quantifier la performance d'une approche de gestion. Pour mesurer la quantité de données au niveau d'un système de gestion nous proposons l'unité : *variable* avec les préfixes usuels en base décimale de 1000 (par exemple 1kv = 1000 variables). La mesure des *variables* est effectuée sur une entité de gestion entre deux instants t_0 et t'. Au niveau de la couche du protocole de gestion entre deux entités de gestion, la quantité de données de gestion se mesure en *Bits par Variable*.

Généralement, une variable de gestion possède des propriétés temporelles relatives à l'entité de gestion qui l'observe. La propriété temporelle représente notamment sa périodicité. Une variable v_i est périodique s'il existe une opération O et un nombre $\Delta > 0$ appelés par une fonction de gestion tel que $O(v_i, t) = O(v_i, t + \Delta)$. Par exemple, une fonction de surveillance à base de scrutation fait appel à des variables périodiques avec des intervalles de surveillance statiques ou adaptatifs. Autrement la variable est apériodique. Par exemple, une variable appelée par une

opération de notification au niveau de gestionnaire n'est pas périodique puisque les intervalles de transfert de l'évènement du dépassement du son seuil qui lui est associé sont aléatoires et différentes. En revanche, la surveillance de cette variable sur l'agent pour détecter ce dépassement du seuil est périodique. Pour chaque variable de gestion de l'état d'un système géré, nous définissons les termes suivants :

- période d'entrée : il s'agit d'une consigne sous forme de l'intervalle de temps entre deux instances d'une opération portant sur une variable.
- résolution de sortie : l'intervalle de temps entre deux réponses des instances d'une opération portant sur une variable.

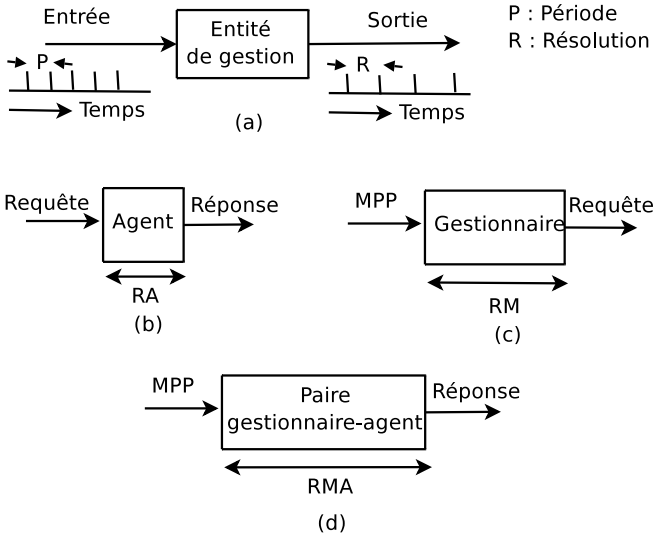


FIG. 4.1 – (a) Illustration des propriétés temporelles d'une variable de gestion. (b) période et résolution d'une variable de gestion. (b) résolution d'un agent. (c) résolution d'un gestionnaire. (d) résolution d'un paire gestionnaire-agent.

Dans une approche gestionnaire-agent, deux facteurs fondamentaux affectent les performances d'une activité de surveillance :

- Période de mise à jour sur l'agent, noté AUP (*Agent Update Period*) : les intervalles de temps prévus pour la mise à jour des valeurs d'une variable sur l'agent.
- Période de scrutation sur le gestionnaire, noté MPP (*Manager Polling Period*) : les intervalles de temps prévus pour la collecte des valeurs de variables.
- Période de surveillance de la variable sur l'agent, noté ANP (*Agent Notification Period*) pour vérifier si sa valeur a dépassé un seuil prédéfini.

Les AUP et les MPP sont les périodes d'entrées d'une approche gestionnaire-agent. L'intensité de demande d'une variable, noté $I(V_i)$, par une fonction de gestion est définie par :

$$I(V_i, \Delta_i) = \frac{1}{\Delta_i} \tag{4.1}$$

avec Δ la période d'entrée de la variable. En revanche, l'intensité de notification d'une variable possède un caractère aléatoire qui se présente sous la forme de la probabilité que sa valeur dépasse un seuil. Dans une approche reposant sur un gestionnaire-agent, on peut distinguer les résolutions de sortie suivantes principalement axés sur les entités de gestion :

- résolution de l'agent, noté RA : l'entrée sont les requêtes provenant de gestionnaire et les sorties sont les réponses portant les valeurs de variables.

- résolution du gestionnaire, noté RM : l'entrée est la MPP qui présente la période de scrutation et les sorties sont les requêtes interrogeant l'agent.
- résolution d'un paire gestionnaire-agent RMA : l'entrée est la MPP et les sorties sont les mises à jour des valeurs des variables sur le gestionnaire.

Il faut noter que dans un mode de notification, ces paramètres sont aussi valides en inversant les rôles joués par le gestionnaire et l'agent pour le transfert de valeurs de variables. L'agent devient la source et le gestionnaire est désormais le récepteur.

À la réception de la valeur d'une variable depuis un agent, le gestionnaire met à jour sa base d'information de gestion. La réception de deux messages différents ne signifie pas que les valeurs qu'ils transportent sont différentes. Cela est dû à une période de mise à jour AUP importante ou que l'agent est incapable de mettre à jour ses variables au cours d'un faible laps de temps. En effet, la résolution réelle, notée RR d'une variable est définie comme étant le temps minimal entre deux instances d'une opération transportants deux valeurs différentes d'une variable depuis un agent.

4.2.2 Capacité de la gestion

La capacité d'une approche de gestion se manifeste sous plusieurs formes. Pour un opérateur réseau la capacité d'un système de gestion se présente [104] comme étant sa :

- capacité à surveiller en temps réel les composants réseaux
- capacité à traiter un très grand nombre d'évènements, d'alarmes, de messages dans un court intervalle de temps
- capacité de corrélation d'alarmes

D'une manière générique, nous proposons la définition suivante :

Définition 4 : *La capacité de la gestion est sa rapidité d'absorption des problèmes qui peuvent survenir sur un système géré.*

Concrètement, cette absorption repose sur la détection, l'isolation et l'élimination des problèmes. En effet, elle nécessite la collection, le stockage et l'analyse des variables de gestion du système géré.

On s'aperçoit que pour mesurer la capacité de la gestion, nous devons définir et instancier un ensemble de métriques qui mesurent sa rapidité pour accomplir ses tâches. Plus l'approche de gestion est rapide, plus son estimation des problèmes est rapide.

Des métriques de rapidité

Cette famille de métriques mesure la rapidité de transfert de variables de gestion entre les différentes entités impliquées dans une approche de gestion. Dans la suite, nous définissons l'ensemble de métriques qui capturent cette rapidité.

Débit de données de la gestion On définit le débit de la gestion entre deux entités e_1 et e_2 à un instant t, noté $\gamma(e_1, e_2, t, T)$, comme étant le nombre de variables transférées correctement entre ces deux entités en réponse à une intensité de demande de gestion durant l'intervalle $[t, t+T]$ divisé par T. Cette métrique dépend des facteurs que nous avons identifiés dans la section précédente portant sur les périodes et les résolutions de variables. Idéalement, des faibles périodes de gestion engendrent des débits importants, mais des importantes résolution introduites sur les entités e_1 et e_2 peuvent entraîner une baisse de ces débits. Le débit est relatif à deux entités de gestion ou à un groupe d'entités de gestion. Entre deux entités de gestion, la métriques

de débit est de type *un-à-un*, d'ailleurs sa définition est celle présentée précédemment. Entre un groupe d'entités de gestion, le débit est de type multi-parties [155]. Une métrique est de type multi-parties, si elle implique plusieurs entités. Dans le cas de sa mesure entre une et plusieurs entités différentes, elle est de type *un-au-groupe*. Dans le cas de sa mesure entre plusieurs et une entité, elle est de type *groupe-au-un*. Par exemple, si l'ensemble des entités est $<e_1, e_2,...,e_N>$, avec e_1 le gestionnaire alors la mesure de débit s'effectue entre $<e_1,e_2>$, $<e_1,e_3>$,...,$<e_1,e_N>$. Ainsi en se basant sur cette terminologie développée dans IPPM [155], le débit est de type : *un-au-groupe* lorsqu'on le mesure entre un gestionnaire et N agents, et si c'est le gestionnaire qui déclenche la fonction de gestion, par exemple dans le cas d'une surveillance par scrutation. En revanche, le débit est de type *groupe-au-un* dans le cas où se sont les agents qui déclenchent la fonction de gestion comme dans le cas d'une surveillance par notifications. La figure 4.2 illustre la relation entre une métrique de type *un-au-groupe*, une métrique de type *un-à-un* et un singleton de mesure (voir section 3.4.2). Cette illustration est aussi valable pour une métrique de type *groupe-au-un*, où les destinataires (les agents) deviennent les sources de variables (les notifications par exemple).

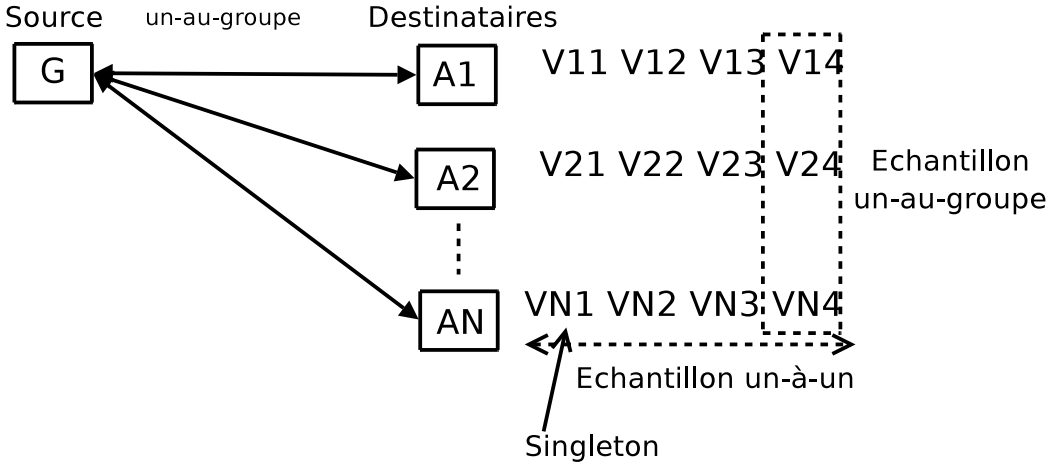


FIG. 4.2 – Illustration de la métrique de type *un-au-groupe* entre un gestionnaire G et un groupe de N agents A_i sous un mode de scrutation d'un ensemble $\{V_{ij}\}$ de variables.

En se basant sur cette définition, le débit d'une approche de gestion est défini comme étant le débit minimum mesuré entre ces différentes entités. Particulièrement, pour une approche centralisée basée sur la scrutation impliquant un gestionnaire G et un groupe $\{A_i\}$ de N agents, le débit de l'approche est une métrique de type *un-au-groupe* [155], noté $\gamma(G,t,\Delta)$ est égale à :

$$\gamma(G,t,\Delta) = min\{\gamma(G,A_i,t,\Delta), i \in \{1..n\}\} \tag{4.2}$$

L'unité de mesure du débit d'une approche gestion ou l'une de ses entités est en variables par seconde (*vps*). La figure 4.3 présente le débit un-au-un mesuré entre un gestionnaire et un agent JMX sous une intensité de supervision de 100 *getAttribute*/seconde. Il faut noter qu'il est indispensable de spécifier le type de l'opération employée lors de la mesure de débit ainsi que sa multiplicité en terme de nombre de variables qu'elle transporte.

Capacité de collecte massive La capacité de collecte massive est la capacité d'une approche de gestion de transférer une quantité importante de données de gestion en une seule opération.

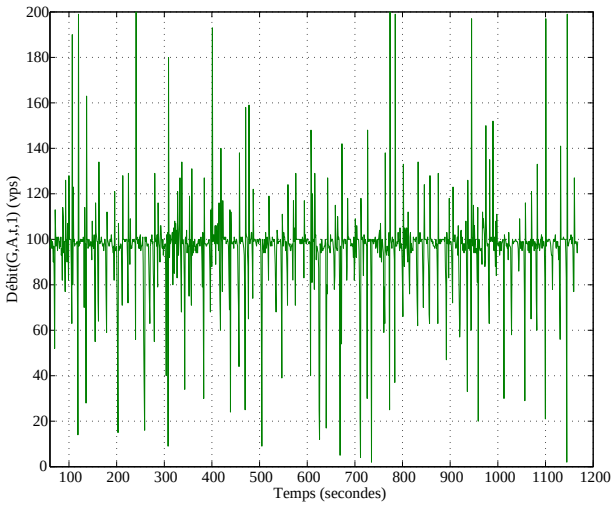


FIG. 4.3 – Débit de type *un-à-un* en variables par seconde (vps) en fonction de temps t entre un gestionnaire, noté G, et un agent JMX, noté A. L'intensité de supervision est de 100 *getAttribute*/seconde. La multiplicité de *getAttribute* est égale à 1.

Dans une gestion par SNMP l'opération de type *GetBulk* effectue ce genre de transfert. Dans une gestion basée sur les web-services [126], avec des passerelles (voir 2.2.1) opérant au niveau objet, l'implantation d'opérations de type *GetifTable* permettant de transférer toutes les données de gestion de la table des interfaces d'un équipement réseau est un autre exemple de collecte massive. La définition de la capacité de collecte massive est la suivante :

$$CCM = \frac{\text{Nombre de variables collectées par les messages de réponse de l'opération de collecte massive}}{\text{Temps écoulé}} \tag{4.3}$$

Le besoin de définir une métrique claire pour mesurer cette capacité est justifié par la diversité des algorithmes de recherche des objets de gestion dans une table. Trois algorithmes sont définis pour la collecte de données de gestion depuis une table [87]. Le premier algorithme opère colonne-par-colonne, le deuxième opère ligne par ligne de la table et le troisième effectue une collecte d'une colonne avec ses différentes lignes. Lors de la mesure de cette capacité de collecte massive la méthodologie doit spécifier l'algorithme sur lequel repose la collecte.

Les métriques des délais Dans un système de gestion, les délais ont une grande importance pour la simple raison que les fonctions de gestion sont toutes dépendantes du temps. Par exemple, une fonction de surveillance dépend du temps nécessaire pour collecter les valeurs de variables. Une fonction de contrôle doit acquérir la valeur d'une variable dans un temps fini sinon, il sera difficile de garder le contrôle sur le système géré. Plusieurs travaux se sont intéressés aux délais que subissent les messages d'un protocole de gestion [142, 139, 91]. Dans notre approche, nous définissons les métriques de délais pour une variable de gestion indépendemment du message qui la transporte. Cela nous permettra de rester dans notre objectif de généricité des métriques que nous nous sommes fixé. Toutefois, lors de la description de la métrique de délais à mesurer, il faut spécifier le type de l'opération qui transporte sa valeur. Le délais d'une variable de gestion est l'intervalle de temps entre le début de l'appel de l'opération qu'elle la transporte et la fin de

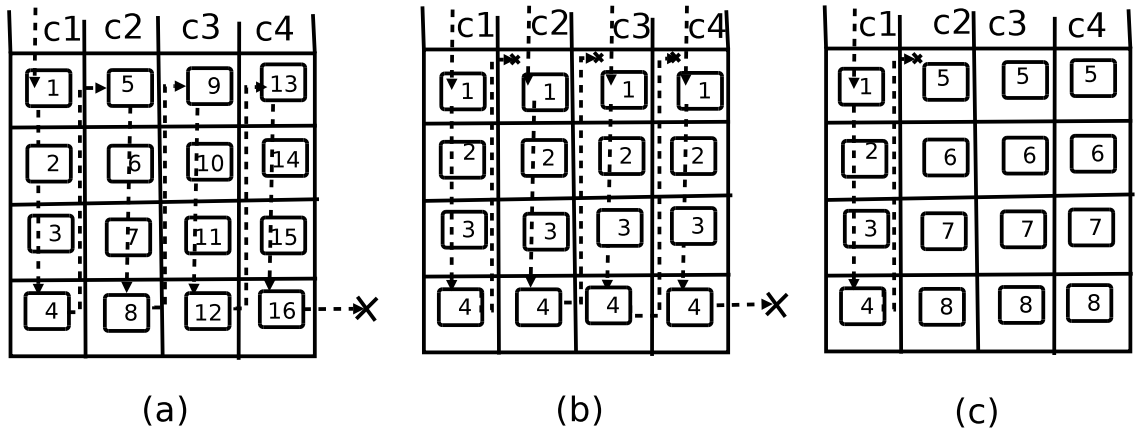


FIG. 4.4 – Différents algorithmes de collecte de données de gestion depuis une table SNMP.(a) colonne-par-colonne.(b) ligne par ligne. (c) colonne plus ligne.

la récupération de sa valeur. Comme indiqué pour les métriques de débit, les métriques de délais sont multi-parties et mesurables entre différentes entités de gestion. Les métriques de délais sont de type *un-à-un*, dans le cas de leur mesure entre deux entités différentes. Ces métriques mesurent le délai que subit une variable par le biais d'une opération spécifique entre ces différentes entités. Les métriques de délai sont de type *un-au-groupe* ou *groupe-au-un* dans le cas de mesure de délais que subit une variable lors de transfert de sa valeur entre plusieurs entités.

Dans une fonction de gestion, on distingue deux familles de délais selon la nature des opérations qui transportent les variables : les délais unidirectionnels des variables et les délais bidirectionelles des variables.

Définition 5 : *Le délai bidirectionel d'une variable est l'écart entre l'instant du départ de sa requête depuis une entité de gestion pour récupérer sa valeur depuis une autre entité de gestion et l'instant du retour de la réponse transportant sa valeur.*

Définition 6 : *Le délai unidirectionnel d'une variable est l'écart entre l'instant du départ de sa valeur depuis une entité de gestion et l'instant de son arrivée sur une autre entité.*

Le délais bidirectionel représente le temps d'aller-retour d'une opération de gestion sur une entité de référence qui facilite sa mesure. En revanche, le délai unidirectionnel représente le temps d'aller-simple d'une opération de gestion entre deux entités ce qui est plus difficile à mesurer à cause des problèmes de synchronisation des horloges entre les deux entités. Il faut noter que la méthodologie de mesure doit préciser à quel niveau les mesures de délais sont effectuées : niveau applicatif, niveau noyau du système d'exploitation ou niveau paquets [13]. Les délais de gestion et leurs caractérisations seront présentés avec plus de détails dans le chapitre 8.

Les pertes À ce niveau de définitions des métriques de la gestion, la perte d'une variable de gestion signifie qu'elle n'a pas acquis une valeur correcte, après sa manipulation par une opération, dans un délai fini choisi d'une façon raisonnable. Une valeur incorrecte d'une variable de gestion est due soit à une erreur au niveau des messages du protocole de gestion sous-jacent générés par l'opération ou soit à une erreur dans la couche de communications en dessous de protocole de gestion. La figure 4.5 représente la perte de la valeur d'une variable entre les instants 6 et 7 secondes. Cette perte est détectée par l'écoulement d'un délai d'une seconde après l'appel d'une

opération sur cette variable sans que la variable possède une valeur correcte. Nous définissons la

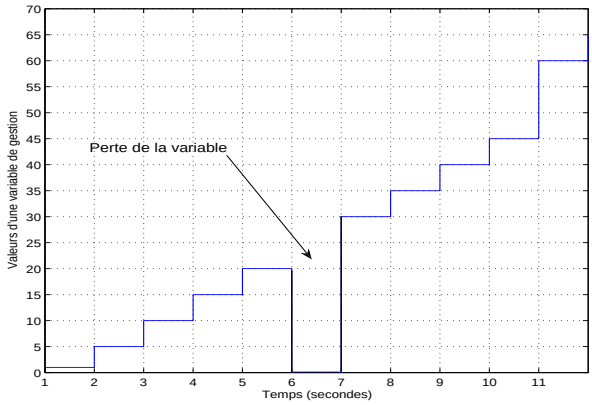


FIG. 4.5 – Description du phénomène de perte d'une variable de gestion en fonction de temps, lors de transfert de ses valeurs sur un gestionnaire par le biais d'une opération de lecture.

perte d'une variable v_i comme étant une fonction f définie de la manière suivante :

$$f(v_i) = \begin{cases} 0 \text{ si } \mathrm{D}(v_i) \leq \Upsilon \\ 1 \text{ si } \mathrm{D}(v_i) > \Upsilon \end{cases}$$

où $D(v_i)$ est son délai respectif et Υ est le seuil de détection de sa perte. À partir de cette définition de la perte d'une variable de gestion, nous pouvons définir certaines statistiques comme le taux de perte défini comme étant le rapport entre le nombre de variables perdues et le nombre total des variables transférées. Nous pouvons aussi identifier des motifs de pertes comme la distance entre les pertes et la période des perte [156]. Il faut noter qu'une méthodologie de mesure de la perte doit spécifier la valeur de Υ pour distinguer entre une perte et un délai très large mais fini. Des valeurs possible de Υ sont soit 255 secondes ce qui représente la durée de vie ou le *hop count* d'un paquet IP [162][55], soit l'intervalle de surveillance de la variable étant donné que l'arrivée de la valeur de cette variable au delà de cette intervalle rend inutile cette valeur.

4.2.3 Coût de la gestion

Le coût[56] de la gestion et de ses fonctions a été étudié par plusieurs travaux [72, 39, 108, 124]. Leurs but était d'avoir une meilleure compréhension de ce coût afin de le réduire [124, 48, 41]. Cependant, malgré la diversité de ces études et leur objectif commun, chacune d'elles propose des métriques différentes de coût en le considérant comme un indicateur d'efficacité sans définition claire et précise. Il est donc nécessaire, avant d'entamer une étude d'optimisation du coût de la gestion, de définir clairement ce concept et de mettre au point les métriques associées. Le coût de la gestion inclut plusieurs aspects. On trouve le coût propre à la gestion introduit par ses activités sur les différentes entités de gestion. On trouve aussi, le coût intrusif de la gestion imposé sur le système géré. Ce coût est dû au partage des ressources entre les activités de gestion et les fonctions primaires destinées aux utilisateurs du système géré. Dans la suite, nous définissons

[55] Le champ TTL d'un paquet IP est interprété comme étant le temps restant avant que le paquet devient obsolète. Sa valeur limite est de 255 sauts ou 255 secondes en raisonnant en terme de durée de vie.

[56] En anglais : *overhead*.

les métriques de coût de la gestion indépendement de sa nature en se basant sur le coût d'une variable de gestion.

Métriques de coût

Une approche de gestion est en compétition au niveau ressources matérielles avec le système à surveiller et à contrôler. Cette compétition est légitime puisqu'elle représente le coût à payer pour le maintien du système géré. Éventuellement, ce coût peut être de nature non intrusif et il est propre à la gestion lorsque la fonction de gestion possède ses propres ressources. Dans ce contexte un réseau overlay est mis en place et dédié aux activités de gestion [55]. En revanche, même dans ce cas, la fonction de gestion doit accéder au système géré pour manipuler l'une des variables qui représente l'un de ses objets réels et elle consommera ainsi des ressources sur ce système géré.

Coûts d'une variable de gestion Le coût d'une variable v_i est la somme de ses coûts de traitement, de stockage et de communication lors du son appel par une fonction de gestion (surveillance, contrôle ou initialisation) par le biais d'une opération [84]. Nous obtenons ainsi :

$$c(v_i) = ct(v_i) + cs(v_i) + cc(v_i) \tag{4.4}$$

Définition 7 : *Le coût de traitement d'une variable est le nombre des cycles processeur consommés sur chaque entité de gestion impliquée lors du son utilisation par une fonction de gestion par le biais d'une opération du protocole de gestion.*

Ce coût est essentiellement introduit par l'encodage et le décodage de la variable pour être transporté dans un message du protocole de gestion ainsi que le temps nécessaire pour accéder à la valeur de la variable sur l'objet géré. Des facteurs comme le type de l'encodage, la granularité de l'opération et la nature de la variable (simple ou tabulaire) affectent ce coût [142].

Définition 8 : *Le coût de stockage d'une variable est l'espace mémoire qu'elle occupe sur chaque entité de gestion.*

Une application de gestion possède une zone mémoire statique allouée au début de son exécution dans laquelle réside ses instructions. Au cours de son exécution, de la mémoire dynamique est allouée à la réception ou l'émission d'un message de gestion transportant une variable et libérée après la transmission ou le traitement d'une réponse.

Définition 9 : *Le coût de communication d'une variable est le nombre de bits de communication qu'elle introduit dans un réseau lors de son appel par une fonction de gestion par le biais d'une opération du protocole de gestion.*

Ce coût est dû aux appels à des opérations au cours de la réalisation d'une fonction de gestion. Il se présente sous la forme d'un nombre de messages du protocole de gestion transportant les valeurs des variables. À ces messages de gestion s'ajoutent aussi les messages propres au protocole de communication sous-jacent. Ceci augmente évidement le trafic réseau reliant les différentes entités de gestion impliquées dans la fonction de gestion. Par exemple, dans une approche JMX reposant sur RMI, l'appel d'une opération *getAttribute* nécessite au moins 4 messages RMI qui sont la requête, la réponse et deux message de vérification de la connexion (voir figure 2.12).

On s'aperçoit que ces coûts ont des dimensions différentes. Afin de remédier à ce problème, nous proposons de considérer ces coûts sous la forme de fractions par rapport à la capacité

de la ressource matérielle sur laquelle ils sont pris. Par exemple, le coût processeur se mesure sous la forme de la fraction d'occupation de CPU, le coût mémoire se mesure sous la forme de la fraction occupée par rapport à la taille de la mémoire. Ces différents coûts primaires dépendent des propriétés temporelles présentées dans la section 4.2.1. Par exemple, une variable impliquée dans une fonction de surveillance en mode scrutation avec un intervalle de surveillance $MPP = \Delta$, qui temporise son rafraîchissement sur un gestionnaire tous les Δ secondes, possède un coût qui est amortit par cet intervalle. En effet, plus la période MPP est importante, plus le coût est faible. Nous remarquons que le coût d'une variable est normalisé par ses propriétés temporelles. Ainsi, le coût du temps de gestion, noté $C(v_i, MPP)$ d'une variable de gestion en fonction de sa propriété temporelle MPP est définie de la façon suivante :

$$C(v_i, MPP) = \frac{c(v_i)}{MPP} \tag{4.5}$$

Il faut noter qu'ici nous avons considéré le coût du temps de gestion en fonction de MPP qui est sa période de surveillance sur un gestionnaire, mais on peut toujours le calculer en fonction de l'une des périodes d'une variable définies dans la section 4.2.

4.2.4 Maintien et mise à jour du système géré dans un état désiré

Tout commence par la Fin, par l'objectif à atteindre. L'objectif ultime d'une approche de gestion est de maintenir un système géré dans un état désiré fixé dans le cadre d'une politique ou d'un contrat de service [62]. Il est très courant de décrire la qualité comme la satisfaction des utilisateurs.

Ce maintien repose sur l'activité d'observation. Le résultat de cette observation est un ensemble de propriétés du système géré (paramètres de sa qualité de service) qui quantifie son état, et obtenue grâce à la capacité du système de gestion décrite précédemment. La question qui s'impose maintenant est : « ***quel est l'écart entre l'état du système géré observé et la réalité ?*** ». La réponse à cette question, nécessite que l'on définisse une famille de métriques qui quantifie ce degré de vraisemblance entre l'état réel du système géré et l'état résultant observé par un système de gestion.

Des métriques de la qualité

La qualité d'une vue observée du système géré dépend notamment du rapprochement des données de gestion de l'état réel du système. En effet, les valeurs des variables de la vue du système géré peut dévier considérablement des d'objets réels qu'elles représentent à cause du bruit introduit lors de sa surveillance ou son contrôle (voir section 2.2.3). Par exemple, une scrutation toutes les 5 minutes d'un réseau IP d'un opérateur est soumise à un *bruit* puisque le gestionnaire ne parviendra pas à accomplir la surveillance des toutes les variables des MIBs des milliers des interfaces réseaux de centaines de routeurs au même instant dès le début de la période de surveillance de 5 minutes. Cela introduira un décalage au niveau des périodes de surveillance. En effet, ces périodes deviennent plus petites ou plus grandes que 5 minutes [106].

Sensibilité La sensibilité exprime la plus petite variation de valeur de l'objet réel géré qu'une variable peut déceler. En effet, elle représente le quotient de la variation de la valeur de l'objet réel observé (son changement), noté VR_i par la variation de la valeur de la variable de gestion, noté V_i, représentant cet objet. On définit la sensibilité d'une variable de gestion de la façon

suivante :

$$S(V_i) = \frac{dV_i}{dVR_i} \tag{4.6}$$

La sensibilité moyenne est la pente de la courbe présentée dans la figure 4.6. En effet, on obtient : $S = tan(\Phi)$.

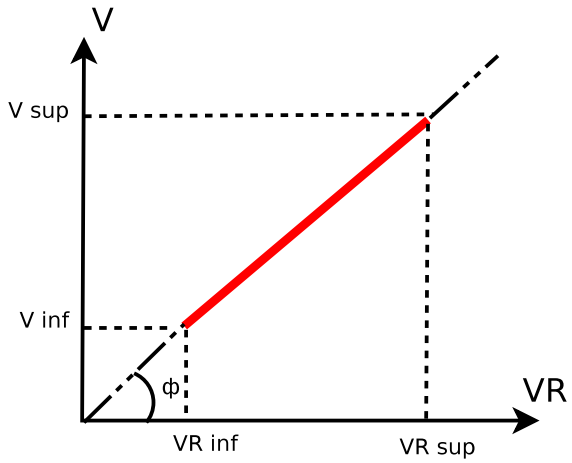


FIG. 4.6 – Illustration de la sensibilité d'une variable de gestion.

Il faut noter que la définition de la sensibilité est valide seulement pour des variables de gestion de types compteur impliqués dans une fonction de surveillance. Cela parce que un compteur est une fonction monotone croissante. Il est d'ailleurs recommandé d'utiliser des compteurs à la place des jauges pour leur représentativité de l'objet géré sous forme de moyenne en fonction de temps [117]. Malgré qu'ils n'apparaîssent pas dans l'équation 4.6, les facteurs qui affectent la sensibilité sont essentiellement, la période AUP de l'agent et le MPP du gestionnaire, parce que ces ce sont ces périodes de mise à jour des valeurs de VR_i et V_i.

Exactitude : précision de données Une variable de gestion est d'autant plus précise que les valeurs qu'elle indique coïncident avec la valeur de l'objet réel qu'elle représente. La précision est définie par l'erreur de précision. Elle s'exprime en unité de grandeur (erreur absolue) ou en pourcentage (erreur relative). La précision d'une variable est exprimée par deux métriques : l'exactitude et la justesse. Dans la suite, nous posons une variable de gestion V et O l'objet réel qu'elle représente. Soit $v(V,t)$ le numéro de version de la variable V à un instant t. Ce numéro est incrémenté à chaque mise à jour de la variable, c'est à dire à chaque rafraîchissement sur le gestionnaire par le biais d'une opération vers l'agent. Soit $v(O,t)$ le numéro de version de O à un instant t. Ce numéro est incrementé à chaque mise à jour de O. Nous pouvons alors définir la divergence entre V et O à un instant donné par la fonction $\delta(V,t)$. Lorsque un rafraîchissement est effectué à un instant t, on aura alors $\delta(V,t) = 0$. Entre deux rafraîchissements consécutifs la valeur de divergence pourra augmenter. En effet, sa valeur dépend des périodes de rafraîchissement et des délais que subissent les opérations. En se basant sur ce concept de divergence, nous définissons l'exactitude d'une variable de la façon suivante :

$$\delta(V_i,t) = \begin{cases} 0 \text{ si } v(V_i(t)) = v(O_i(t)) \\ 1 \text{ si } v(V_i(t)) \neq v(O_i(t)) \end{cases}$$

Autrement, nous pouvons mesurer la divergence par la différence de version : $\delta(V,t) = v(V_i(t)) - v(O_i(t))$ ou la déviation de valeurs $\delta(V,t) = \Delta(V(t), O(t))$, où $\Delta(V_1, V_2)$ est une fonction qui quantifie la différence entre deux version de la variable de gestion et l'objet réel. La valeur de $\Delta(V_1, V_2)$ représente le bruit de la variable de gestion par rapport à l'objet réel qu'elle représente. Il faut noter que cette dernière métrique de précision est souvent difficile à mesurer d'une manière empirique à cause de problème de synchronisation des horloges entre deux entités de gestion (un gestionnaire et un agent par exemple). Elle est souvent utilisée dans un cadre analytique. Dans [112], Stadler et al ont utilisé cette métrique de déviation des valeurs pour évaluer une approche de gestion basée sur l'agrégation et le filtrage des variables. Cette métriques a été aussi utilisée dans le travail de Zaho et al [106] portant sur l'estimation des matrices de trafic réseau avec des données imparfaites d'estimation.

Précision temporelle La précision temporelle[57] [76, 85] est une manière de mesurer le bruit que subit une variable au cours de son rafraîchissement (surveillance ou contrôle). Elle caractérise la temporalité des opérations que subit une variable pour acquérir ses valeurs au cours d'une fonction de gestion. Une variable de gestion possède une forte précision temporelle si le délai qu'elle subit, noté $D(V_i)$, pour acquérir une valeur par la biais d'une opération dans une fonction de gestion est inférieur ou égal à un délai de tolérance, noté τ. Ce délai de tolérance représente la durée de vie de la valeur actuelle de la variable. Au delà de ce délai cette valeur obsolète et elle n'est plus utile pour le système de gestion. Évidemment les délais sont mesurés par l'une des métriques que nous avons proposées dans la section 4.2.2. En effet, la précision temporelle, noté θ, est définie de la manière suivante :

$$\theta(V_i) = \begin{cases} 0 \text{ si } D(V_i) \leq \tau \\ 1 \text{ si } D(V_i) > \tau \end{cases}$$

Le délai de tolérance est une contrainte à définir qui dépend de l'application de gestion et de son besoin de la précision temporelle. Par exemple, des applications de gestion temps réel [90] nécessitent des délais de variables de gestion bornés et très faibles de l'ordre de millisecondes. La figure 4.7 illustre l'évolution de la précision temporelle en fonction des temps d'une variable de surveillance dans une approche JMX entre un gestionnaire et 700 agents.

4.2.5 Compromis coût-qualité

Entre ces différentes familles de métriques primaires que nous avons identifiées pour la mesure de performances de la gestion, un compromis [112, 41] est nécessaire. Ce compromis est dû au fait que la qualité et le coût d'une approche de gestion ne sont pas uniformément améliorables. Un conflit apparaît à chaque fois qu'on doit privilégier l'un de ces deux critères. En effet, la réduction du coût d'une variable qui signifie éventuellement une augmentation de l'une de ses périodes MPP sur le gestionnaire ou AUP sur l'agent, peut entraîner une réduction de sa qualité en terme de l'exactitude de ses valeurs et de sa sensibilité. Cette réduction de la qualité n'est pas forcement souhaitable pour des applications de gestion qui nécessite une parfaite synchronisation entre la variable observée et l'objet réel. D'autre part, l'augmentation de la qualité de la variable qui signifie une diminution de l'une de ses périodes de surveillance entraîne une augmentation du coût qui n'est pas toujours bon pour certaines applications de gestion restreintes au niveau des ressources matérielles.

[57] En anglais : *timeliness*

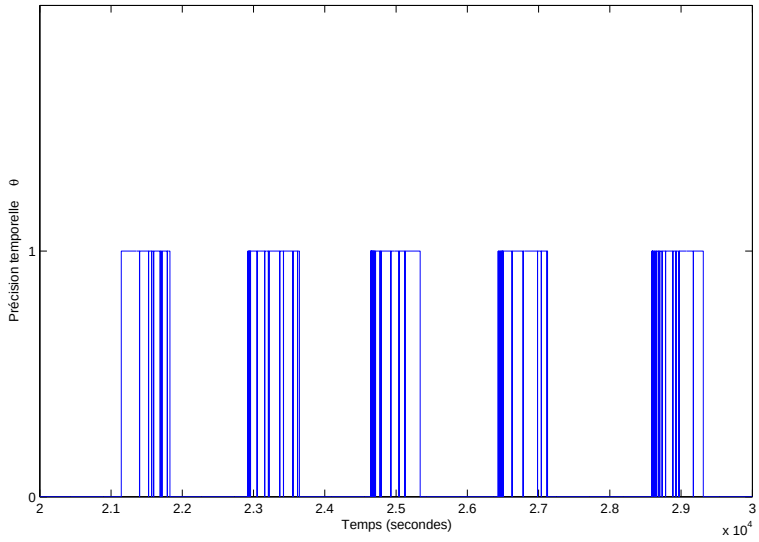


FIG. 4.7 – Précision temporelle d'une variable en fonction du temps. La variable est surveillée sur un ensemble de 700 agents avec une période MPP=1 seconde par agent. Le délai de tolérance τ= 1 seconde.

Pour mesurer ce compromis entre le coût et la qualité d'une variable de gestion nous identifions le ratio d'accès à la mise à jour d'une variable AUR (*Access Update Ratio*) qui est défini de la façon suivante :

$$AUR(V_i) = \frac{MPP(V_i)}{RR(V_i)} \tag{4.7}$$

où RR est la résolution réelle minimale de la variable V_i (voir 4.2.1) qui est définie à partir d'un échantillon $\{RR_j\}$ par la formule suivante : $RR(V_i) = Min(\{Max(AUP, RR_j(V_i))\})$. Le ratio AUR capture notamment l'utilité de la surveillance de la variable puisque toutes les $\{RR_j(V_i)\}$ secondes on détecte un changement de la valeur de cette variable. En effet, si la valeur de RR est importante, cela signifie que la variable ne change pas fréquement et il n'est pas utile de la surveiller avec une faible période MPP. Le paramètre MPP capture le coût de la variable car toutes le MPP secondes un coût est introduit sur les entités de gestion. Selon la valeur que prend AUR, trois cas se présentent :

- $AUR < 1$: cela signifie que $RR(V_i) > MPP(V_i)$ et que la variable est en état de sur estimation de l'objet réel qu'elle représente puisqu'elle introduit un surcoût inutile.
- $AUR = 1$: cela signifie que $RR(V_i) = MPP(V_i)$ la variable observée est synchrone avec l'objet réel qu'elle représente et son coût est optimal.
- $AUR > 1$: cela signifie que $RR(V_i) < MPP(V_i)$ la variable est en état de sous estimation et éventuellement la précision de la variable observée est mauvaise.

Un algorithme d'adaptation de la période du surveillance MPP à la résolution réelle de la variable a été proposé par Raz et al [124]. L'algorithme proposé nommé *Simple-rate* ajuste la période de surveillance de telle manière qu'elle détecte le prochain dépassement de seuil en se basant sur l'estimation de sa résolution réelle comme nous l'avons définie. Ce ratio AUR garantit seulement une synchronisation temporelle entre la variable de gestion et l'objet réel qu'elle représente, mais dans le cas de faibles périodes de MPP ou de AUP, le coût devient important et les entités de gestion deviennent surchargées et la précision temporelle de la variable se dégrade

considérablement.

4.3 L'efficacité d'une approche de gestion

Les métriques de rapidité, de coût et de qualité présentées précédemment sont de type primaire selon la terminologie définie par IPPM. En revanche, d'autres métriques sont indispensables pour avoir une meilleure compréhension de l'efficacité d'une approche de gestion. A ce stade de définition de métriques de performances de gestion, nous pouvons analyser séparément les différentes métriques primaires ou on peut s'appuier sur seulement l'une d'elles pour identifier l'efficacité de l'approche. Une telle méthode d'analyse de ces métriques donne une vision partielle de l'efficacité de l'approche. En effet, une analyse partielle ne permet pas de juger l'efficacité d'une approche de gestion d'une façon pertinente puisque elle ne capture pas le compromis qui existe entre les trois familles de métriques. La compréhension globale de l'efficacité nécessite la mise en relation des métriques primaires que nous avons définies. Nous définissons l'efficacité d'une approche de gestion comme un ratio entre la valeur du travail accompli et son coût d'accomplissement. Nous obtenons, ainsi, l'efficacité de l'approche de gestion, notée $E(k)$, sous un facteur d'évaluation k :

$$E(k) = \frac{R(k) \times Q(k)}{C(k)} \tag{4.8}$$

Cette métriques est inspirée de travail de Woodside et al [23] dans un contexte d'évaluation de systèmes distribués. La métrique $R(k)$ est l'une des métriques qui décrit la rapidité, $Q(k)$ est l'une des métriques qui décrit la qualité et $C(k)$ est l'une des métriques qui décrit le coût. Le facteur k permet d'indexer un vecteur de valeurs d'un facteur d'évaluation. Par exemple, k peut désigner l'index d'un vecteur de périodes de surveillance dans lequel on varie les valeurs de ces périodes pour évaluer le comportement d'une fonction de surveillance. Cette métrique composite nous servira de base dans les chapitres 6 et 7 pour quantifier le passage à l'échelle d'une approche de gestion ainsi que son incidence sur la performance d'un système géré.

4.4 Besoins des applications de gestion en terme de performances

Les applications de gestion ont des besoins différents en terme de performances et elles introduisent des coûts différents lors de la réalisation des fonctions de gestion [48]. Par exemple une application de collecte des données de comptabilité (*accounting*) a un faible besoin au niveau des délais qui peuvent varier entre 10 secondes à plusieurs heures. Par contre elle va effectuer des transferts de gros volumes de données de gestion (activités des utilisateurs pour la facturation, etc), et les pertes doivent être quasiment nulles puisque les informations sont précieuses et l'enjeu financier est énorme. En revanche, une application de surveillance temps réel [90] a besoin de faibles délais pour avoir une vue suffisament récente du système géré, idéalement elle doit introduire un coût faible pour ne pas perturber les activités des utilisateurs et posséder une perte moyenne. Le tableau 4.1 présente des besoins en terme de performances pour certaines applications de gestion. Les valeurs présentées dans ce tableau sont à titre indicatif. Nous les avons obtenues par analogie avec d'autres applications indépendamment de la gestion [64]. Toutefois, il faut noter que les besoins de l'application de gestion en terme de performances et le coût qu'elles introduisent dépendent fortement de la nature du système géré et de sa mission [132]. Il existe une différence entre une même application de surveillance en temps réel d'un laboratoire académique de recherche et une banque. Cela dépend notamment de l'enjeu financier de la disponibilité de l'information qui est plus important dans une banque que dans un laboratoire académique.

Applications \ Métriques	Rapidité			Précision		Coût
	Débit	Délais	Pertes	Exactitude	précision temporelle	
Surveillance temps réel	constant : MPP $<$ 100 ms	$<$ 150 ms	faible	moyenne	importante	important
Collecte statistique	important	10 secondes à quelques heures	zéro	importante	faible	important
Configuration	variable	2 à 5 secondes	zéro	importante	faible	variable

TAB. 4.1 – Performances réquisent par certaines applications de gestion ainsi que le coût qu'elles introduisent.

4.5 Synthèse

Dans ce chapitre nous avons proposé un ensemble de métriques déterministes, claires et précises pour mesurer la performance d'une approche de gestion. Ces métriques sont présentées sous la forme de trois familles : rapidité, coût et qualité. La rapidité mesure le débit en terme de variables par secondes, les délais que subissent les variables manipulées par le biais d'une opération dans le cadre d'une fonction de gestion. Le coût mesure les ressources matérielles (processeur, mémoire et communication) que consomme une variable lors de sa manipulation par une opération de gestion. La qualité mesure essentiellement la qualité d'une variable de gestion en terme de sa déviation par rapport à l'objet réel qu'elle représente et en terme de sa précision temporelle après sa manipulation par une opération de gestion. À partir de ces métriques primaires, nous avons proposé une métrique dérivée pour quantifier l'efficacité d'une approche de gestion qui met en relation ces trois familles des métriques. Maintenant que ces métriques sont définies, nous avons besoin dans la suite d'identifier une méthodologie pour leurs mesures dans un cadre cohérent des paramètres qui les affectent.

Chapitre 5

Méthodologie de mesure de la performance de la gestion

Sommaire

5.1 Introduction

Une fois le travail de définition des métriques de performances inhérentes à une approche de gestion terminé, nous nous sommes intéressé à la manière de les mesurer. Pour ce faire, nous avons développé une méthodologie de mesure de ces métriques. Le travail effectué dans cette voie a tout d'abord consisté à abstraire les paramètres d'une approche de gestion. Pour cela, nous avons défini la notion de pratique de gestion qui représente un arrangement d'un ensemble des paramètres de l'approche pour son évaluation. Une fois la pratique sous test identifiée, ses paramètres sont déployés sur les entités de gestion et ses performances sont mesurées sous un facteur d'évaluation. Dans ce chapitre nous détaillons tout d'abord, la définition d'une pratique de gestion et son intérêt pour élaborer une méthodologie de mesure répondant aux critères de continuité et de répétabilité. En se basant sur cette notion de pratique, nous décrivons la mise en œuvre d'un banc de mesure dédiée pour l'évaluation des approches de gestion nommé, *MAGON : Management trAffic GeneratOr and beNchmark* [58] qui implantent des pratiques de gestion pour mesurer leurs performances selon les métriques spécifiées dans le chapitre 4. Les difficultés rencontrées lors de cette mise en œuvre sont présentées, ainsi que leurs solutions respectives. Dans la dernière partie de ce chapitre, nous présentons un modèle de performances d'une paire gestionnaire-agent JMX afin de calibrer ces entités dans nos expérimentations que nous présenterons dans la dernière partie de ce manuscrit.

[58] Dans l'histoire, Magon est un général carthaginois, frère d'Hannibal. Actuellement, ce nom est l'appellation d'un excellent vin rouge Tunisien.

5.2 Définition des pratiques de gestion

Comparer la performance de deux approches de gestion d'une façon objective est pratiquement impossible parce qu'il n'existe pas deux réseaux à gérer identiques, et que la bonne performance d'une approche pour gérer un type de réseau, peut être vue comme médiocre pour un autre type. Néanmoins, toute approche de gestion définit un ensemble de pratiques implantées en accord avec un modèle de gestion. Ainsi, la mesure de la performance d'une approche de gestion d'une manière cohérente, répétitive et continue nécessite l'isolation et la définition des pratiques qui affectent sa performance. Tout d'abord, nous proposons la définition suivante d'une pratique de gestion.

Définition 10 : *Une pratique de gestion est un arrangement des paramètres relatifs à une approche de gestion pour accomplir une ou plusieurs fonctions de gestion.*

La figure 5.1 présente une description des paramètres communs des approches de gestion traditionnelles. L'identification d'une pratique repose sur la sélection de ses paramètres en parcourant verticalement ce diagramme. La définition de sa charge d'évaluation s'effectue en variant horizontalement les valeurs de l'un ou plusieurs de ses paramètres. En effet, ce diagramme présente clairement la complexité de l'évaluation des approches de gestion pour couvrir toutes ses sources de variation.

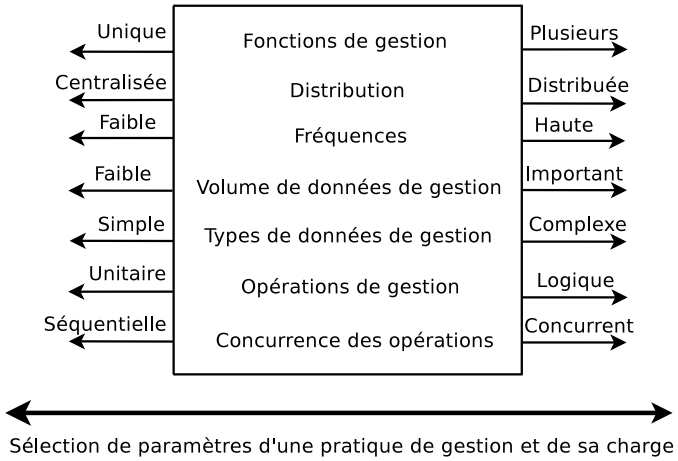


FIG. 5.1 – Sélection de différents paramètres d'une pratique de gestion et de leurs valeurs.

Cette évaluation par pratique de gestion nous permet d'assurer la continuité de la mesure et sa répétabilité [160]. En effet, la répétition de la mesure de performances d'une pratique avec les mêmes valeurs de ses paramètres, nous fournit des résultats similaires. La continuité signifie que l'évaluation d'une pratique avec une petite variation des valeurs de ses paramètres se traduit par une petite variation des résultats. Les paramètres d'une pratique de gestion sont ceux présentés dans le chapitre 2 dans la section 2.2.3 portant sur la caractérisation des approches de gestion. Chaque test de mesure repose sur la spécification de l'ensemble de paramètres de la pratique sous test ainsi que les valeurs qu'ils prennent. Un exemple d'une pratique de gestion est le suivant :

- Approche gestionnaire-agent
- Technologie JMX
- Scénario : 1 gestionnaire, 1 agent
- Agent en mode pilote (*driver*)

- Instrumentation interne
- Fonction de surveillance
- Opérations concurrentes
- Opération : *getAttribute*
- Variable : $getFreeMemory$ dans un MBean standard
- Intensité de demande de surveillance : [1 :50 :500] *getAttribute*/s[59].

Le facteur d'évaluation de cette pratique de gestion est l'intensité de la surveillance qui prend ses valeurs dans le vecteur spécifié.

5.3 Mise en œuvre du banc de mesure *MAGON*

Dans cette section, nous présentons la conception d'un banc de mesure des approches de gestion modelées autour du concept gestionnaire-agent. Ce banc a pour objectif l'évaluation de performances des pratiques de gestion basées sur l'ensemble des paramètres du modèle gestionnaire-agent tel que nous l'avons présenté dans le chapitre 2. Les objectifs de notre banc de mesure sont les suivants :

- Évaluer le coût d'une pratique de gestion d'une approche spécifique en utilisant des métriques définies dans le chapitre 4. Cette évaluation nécessite l'abstraction et le déploiement de l'ensemble des paramètres de la pratique de gestion sous test ;
- Élaborer des campagnes de mesures qui identifient systématiquement des problèmes de performances dans une pratique de gestion en variant un paramètre sur un vecteur de valeurs. Par exemple, une instance du banc de mesure reposant sur la technologie JMX nous a permis d'identifier qu'un MBean avec un nombre considérable d'attributs engendre un problème de performances au niveau de l'agent [5] ;
- Comparer les performances de différentes pratiques de gestion. Cette comparaison permet d'identifier la pertinence, en terme de performances, d'une pratique de gestion pour gérer un système (service ou application) dans un environnement spécifique. Par exemple, évaluer la pertinence d'une pratique de supervision pour gérer des services déployés sur une grille en utilisant un mode de scrutation ;
- Développer une suite de tests automatisée, flexible et modulaire, supportant différentes pratiques de gestion et différentes technologies d'implantation.

Les métriques mesurées sur le banc de mesure sont celles présentées dans le chapitre 4. Pour atteindre ces objectifs nous avons dû résoudre certaines difficultés dont les solutions sont présentées dans la section suivante.

5.3.1 Difficultés et solutions

Notre approche de mesure se base sur la terminologie utilisée dans [21][page 61] où notre système sous test (SUT) est une pratique de gestion avec l'ensemble de ses paramètres associés. Au cours de la conception d'un banc de mesure pour les approches de gestion nous avons été confronté aux difficultés suivantes :

- l'abstraction des paramètres d'une pratique de gestion sous test et leurs répartitions sur les différentes couches d'une approche de gestion ;
- la génération et injection des charges liées à une pratique de gestion ;

[59] Nous utilisons getAttribute/se au lieu de variables/s parce que dans ce cas ils sont équivalents à cause de la multiplicité égale à 1 de *getAttribute*.

– l'élaboration d'une méthodologie de mesure des métriques pour capturer la performance et le coût d'une pratique de gestion.

Abstraction et déploiement de paramètres d'une pratique de gestion

Une étape importante lors de mesures de performances d'une pratique de gestion consiste à définir les paramètres de test incluant aussi les facteurs d'évaluation. Nous avons défini ces paramètres d'une façon générique pour qu'on puisse comparer les performances d'une pratique sous différentes technologies de gestion (JMX, SNMP). Dans ce but, ces paramètres sont relatifs aux trois couches indispensables dans une approche de gestion traditionnelle qui sont : l'instrumentation, l'agent et le gestionnaire.

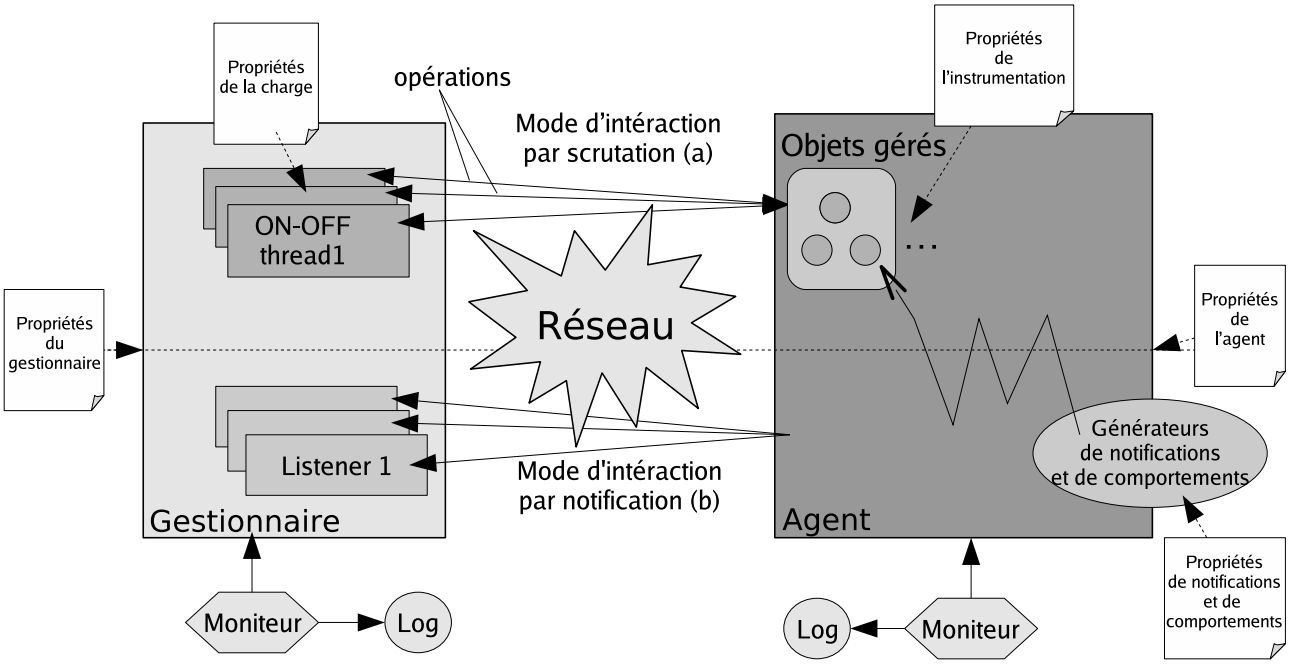


FIG. 5.2 – Description du banc de mesure de performances d'une approche de gestion basée sur un modèle gestionnaire-agent reposant sur une technologie JMX.

Au cours de la réalisation d'un test de mesure sur la plate-forme physique, ces paramètres sont déployés sur les entités de gestion (les gestionnaires et les agents) sous la forme des fichiers de propriétés. Nous distinguons ainsi, 4 types de fichiers de propriétés relatifs à une pratique de gestion :

– Les propriétés de la couche instrumentation incluant le volume de données de gestion et leurs types (simple ou tabulaire). Par exemple, pour JMX, il s'agit du nombre de MBeans (un ou plusieurs), le nombre d'attributs exposés par chaque MBean, le type de ces MBeans et le type de leurs attributs ;
– Les propriétés de la couche agent incluant les périodes de mise à jour des variables, le connecteur qu'il expose pour les gestionnaire et son modèle d'intégration au sein du système géré. Pour un agent JMX on doit définir le type de connecteur (RMI, SOAP), le lien d'invocation du service JMX sur le serveur MBean et son modèle d'intégration dans le service géré ;

– Les propriétés de la couche gestionnaire incluant le nombre d'agents, les propriétés du connecteur pour chaque agent (type et lien de service), le mode d'interaction (scrutation ou notification), le modèle de distribution de la fonction de gestion (centralisé, faiblement centralisé ou fortement centralisé) et le niveau de concurrence des opérations ;
– Les propriété de la charge d'évaluation incluant la fréquence d'injection de la charge de chacune de variables mise en jeu, les types des opérations de gestion employées, leurs nature (unitaire ou logique) et leurs multiplicité (nombre de variables par opération).

Il faut noter que les propriétés de la charge sont déployées sur les entités de gestion responsables de son injection. L'architecture d'un banc de mesure pour le modèle gestionnaire-agent est représentée sur la figure 5.2.

Tout d'abord, elle montre les différents composants impliqués dans la conception du banc de mesure. Les injecteurs de charges, les générateurs de notifications et de comportements ainsi que le déploiement de différents paramètres d'une pratique sur ses différentes entités et couches. Ensuite, le positionnement des points de mesure au niveau des entités impliquées. Dans la suite nous allons présenter la conception de ces différents composants pour répondre aux objectifs que nous nous sommes fixés.

Caractérisation du trafic de gestion

Une fonction de gestion repose sur un ensemble de variables possédant des propriétés temporelles (leurs fréquences de surveillance, de contrôle, etc). Cette fonction implique une ou plusieurs entités de gestion sur lesquelles une ou plusieurs sessions sont maintenues pour le transfert de valeurs de variables. La figure 5.3 représente un modèle d'une session de gestion entre deux entités. Elle illustre les différents niveaux d'un trafic de gestion.

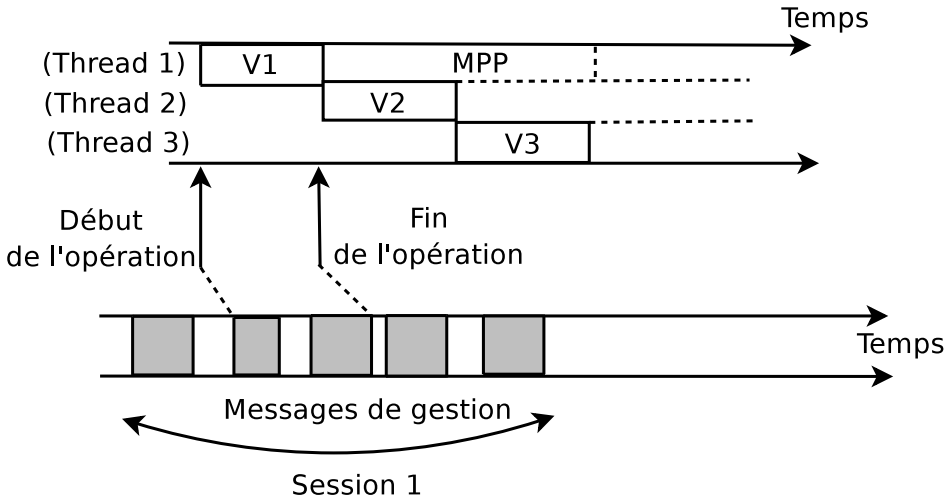


FIG. 5.3 – Modèle d'une session de supervision comme implanté par le banc de mesure *MAGON*.

Une fonction de gestion est vue sous trois niveaux différents : sessions, cycles, messages et variables. Une session entre deux entités de gestion est la durée d'une activité de gestion. Une session est caractérisée par une paire <*source, destination*> représentant les adresses des entités de gestion impliquées. Par exemple, la durée d'une activité de configuration (*provisioning*) ou de surveillance de la vue d'un réseaux de taille donnée sont des exemples de session. Une session est composée d'un ou plusieurs cycles qui représentent les intervalles de temps entre différentes

instances d'une opération portant sur une ou plusieurs variables. Les cycles peuvent se chevaucher ou être séparés, selon le modèle de concurrence des opérations de gestion Au cours d'un cycle, une opération de gestion génère un ou plusieurs messages de gestion, selon sa nature (simple ou logique) et le nombre de variables transportées par message. Il faut noter que, dans notre terminologie, une variable de gestion n'est pas forcément un attribut de gestion simple (par exemple un OID SNMP), mais peut être toute sorte de variables nécessaire pour refléter l'état d'un système géré. Le modèle d'injection d'une charge de gestion sur notre banc de mesure doit prendre en considération ces différents niveaux du trafic de gestion.

Comportement de l'application gérée

Le comportement d'un système quelconque est la partie de son activité qui se manifeste à un observateur[60]. Une pratique de gestion intervient dans plusieurs points du cycle de vie d'une application gérée [50]. Cette intervention, selon la pratique de gestion et sa fonction associée, se manifeste sous plusieurs formes. Par exemple, les pratiques de gestion interviennent pour la configuration de certaines variables de l'application gérée ou pour la collection périodique des ses variables d'états. Chacune de ces interventions est fortement liée au comportement en fonction du temps de l'application gérée. En effet, la détection de problèmes surviendra seulement si le comportement de l'application ou de certaines de ses variables observées présentent des dépassements de seuils prédéfinis.

Une façon de modéliser le comportement de l'application gérée, est de le définir comme étant une trajectoire dans un espace multidimensionnel où chaque dimension représente une variable de son état. Lors du déplacement de la performance de l'application gérée sur cette trajectoire, sa qualité de service acceptable (QdS) est limitée par une région spécifique de cet espace. Si la performance de l'application gérée se déplace hors de cette région, les pratiques de gestion interviennent à nouveau pour le ramener dans cette région spécifique. Dans notre banc de mesure d'une approche de gestion, le comportement d'une application gérée est présent ou absent selon l'objectif de la mesure. Dans des tests de mesures synthétiques [21][page 50] d'une pratique de gestion, où seule la performance propre de la pratique de gestion est considérée, ce comportement est absent. Ces tests **synthétiques** permettent essentiellement d'identifier les problèmes de performances qui peuvent émerger dans une pratique de gestion sans prise en considération de l'état du système géré. En revanche, dans des tests de mesure plus réalistes la présence de ce comportement permet d'identifier l'impact de l'état du système géré sur la performance de la pratique de gestion. Par exemple, si le système géré est chargé (charge fonctionnelle exercée par les utilisateurs du service), la performance d'une pratique de gestion ne sera pas la même que dans le cas d'un système géré non chargé. Dans le paragraphe suivant nous décrivons la façon d'implanter dans notre banc de mesure ces comportement liés à une application gérée.

Générateur de trafic et des comportements

Tout travail de mesure nécessite la définition du système d'injection de la charge (*workload*) [21]. Nous avons utilisé une approche similaire à celle de [12] pour développer nos systèmes d'injection de la charge relative à une pratique de gestion. L'approche est basée sur l'utilisation des *threads* d'injections où chaque *thread* représente une opération unique de gestion. Une opération peut être soit une opération de lecture, d'écriture, d'invocation ou de génération d'une notification. Cette approche multi-threads est suggérée dans un document de standardisation [87]

[60] Source wikipedia : http ://www.wikipedia.org

relatifs au protocole SNMP, elle utilisée dans plusieurs outils de gestion de réseau (Nagios, Cricket, MRTG, HP OpenView) [43] et adoptée par plusieurs des outils de supervision développés dans notre équipe de recherche.

Dans le cas d'un mode d'interaction par scrutation les threads d'injection sont implantés au niveau du gestionnaire. Chaque *thread* est responsable de l'envoi d'une opération par unité de temps. En effet un thread d'injection met en œuvre un processus de type ON/OFF qui alterne entre des périodes d'activités où il déclenche des opérations et des périodes d'inactivités où aucune activité n'est générée. Le pseudo code d'un thread d'injection des variables de gestion implanté sur le gestionnaire est représenté par l'algorithme 1 : Ce pseudo code permet la génération d'un

Algorithme 1 Pseudo code d'un thread d'injection de variables de gestion

```
Require: MPP, VARIABLES_List, V_COUNT
  while TRUE do
    SLEEP MPP item
    while VARIABLES remain in VARIABLES_List do
      I = NEXT V_COUNT item
      while I > 0 do
        V = V, NEXT VARIABLES_List item
        I = I -1
      end while
      OPERATION V
      SLEEP NEXT MPP item
    end while
  end while
```

trafic de gestion ainsi que la simulation de différentes périodes temporelles d'une variable. Chaque thread nécessite la définition des vecteurs suivants : $VARIABLES\_List$, MPP, V_COUNT. Le vecteur $VARIABLES\_List$ représente la liste de variables à transférer par le thread par le biais d'une opération. Le vecteur MPP contient les périodes de surveillance, de contrôle ou de notifications de valeurs de variables. Le dernier vecteur nommé V_COUNT représente le nombre de variables à transférer au cours d'une période MPP. Il indique la multiplicité de l'opération de gestion. Ainsi, à chaque *thread* on rattache cet ensemble de vecteurs défini dans le fichier de la charge relatif à l'entité qui joue le rôle de l'injecteur des opérations de gestion. Ces vecteurs peuvent être produits d'une manière synthétique (à la main) ou éventuellement générés à partir d'une analyse de traces réelles d'un trafic de gestion. La figure 5.4 présente un exemple d'une trace des opérations JMX ainsi générées. La figure 5.5 représente le séquencement de la génération de différent vecteurs rattachés à un *thread*.

Ainsi, le nombre total de *threads* présents sur une entité de génération de la charge représente le degré de concurrence de la pratique de gestion sous test. Ce degré est défini dans le fichier de propriétés de la charge déployée sur cette entité. Dans le cas d'un mode d'interaction par notification, les threads de génération sont présents sur l'agent. Chaque *thread* est responsable de l'envoi d'une notification par unité de temps. Ainsi, le taux d'injection est une fonction (produit) du nombre total de threads et du nombre de *listeners* enregistrés présents sur les gestionnaires. Selon le modèle de distribution de la fonction de gestion entre les agents et les gestionnaires, les threads d'injection sont répartis sur des gestionnaires intermédiaires ou sur certains agents pour accomplir certaines fonctions de gestion portant sur les valeurs des variables de gestion (somme, moyenne, maximum ou minimum).

Nous avons utilisé un autre ensemble de threads pour modéliser le comportement de l'appli-

```
#Time            #Variable  #Operation
1167827485.51068 BootTime  getAttribue
1167827486.51406 MaxThreads  getAttribute
1167827488.52592 MaxTimesConnectionUse  getAttribute
1167827489.53385 HostName  getAttribute
1167827490.53805 BootTime  getAttribute
1167827492.55465 ServerSocketInfo  getAttribute
1167827493.55788 AccessLogged  getAttribute
1167827494.56185 SessionsCount  getAttribute
1167827495.56984 ThreadGroupName  getAttribute
1167827496.5778  SessionsCount  getAttribute
```

FIG. 5.4 – Exemple d'une trace des instants de surveillance d'un ensemble de variables depuis un serveur web par le biais de l'opération JMX *getAttribute*.

cation gérée et l'impact de la gestion sur ce comportement. Ils sont implantés soit dans le même processus que l'agent dans le cas où le modèle d'intégration est de type composant ou noyau selon la terminologie de [73] ou dans un processus séparé dans le cas où le modèle d'intégration est de type démon. Ces threads ont pour objectif de générer les valeurs des objets réels qu'elles représentent les variables de gestion. Les valeurs de ces objets réels dépendent du temps et présentent un caractère aléatoire [124]. Raz et al [124] dans leur travail portant sur l'efficacité de certaines approches réactives de supervision ont présenté un ensemble de modèles de comportements d'un objets géré O_i :

- Comportement à changement uniforme : la valeur de l'objet O_i à un instant $t+1$ est définie par : $O_i^{t+1} = O_i^t + y_i^t$, où y_i^t est une variable aléatoire uniformément distribuée sur un intervalle $[-k..k]$. En effet, la valeur de l'objet à l'instant $t+1$ dépend uniquement de sa valeur précédente, et de son changement uniforme. Le choix de la valeur de k est définie depuis la plage de variation de l'objet O_i. La consommation CPU d'une application gérée est un exemple de ce comportement à changement uniforme. La figure 5.6(a) représente la génération d'un comportement uniforme d'un objet géré avec une plage de valeurs entre $[0..100]$ et $k = 25$.
- Comportement à changement normal : la valeur de l'objet O_i à un instant $t+1$ est aussi définie par : $O_i^{t+1} = O_i^t + y_i^t$, où y_i^t est une variable aléatoire de distribution normale avec une valeur moyenne de 0 et un écart type k. La figure 5.6(b) représente la génération d'un comportement suivant une distribution normale de moyenne 0 et d'écart type 1.
- Comportement fractal (*self-similar*) : C'est un comportement avec une forte dépendance temporelle reposant sur des mécanismes de génération de type ON/OFF [10]. En effet un comportement self-similar est dû à la superposition d'un nombre considérable de sources ON/OFF. Une description des comportement self-similaire est présentée dans [80]. La figure 5.6(c) représente la génération d'un comportement self-similair due à la superposition de 10 sources ON/OFF avec des périodes suivants des distributions Pareto.

Il faut noter que les valeurs que prennent les objets réels sont positives et généralement bornées par une valeur maximale (limité par son codage et son espace mémoire). En revanche, le processus aléatoire pourrait générer des valeurs en dehors de ces bornes. Dans ce cas, la valeur aléatoire générée est ajustée pour qu'elle soit entre ces bornes. Afin de modéliser le comportement de type compteur, la valeur de l'objet géré est remise à zéro après un certain nombre d'unités de temps prédéfini.

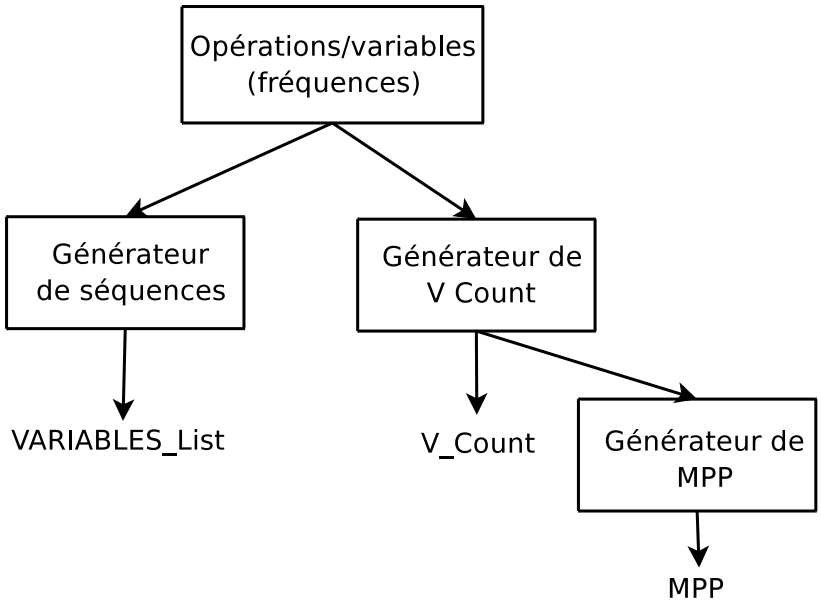


FIG. 5.5 – Génération de vecteurs de la charge rattachés aux threads d'injection de variables de gestion.

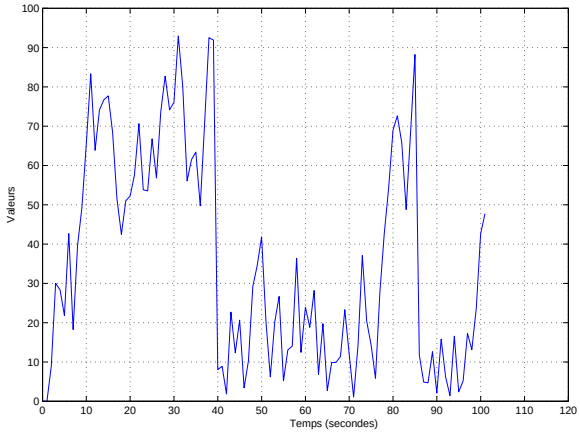
5.3.2 Instance JMX de MAGON

Pour illustrer notre banc de mesure *MAGON*, nous avons implanté une instance dédiée à la gestion basée sur la technologie JMX. Nous avons choisi JMX présenté dans la section 2.3 du chapitre 2 car c'est une proposition reconnue et standard dans le monde des applications et services basés sur la technologie Java. En outre, ce protocole offre une vaste variété de pratiques de gestion ce qui en fait un bon candidat dans le cadre de notre travail. Dans les paragraphes suivants nous détaillons certaines difficultés techniques reliées à la méthodologie mise en œuvre par cette instance de MAGON dédiée à l'approche JMX.

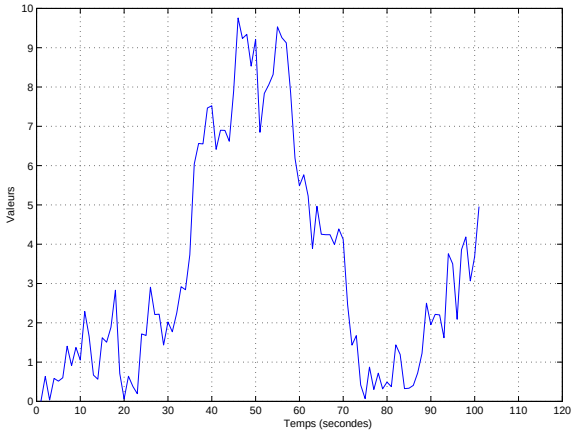
Problème *de démarrage à froid* (Cold Start)

Lors de la mesure de performances d'une pratique de gestion, la charge de gestion ainsi que le trafic qui lui correspond doit être injecté d'une façon progressive et non brutale. En effet, des technologies de gestion comme JMX ou d'autre venant du monde Java, implantent différents mécanismes de cache, des buffers pour les requêtes ainsi que des threads. Pour que ces différents mécanismes se mettent en place et deviennent efficaces, il faut l'injection d'un faible ensemble de requêtes. Parce qu'une injection instantanée avec un nombre considérable de *threads* rend ses mécanismes inefficaces et fait écrouler le banc de mesure rapidement. Cela nous empêche de mesurer un échantillon représentatif d'une métrique de performances d'une pratique de gestion sous test. Afin de remédier à ce problème, comme il a été recommandé dans [51], il faut injecter la charge d'une façon progressive jusqu'au l'obtention de la valeur désirée de *threads* d'injection spécifiée dans le fichier de propriétés de la charge. La durée de cette injection progressive représente la période d'amorçage de test (*warm-up*).

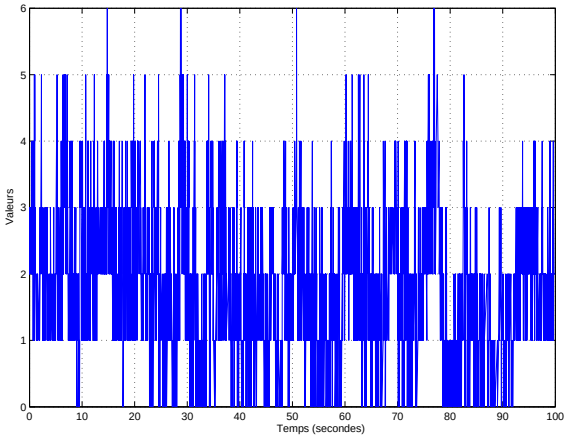
La figure 5.7 illustre l'évolution de l'intensité d'injection en fonction de la période d'amorçage. Cette période doit être suffisamment longue pour amorcer convenablement les threads d'injection. Actuellement la valeur de cette période venant de l'expérience est configurée manuellement.



(a)



(b)



(c)

FIG. 5.6 – Différents type de comportements d'un objet géré. (a) comportement à changement uniforme $O^{t=0} = 0$,$k = 25$,$O_{max} = 100$. (b) comportement à changement normal avec $O^{t=0} = 0$,k=1,$O_{max} = 100$. (c) comportement fractale (*self-similar*) d'une superposition de 10 sources de type ON-OFF avec des distributions Pareto de leurs périodes.

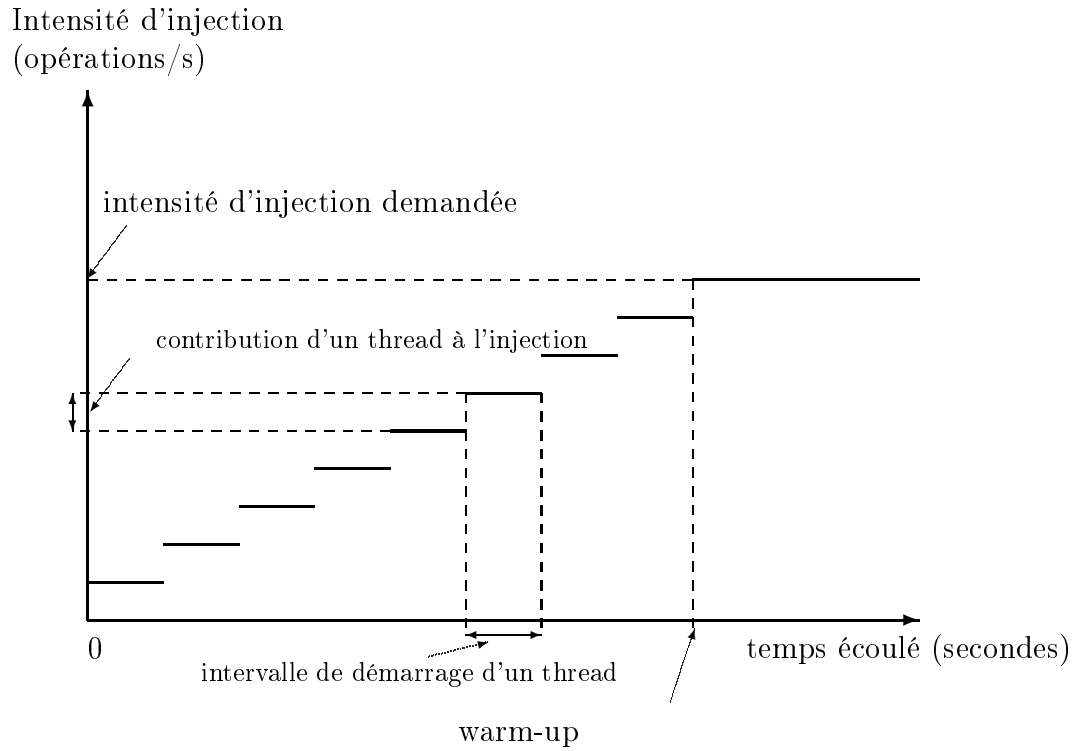


FIG. 5.7 – Illustration de l'effet de l'amorçage *rampup* sur l'intensité d'injection demandée.

Quantification des incertitudes temporelles

L'une des difficultés que nous avons rencontrées lors du développement de notre banc de mesure concerne des imprécisions temporelles des méthodes utilisées pour mesurer les délais ainsi que celles pour temporiser les opérations (méthode *SLEEP* de l'algorithme 1 de la page 81). Ces incertitudes présentées dans la section 3.4.4 du chapitre 3 et comme il a été recommandé par les documents IPPM doivent être quantifiées. Dans cette instance JMX de MAGON, nous avons utilisé pour mesurer les délais, la méthode Java *System.currentTimeMillis*(). Cette méthode permet d'obtenir le temps actuel en millisecondes. Nous avons mené quelques tests pour quantifier l'erreur qu'elle introduit ce qui constitue sa résolution. Sur un système d'exploitation Linux, qui est celui sur lequel repose nos mesures, l'erreur est de l'ordre de 1 ms. Ainsi, les délais inférieurs à 1 ms ne peuvent pas être mesurés par cette méthode. Une solution plus efficace pour mesurer les temps est l'utilisation de l'instruction *RDTSC* sur les plates-formes Intel, l'appel système *gethrtime()*[61] pour les plates-formes Solaris ou *QueryPerformanceCounter()*[62] pour les plates-formes Windows.

Une autre source d'erreurs et d'incertitudes temporelles est due aux temporisations. Par exemple, comme le montre la figure 5.8 l'utilisation de la méthode Java *sleep* présente un phénomène de latence importante entre la temporisation théorique désirée qui est de 10 ms et celle réellement obtenue par la méthode *sleep* qui varie entre 16 et 17 ms. Si celui-ci est à peine visible lorsque la temporisation est importante, il n'est pas de même lorsque on désire obtenir des temporisation faibles.

[61] http ://docs.sun.com/app/docs/doc/806-0627/6j9vhfmt5
[62] http ://support.microsoft.com/kb/172338

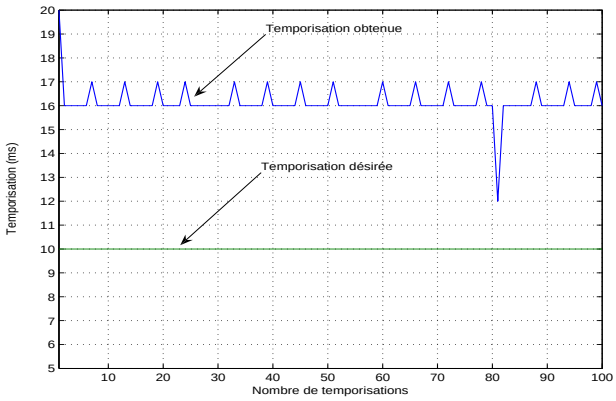


FIG. 5.8 – Erreur de la temporisation de la méthode Java *sleep*. La temporisation théorique est de 10 ms, et la temporisation obtenue est de l'ordre de 16 ms.

Points de mesure des métriques

Comme présenté dans le chapitre précédent, nous évaluons une pratique de gestion selon les trois familles de métriques : rapidité, coût et qualité. Pour leur mesure nous avons projeté ces trois familles de métriques sur les différentes entités impliquées dans la pratique sous test. Les métriques liées à la couche instrumentation mesurent le coût intrusif de JMX sur le système géré. Elles incluent essentiellement, la consommation de ressources lors de l'exposition de valeurs de variables représentant des objets gérés. Cette consommation inclut essentiellement le processeur et la mémoire, éventuellement réseau dans le cas de l'existence d'une communication entre l'agent et l'application gérée. Les métriques mesurées sur l'entité agent sont son débit en terme du nombre de variables de gestion par unité de temps correctement servies par l'agent suite à une charge injectée sous différents modes d'interactions (scrutation ou notification) et sous un nombre variables d'objets gérés. Le coût d'un agent est mesuré en utilisant un outil externe qui collecte la consommation processeur et le trafic réseau échangé entre l'agent et le gestionnaire. Dans ce travail nous avons utilisé un outil nommé SAR[63] qui fournit des mesures en fonction du temps de la consommation de ressources d'un processus sur une machine physique. Les métriques mesurées sur l'entité gestionnaire sont les suivantes : le débit en terme de variables correctement reçues en fonction de nombre d'agents, les délais des variables par type d'opération et la précision de données de gestion collectées. La mesure de performances d'une pratique de gestion, nécessite la sélection des métriques adéquates à son évaluation qui permettent de capturer sa performance et son coût. Cette sélection dépend essentiellement de l'objectif de l'évaluation (comparaison, surcoût ou pertinence) et des exigences en terme de performances de son algorithme de gestion comme indiqué dans la section 4.4. Par exemple, la mesure de la performance d'une pratique de supervision utilisant la scrutation pour la collecte de données de gestion à des fins statistiques nécessite l'ensemble des métriques relatives aux deux familles coût et rapidité. Cependant, les métriques de la famille qualité prennent de l'importance dans le cas d'une pratique de supervision réactive où la précision des donnée collectées ainsi que la qualité des délais deviennent critiques pour déclencher les opérations de gestion nécessaires à la résolution d'un problème.

[63] SAR acronyme de System Activity Report : http ://perso.orange.fr/sebastien.godard/index.html, visité en juillet 2007.

Mesure au niveau applicatif versus niveau réseau

Les métriques de débit, de la qualité et le trafic réseau généré sont mesurées soit au niveau applicatif soit au niveau réseau [13]. Au niveau réseau, la mesure est effectuée sur le lien entre un gestionnaire et un ou plusieurs agents sur le réseau qui les connecte. Au niveau applicatif, les méthodes de mesure font partie du code de l'application de gestion. La métrique de délai est celle la plus sensible à son niveau de mesure (applicatif ou réseau). En effet, la mesure des délais au niveau réseau ne présente pas une vue complète des délais que subit une variable lors de son transfert d'une entité de gestion à une autre. Cela est dû au fait que les délais réseau n'incluent pas des latences associées à l'encodage, décodage des messages, multiplexage de thread au niveau du gestionnaire et la bufferisation des notifications sur un agent. En revanche, la mesure de délais au niveau applicatif permet de capturer l'ensemble de latences et elle nous fournit le délai de bout-en-bout que subit une variable de gestion. Elle représente les délais vus par les algorithmes de gestion résidant dans le entités.

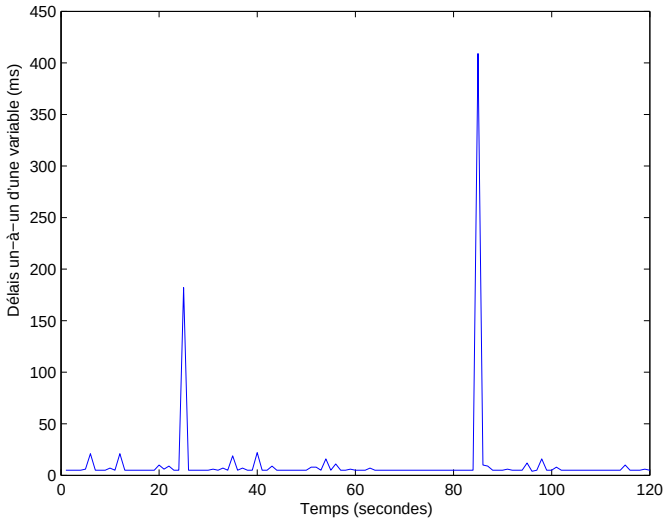


FIG. 5.9 – Mesure au niveau applicatif du délais un-à-un d'une variable de gestion sous une période de surveillance d'1 seconde.

La figure 5.9 représente une série de mesures de délais de la surveillance d'une variable au niveau applicatif sur le gestionnaire. On observe des piques périodiques toute les 60 secondes où le délai du transfert de la valeur de la variable entre le gestionnaire et l'agent augmente considérablement. Ce phénomène est dû aux activités du ramasse miettes GC (*garbage collector*) de la JVM déclenché toute les 60 seconde. Seule une mesure applicative au niveau du gestionnaire permet de détecter un tel phénomène. En revanche, une mesure au niveau applicatif introduit un surcoût de collecte de mesures et leurs stockages dans des fichiers et provoque la perte des informations des différentes composantes constituants le délai mesuré. Afin de combler ces lacunes, nous avons opté dans MAGON pour l'utilisation de l'un, de l'autre ou des deux niveaux selon le choix de l'utilisateur du banc de mesure.

5.3.3 Un dissecteur de trafic de supervision

La génération d'une charge relative à une pratique de gestion nécessite la collecte de traces de trafic pour la construction des vecteurs de paramètres d'injection rattachés aux threads. De plus, la mesure des métriques de rapidité et du coût de communication au niveau paquets nécessite que le point de mesure soit au niveau réseau. Pour répondre à ces besoins, il est indispensable d'utiliser ou mettre en œuvre des outils dédiés à l'analyse du trafic de gestion (des dissecteurs) pour la mesure de ses métriques et l'analyse de traces de trafic de gestion sous différents protocoles. Dans ce contexte, une première initiative visant à développer un dissecteur de trafic SNMP a lancée par Schoenwaelder et al [116]. Dans notre contexte de travail, nous avons développé un outil nommé *RmiDump* [56] qui permet d'analyser le trafic RMI employé par la couche de communication JMX. Cet outil repose sur sa propre déserialisation des messages RMI pour l'extraction des variables de gestion transportées dans chaque message. Les métriques sont ainsi mesurées pour chaque variable au niveau réseau. Ce genre d'outil nous permettra de mesurer le délai que subit une variable (voir section 4.2.1) au niveau de l'agent sans perturber son fonctionnement et de vérifier le bon fonctionnement d'un gestionnaire en comparant son débit en termes de variables par seconde au niveau applicatif à celui au niveau de réseau. Notre outil couplé avec un analyseur de trafic comme *Wireshark*[64] ou *tcpdump* permettra l'analyse des traces réseau d'un trafic JMX reposant sur RMI et d'extraire les paramètres relatifs aux pratiques de gestion.

5.3.4 Calibrage de la performance de la couche transport

L'objectif de notre banc de mesure est d'identifier des problèmes de performances propres à la pratique de gestion sous test indépendemment de son protocole de transport sous-jacent. Pour identifier clairement la performance propre de la couche de gestion, il est faut que le protocole de transport ne pose pas de limitations ou de problèmes de goulot d'étranglement. Les pratiques de gestion basées sur JMX que nous avons évaluées reposent sur le protocole TCP. Afin de vérifier que ce protocole ne pose aucun problème de performances dans nos tests, nous avons utilisé l'outil *netperf*[65] pour calibrer ses paramètres, essentiellement les tailles de ses buffers d'émission et de réception. Cet outil permet de simuler des flux TCP pour mesurer le débit de transmission massive du réseau (*Bulk transfert*) ainsi que la simulation d'un ensemble de transactions TCP de type requête/réponse TCP. Nous avons utilisé ces deux scénarios pour vérifier que la couche transport de système d'exploitation de la plateforme physique de test ne pose aucun problème. Notre plateforme de test repose sur un réseau Ethernet de capacité 1 Gigabits/s. La figure 5.10 montre les résultats de ces tests.

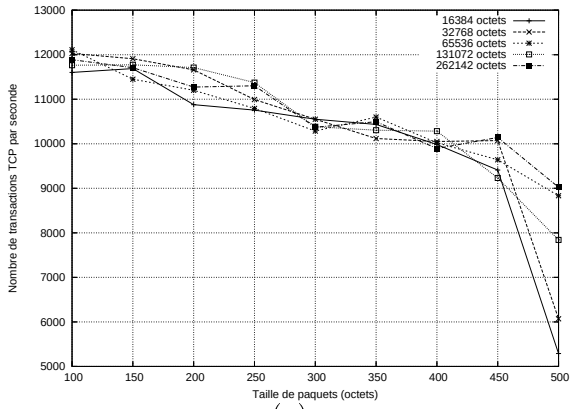
Dans ces tests, nous avons mesuré la performance de TCP en fonction de différents tailles de ses buffers ainsi que différents tailles de paquets. Nous avons aussi trouvé que l'effet de l'algorithme Nagle[66] sur notre environnement de mesure est négligeable.

Nous avons constaté que la couche transport TCP de la plate-forme de test ne pose aucun problème et ses débits sont de l'ordre de 11000 transactions par seconde et de 650 MBits/s pour des paquets de taille 200 octets . Ces débits sont largement au dessus de l'intensité de trafic de gestion demandé lors de nos tests.

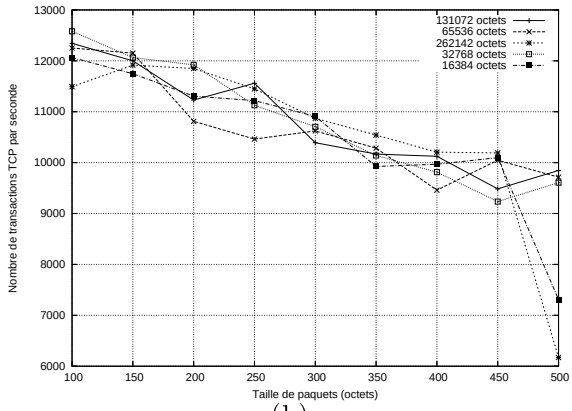
[64] Anciennement connu sous le nom d'Ethereal : http ://www.wireshark.org.

[65] http ://www.netperf.org/netperf/

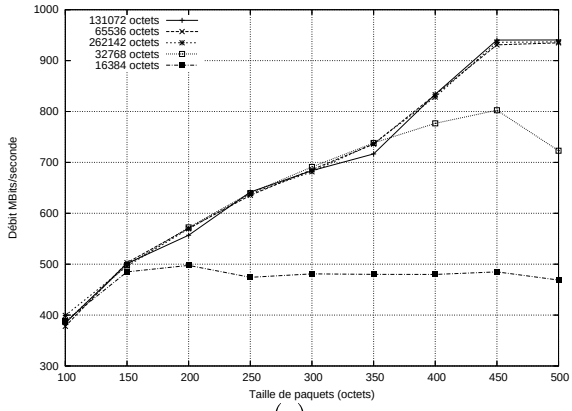
[66] Cet algorithme défini dans le RFC 896 propose de ne pas envoyer de données tant qu'on attend un acquittement TCP. Concrètement, il s'agit de positionner à vrai la valeur de l'option *TCP_NODELAY* d'une socket pour désactiver ce mécanisme.



(a)



(b)



(c)

FIG. 5.10 – Résultats de mesure de performances du protocole TCP en fonction de tailles de paquets transmis obtenus par l'outil NetPerf. (a) Scénarios Requêtes/Réponses TCP.(b) Scénarios Requêtes/Réponses sans Nagle.(c) Transfert massif TCP

5.3.5 Calibrage du support Multithreading de Java

L'instance JMX de notre banc de mesure repose sur l'utilisation du support multithread de Java. Ce support se présente sous la forme d'une API java pour la création et le contrôle des threads. Ce support, notamment la façon avec laquelle les threads sont ordonnancés dépend étroitement de l'implantation de la JVM sous-jacente. Afin de comprendre ce support d'ordonnancement et son effet dans nos mesures, nous avons effectué des tests pour collecter des informations sur la manière dont les threads sont ordonnancés dans la JVM sur laquelle repose notre banc de mesure. Nous utilisons une JVM BEA WebLogic JRockit reposant sur une JDK 1.4.2_04 déployée sur un système d'exploitation Linux RedHat avec un noyau 2.4.21-32.0.1.EL. Cette JVM repose sur un support natif de thread en s'appuyant directement sur les threads au niveau noyau du système d'exploitation. Le programme de test développé instancie un ensemble de threads de différentes priorités. Chacun des *threads* implante une boucle sans fin qui incrémente un compteur tant qu'il n'a pas été interrompu. La seule limitation dans ce cas est le processeur (CPU) puisque aucun thread n'effectue d'opérations d'entrée/sortie. Ainsi dans ce cas les seuls changements de contexte sont dus à l'ordonnancement des threads. La figure 5.3.5 présente les résultats de ce test.

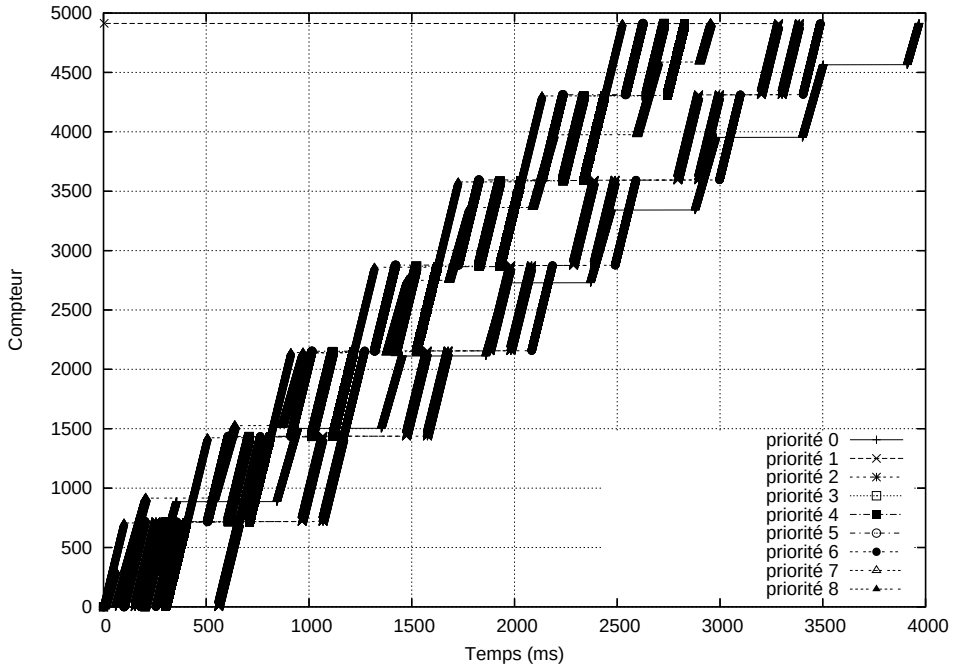


FIG. 5.11 – Ordonnancement des *threads* Java sur un système d'exploitation Linux.

On observe que la période d'exécution d'un *thread* avant sa préemption est de l'ordre de 100 millisecondes. La période d'inactivité du thread dans un état d'attente d'exécution est de l'ordre de 400 millisecondes. On s'aperçoit aussi que les priorités de threads n'affectent pas considérablement l'exécution des *threads* puisque l'ensemble des compteurs des *threads* atteignent des valeurs importantes. L'ordonnanceur de notre JVM de test repose sur un algorithme d'ordonnancement de nature tourniquet (Round Robin). Après cette phase de test de l'ordonnancement des threads fournit par notre JVM sur laquelle repose notre banc de mesure, nous avons gardé la priorité par défaut qui est égale à 5.

5.3.6 Qualités et limitations de banc de mesure MAGON

Notre banc de mesure satisfait la majorité des propriétés qui qualifient une bonne technique de mesure. En effet, si les paramètres et leurs valeurs introduits sur le banc de mesure sont ceux d'une pratique de gestion réelle collectées depuis un ensemble de traces d'une vraie application de gestion, alors les résultats produits sont *représentatifs* d'une réelle application de gestion. Le travaux de Schoenwaelder et al pourraient être utilisés pour définir les valeurs des paramètres liés aux pratiques réelles de gestion comme celles de SNMP. La méthodologie et la conception de notre banc de mesure est applicable à différentes approches de gestion parce qu'ils sont basés sur des caractéristiques communes et intrinsèques de ces approches.

Les résultats et les expérimentations produites par le banc de mesure sont *reproductibles* puisqu'ils reposent sur l'évaluation de pratiques de gestion bien définies et bien détaillées. De plus, les résultats d'évaluation de pratiques de gestion implantées sous des technologies différentes sont *comparables* dans la mesure où on garde les mêmes paramètres, facteurs d'évaluation et métriques. Les résultats obtenus par le banc de mesure peuvent contribuer à l'amélioration ainsi qu'au calibrage de certaines pratiques de gestion sur lesquelles reposent certaines applications de gestion largement déployées. Les éventuelles améliorations d'une approche de gestion reposeront sur l'identification de problèmes de performances que pourraient exposer certaines pratiques.

En revanche, notre banc de mesure souffre de certaines limitations. Son architecture actuelle capture essentiellement le modèle gestionnaire-agent ; ainsi des approches basées sur d'autres types de modèles (auto-gestion, pair-à-pair, agent mobiles) ne sont pas encore supportés. Les pratiques actuellement implantées ne supportent qu'une seule fonction de gestion à la fois. En effet, une prochaine étape sera d'inclure l'évaluation d'un processus de gestion qui met en œuvre plusieurs fonctions de gestion reposant sur différentes pratiques pour converger vers un objectif commun. Par exemple, un processus de supervision mettant en œuvre les fonctions de surveillance et de contrôle reposant sur différentes pratiques de gestion pour garantir une SLA (*Service Level Agreement*). Les algorithmes de gestion actuellement développés sont simples, ainsi il est indispensable d'inclure des algorithmes plus complexes pour analyser leurs performances. Par exemple, inclure les algorithmes de filtrage et d'agrégation de données de gestion élaborés par Stadler et al [149] dans notre banc de mesure afin de les comparer sous différentes technologies (qui utilisent leur propre protocole de gestion nommé TCA-GAP [67]). Il serait aussi intéressant d'inclure des modèles de variables de gestion avec des propriétés temporelles dynamiques où ses fréquences suivent une certaine distribution statistique.

5.4 Performance d'un agent JMX de supervision

Dans cette section, nous détaillons un modèle de performances d'un agent de supervision en utilisant l'instance JMX de MAGON. Les tests reposent sur deux implantations différentes de JMX. L'objectif de cette partie est d'identifier la capacité de l'agent ainsi que l'implantation la plus performante qui sera l'implantation de référence dans la suite. Nous présentons aussi une formulation analytique des tailles des messages JMX reposant sur le protocole RMI.

5.4.1 Étude analytique des tailles de messages JMX/RMI

Nous ne nous intéressons qu'au messages *Call* envoyés par le client vers l'agent pour invoquer une méthode et au message RMI *ReturnData* qui transporte la réponse de l'agent (voir section 2.3.3). Seuls ces deux messages transportent respectivement les données d'appel et de réponse.

[67] TCA-GAP : Threshold Crossing Alerts based on Generic Aggregation Protocol.

Comme nous l'avons expliqué dans la section 2.3.3, le protocole RMI utilise le mécanisme de sérialisation des objets java pour pouvoir donner une représentation binaire d'un objet avant de le transférer sur le réseau. Les types primitifs *int, byte, long, float, boolean,...* sont encodés dans une représentation en *ByteCode* java définie par la spécification standard du langage java. Les chaînes de caractères sont codées suivant l'encodage *UTF-8* qui supporte des caractères codés sur 1, 2 ou 3 octets. Les objets utilisateurs nécessitent la définition de leurs méthodes de sérialisation. Nous avons utilisé l'outil *wireshark* pour examiner les tailles moyennes de deux messages *Call* et *ReturnData* sur une trace de trafic collectée sur notre banc de mesure. Ainsi, pour une requête de type *getAttribute* portant sur un attribut renvoyant une chaîne de caractères, les tailles moyennes des messages *RMI Call* et *RMI ReturnData* sont respectivement 118 octets et 91 octets. La taille de l'entête du protocole *RMI* pour ces deux messages est de 1 octet spécifiant le type de flux (*Call* ou *ReturnData*).

La taille d'un message de type *Call* est définie de la façon suivante : $L_{Call} = 1+4+(2+34)+ n \times (serialized\_object_{length}|primitive\_data\_type_{length})$. La variable n représente le nombre d'arguments de la méthode à invoquer à distance. La valeur 4 est la taille de l'entête de sérialisation définie par le protocole de sérialisation java [66]. La valeur $(2+34)$ représente la taille de la sérialisation du bloc de donnée *ObjectIdentifier,operation et hashcode* que utilisent leurs propres fonctions de sérialisation.

Deux types de paramètre peuvent être passés à une méthode, soit un objet java, soit un paramètre de type de données primitif. La variable $serialized\_object$ représente la forme sérialisée de l'objet passé par valeur à la méthode invoquée par le client. Dans le cas du passage d'une valeur de type primitive en paramètre d'un appel de méthode, la variable $primitive\_data\_type$ représente une copie de ce paramètre.

Hericko et al [19] ont évalué le coût de la sérialisation d'un objet java. Ils montrent que la taille d'un objet sérialisé est exprimée par la formule suivante : $serialized\_object_{length} = 189 + m \times object\_data_{length}$ où m représente le nombre des instances de l'objet à serialiser.

L'invocation de la méthode *getAttribute* prend deux paramètres : l'identifiant d'un MBean et le nom de l'attribut. L'identifiant d'un MBean est de type *ObjectName* et l'attribut est de type *java.lang.String*. Nous avons examiné la taille moyenne de données sérialisées de l'objet *ObjectName*. Nous avons trouvé que cette taille est de 67 octets. Le nom de l'attribut prend comme taille moyenne de donnée $Length_{AttrName} + 8$ octets [33] où $Length_{AttrName}$ représente la taille en nombre de caractères du nom de l'attribut. En effet, nous aurons la taille moyenne d'un *CallData* pour un nom d'attribut de 10 caractères égale à :

$$\begin{aligned} L_{getAttribute} &= 41 + serialized\_ObjectName_{length} \\ &\quad + serialized\_AttrName_{length} \\ &= 41 + 67 + 18 \\ &= 126 \text{ octets} \end{aligned} \tag{5.1}$$

Pour une invocation de la méthode *getAttributes* nous obtenons comme taille moyenne du *Call* RMI associé :

$$\begin{aligned} L_{getAttributes} &= 41 + serialized\_ObjectName_{length} \\ &\quad + n.serialized\_AttrName_{length} \\ &= 108 + n.18 \text{ octets} \end{aligned} \tag{5.2}$$

Où n représente le nombre d'attributs sollicités par la fonction *getAttributes*. La taille d'un *ReturnData* est exprimée avec la formule suivante :

$$\begin{aligned} L_{ReturnData} &= 1 + 4 + (2 + 14) + 1 \\ &\quad + k.n.(serialized\_object_{length}|primitive\_data\_type_{length}) \\ &\quad + (1 - k).(serialized\_exception_{length}) \end{aligned} \tag{5.3}$$

La variable k caractérise le code de retour renvoyé par l'appel *Call* RMI pour indiquer un retour normal dans ce cas $k = 1$ ou une exception java où k prendra la valeur 0. Dans le cas d'un retour normal d'un type primitif, le message *ReturnData* d'un appel de *getAttribute* retourne un objet java de nature *Integer, Long, Double, Float, Boolean* ou une chaîne de caractères. La taille de ce message dans ce cas est égale à :

$$ReturnData_{getAttribute} = 22 + serialized\_object_{length} \tag{5.4}$$

Le tableau 5.1 présente la taille en octets de ces différents types d'objets sérialisés encapsulant[68] les types primitifs, retournés suite à un appel de la fonction *getAttribute*.

Type d'objet retourné	$serialized\_object_{length}$ (octets)
Integer	81
Long	82
Double	84
Float	79
String(length)	length+8
Boolean	47

TAB. 5.1 – Tailles en octets de différents types d'objets sérialisés emballants des types primitifs.

La fonction *getAttributes* retourne un objet de type *AttributeList* qui possède une taille moyenne sérialisée de $278+(n-1).21$ octets où $n \geq 1$ représente le nombre d'attributs retournés. Ainsi pour cette méthode nous obtenons comme taille du message *ReturnData* égale à :

$$ReturnData_{getAttributes} = 300 + (n-1).21 \tag{5.5}$$

Dans le cas du retour d'une exception, après l'appel de la fonction *getAttribute*, cette fonction génère l'une des trois classes d'exceptions suivantes : *AttributeNotFoundException, InstanceNotFoundException, MBeanException, ReflectionException, IOException* pour un Mbean Standard ou dynamique et *RuntimeOperationsException* pour un MBean modèle ou ouvert. Le tableau 5.2 résume la taille de la forme sérialisée de ces exceptions. Nous constatons que la taille sérialisée d'une exception java est considérable en la comparant à un simple objet, mais leur impact sur les performances est nul dans le cas d'un retour normal des fonction *getAttribute* ou *getAttributes*. Ces résultats analytiques de tailles de messages JMX/RMI viennent compléter ceux définis par Pras et al [142] pour les messages du protocole SNMP.

5.4.2 Scénarios de tests

Dans notre contexte, le système sous test est l'agent JMX soumis à un jeu de tests synthétiques. Nous présentons ici deux jeux de tests relatifs aux métriques présentées précédemment. Le premier scénario est le plus simple puisque il s'agit d'un *MBeanServer* contenant un seul MBean. L'objectif de ce scénario est de connaître la relation qui existe entre le nombre d'attributs exposés par un seul MBean et le débit de l'agent. Le deuxième scénario consiste à mesurer l'impact du remplissage d'un *MBeanServer*. L'agent comportera un nombre variable de MBeans munis d'un seul attribut. Par conséquent les mesures tiendront compte du nombre de MBeans enregistrés dans l'agent et le débit de l'agent lors de l'accès à l'unique attribut d'un MBean choisi aléatoirement. Nous avons fait varier le type d'implantations de l'agent. Les deux implantations testées sont SUN-RI [98] et MX4J [82]. Les types de MBeans varient selon les types définis par JMX i.e. le MBean standard, le MBean dynamique, le MBean modèle et le MBean ouvert.

[68] Il s'agit d'objets emballants (*wrapper*) les types primitifs qui ne sont pas d'objets Java.

Nom d'exception	$serialized\_exception_{length}$ (octets)
AttributeNotFoundException	546
MBeanException(Exception)	567
MBeanException(Exception,msg)	569+$length_{msg}$
ReflectionException(Exception)	572
ReflectionException(Exception,msg)	574+$length_{msg}$
InstanceNotFoundException(msg)	547+$length_{msg}$
IOException	428
RuntimeOperationsException(Exception)	641
RuntimeOperationsException(Exception,msg)	643+$length_{msg}$

TAB. 5.2 – Tailles en octets des *exceptions* de retour des appels JMX/RMI.

5.4.3 Environnement logiciel

Nous avons testé les deux implantations de JMX que sont SUN-RI v1.2.8 et MX4J v2.0. Toutes les deux se basent sur la spécification 1.2 de JMX et fournissent un support du *JMX REMOTE* (JSR 160) [99]. Pour tous les tests nous avons utilisé la JVM de Sun sous forme du JDK 1.4.0_01 pour Linux. Nous avons choisi une distribution *Slackware 9.1* comme système d'exploitation qui utilise un noyau Linux en version 2.4.22. Nous avons réalisé une installation minimale afin de limiter le nombre des services actifs.

5.4.4 Plate-forme matérielle

Nous avons utilisé quatre machines pour réaliser nos expérimentations. Chaque injecteur est composé d'un PII 350MHZ avec 128Mo de RAM. L'agent tourne sur un bi-processeur PIII 550MHZ avec 512Mo de RAM. Toutes les machines sont équipées d'une carte réseau Ethernet 100Mbps et inter-connectées via un switch *Cisco 2600 series* où seuls nos tests génèrent du trafic.

5.4.5 Résultats préliminaires

Dans un premier temps, nous avons réalisé une série des tests unitaires en faisant varier l'intensité d'injection pour positionner la capacité de l'agent et déterminer le débit maximal de l'agent en terme de nombre de variables par seconde. Celle-ci se manifeste par l'écroulement du nombre de réponses de l'agent par rapport à l'intensité de demande théorique de l'injecteur en attributs par seconde. Nous serons ainsi en mesure de dégager la plage optimale de l'utilisation de l'agent.

Un MBean dans un MBeansServer

Dans ce scénario de test nous avons considéré un MBean dans un *MBeanserver* avec un nombre d'attributs variant entre 1 et 1000. Nous avons déroulé ce test pour le deux implantations MX4J et SUN-RI. Par exemple avec l'implantation *SUN-RI* et pour un MBean possédant un seul attribut, une première analyse nous a permis de positionner la zone d'instabilité de l'agent entre 600 et 725 attributs par seconde. Ce seuil fixant le régime linéaire est variable selon le nombre d'attributs exposés par le MBean. Les courbes de la figure 5.12 montrent les résultats de ce test. Nous observons que le nombre d'attributs à une influence directe sur les performances. La perte s'avère plus perceptible entre les deux implantations pour un nombre considérable d'attributs. Ainsi, pour MX4J avec 1000 attributs le débit maximal se situe à 800

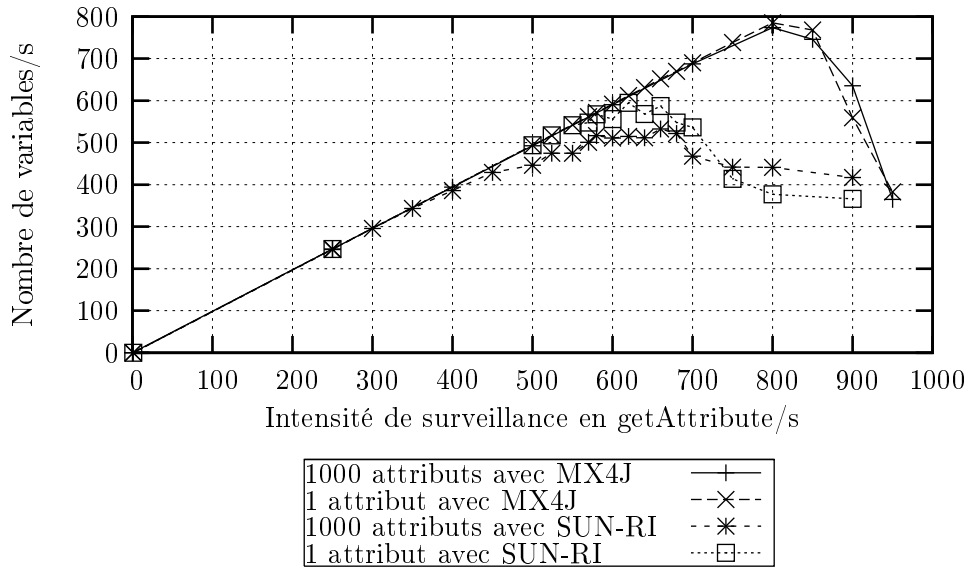


FIG. 5.12 – Comparaison des débits de l'agent en fonction de l'intensité de la demande en *getAttribute*/seconde sur un MBean standard comportant un nombre variable d'attributs pour les deux implantations MX4J et SUN-RI.

attributs/seconde, celui-ci chute à 350 attributs/seconde pour *SUN-RI*. Nous constatons une perte de 50% du débit de l'agent entre les deux implantations. Pour expliquer ce gain de MX4J sur SUN-RI nous avons examiné le code source des deux implantations, et nous nous sommes rendu compte que MX4J utilise un cache pour récupérer les références faibles (*WeakReference*) des instances des identifiants des MBean déjà allouées (définis par la classe *ObjectName*) sans avoir à les instancier de nouveau pour chaque requête. Contrairement, l'implantation de SUN-RI crée un nouvel identifiant du MBean pour chaque requête. Ceci engendre un coût considérable en terme de mémoire et de CPU puisque la création d'un nouveau objet *ObjectName* nécessite le *parsing* d'une chaîne de caractères identifiant le MBean. Ce cache utilisé par MX4J explique la stabilité au niveau de l'utilisation mémoire pour un MBean standard avec 1 attribut et 1000 attributs (voir figure 5.13), ce qui n'est pas le cas pour l'implantation de SUN-RI.

Plusieurs MBeans dans un MBeansServer

Dans le deuxième scénario nous avons mesuré l'impact du remplissage du MBean server sur le débit maximal de l'agent. L'agent comporte donc un nombre variable de MBeans munis d'un seul attribut. Nous avons déroulé ce test avec des MBeans standards en utilisant l'implantation MX4J. Nous constatons que le nombre de MBeans enregistrés sur le *MBeanServer* a peu d'impact sur le débit maximal de l'agent qui se situe aux alentours de 850 attributs/seconde que se soit pour un *MBeanServer* avec 50 ou 1000 MBeans standards (voir figure 5.14). À travers ces tests, nous avons constaté que l'impact sur les performances du nombre d'attributs dans un MBean est plus important que le nombre des MBeans enregistrés dans le *MBeanServer*. Nous pouvons expliquer cet impact par le coût important de l'introspection Java (voir figures 5.15 et 5.13) et de la sérialisation dans le cas d'un MBean standard avec un nombre considérable d'attributs [81]. En effet, dans le cas d'un MBean standard le lien entre le nom de l'attribut sollicité par la requête et celui exposé par le MBean ce fait grâce à une convention où le nom de la méthode dans le MBean

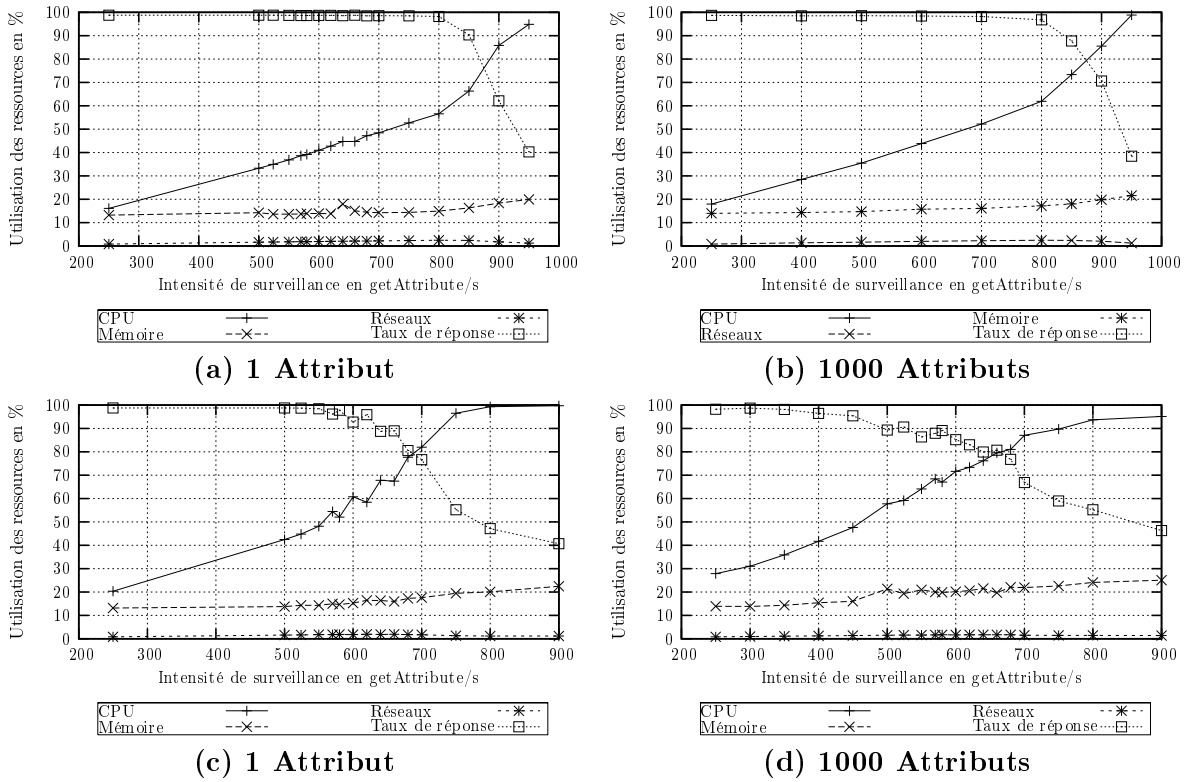


FIG. 5.13 – Utilisation de ressources par l'agent pour un MBean standard avec 1 attribut et 1000 attributs en utilisant l'implantation MX4J ((a),(b)) et SUN-RI ((c),(d)) pour le service getAttribute.

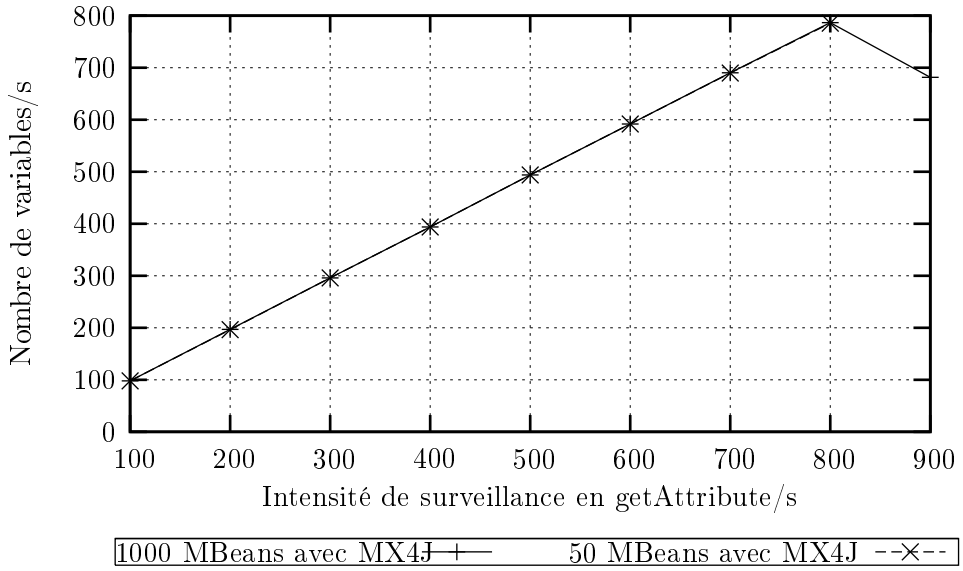


FIG. 5.14 – Débit de l'agent en fonction de l'intensité de demande en *getAttribute*/s avec un nombre variable de MBeans standards et en utilisant l'implantation MX4J.

correspond à celui de l'attribut. L'accès à cette méthode s'effectue par introspection Java. En revanche, cet impact diminu en utilisant des MBeans dynamiques au lieu des MBeans standards. Dans le cas des MBeans dynamiques, le débit n'est pénalisé que par le coût de la sérialisation. Il faut noter que l'accès à un attribut d'un MBean dynamique est codé par une simple imbrication de *if-then-else*. Les résultats de notre étude préliminaire sur les performances de l'agent JMX, nous

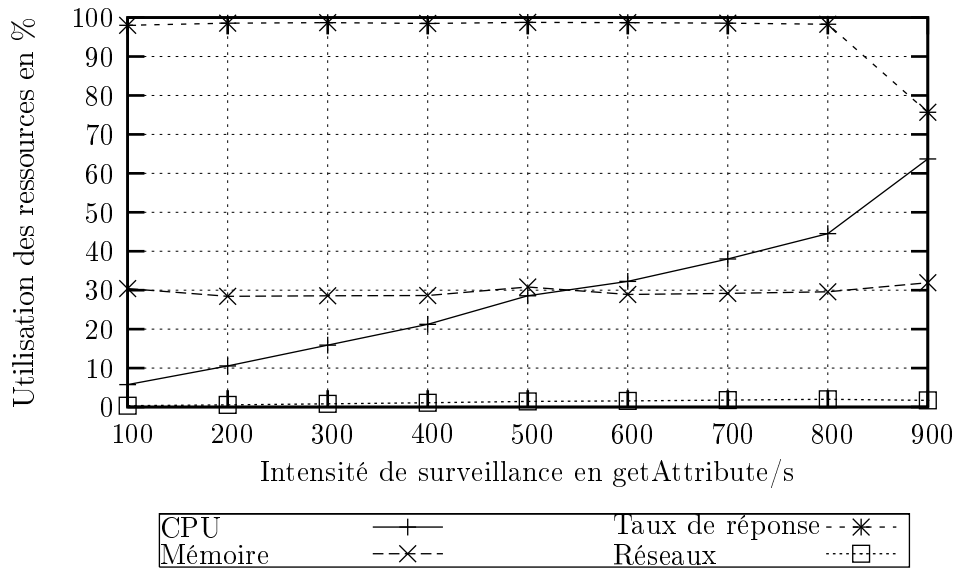


FIG. 5.15 – L'utilisation de ressources par l'agent avec un MBeanServer contenant 1000 MBeans standard et en utilisant l'implantation MX4J pour le service getAttribute.

ont permis d'identifier certaines indications sur le choix du type de MBeans et le fournisseur de l'implantation JMX. Nous constatons que l'implantation MX4J est plus robuste que celle de SUN-RI. Cette efficacité est due aux mécanismes d'optimisation qu'utilise MX4J. Pourtant, l'implantation SUN-RI est une implémentation de référence, que nous devons l'examiner avec d'autres scénarios de tests avec des MBeans dynamiques au lieu des MBeans standards. Nous constatons aussi que le remplissage du *MBeanServer* avec des MBeans consomme plus de mémoire que le remplissage d'un MBean avec des attributs. En revanche, le remplissage du *MBeanServer* consomme moins de CPU (10%) que le second scénario.

5.5 Synthèse

Dans ce chapitre nous avons présenté notre méthodologie de mesure d'un système de gestion en se basant sur la notion de pratiques de gestion. Ce concept nous a permis d'abstraire une approche de gestion en un ensemble de pratiques évaluables par mesure. À partir de la spécification des pratiques, un banc de mesure est développé qui permet de mettre en œuvre ces pratiques pour leurs évaluations. La conception d'un banc de mesure d'une approche de gestion doit garantir la représentativité des tests et leur reproductibilité ainsi que leur comparabilité. Toutes ces garanties sont présentes à travers le banc de mesure que nous avons conçu et instancié pour des approches reposant sur un modèle gestionnaire-agent. Nous avons aussi introduit sur notre banc de mesure, l'utilisation de threads pour la génération d'un trafic de gestion entre deux entités différentes. Chaque thread représente une opération de gestion à laquelle est attachée des vecteurs de variables et des propriétés temporelles. Ces vecteurs sont produits d'une manière syn-

thétique ou éventuellement à partir d'un ensemble de traces réelles d'un trafic de gestion ou d'une manière synthétique. Nous avons aussi eu recours à un ensemble de threads implantés au niveau de l'agent ou sur un processus séparé afin de capturer le comportement d'une application gérée. Ces threads sont responsables de la simulation des valeurs des objets gérés représentés par les variables de gestion. Pour la mesure d'un ensemble de métriques d'une pratique il est nécessaire d'identifier leurs points de mesure et de projeter les paramètres de la pratique sur les différentes entités. Nous avons instancié ce banc de mesure sur une approche de gestion basée sur le protocole JMX. Des résultats préliminaires de la performance d'un agent JMX de supervision issus de cette projection ont été présentés. Nous avons mené un autre ensemble d'expérimentations plus avancées que nous présenterons dans la partie suivante de ce manuscrit.

Troisième partie

Expérimentations

Introduction

Objectifs

Dans les chapitres précédents, nous avons développé **une méthodologie de mesure et un ensemble de métriques pour permettre la compréhension de l'efficacité d'une approche de gestion**. Dans l'objectif d'illustrer cette méthodologie de mesure et ces métriques, nous avons évalué la performance du protocole JMX dans une approche gestionnaire-agent. Durant cette évaluation, nous avons couvert les sources majeures de variations de sa performance indiquée dans la section 2.3 du chapitre 2. Ces sources sont notamment le nombre d'agents, l'intensité de la supervision et les modèles d'intégration d'un agent JMX au sein d'une application gérée. Notre évaluation s'est focalisée sur l'ensemble des métriques primaires définies dans le chapitre 4 notamment le débit, les délais et les coûts processeur, mémoire et communication. Nous avons aussi illustré des métriques dérivées basée sur l'efficacité. Ces métriques portent sur le degré du passage à l'échelle de l'approche JMX et l'incidence de JMX sur les performances aperçues par un utilisateur d'un service web. Afin d'avoir une étude comparable et répétitive de cette incidence de la supervision, nous avons étendue son analyse sur un serveur Web sous différents modèles d'intégration de son agent de supervision.

Plan de cette partie

Dans cette partie, nous présentons les résultats des expérimentations que nous avons effectuées sur la mesure de la performance du protocole JMX reposant sur une approche centralisée du modèle gestionnaire-agent. Les expérimentations sont effectuées sur le banc de mesure que nous avons développé pour l'évaluation de ce protocole. Deux stages ont contribué à ce développement. Julien Delove [8] a travaillé sur les premiers développement du banc de mesure pour un scénario 1 gestionnaire-1 agent avec un nombre variable de MBeans, de nombre d'attributs qu'ils exposent et leurs types (standard, dynamique et modèle). Laurent Andrey et moi même nous avons fait évoluer le banc de mesure pour qu'il supporte plusieurs agents dans des environnements de type cluster et grid. J'ai aussi développé en langage PERL les scripts d'analyse de données issues de mesures. Anca Ghitescu [4] a implanté les trois modèles d'intégration d'un agent au niveau d'un serveur web et elle a effectué les campagnes de mesure et l'analyse de données issues de ces mesures.

Cette partie est divisée en trois chapitres. Le premier chapitre présente l'illustration d'une métrique unique de quantification du passage à l'échelle d'une approche de supervision. Cette métrique repose sur la métrique d'efficacité que nous avons défini dans le chapitre 4. Grâce à cette métrique nous avons identifié les limites du passage à l'échelle d'une approche centralisée JMX en tenant compte des différentes métriques primaires.

Le deuxième chapitre concerne la quantification de l'impact d'une activité de supervision sur la performance d'un serveur web sous une charge utilisateur. Nous avons montré dans ce chapitre, que cet impact n'est pas négligeable sous des intensités de supervision élevées.

Dans le troisième chapitre, nous avons analysé notamment les métriques de délais que subissent les opérations JMX. Cette analyse a porté sur la distribution statistique des métriques de délais rarement traitées par les études de performances dans le domaine de la gestion.

Description des environnements de mesure

Les environnements de mesures que nous avons utilisé dans les expérimentations qui suivent sont les suivants : un environnement constitué de 3 PC de bureau reliés par un réseau Ethernet à 100 MBits/s pour les expérimentations à petite échelle. Une grappe de 100 machines nommée I-Cluster2 localisée à Grenoble[69] pour les expériences d'une supervision à grande échelle dans un environnement haut débit à faible latence réseau. La plate-forme Grid5000[70] a été utilisée pour les expérimentations d'une supervision multi-sites à grande échelle avec des délais réseaux plus importants.

[69] http ://i-cluster2.inrialpes.fr/
[70] http ://www.grid5000.fr

Chapitre 6

Passage à l'échelle de la supervision

Sommaire

6.1 Introduction

Un critère important lors de l'évaluation de la pertinence en terme de performance d'une approche de gestion dans son environnement d'exécution est sa capacité à résister au facteur d'échelle. L'évaluation du passage à l'échelle est vue sous plusieurs angles selon la nature du facteur d'échelle considéré. Un facteur de « grande échelle » est relatif au nombre d'éléments de gestion (agents et gestionnaires), le nombre de composants gérés, la quantité de données de gestion sur laquelle les protocoles doivent opérer.

Le processus d'évaluation du passage à l'échelle d'une approche de gestion passe par la question : ***À quel degré mon approche de gestion passe à l'échelle ?*** Cette question est fondamentale en phase de conception du système de gestion car il s'agit d'une part, d'éviter le sur-dimensionnement coûteux et d'autre part, de lui donner les moyens d'atteindre un certain niveau de satisfaction, ou même plus simplement de garantir sa performance en augmentant ou en réduisant un facteur d'échelle. Ceci faciliterait l'analyse des tests de performances et permettrait des médiations plus judicieuses sur le réglage du fonctionnement d'un système de gestion.

L'objectif de ce chapitre est de présenter une analyse du passage à l'échelle d'une approche de gestion en se basant sur une métrique unique. Cette métrique repose sur le travail de Woodside et al [23] dans le contexte des systèmes distribués. Elle quantifie la dégradation de l'efficacité d'une approche de gestion sous la variation de l'un de ses facteurs d'échelle. Tout d'abord, nous présentons une définition du passage de l'échelle d'une approche de gestion ainsi que l'ensemble des facteurs qui l'affectent. Ensuite, nous introduisons le cadre formel de la métrique qui représente le degré du passage à l'échelle. Nous illustrons alors cette métrique analytiquement sur une pratique de gestion reposant sur le modèle gestionnaire agent en nous basant sur des modèles de performances existants notamment ceux de Thomas Chen [108]. Enfin, une application numérique de cette métrique sera présentée en se basant sur des résultats obtenus de notre banc de

mesure dédié à l'approche JMX.

6.2 Définitions et facteurs impliqués

Dans cette section, nous établissons la notion du passage à l'échelle d'une approche de gestion. Nous identifions, ensuite, les facteurs qui l'affectent et la métrique dérivée nécessaire pour sa mesure.

6.2.1 Définition du passage à l'échelle

Le problème du passage à l'échelle a été bien analysé dans les domaines du calcul parallèle et des systèmes distribués [23, 17]. Ces études font référence à des architectures génériques ou des services spécifiques comme le web, où plusieurs métriques composites ont été développées pour capturer leurs passages à l'échelle. Cependant, dans le domaine de la gestion de réseaux et de services, le passage à l'échelle a été analysé comme un problème de performances où des métriques assez diverses ont été envisagées. Ces métriques se concentrent autour de la mesure des temps de réponse, de la consommation de ressources et du nombre de messages échangés entre les nœuds de gestion [133, 134, 146]. Le facteur d'échelle le plus considéré est le nombre de nœuds.

Par sa définition la plus générale, le passage à l'échelle est : *l'aptitude d'une solution à un problème de fonctionner même si la taille du problème augmente.* Dans notre contexte, cette définition est insuffisante puisque le passage à l'échelle ne signifie pas seulement que le système de gestion fonctionne, mais qu'il doit fonctionner efficacement (d'une façon continue) et avec un certain niveau de qualité sous différentes échelles. A l'heure actuelle, le passage à l'échelle d'un système de gestion est évalué à grande (*scaling up*) ou à petite (*scaling down*) échelles. Un système de gestion passe à grande échelle, s'il maintient sa performance à l'introduction d'un nouveau équipement ou service à gérer. Cependant, il passe à petite échelle, s'il maintient sa performance après une limitation de consommation de certaines ressources (faible mémoire et calculs). L'aspect de petite échelle est de plus en plus important en raison de l'émergence des équipements et de services à faible ressources dans des environnements contraints. Dans ce travail, nous sommes concentré sur l'aspect grande échelle.

Le passage à l'échelle d'un système de gestion doit être vu sous plusieurs dimensions : (i) quand le nombre de nœuds de gestion impliqués dans un domaine ou à travers plusieurs domaines augmente, (ii) quand la taille des objets de gestion et des objets gérés sur laquelle les agents opèrent augmente, (iii) quand la diversité des agents de gestion augmente. Cette dernière dimension est importante et elle inclut les aspects de conception des agents de gestion qui sont de plus en plus divers (agents orientés données, agents orientés commandes, agents orientés objets, agents orientés document). Cette diversité affecte la densité et la granularité des données de gestion, et dans ce cas le passage à l'échelle sera lié aux méthodologies de conception et d'analyse des systèmes de gestion. Par exemple, une décision critique lors de la conception des agents orientés objets est la granularité des objets et leur interdépendance [115]. Cette diversité inclut aussi la technologie utilisée pour le développement du système de gestion (CORBA, Client-Serveur, RMI, agents mobiles, etc) et les protocoles sous-jacents de la gestion (SNMP, JMX, CMIP, etc). Tous ces aspects combinés avec les caractéristiques des plates-formes matérielles déterminent le passage à l'échelle d'un système de gestion.

6.2.2 Facteurs impliqués

Plusieurs facteurs sont impliqués lors de l'évaluation du passage à l'échelle d'un système de gestion. Nous les avons classifiés en deux ensembles, selon leur type d'impact sur la performance du système à évaluer en s'inspirant de la méthodologie décrite dans [23]. Le première ensemble contient les facteurs pénalisant (*disablers*) qui dégradent le passage à l'échelle du système de gestion en augmentant leurs valeurs associées. Le deuxième ensemble contient les facteurs d'amélioration (*enablers*) qui sont des variables d'ajustement ayant pour rôle de maintenir ou même d'améliorer la performance du système de gestion lorsqu'on agit sur elles. Le tableau 6.1 donne une liste non exhaustive de ces deux ensembles pour une approche de gestion. Au cours de l'évaluation d'une approche de gestion, les facteurs d'échelle sont choisis parmi les facteurs pénalisants.

Facteurs pénalisant	Facteurs d'amélioration
– Intensité de la demande de variables de gestion par une fonction de gestion ; – Nombre d'agents de gestion ; – Nombre d'objets de gestion ; – Nombre d'applications de gestion ; – Niveau de sécurité de gestion : sans sécurité, authentification, authentification et confidentialité ; – Modèle de concurrence des opérations de gestion : concurrent ou séquentiel [46].	– Modèle de distribution des informations de gestion : protocole d'extensibilité d'agent (AgentX [58]). – Modèle de distribution de fonctions de gestion : centralisé, décentralisé [108]. – Modèle d'intégration d'agent : démon, composant ou noyau [73]. – Modèle d'interaction : scrutation ou notification.

TAB. 6.1 – Classification de facteurs d'impact sur le passage à l'échelle d'un système de gestion.

6.2.3 Besoin d'une métrique unique de passage à l'échelle

La difficulté principale pour juger la capacité de passage à l'échelle d'une approche de gestion, en analysant séparément des métriques primaires, est due au compromis (*tradeoff*) nécessaire comme présenté dans la section 4.2.5. La majorité des études portant sur le passage à l'échelle des approches de gestion ne considèrent généralement qu'un sous ensemble de métriques de base avant de juger la capacité de passage à l'échelle de l'approche sous test. Cela est insuffisant pour identifier clairement le degré de passage à l'échelle du système. En effet, en variant un facteur d'échelle, l'une de métriques primiares mesurées peut prendre des valeurs optimales. En revanche, une autre métrique qui n'est pas forcément mesurée ou prise en considération peut prendre des valeurs inappropriées pour juger que le système passe à l'échelle.

6.3 Une métrique pour quantifier le passage à l'échelle

Afin de répondre aux besoins présentés précédemment, nous nous sommes posés la question suivante : ***Comment quantifier le passage à l'échelle d'une pratique de gestion ?***

Dans la section 4.3, nous avons présenté une métrique quantifiant l'efficacité d'une approche de gestion. Cette métrique d'efficacité met en relation sous un facteur d'évaluation les métriques de base de la rapidité, du coût et de la qualité de la pratique sous test. En effet, en variant la valeur du facteur d'échelle tout en mesurant l'efficacité de la pratique pour chacune de ses

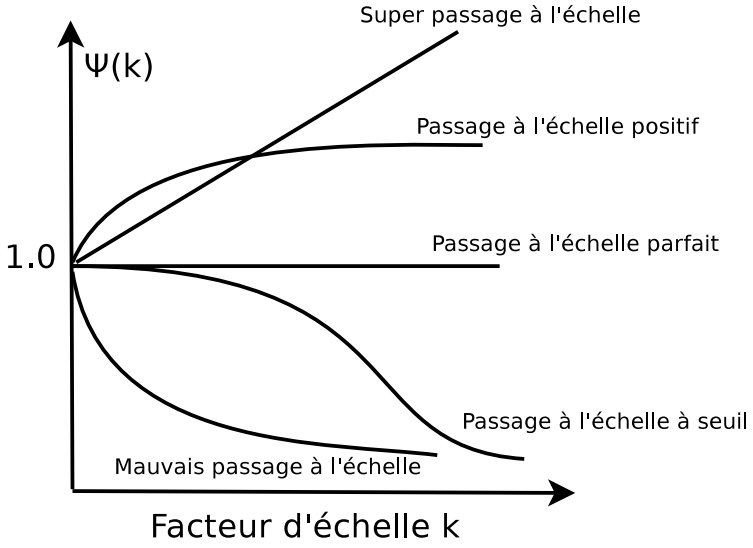


FIG. 6.1 – Illustration de différents comportement du degré de passage à l'échelle $\Psi(k)$ en fonction d'un facteur d'échelle.

valeurs, on aperçoit que l'efficacité diminue ou augmente selon la nature du facteur (pénalisant ou d'amélioration). Cela nous a permis d'élaborer une métrique qui mesure cette variation de l'efficacité sous l'effet d'un facteur d'échelle. Ainsi, le degré du passage à l'échelle, noté $\Psi(k_1, k_2)$, d'une approche de gestion sous un facteur d'échelle spécifique prenant des valeurs k_1 et k_2 est donc :

$$\Psi(k_1, k_2) = \frac{E(k_2)}{E(k_1)} \tag{6.1}$$

où, $E(k_i)$ est l'efficacité du système sous une valeur k_i du facteur d'échelle considéré. La valeur k_1 du facteur d'échelle est, généralement, fixée à une valeur appropriée qui capture l'efficacité du système de gestion sous une pratique de base. Le degré du passage de l'échelle est ainsi défini en fonction de la valeur k du facteur d'échelle considéré. Un système de gestion passe du facteur d'échelle k_1 au k_2 si $\Psi(k_1, k_2) \geq 1$ ou $\Psi(k_1, k_2) \approx 1$. La métrique $\Psi(k_1, k_2)$ permet ainsi de capturer la dégradation et éventuellement l'amélioration de l'efficacité d'un système de gestion, en évaluant une pratique de gestion sous des valeurs différentes k_1 et k_2 d'un facteur d'échelle considéré. Comme l'indique la figure 6.1, le degré du passage à l'échelle possède différentes allures en fonction du rapport des efficacités de l'approche de gestion sous deux valeurs différentes d'un facteur d'échelle spécifique.

6.4 Application à une approche centralisée de supervision

Nous considérons par la suite une étude de cas sur un système de supervision d'un réseau basé sur une approche centralisée. Le système comporte un gestionnaire qui récupère les valeurs d'un attribut depuis un ensemble d'agents déployés sur les N nœuds du réseau en utilisant un mode de scrutation. Cette approche centralisée est utilisée par plusieurs outils de supervision de réseaux tels Cricket [71] ou MRTG [72].

[71] http ://cricket.sourceforge.net
[72] http ://oss.oetiker.ch/mrtg

6.4.1 Approche gestionnaire-agent

Le modèle de performance de l'approche de gestion considéré est celui présenté dans l'article de Thomas Chen [108]. Il s'agit de l'approche gestionnaire-agent sur laquelle repose le protocole SNMP [53] pour la gestion de réseaux et le framework JMX [97, 99] pour la gestion de services et des applications Java, comme nous l'avons présenté dans le chapitre 2. Dans cette approche le gestionnaire central interroge à intervalles de temps réguliers, noté Δ, un groupe de N agents situés sur les différents nœuds du réseaux. Il collecte, sur chacun des agents les valeurs de m variables, puis analyse et corrèle les données récoltées avant de prendre éventuellement des mesures correctives. Dans ce travail, nous considérons un facteur d'échelle k proportionnel à N.

Dans le modèle de performance proposé par Chen [108], les délais que subissent les paquets dans le réseau géré sont considérés comme non uniformes. Ces délais sont modélisés par une variable aléatoire de paramètre N avec une valeur moyenne notée $\bar{l_p}(N)$. D'après lui, le temps moyen nécessaire à l'algorithme de supervision pour détecter un changement d'une variable depuis un agent, noté $T_{CS}(N)$ est égale à :

$$T_{CS}(N) = \frac{\Delta}{2} + \alpha + \bar{l_p}(N) \tag{6.2}$$

où α est la somme des délais fixes de traitements des messages (une paire de requêtes/réponses) sur le gestionnaire et un agent. La valeur $\frac{\Delta}{2}$ représente le temps moyen jusqu'à la réception d'une requête par l'agent après l'occurrence d'un changement de la valeur de la variable surveillée. La variable scrutée possède un modèle de changement de ses valeurs de nature poissonnienne de paramètre λ. La valeur de λ représente la densité du processus poissonnien et elle est égale au nombre moyen de changements par unité de temps. Dans son modèle de temps de détection, Chen n'a pas considéré une métrique de qualité importante qui est la sensibilité de la variable surveillée au niveau des changements détectés par rapport aux changements réellement produits sur la variable. Cette sensibilité, noté S, est définie par la probabilité d'occurrence au plus d'un changement sur un intervalle de scrutation Δ. Si on considère le processus de comptage $\{n(t); t \geq 0\}$ de nombre des changements des valeurs de la variable en fonction de temps, on a comme propriété induite de processus de poisson : $P(n(t) = j) = \frac{e^{-\lambda t}(\lambda t^j)}{j!}$. En effet, la sensibilité de la variable surveillée est définie par la somme des probabilités $P(n(\Delta) = 0)$ et $P(n(\Delta) = 1)$. En remplaçant ces probabilités par ses valeurs, on obtient ainsi :

$$S = e^{-\lambda\Delta}(1 + \lambda\Delta) \tag{6.3}$$

On s'aperçoit que la sensibilité dépend notamment des valeurs respectives de la fréquence de changement de la variable et de l'intervalle de sa surveillance. Comme nous l'avons indiqué dans la section 4.2.4, elle mesure la variation des valeurs de l'objet réelle par rapport à celle de la variable observée sur le gestionnaire. Par l'équation 6.3, nous avons défini une instance analytique de cette sensibilité. En appliquant le théorème de Shanon [89] qui stipule que la fréquence d'échantillonnage qui est égale à $\frac{1}{\Delta}$ doit être double au supérieur à la fréquence du signal réel qui est égale à λ. Ainsi, dans ce cas on obtient une valeur de la sensibilité proche de 1.

6.4.2 Étude analytique

La métrique de rapidité considérée est le nombre moyen de changements détectés, noté $R(N)$, par le gestionnaire d'une variable depuis un agent choisi aléatoirement. La fonction $R(n)$ est ainsi égale à $\frac{1}{T_{CS}(N)}$. La fonction de qualité $Q(N)$ est égale à la sensibilité définie par l'équation 6.3.

La fonction coût, noté $C(N)$, est définie comme la somme de la consommation processeur sur le gestionnaire et les N agents et la quantité de trafic générée entre eux. Nous obtenons, ainsi [108] :

$$C(N) = \frac{N}{\Delta} \times c \tag{6.4}$$

où c est le coût supposé fixe d'une paire requête-réponse par unité de temps entre le gestionnaire et un agent. En effet, le coût c est une moyenne des fractions d'utilisation des différents types de ressources. D'après l'équation d'efficacité présentée dans la section 4.3 du chapitre 4, nous obtenons comme efficacité de cette approche de gestion la suivante :

$$E_{CS}(N) = \frac{\Delta \times e^{-\lambda\Delta}(1+\lambda\Delta)}{T_{CS}(N) \times N \times c} \tag{6.5}$$

Nous constatons que si la sensibilité est proche de 1, l'efficacité se réduit à la formule suivante :

$$E_{CS}(N) = \frac{\Delta}{T_{CS}(N) \times N \times c} \tag{6.6}$$

Dans ce cas, l'efficacité dépend seulement de temps moyen de détection et de l'intervalle de surveillance. Si on considère que le temps moyen de détection est borné par Δ qui signifie que la précision temporelle comme nous l'avons définie dans 4.2.4 est aussi proche 1, alors l'efficacité aura une borne inférieure approximative de $\frac{1}{N^* \times c}$. La valeur N^* représente le nombre optimal d'agents de gestion qui satisfait cette échéance ($T_{CS}(N^*) \leq \Delta$). En se basant sur l'équation 6.1, le degré du passage à l'échelle de cette approche de gestion pour un réseau géré de taille N_1 à un réseau de taille N_2 est égal à :

$$\Psi(N_1, N_2) = \frac{N_1}{N_2} \times \frac{T_{CS}(N_1)}{T_{CS}(N_2)} = \frac{1}{k} \times \frac{t_0 + \bar{l_p}(N_1)}{t_0 + \bar{l_p}(N_2)} \tag{6.7}$$

avec $t_0 = \frac{\Delta}{2} + \alpha$ représente un délai fixe subit par chaque détection du changement d'une variable et k est égale à $\frac{N_2}{N_1}$, si on suppose que $N_2 = k \times N_1$. Nous constatons, ainsi, que dans le modèle de performance présenté dans [108], le passage à l'échelle de l'approche gestionnaire-agent est limité par les délais que subissent les messages échangés entre le gestionnaire et les N agents dans le réseau géré et le facteur de passage à l'échelle k. Cette limitation est déjà optimiste, puisque dans son modèle de performance le temps de traitement α d'un message de gestion sur le nœuds de gestion, est considéré comme constant ce qui n'est pas vrai puisqu'il dépend de la charge exercée sur le processeur des nœuds de gestion impliqués dans la tâche. Nous constatons aussi qu'un intervalle de scrutation suffisamment grand par rapport à $\bar{l_p}(N)$ permettra d'absorber l'augmentation de délais que subissent les messages sur un réseau géré de grande taille. Cette dernière optimisation a été proposée dans le travail [79], où les auteurs adaptent l'intervalle de scrutation selon la variation des délais mesurés sur le gestionnaire de requêtes de supervision. Dans le travail [46], les auteurs affirment qu'avec un intervalle de scrutation de l'ordre de 5 minutes leur outil arrive à gérer un réseau de grande taille avec 5000 variables à surveiller. Nous constatons aussi qu'avec des délais importants de transit dans le réseau géré ($\bar{l_p}(N_1)$ et $\bar{l_p}(N_2)$) sont importants) le degré de passage à l'échelle dépendra que de la valeur de facteur de passage à l'échelle k. Ainsi, à première vue il semble que si le facteur d'échelle n'augmente pas considérablement entre deux tailles différentes d'un réseau géré, l'éloignement des agents de la station centrale est bénéfique pour maintenir le degré du passage à l'échelle du système de supervision stable. Cela met en évidence le besoin de caractériser ses délais sous un facteur d'échelle afin de mieux comprendre le comportement de la métriques de passage à l'échelle d'une approche de gestion. Cette caractérisation de délais de supervision est effectuée dans le chapitre 8.

6.4.3 Résultats numériques

En se basant sur notre banc dédié à la mesure de performances du *framework* de gestion JMX, nous avons validé par mesure l'approche analytique présentée dans le paragraphe précédent. Au cours de nos mesures, nous avons varié le nombre d'agents entre 70 et 700 par palier de 70. Ceci nous donne un facteur d'échelle k dans un intervalle de [1..10]. Nous avons considéré une période de scrutation Δ de 1 seconde. La plate-forme matérielle de mesure est une grappe de 100 machines (900Mhz bi-processeur Itanium2) [73] inter-connectées via un commutateur Gigabit Ethernet. La

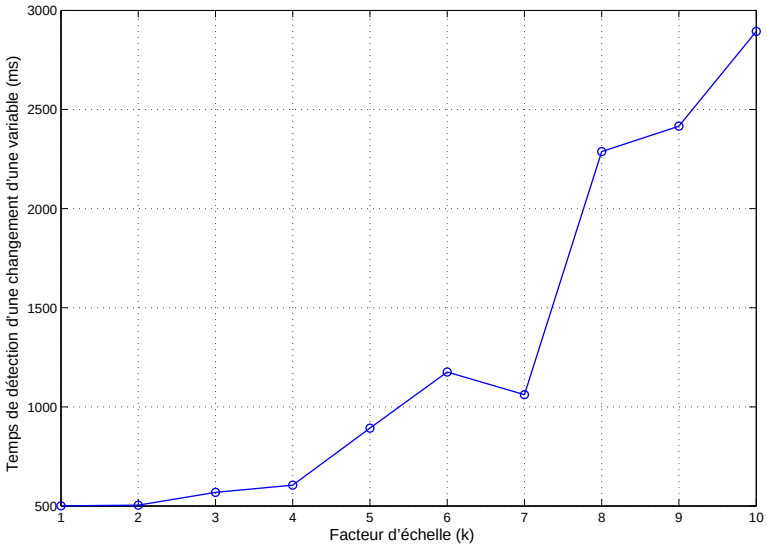


FIG. 6.2 – Temps de détection de changement d'une variable depuis un agent en fonction du facteur d'échelle représentant le nombre d'agents.

figure 6.2 montre l'évolution du temps moyen de détection du changement d'une variable depuis un agent mesuré sur le gestionnaire en fonction du facteur d'échelle. Nous observons que ce temps suit une fonction quadratique avec un coefficient de détermination [21, Page 227] de $R^2 = 0,95$. Ainsi, d'après l'équation 6.7 le degré du passage à l'échelle se décline approximativement en $\frac{1}{k^2}$ comme le montre la figure 6.3.

Nous supposons que la limite du degré du passage à l'échelle de l'approche est définie pour un temps de détection moyen $T_{CS}(k) \leq \Delta$. On obtient, ainsi, comme valeur limite du degré de passage à l'échelle de cette approche : $\Psi_{lim}(k) \geq 0.1$ qui correspond à une valeur de $k = 5$ et un nombre d'agents $N = 350$, pour notre cas d'étude. Au delà de cette valeur, le gestionnaire central sature et effectivement il perd le contrôle du réseau géré. À partir de cette valeur de facteur d'échelle ($k = 5$), le maintient ou l'amélioration de l'efficacité du système de gestion nécessite l'introduction d'un facteur d'amélioration (voir tableau 6.1). En effet, l'intérêt de cette métrique de passage à l'échelle dans le cadre de la conception de plates-formes de gestion, est de déterminer une stratégie d'amélioration de leur passage à l'échelle. Cette améliortaion repose sur l'introduction pour chacune des valeurs limites de cette métrique un facteur parmi les facteurs

[73] http ://i-cluster2.inrialpes.fr

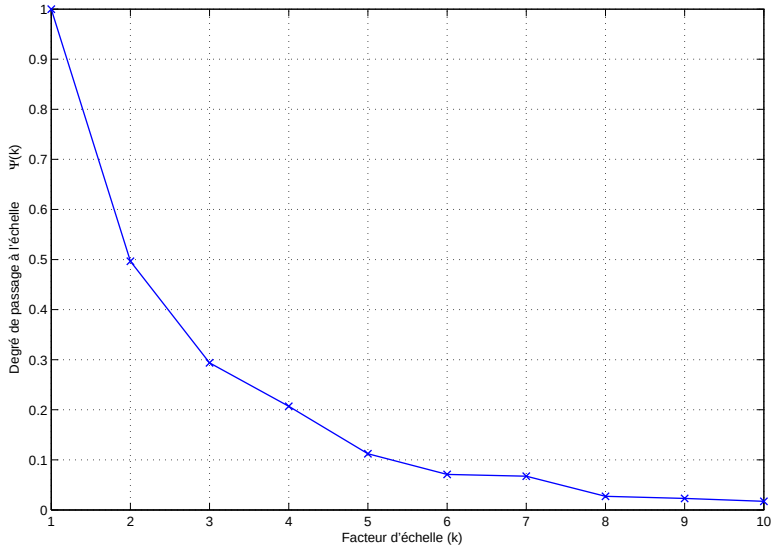


FIG. 6.3 – Degré de passage à l'échelle d'une approche JMX de supervision reposant sur un modèle 1 gestionnaire et N agents.

d'amélioration décrits précédemment.

6.5 Synthèse

Dans ce chapitre nous avons étudié le problème du passage à l'échelle des systèmes de gestion. Les études précédentes ont analysé cet aspect sous la forme d'un problème de performance en analysant séparément un ensemble de métriques (délais, trafic, charge processeur). Pour avoir une idée plus précise sur le passage à l'échelle d'un système de gestion, nous avons présenté dans la section 4.3 une métrique unique de l'efficacité d'une approche de gestion qui met en relation trois familles de métriques primaires. Nous l'avons illustrée ici sur l'étude d'une approche de supervision centralisée impliquant un nombre d'agents considérable. Nous avons montré que cette approche centralisée possède un mauvais passage à l'échelle, où la métrique que nous avons définie décroît selon un facteur de $\frac{1}{k}$. Cette décroissance est due à une augmentation de coût proportionnelle au facteur d'échelle k selon le modèle de performance présenté dans [108]. La difficulté rencontrée lors de l'application de notre métrique de passage à l'échelle est l'identification d'une métrique primaire de qualité adéquate pour capturer les différents aspect de la qualité d'une approche de gestion. Dans ce travail nous avons utilisé comme métrique de qualité la sensibilité, qui est insuffisante puisque elle ne dépend pas de facteur d'échelle. Nous avons aussi mis l'accent sur l'importance des délais, que subissent les messages de gestion dans un réseau, sur le degré de passage à l'échelle d'une approche de gestion centralisée. Cela justifiera nos travaux du chapitre 8 où nous allons analyser pertinemment ces délais avec des modèles statistiques. Enfin, notre analyse du passage à l'échelle d'une approche centralisée est basée sur la travail présenté dans [108], où les auteurs ont associé des modèles de performances pour plusieurs types d'approches de gestion. Nous confirmons ainsi qu'il est indispensable de définir des modèles de performances

analytiques relatifs aux approches de gestion pour qu'on puisse juger leur degré du passage à l'échelle. Cette métrique de mesure de l'efficacité d'une approche de gestion nous sera utile pour quantifier l'impact de la supervision sur une application gérée, notamment sur sa performance.

Chapitre 7

Incidence de la supervision sur la performance d'un système géré

Sommaire

7.1 Introduction

L'efficacité d'une approche de gestion ne signifie pas seulement la mesure de sa propre performance, mais aussi la mesure de son incidence sur la performance du système géré. Cet impact incidence est dû au caractère intrusif des activités de gestion sur les ressources du système géré. Généralement, les activités de gestion partagent des ressources avec le système géré sous la forme d'un surcoût à payer afin de maintenir son état. Ce surcoût se traduit par une dégradation de la performance du système géré, notamment, celle perçue par les utilisateurs sous la forme du débit (*throughput*) et de la latence de leurs requêtes. Si ce surcoût peut être minimal pour des systèmes avec des ressources abondantes, il est dramatique dans le cas des systèmes à faibles ressources. Afin de garantir la qualité des services utilisateur du système géré, il est indispensable de quantifier cette incidence sous la forme d'une métrique qui met en relation l'efficacité du système géré et celle de l'approche de gestion.

Dans ce chapitre nous présentons tout d'abord, une métrique qui capture l'incidence des activités JMX de gestion sur un serveur applicatif. Notre choix c'est porté sur le serveur JBoss. Ce serveur est un bon candidat puisqu'il est distribué en *Open Source*. Il repose sur un modèle d'intégration de type pilote et il présente une forte intégration du plan de gestion au sein du son plan fonctionnel. Cette première expérimentation, nous a permis d'identifier un facteur important qui affecte la performance d'un serveur géré. Ce facteur est le modèle d'intégration de l'agent JMX au sein d'une application gérée. Ainsi, nous avons étendu nos expérimentations en analysant l'effet de chacun des modèles d'intégration présentés dans la section 2.2.3 sur un serveur Web. Ces expérimentations montrent que chacun de ces modèles possède son propre surcoût sur le

serveur géré ainsi que sa propre performance de point de vue de la supervision.

7.2 Une métrique d'incidence

La métrique que nous proposons mesure la variation de performances d'un système géré sous une pratique de gestion spécifique. Cette variation est due à un facteur d'incidence relatif à la pratique de gestion employée pour gérer le système. Ces facteurs sont ceux présentés dans la section 5.2. Cette métrique nommée MIM (*Management Impact Metric*) est une fonction qui traduit un facteur d'incidence k en une valeur dans l'intervalle [0..1]. Cette fonction indique la présence d'une dégradation de performance du système géré ainsi que le degré de cette dégradation. La fonction $MIM(k)$ nous permet de distinguer entre une dégradation inacceptable où sa valeur est proche de 1 et une dégradation acceptable où sa valeur est proche de 0.

Cette métrique repose sur la définition de l'efficacité que nous avons proposé pour une approche de gestion dans la section 4.3. Cette définition est aussi valable pour le système géré où nous définissons son efficacité, appelée aussi productivité dans [23] :

$$F(k) = \frac{\lambda(k) \times f(k)}{C(k)} \tag{7.1}$$

où $\lambda(k)$ est le débit (*throughput*) du système géré en terme de réponses par secondes, $f(k)$ est une fonction qui qualifie la qualité de ces réponses, et $C(k)$ est une fonction de coût du système géré. Ce coût n'est pas forcément lié à la consommation de ressources matérielles mais il peut être un coût financier d'exploitation (consommation d'énergie par exemple) ou autre. La fonction $f(k)$ est définie sous la forme des différentes métriques relatives à la qualité de service (délais, gigue, probabilité de dépassement d'un délai seuil, etc). Dans ce travail, nous considérons la fonction suivante pour définir la qualité des réponses du système géré [23] :

$$f(k) = \frac{1}{1 + \frac{T(k)}{\bar{T}}} \tag{7.2}$$

où $T(k)$ est le temps moyen de réponse mesuré sous un facteur k et $\bar{T}$ est le temps de tolérance maximal établit par la SLA[74] du service géré. Par exemple, pour un service web ce temps de tolérance est de l'ordre de 2 à 5 secondes, pour un service VOIP ce temps est de l'ordre de 150 ms [64]. Dans le cas où ce temps de tolérance n'est pas spécifié, la fonction f(k) est définie par : $f(k) = \frac{1}{T(k)}$. Dans ce cas l'efficacité du système géré devient :

$$F(k) = \frac{\lambda(k)}{T(k) \times C(k)} \tag{7.3}$$

De la même manière nous présentons l'efficacité $G(k)$ de la pratique de gestion associée au système géré. Cette efficacité est définie dans la section 4.3. Elle est exprimée en fonction de la rapidité, la qualité et le coût de la pratique de gestion sous la forme suivante :

$$G(k) = \frac{R(k) \times Q(k)}{C(k)} \tag{7.4}$$

Jusqu'à maintenant nous avons défini l'efficacité de la pratique de gestion et celle du système géré l'une indépendemment de l'autre. En revanche l'efficacité du système géré dépend de celle

[74] Service Level Agreement

de sa pratique de gestion puisque c'est elle qui la maintient ou la dégrade. Nous définissons ainsi le rendement d'un système géré comme étant :

$$E(k) = \frac{F(k)}{F(k) + G(k)} \tag{7.5}$$

Celle-ci mesure l'efficacité fonctionnelle du système géré par rapport à l'efficacité totale pour l'atteindre ($F(k) + G(k)$). Sous différentes valeurs k et sous k_0, une valeur de référence d'un facteur d'impact quelconque, la métrique d'incidence reliant les rendements du système géré sous ces deux valeurs est définie de la manière suivante :

$$MIM(k_0, k) = 1 - \frac{E(k)}{E(k_0)} \in [0, 1] \tag{7.6}$$

L'intention derrière la définition de cette métrique est de capturer le comportement du rendement du système géré par rapport à un rendement de base $E(k_0)$. En effet, on est capable d'identifier les cas où ce rendement est inacceptable sous une valeur k d'un facteur d'incidence par rapport à un rendement de référence sous une valeur k_0. Une manière de définir ce rendement de référence est de considérer $k_0 = 0$ où on suspend toute activité de gestion et on mesure les performances du système géré avec un ensemble de métriques appropriées (débit, délais et coût). Dans cas en utilisant l'équation 7.5, nous avons $G(k_0 = 0) = 0$, ainsi que $E(k_0 = 0) = 1$. Dans ce cas particulier, la métrique d'incidence $MIM(k)$ est définie de la manière suivante :

$$MIM(k) = 1 - E(k) = 1 - \frac{F(k)}{F(k) + G(k)}; \text{où } k \geq 1 \tag{7.7}$$

D'un point de vue gestion, un système est géré efficacement si son efficacité fonctionnelle $F(k)$ augmente au moins au même taux que celle de la pratique de gestion. Soit α le rapport entre les efficacités respectives $G(k)$ et $F(k)$ de la gestion et du système géré. La valeur α représente la fraction de la productivité du système géré attribuée aux activités de gestion. On obtient ainsi, $G(k) = \alpha F(k)$ et $MIM(k) = \frac{\alpha}{1+\alpha}$.

La figure 7.1 représente le comportement de la métrique $MIM(k)$ dans le cas d'une relation de proportionnalité entre $G(k)$ et $F(k)$. On s'aperçoit que lorsque $\alpha = 1$, le travail accompli par la gestion possède le même taux que celui fourni par le système géré à ces utilisateurs. Cette valeur de $\alpha = 1$ représente la valeur limite d'une gestion efficace d'un système géré, au-delà de cette valeur les activités de gestion deviennent pénalisantes et son incidence devient importante.

7.2.1 Méthodologie de calcul de la métrique d'incidence

Dans cette section, nous présentons un algorithme de calcul de la métrique d'incidence que nous venons de présenter. En effet, le calcul de cette métrique dépend essentiellement de la technique d'évaluation de performance utilisée pour mesurer les différents paramètres qu'elle inclut. Il s'avère que les techniques de simulation et analytiques sont plus adéquates grâce à leur flexibilité au niveau de la variation des valeurs du facteur impact. En revanche, cela nécessite l'identification d'un ensemble de modèles analytiques de performances pour le système géré et sa pratique de gestion associée. Ces modèles ne sont pas forcément disponibles pour des systèmes distribués et complexes.

Nous identifions les étapes suivantes pour le calcul de la métrique d'incidence. Tout d'abord, il faut définir la performance de base du système géré, c'est à dire $E(k_0)$. Cette performance est définie sous un ensemble de paramètres du système géré fixes (nombre d'utilisateurs, nombre de

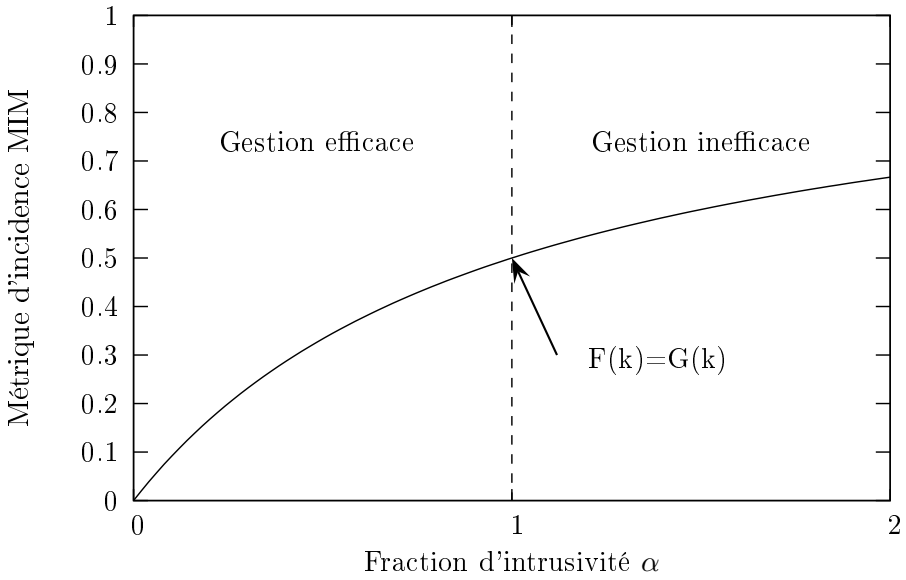


FIG. 7.1 – Illustration du comportement de la métrique d'incidence dans le cas d'une relation de proportionnalité entre l'efficacité fonctionnelle $F(k)$ et celle $G(k)$ de la gestion.

requêtes utilisateurs par seconde, nombre de serveurs, etc). Ensuite, nous définissons la pratique de gestion et le facteur d'impact à varier. Les fonctions $F(k)$ et $G(k)$ relatives respectivement au système géré et la gestion sont calculées de la manière suivante :

1. Fixer les paramètres d'une pratique de gestion comme indiqué dans la section 5.2.
2. Choisir un facteur d'impact et fixer les valeurs des autres paramètres.
3. Définir une charge utilisateur à imposer sur le système géré.
4. Mesure de l'efficacité fonctionnelle du système géré $F(k)$: il s'agit de mesurer le débit du système en termes du nombre de réponses par seconde et le temps moyen de réponse. Le coût à considérer est celui de la consommation des ressources matérielles (CPU, mémoire, réseau).
5. Mesure de l'efficacité de la pratique de gestion : il s'agit d'utiliser les métriques définies dans la section 4.2 et leurs méthodologies de gestion définies dans le chapitre 5.
6. Calculer la métrique d'incidence en se basant sur l'équation 7.1 dans le cas de la spécification de la fonction $f(k)$ sinon l'équation 7.3 dans le cas où on considère seulement le temps de réponse $T(k)$.
7. Varier la valeur du facteur d'impact et aller à l'étape 3.

7.3 Incidence d'une activité de surveillance sur la performance d'un serveur applicatif

Dans cette section nous allons illustrer la métrique d'incidence dans le cadre d'une activité de surveillance par JMX du serveur applicatif JBoss [63]. Les paramètres de la pratique de gestion qui nous associons à ce serveur sont présentés dans le tableau 7.1. Le facteur d'impact que nous considérons est l'intensité de demande de la surveillance en terme de nombre de *getAttribute*/seconde. Cette intensité varie de 1 à 350 *getAttribute*/seconde. Cette plage d'utilisation

Paramètres de la pratique de gestion
Approche gestionnaire, agent
Agent pilote
Technologie JMX
Fonction de surveillance
Mode scrutation
Opération *getAttribute*, multiplicité égale 1
Opérations concurrentes
Instrumentation interne
Attribut : *FreeMemory* exposé par le MBean *ServerInfo* qui retourne la mémoire libre du JVM qui tourne sur le serveur en faisant appel à la fonction Java *Runtime.freeMemory*

TAB. 7.1 – Paramètres de la pratique de gestion associée à la gestion du serveur JBoss.

de l'agent JMX est celle que nous l'avons identifiée dans les expérimentations décrites dans la section 5.4.5. Cette plage nous garantie une utilisation nominale de l'agent sans qu'il s'écroule.

7.3.1 Description des environnements logiciels et physiques

Nous avons utilisé le serveur JBoss version 3.2.1 reposant sur un environnement d'éxecution Java Sun JDK v1.4.2. Pour émuler une charge fonctionnelle à appliquer sur le serveur JBoss nous avons utilisé la suite de test RUBIs [30]. Cette suite de benchmarking des serveurs J2EE permet à la fois la génération de trafic web et la mesure de leurs performances sous une telle charge. Elle modélise une application J2EE de type site d'enchère à la manière d'*eBay*[75]. Dans nos tests, nous avons utilisé une variante sans état des EJB qui implantent les différents composants de cette application J2EE. Cette variante se présente comme la plus performante selon l'étude de Cecchet et al [12]. Dans nos tests, nous avons émulé 100 clients web avec un temps de réflexion[76] de 7 secondes. Ces clients exécutent seulement des transactions *http* de type consultation (*browsing*). Pour la charge de surveillance appliquée sur le serveur, nous avons utilisé notre banc de mesure dédié pour l'approche JMX. Chacune des mesures possède une durée de 15 minutes et une période d'amorçage de 2 minutes. Ces différentes suites logiciels sont déployées sur des machines bi-processeur PIII cadencés à 500MHZ avec 512 Mo de RAM et un système Linux d'exploitation Slackware 9.1. Les différentes machines sont connectées par un réseau Ethernet 100Mbits sans aucun trafic autre que celui généré par nos tests. La méthodologie de mesure des différentes métriques est celle présentée dans le chapitre 5.

7.3.2 Résultats expérimentaux

En première étape, nous avons mesuré la performance de référence du serveur JBoss sans qu'il soit soumis à une quelconque activité de gestion. Seule la charge utilisateur générée par la suite *RUBis* est exercée sur le serveur. Le tableau 7.2 représente les résultats de ce test en terme du nombre moyen de réponses/s, le temps moyen de réponse en seconde, l'utilisation moyenne de CPU, de la mémoire et du réseau en %. La valeur de l'efficacité correspondante est celle définie par l'équation 7.3. Dans une deuxième étape, nous avons varié l'intensité de gestion et nous avons mesuré l'efficacité $F(k)$ du serveur, ainsi que celle de la partie de surveillance $G(k)$. La figure 7.3 (a) représente la décroissance de la valeur de l'efficacité du serveur en fonction de

[75] http ://www.ebay.com
[76] En anglais : *thinking time*.

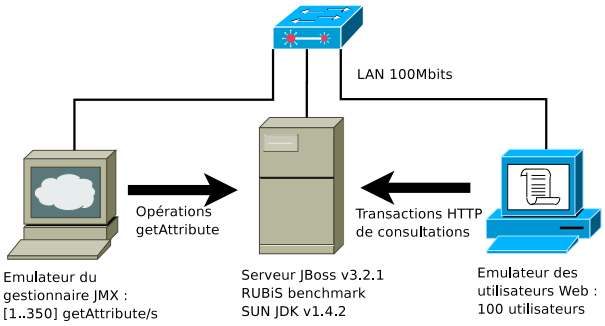


FIG. 7.2 – Architecture de testbed de mesure de l'incidence de la supervision sur le serveur applicatif JBoss.

Débit (réponses/s)	Temps de réponse (seconde)	Utilisation de ressources (%)			Efficacité $F(k_0)$
		CPU	mémoire	réseau	
15	0,466	88	24	2,6	3688,84

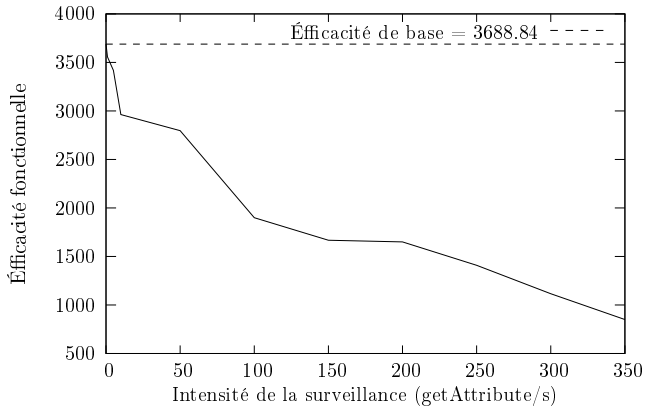
TAB. 7.2 – Résultats de l'efficacité de référence $F(k_0)$ du serveur JBoss sans aucune activité de surveillance.

l'intensité de demande de la surveillance. La figure 7.3 (b) montre que cette dégradation varie entre 24% pour une intensité de demande de surveillance de l'ordre de 50 *getAttribute*/s et de 74% pour une valeur de 350 *getAttribute*/s. Cette dégradation est notamment due à l'augmentation du temps de réponse $T(k)$ du serveur pour servir les requêtes clientes. Cette augmentation du temps de réponse a engendré une diminution de son débit $\lambda(k)$ comme le montre la figure 7.4. La figure 7.5 montre la croissance de l'efficacité de la gestion lorsque on augmente l'intensité de la demande de la surveillance. Comme le montre la figure 7.6, l'augmentation de l'efficacité de gestion et la diminution de l'efficacité fonctionnelle de serveur ont augmenté la métrique d'incidence MIM. Elle devient très proche de 1 dès que la valeur de l'intensité de supervision dépasse le 50 *getAttribute*/s. Il faut noter que dans le cas de JBoss, le serveur des MBeans de l'agent JMX est le noyau du serveur lui même. Il représente le conteneur du serveur applicatif mais pas au niveau EJB. Ainsi, dans ce modèle l'incidence des activités de gestion est nettement visible puisque les opérations de gestion sont en concurrence directe avec les requêtes utilisateur.

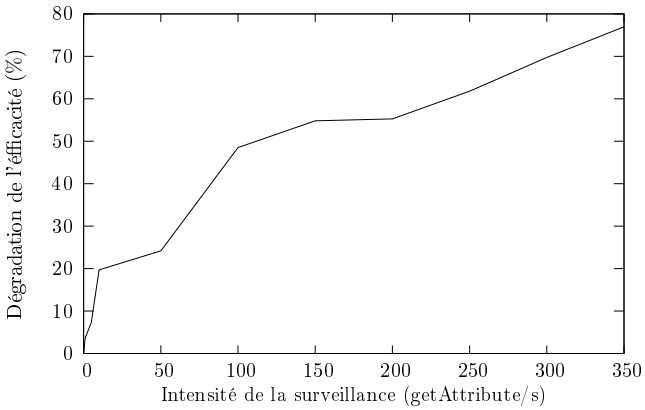
7.4 Incidence du modèle d'intégration d'un agent JMX

Dans la section précédente, nous avons présenté l'incidence d'une activité de surveillance sur un serveur applicatif reposant sur un modèle d'intégration de type pilote de son agent JMX. Cela nous a montré que cette incidence est inévitable si le serveur est soumis à une activité de supervision. En revanche, les questions qui s'imposent maintenant sont les suivantes : ***Est-ce que les trois modèles d'intégration ont le même impact d'un point de vue performance sur un système géré ? Est-ce qu'ils impactent aussi la performance de la partie gestion ? Quel est le modèle le plus approprié et dans quel contexte ?***

L'objectif de cette section est de répondre à ces questions en se basant sur un travail de mesure

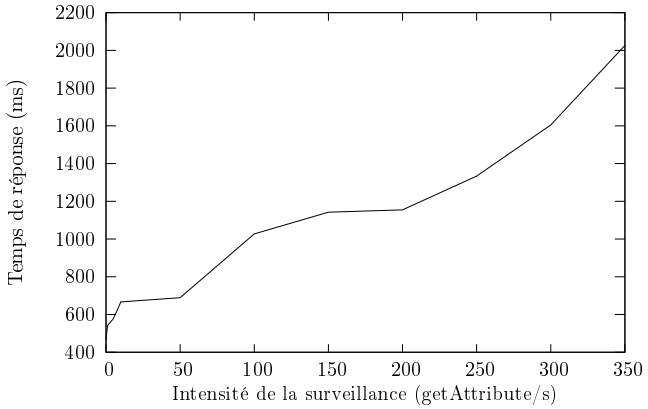


(a) Efficacité de serveur JBoss

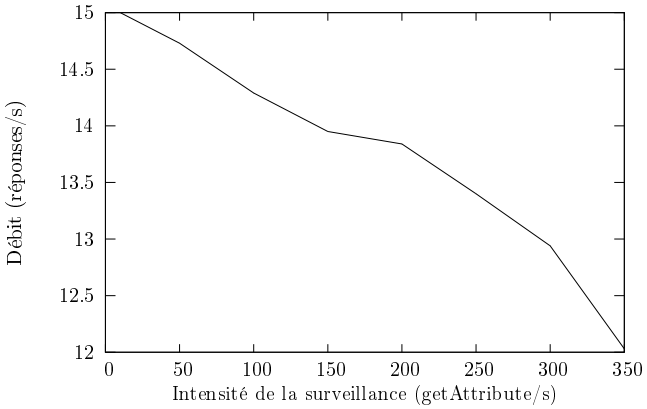


(b) Dégradation de l'efficacité du serveur JBoss

FIG. 7.3 – Décroissance de l'efficacité fonctionnelle de serveur JBoss en fonction de l'intensité de demande de sa surveillance en termes de *getAttribute*/s.



(a) Temps de réponse



(b) Débit

FIG. 7.4 – Effet de l'intensité de la supervision sur le temps de réponse et le débit fonctionnels de serveur JBoss sous à une charge stationnaire de 100 utilisateurs.

de la performance de trois modèles d'intégration d'un agent JMX présentés dans la section 2.2.3.

7.4.1 Méthodologie de mesure

Le banc de mesure que nous avons développé pour la gestion JMX, repose sur des agents synthétiques qui ne gèrent aucune application réelle. Ce type d'agent ne nous permet pas de comparer les trois modèles, puisque nous avons besoin d'une application réelle dans laquelle nous intégrons l'agent selon les trois modèles. Cette application réelle à superviser doit être de préférence développée en Java et facilement pourvue d'un générateur de charge fonctionnelle. Notre choix s'est porté sur un serveur web libre source, nommé TJWS (Tiny Java Web Server) [77] puisque il s'agit d'une application très répandue et que plusieurs injecteurs web existent pour ce type d'application.

[77] http ://tjws.sourceforge.net

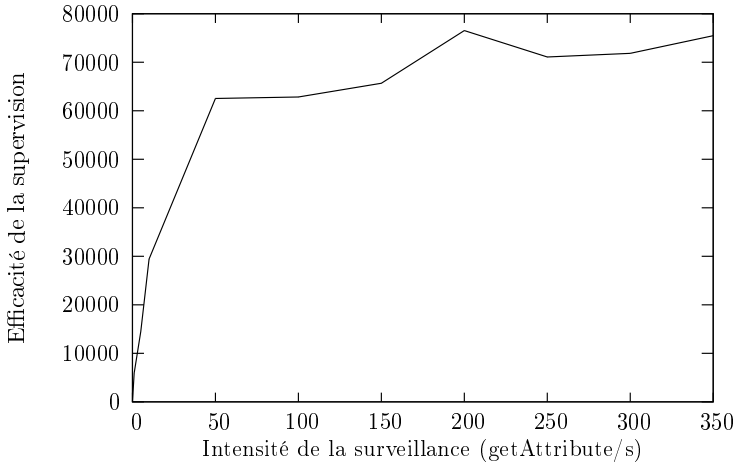
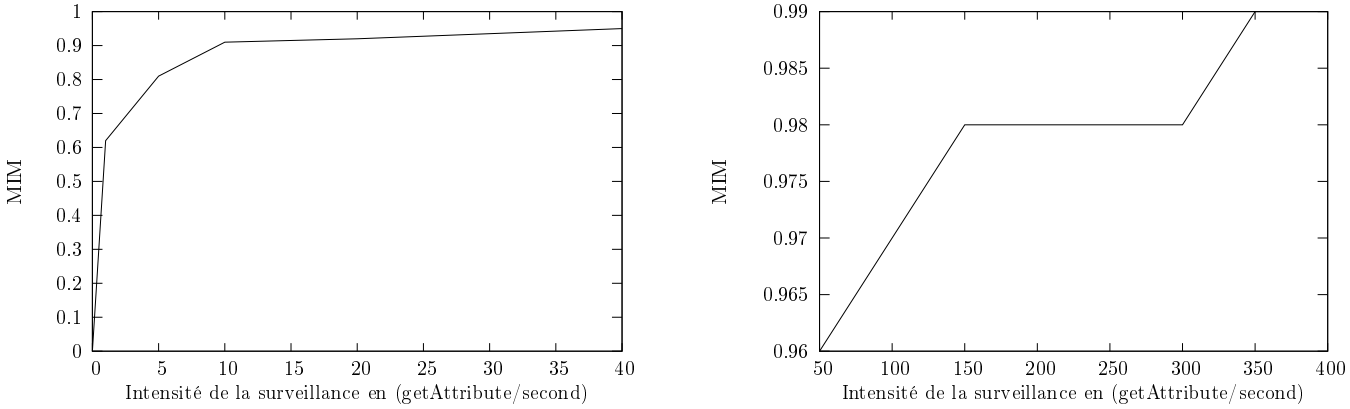


FIG. 7.5 – Augmentation de l'efficacité de gestion en fonction de son intensité en terme de *getAttribute*/s.



(a) Faible intensité de surveillance **(b) Haute intensité de surveillance**

FIG. 7.6 – Croissance de la métrique d'incidence MIM en fonction de l'intensité de surveillance du serveur applicatif JBoss.

Instrumentation du serveur web

Le serveur web TJWs est instrumenté via des MBeans dynamiques puisqu'ils présentent de meilleures performances que les MBeans standards (voir section 5.4.5). Nous avons instrumenté un ensemble de composants de serveur web. Cette instrumentation se compose des MBeans suivants :

- *ThreadPool* exposant 3 attributs. Il présente des informations concernant l'ensemble des threads employés par le serveur pour servir les requêtes HTTP.
- *HTTPSession* exposant 2 attributs en lecture seule. Il présente un compteur de nombre des sessions HTTP ouvertes sur le serveur et leurs valeurs de *timeout*.
- *Host* exposant 3 attributs en lecture seule. Il expose des informations de base à propos de l'hôte du serveur.
- *Server* exposant 4 attributs en lecture seule et 2 opérations. Il expose des informations d'exécution du serveur et des opérations pour son démarrage et son arrêt.
- *Connector* exposant 6 attributs en lecture seule et 2 attributs en lecture-écriture. Il expose

les paramètres des connexions TCP.

– *RequestStatistics* exposant 2 attributs en lecture seule. Il expose les compteurs de requêtes.

Dès leur création chacun des MBeans possède une référence mémoire vers le composant à gèrer. Les attributs de ces MBeans sont classés en trois catégories : des attributs statistiques qui changent au cours du cycle de vie du serveur, comme le nombre de connexions TCP et la taille du *pool* de *threads* ; des attributs de configuration qui sont constants comme le numéro de port TCP ou le nom du serveur ; et l'ensemble des attributs collectés à partir des *logs* du serveur. Les deux premières catégories des attributs sont internes et ceux de la troisième sont externes.

Extension du banc de mesure

Nous avons étendu notre banc de mesure décrit au chapitre 5 pour qu'il supporte le serveur web TJW ainsi que les différents modèles d'intégration des agents. En effet, lors de la configuration d'un test sur le banc de mesure, l'utilisateur a le choix entre des agents synthétiques ou des agents réels qui supervisent le serveur web TJWS. Dans ce cas, l'utilisateur doit choisir le modèle d'intégration de l'agent parmi les trois modèles disponibles : démon, composant ou pilote. Dans le modèle démon, l'agent possède son propre processus système. Un protocole de communication *ad hoc* est interposé entre lui et le serveur pour collecter les informations qu'il expose via les attributs des MBean. Nous avons opté pour un *pool* de sockets TCP entre le serveur et l'agent démon pour leurs inter-communication.

La partie gestionnaire de banc de mesure ne nécessite aucune modification puisqu'elle est générique. Dans nos tests nous avons utilisé une interaction par scrutation via l'opération *getAttribute* entre le gestionnaire et l'agent. Nous avons rajouté sur le banc de mesure un outil

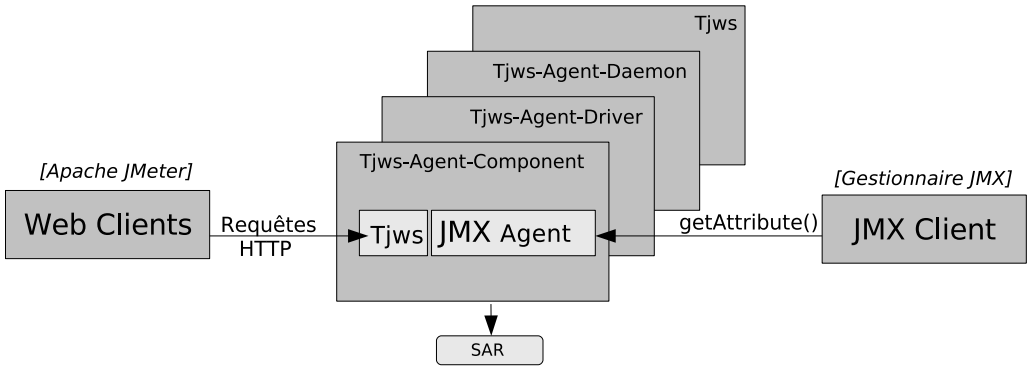


FIG. 7.7 – Architecture de test de TJWS couplé aux différents modèles d'intégration d'un agent JMX.

d'injection de charge web appliquée à la partie fonctionnelle de serveur TJWS. Nous avons ainsi intégré l'outil Apache JMeter[78] pour générer cette charge. JMeter est un outil de mesure de résistance à la charge, développé en Java. L'utilisateur définit un plan de test qui est la composition d'un ensemble de threads d'injection pour simuler des utilisateurs concurrents. Dans nos tests, chaque thread génère des requêtes HTTP vers le serveur. Chaque requête sollicite une page web statique ou un servlet hébergé par le serveur TJWS. Le temps de réflexion d'un utilisateur web suit une loi uniforme avec un temps moyen de 7 secondes et une plage de 10 secondes. Chaque utilisateur web est émulé par un *thread*.

[78] http ://jakarta.apache.org/jmeter

Description des environnements logiciels et physiques

Comme le montre la figure 7.7 chacun des composants de test (TJWS, JMeter, gestionnaire JMX) s'exécutent sur une machine séparée. Une console de test, non représentée sur la figure, a la charge de lancer les tests en déployant les différents composants puis la collecte des traces de mesure à la fin de chaque test. Ces tests sont réalisés sur la même plate-forme physique décrite dans la section 8.3.2. Nous avons utilisé 4 machines du cluster pour réaliser nos tests.

7.4.2 Scénarios de tests

Scénario / Charge	Aucun agent	Démon	Composant	Conducteur
Charge web seule	X	X	X	X
Charge JMX seule	-	X	X	X
Charges web et JMX concurrents	-	X	X	X

TAB. 7.3 – Scénarios de mesure de l'incidence de trois modèles d'intégration d'un agent sur un serveur web.

Nous avons identifié 10 scénarios de test indiqués sur le tableau 7.3. Le premier scénario représente le serveur web TJWS sans appliquer aucune charge JMX. Il s'agit du scénario de référence. Les trois autres scénarios correspondent au trois différents modèles d'intégration des agents. Nous avons varié séparement ou d'une manière concurrente le nombre des utilisateurs web (Injection par JMeter) et la charge JMX en terme de *getAttribute*/s. Chacune de ces charges est variée de 20 et 1000 par un palier de 20. Dans le cas de deux charges JMX et Web concurrentes, le nombre de clients et l'intensité de demande de la surveillance sont proportionnelles, c'est-à-dire si 20 utilisateurs web exercent une charge sur le serveur web, alors l'intensité de supervision est de 20 *getAttribute*/s. La durée de chaque test est de 20 minutes avec une période d'amorçage de 1 minute.

Pour la partie JMX, nous avons mesuré les délais que subit un attribut par le biais de l'opération *getAttribute* pour transférer sa valeur entre l'agent et le gestionnaire. Le débit mesuré sur le gestionnaire est en nombre d'attributs[79] par seconde. Nous avons aussi mesuré les consommations de ressources en terme de processeur et mémoire.

Pour la partie web, nous avons mesuré le nombre de transactions[80] HTTP par seconde et leurs délais.

7.4.3 Résultats expérimentaux

Analyse des débits

La figure 7.8 (a) présente le nombre moyen d'attributs scrutés par seconde mesuré sur le gestionnaire JMX, où aucune charge web n'est appliquée sur le serveur. Nous observons que les trois modèles possèdent les mêmes débits. Le coude du débit (*Knee capacity*) [21] est aux alentours de 300 attributs/s. Ce débit représente la limite au delà de laquelle la pente de la courbe de débit commence à dévier de l'unité. Cela signifie qu'à partir de ce point le gestionnaire

[79] Nous utilisons attribut au lieu de variable parce qu'une variable de gestion dans l'approche JMX est appelée attribut.

[80] Dans notre contexte une transaction HTTP est une paire requête/réponse.

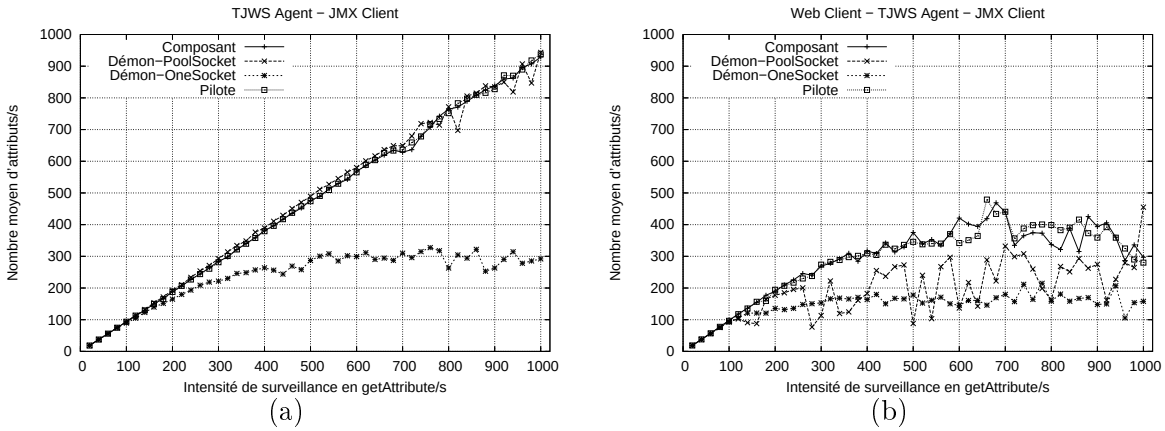


FIG. 7.8 – Débit JMX en terme d'attributs par seconde pour les trois différents modèles d'intégration d'un agent. (a) Charge JMX seule. (b) Charges web et JMX concurrentes.

commence à saturer. On observe aussi que le débit du modèle démon utilisant une seule socket de communication avec le serveur géré, sature plus rapidement que celui avec un *pool* de sockets.

À l'injection de la charge web comme l'indique la figure 7.8 (b), sur le serveur web, le débit JMX décroît pour les trois modèles. Le débit du modèle démon décroît au maximum de 50 % lorsque une charge web est exercée sur le serveur. Cette limitation de débit est due au coût introduit par la communication entre l'agent et le serveur. En revanche, les deux autres modèles répondent plus correctement même sous des intensités de surveillance importantes. Ces deux modèles atteignent un coude de débit de l'ordre de 200 attributs par seconde, mais ils continuent à répondre d'une manière satisfaisante jusqu'au une intensité de 500 getAttribute/s en retardant 10% des opérations. En effet, on conclut que le modèle démon est moins performant que les deux autres (composant et pilote) du point de vue de la supervision. Comme la montre la figure

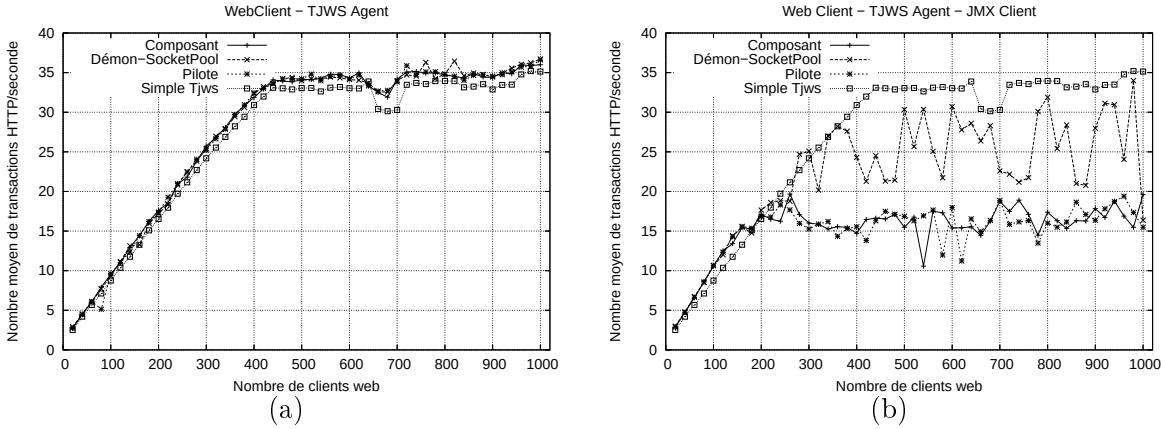


FIG. 7.9 – Débit web en terme de nombre de transactions HTTP par seconde pour les trois différents modèles d'intégration d'un agent. (a) Charge JMX seule. (b) Charges web et JMX concurrentes.

7.9 (a), la simple activation de l'instrumentation JMX sans lui appliquer de charge, ne perturbe pas le débit de serveur web en nombre de transactions HTTP servies par seconde. Le débit de

référence du serveur, representé sur la figure 7.9 par l'étiquette *Simple Tjws*, est de l'ordre de 35 transactions HTTP par seconde. En revanche, à l'injection d'une charge de supervision JMX, comme la montre la figure 7.9 (b), ce débit web diminue pour les trois modèles. Mais, on observe que les modèles composant et pilote ont plus d'incidence sur le débit de serveur web que le modèle démon. Cela est dû à la séparation de processus système démon de celui de serveur web.

Analyse des délais

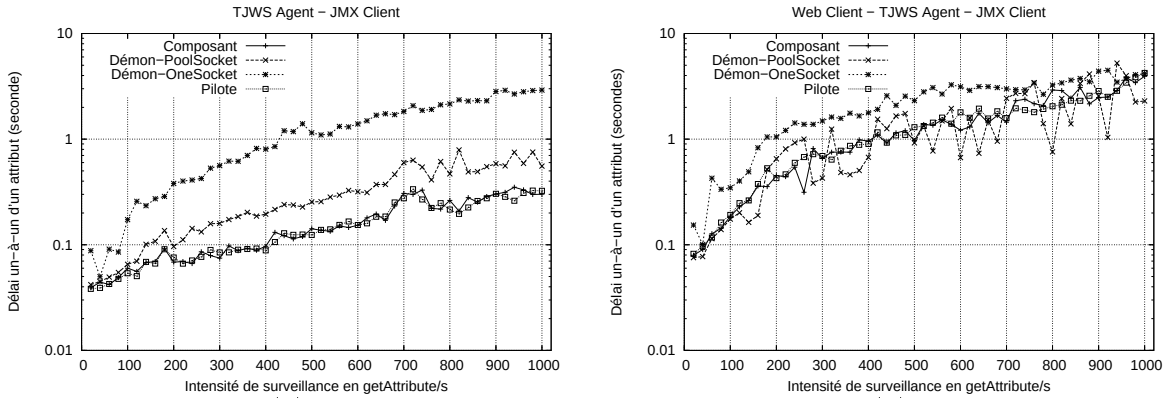


FIG. 7.10 – Délais un-au-un moyens d'un attribut pour les trois différents modèles d'intégration d'un agent. (a) Charge JMX seule. (b) Charges web et JMX concurrentes.

D'un point de vue délais, la figure 7.10 (a) montre que sans aucune charge web, les délais d'un attribut mesurés sur le gestionnaire sont au dessous de la seconde. Au delà d'une intensité de surveillance de 200 getAttribute/s, les délais que subissent les attributs sous le modèle démon commencent à croître plus que ceux des deux autres modèles. Cela est dû, comme nous l'avons mentionné précédemment, aux délais introduits par le processus d'inter-communication entre l'agent et le serveur web. Évidement à l'application d'une charge web sur le serveur (figure 7.10 (b)), les délais de supervision augmentent considérablement pour les trois modèles. Ceci est critique pour le modèle démon qui voient ses délais dépasser la seconde à partir d'une intensité de 400 *getAttribut*/s.

La figure 7.11 (a) montre que les délais des transactions HTTP ne sont pas affectés lorsque aucune charge JMX n'est exercée. En effet, avec un nombre de clients web de l'ordre de 300, les délais de transactions web sont au dessous d'une seconde. En revanche, comme l'indique la figure 7.11 (b), lorsque le serveur web est confronté à une charge de supervision ces délais deviennent plus fluctuants et plus importants. Les modèles composant et pilote affectent plus particulièrement le plan fonctionnel web du serveur TJWS. Dans ces deux cas, les délais des transactions HTTP augmentent de 2 à 3,5 fois que ceux du cas d'un serveur web sans aucune charge de supervision. Cela est dû à leur forte intégration au niveau de serveur web et leurs concurrence directe avec son plan fonctionnel. Le modèle démon affecte moins les délais que subissent les requêtes HTTP, avec un facteur de l'ordre de 2.

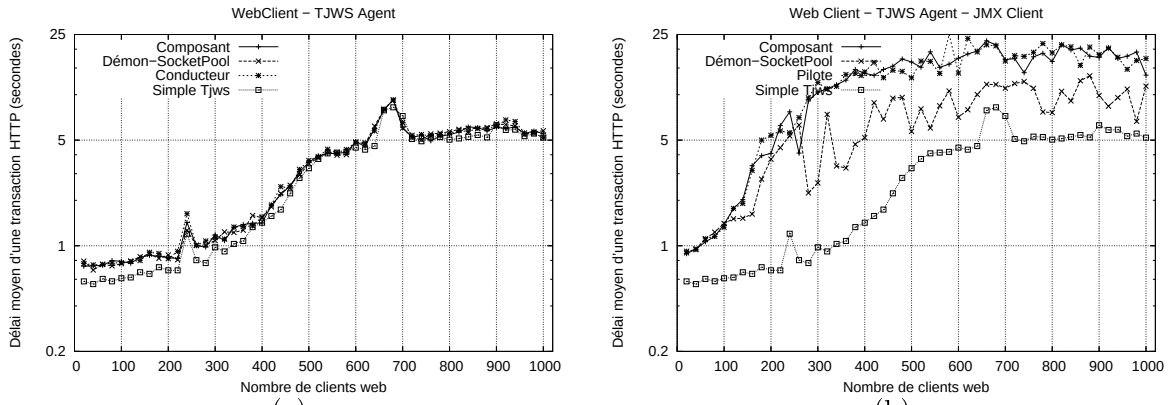


FIG. 7.11 – Délais de transactions HTTP pour les trois différents modèles d'intégration d'un agent. (a) Charge JMX seule. (b) Charges web et JMX concurrentes.

Utilisation des ressources

Nous avons analysé les ressources consommées en termes d'occupation processeur et mémoire sur le serveur web pour les différents modèles d'intégration et sous les différents scénarios de test présentés dans le tableau 7.3. La figure 7.12 présente les différents résultats de ces scénarios en terme de taux moyen d'occupation de processeur et d'occupation mémoire sur le serveur sous ces différents scénarios. Dans tous ces tests, l'occupation mémoire est quasiment constante située entre 10 à 20% de la mémoire disponible sur le serveur. Cela signifie qu'une charge JMX ne nécessite pas le chargement d'un nombre important d'objets Java dans la mémoire. En revanche, d'un point de vue occupation de processeur les résultats sont différents. Lorsque seule une charge web est exercée sur le serveur, la consommation processeur est faible de l'ordre de 5% du temps processeur disponible. En revanche, si seule une charge JMX est appliquée au serveur cette consommation processeur augmente nettement. Elle atteint une valeur proche de 40 % mais sans la dépasser. Ainsi, il s'avère que la couche RMI de JMX et les sessions TCP affectent sévèrement la consommation processeur. Cela est notamment visible pour le modèle démon qui consomme plus de processeur que les autres à cause de sa communication via des sockets TCP avec le serveur web. Les deux charges JMX et web concurrentes appliquées sur le serveur affectent considérablement sa consommation processeur. Cette consommation atteint rapidement des valeurs de l'ordre 50% pour les trois modèles d'intégration. On observe que pour les deux modèles composant et pilote cette consommation se stabilise au niveau de cette valeur. En revanche, celle du modèle démon est plus fluctuante et elle varie entre 15 % et 50 %. En effet, il apparaît qu'une activité JMX sur un serveur web sous une charge utilisateur affecte considérablement sa consommation processeur, sans qu'elle impose un surcoût au niveau de la mémoire.

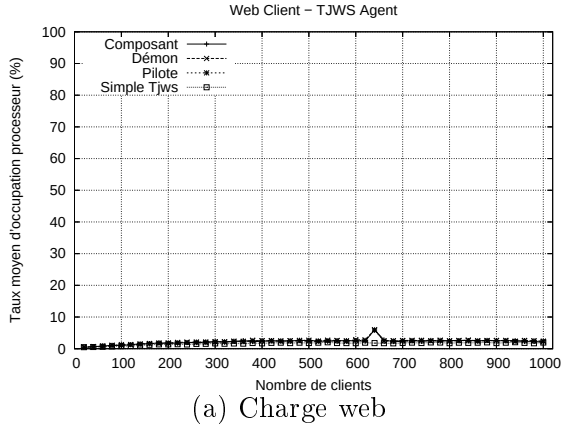
Effet de la nature de l'attribut surveillé

Comme nous l'avons indiqué dans la section 7.4.1, les attributs surveillés, par un gestionnaire JMX depuis un agent d'un serveur web, sont de nature différentes. L'attribut *Port* est statique au cours de cycle de vie de serveur, l'attribut *SessionCount* est en revanche dynamique. Ces deux attributs sont accédés en interne depuis le serveur géré. L'attribut *LogRequestCount* est collecté par l'agent depuis les fichiers de logs de serveur. Les figures 7.13 ,7.14 et 7.15 présentent les délais que subissent ces attributs par le biais de l'opération getAttribute pour récupérer leurs

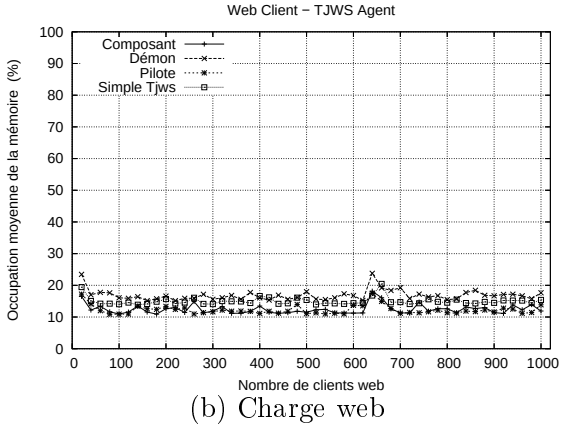
valeurs depuis les différents modèles d'intégration des agents. Ces résultats confirment que les attributs collectés à partir des logs de serveur subissent des latences importantes et leurs délais un-à-un mesurés au niveau de gestionnaire deviennent considérables et dépassent rapidement la seconde dès que la demande de surveillance est au delà de 180 *getAttribute*/s.

7.5 Synthèse

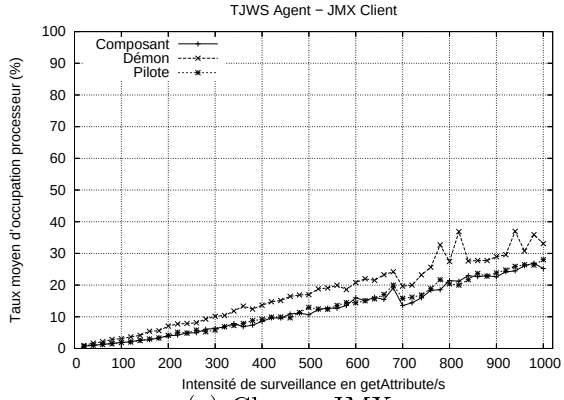
Dans ce chapitre nous avons présenté une métrique pour capturer l'incidence des activités de gestion sur la performance d'un système géré, notamment son efficacité fonctionnelle. Cette métrique représente les rapports des rendements du système géré sous deux valeurs différentes d'un facteur d'impact relatif à la pratique de gestion appliquée au système géré. Nous avons illustré cette métrique sur la supervision d'un serveur applicatif, nommé JBoss pour les applications Java. Ce serveur applicatif repose sur un modèle pilote de son agent JMX. Nos tests de mesure sur ce serveur, en lui appliquant une intensité de la surveillance, ont montré que l'incidence de la supervision est considérable. Cette incidence sur son efficacité de référence varie entre 24% à 74% selon la valeur de l'intensité de la surveillance en nombre de getAttribute/s à laquelle est soumis son agent JMX. Cette étude sur ce serveur applicatif avec son modèle d'agent fortement intégré, nous a amené à étendre notre banc de mesure dédié à l'approche JMX pour identifier d'une manière expérimentale l'effet de chacun de trois modèles d'intégration (démon, composant et pilote) sur un serveur web. Nos résultats ont montré que le modèle démon affecte particulièrement la performance de la partie supervision plus que les deux autres modèles. La performance du plan fonctionnel en terme du nombre de transactions HTTP servies par le serveur web diminue par un facteur qui varie entre 1,5 à 2,3. Ainsi que les délais de ses transactions augmentent d'un facteur qui varie entre 2 à 3,5. Les résultats obtenus dans ce chapitre, prouvent qu'une activité de gestion non calibrée et non contrôlée engendre des sérieux problèmes de performances sur l'application gérée. En revanche, cette dégradation de l'efficacité du serveur n'est visible que pour des intensités de surveillance importantes de l'ordre de centaines de getAttribute/s. Cela nous amène à poser la question sur l'existence de telles intensités dans des applications réelles de supervision.



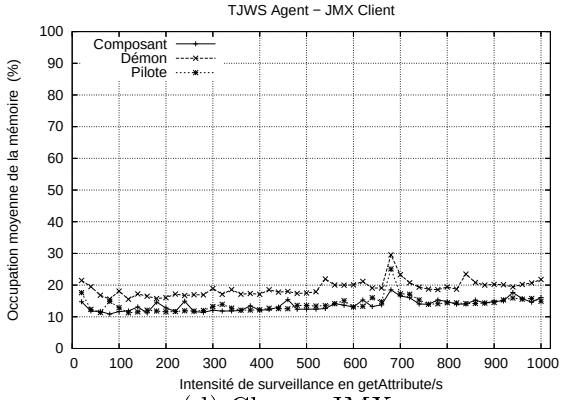
(a) Charge web



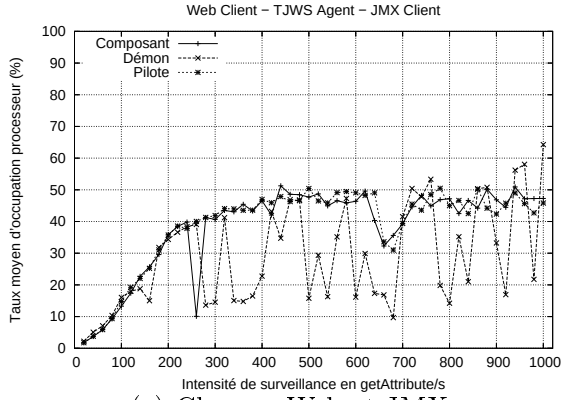
(b) Charge web



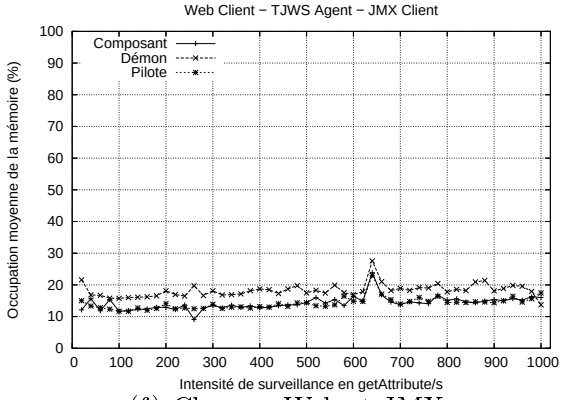
(c) Charge JMX



(d) Charge JMX



(e) Charges Web et JMX



(f) Charges Web et JMX

FIG. 7.12 – Utilisation de ressources processeur et mémoire sur le serveur web pour les trois modèles d'intégration d'un agent JMX.

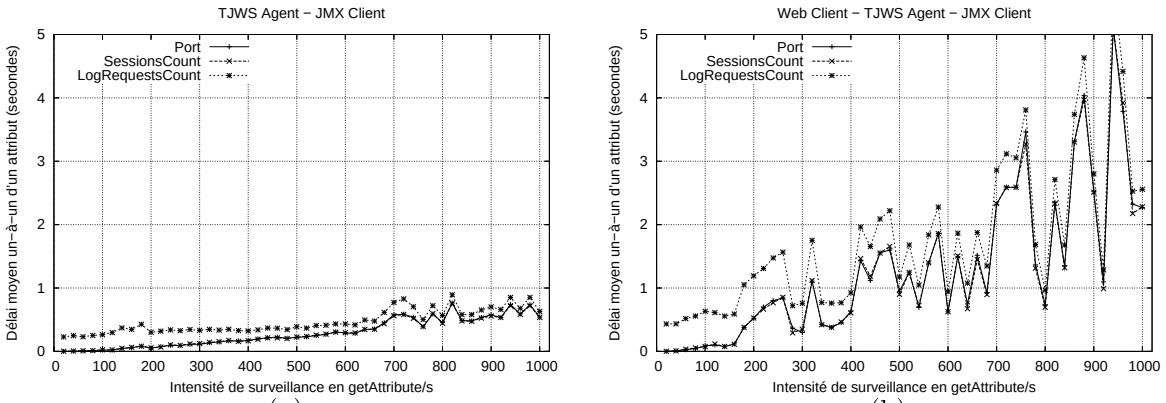


FIG. 7.13 – Délais un-à-un d'un attribut mesurés sur le gestionnaire pour les trois différents types d'attributs (statique, dynamique, externe) exposés par un agent de type démon. (a) Charge JMX seule. (b) Charges web et JMX concurrentes.

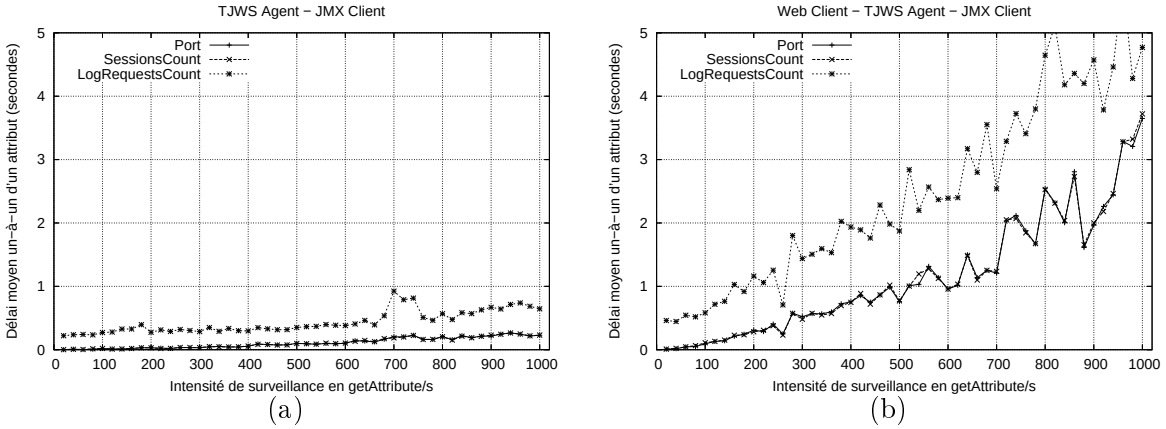


FIG. 7.14 – Délais un-à-un d'un attribut mesurés sur le gestionnaire pour les trois différents types d'attributs (statique, dynamique, externe) exposés par un un agent de type composant. (a) Charge JMX seule. (b) Charges web et JMX concurrentes.

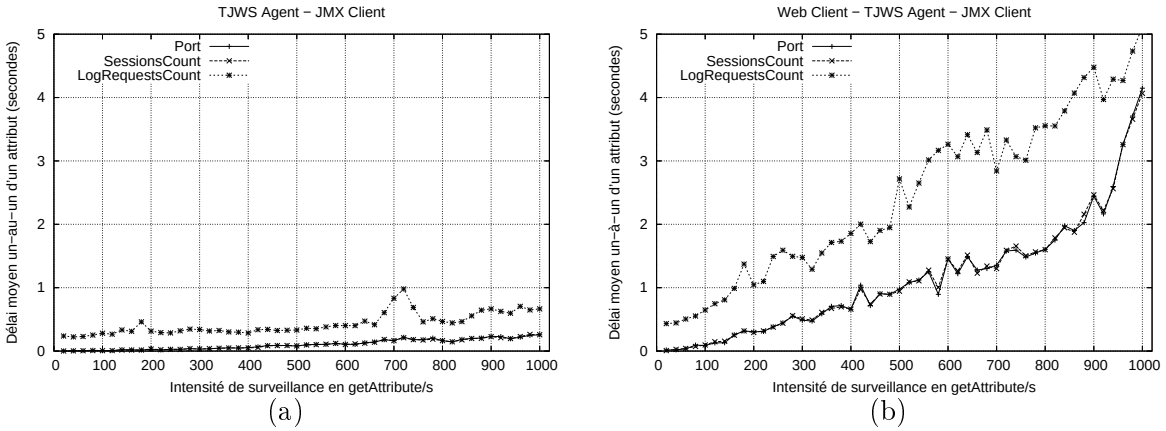


FIG. 7.15 – Délais un-à-un d'un attribut mesurés sur le gestionnaire pour trois différents types d'attributs (statique, dynamique, externe) exposés par un agent de type pilote. (a) Charge JMX seule. (b) Charges web et JMX concurrentes.

Chapitre 8

Caractérisation des délais de la supervision

Sommaire

8.1 Introduction

Les délais représentent la métrique clé d'une approche de gestion. D'autres métriques comme les pertes ou la précision temporelle dépendent étroitement de ces délais. Cela est dû notamment au caractère temporel des activités de gestion. Par exemple, une activité de surveillance repose sur des périodes de collecte des variables et de mise à jour de leurs valeurs. Les délais de gestion se manifestent lors de transferts de valeurs de ces variables. Des délais importants introduits lors de ces transferts affectent considérablement le comportement des fonctions de gestion ainsi que la qualité de leurs données. Une application typique de gestion qui est sensible aux délais est la supervision des objets à durée de vie courte. Un exemple de ces objets sont les transactions SIP[81] dans une infrastructure voix sur IP. Ces transactions sont de très courtes durées. Si les opérations de gestion subissent des délais importants lors de transferts des variables représentants ces objets, les données obtenues deviennent obsolètes. Cela rend la tâche de diagnostic des problèmes qui peuvent surgir sur ces infrastructures, quasiment impossible [42]. Ainsi, il est indispensable de caractériser ces délais et d'avoir une meilleure compréhension des facteurs qui les affectent.

Dans ce chapitre, nous présentons les résultats des campagnes de mesure de délais que subissent les variables dans une pratique de surveillance sous différents scénarios. Les mesures sont effectuées sur le banc de mesure *MAGON* que nous avons développé pour l'évaluation de performances des approches de gestion. Nos tests sont menés sur son instance reposant sur la technologie JMX.

[81] SIP : Session Intiation Protocol.

Ce chapitre est organisé de la façon suivante. Tout d'abord, nous présentons les différents composants du délais bout-en-bout que subit une variable lors de son transfert entre deux ou plusieurs entités de gestion. Ensuite, nous présentons les résultats de la mesure des métriques de délais présentés dans le chapitre 4, pour mieux comprendre leurs distributions statistiques dans le contexte d'une pratique de surveillance sous un mode de scrutation. Cette caractérisation nous a montré des propriétés fondamentales de la supervision ainsi que ces limitations en terme de qualité.

8.2 Différents composants des délais de la gestion

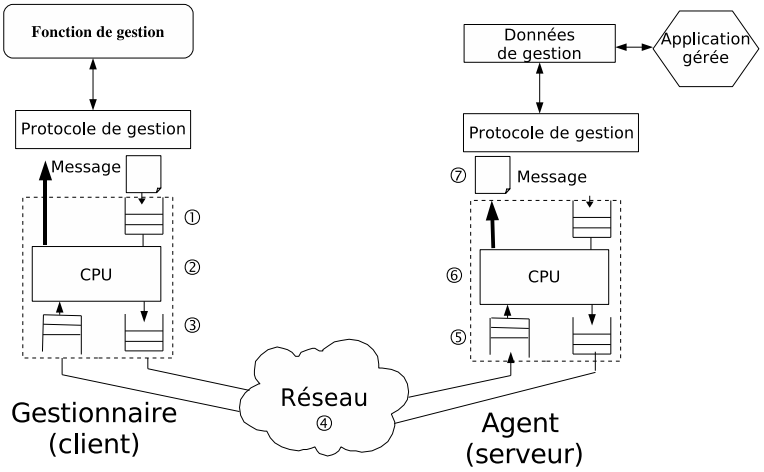


FIG. 8.1 – Décomposition des délais de bout-en-bout entre un gestionnaire et un agent.

Généralement, les délais de bout-en-bout sont exprimés sous la forme de deux composantes, une composante fixe et une autre variable [47]. La composante fixe se présente sous la forme du temps de transmission sur un nœud et le temps de propagation sur le lien de communication. Les composantes variables sont le temps de traitement et le temps d'attente dans les files d'un nœud. Le temps de transmission est linéaire en fonction de la taille des paquets. Le temps d'attente augmente d'une façon linéaire en fonction du temps d'inter-arrivées de paquets, mais varie rapidement dès que ces temps sont faibles.

Dans le contexte d'un modèle gestionnaire-agent comme indiqué sur la figure 8.1, nous décomposons le délai que subit une variable entre deux entités de gestion différentes communicant via un réseau en trois composantes : délais au niveau du gestionnaire, délais de messages et délais au niveau de l'agent. Les délais des messages représentent les délais liés à la couche transport et réseau. Ils incluent les délais de propagation et de transmission des paquets de protocole de transport sous-jacent au protocole de gestion. Le délai que subit une variable sur une entité de gestion se caractérise par son temps de séjour dans la couche du protocole de gestion présent sur une entité. Ce délai correspond au temps de traitement et d'attente des opérations de gestion au niveau de l'entité. De plus sur un agent, le temps d'accès à l'objet géré n'est pas négligeable et doit être considéré lors de l'analyse des délais au niveau des entités de gestion [142]. Par conséquent, le délai que subit la variable au cours d'une opération se traduit par la formule suivante :

$$D(V_i) = D_{Gestionnaire}(V_i) + D_{Message}(V_i) + D_{Agent}(V_i) \tag{8.1}$$

Les composantes relatives aux entités de gestion sont aussi la composition d'une partie fixe qui représente le délai minimal que subit la variable par le biais d'une opération et une partie variable qui dépend de plusieurs facteurs. Pour une approche de gestion, nous avons identifié les facteurs suivants qui affectent les délais que subissent les variables :

- Le nombre d'agents ;
- Le nombre d'objets gérés sur chaque agent ;
- Les périodes temporelles des variables : MPP et AUP présentées dans la section 4.2.

D'autres facteurs, comme la sécurité [136] et le facteur hétérogénéité des agents [142] affectent aussi considérablement les délais de gestion. Cette liste de facteurs n'est pas exhaustive mais représente les facteurs que nous avons considérés lors de notre caractérisation de délais de supervision. Nous nous sommes essentiellement intéressé aux deux facteurs que sont le nombre d'agents depuis lesquelles une variable est surveillée et la variation de ses périodes de surveillance. Ces deux facteurs sont retenus à cause de leur effet sur le passage à l'échelle d'une approche de gestion. Dans la suite, nous caractérisons le délai de bout-en-bout d'une variable sans sa décomposition sur les composantes présentées précédemment. Ce choix nous permet de caractériser le délai que perçoivent les algorithmes de gestion résidant sur une entité de référence.

8.3 Délais d'un scénario gestionnaire-agent

Nous nous somme focalisé sur la mesure de délais du scénario gestionnaire-agent. Dans cette section nous adressons les questions suivantes : ***quelles sont les métriques de délais de bout-en-bout relatifs à ce scénario ? quelles sont leurs propriétés et leurs limitations dans ce scénario ?***

8.3.1 Métriques de délais d'une variable

Comme indiqué dans la section 4.2.2, les métriques de délai sont de type un-à-un ou un-au-groupe selon le nombre d'entité impliquées dans la fonction de gestion. Le mode d'interaction spécifie la nature des délais. Dans un mode de scrutation, nous mesurons des délais bidirectionnels (*Round-Trip-Delay*). Dans un mode de notification, ces délais sont unidirectionnels (*One-Way-Delay*).

Délai un-à-un d'une variable : Ce délai est le temps total que subit une variable pour retrouver ou altérer sa valeur sous un mode d'interaction spécifique entre deux entités de gestion. Ce délai est mesuré en unité de temps sur une entité représentant son point de mesure.

Délai un-au-groupe d'une variable : Ce délais représente la quantité de temps nécessaire pour retrouver/altérer la valeur d'une variable entre une entité de gestion de référence (un gestionnaire par exemple) et un groupe d'agents. On s'aperçoit qu'un échantillon spatial de cette métrique est la composition de singletons où chacun d'eux représente un délai un-à-un de la variable depuis des agents différents. Les échantillons de ces délais sont résumés par l'une des statistiques usuelles (moyenne, écart type) ou robustes présentées dans la section 1.5.4. Chacune de ces métriques est identifiée par le nom de la variable, le type de la métrique, l'opération de la gestion et sa multiplicité. Nous ne spécifions pas le type de délai (unidirectionnel ou bidirectionnel) puisque il est identifié par l'opération de gestion utilisé. Par exemple, une opération de notification implique un délai unidirectionnel. En revanche, une opération *getAttribute* implique un délai bidirectionnel.

8.3.2 Pratique sous test et sa méthodologie de mesure

Nos expérimentations se sont déroulées sur l'instance JMX de MAGON dans un scénario gestionnaire-agent avec un mode de scrutation. En effet, la pratique de supervision sous test possède les paramètres suivants :

- Approche : centralisée, gestionnaire-agent ;
- Fonction : surveillance ;
- Mode d'interaction : scrutation ;
- Mode de concurrence : concurrent ;
- Technologie : JMX ;
- Donnée de gestion : un attribut dans un MBean.

L'opération utilisée pour surveiller la variable est *getAttribute* possède une multiplicité de 1. L'évaluation de cette pratique repose sur des tests synthétiques où aucun comportement d'une application gérée n'est déclenché. La méthode utilisée pour mesurer le délai est l'instruction Java *currentTimemillis*. Les temps sont mesurés au niveau applicatif sur le gestionnaire sous la forme de timestamps retournés par cette méthode avant (T1) et après l'appel de l'opération *getAttribute* (T2). Ces différents timestamps sont stockés dans un fichier relatif à un agent. Le format d'une entrée de ce fichier est présenté dans la figure 8.2. Les délai que subit une variable

```
T1
T1 T2
```

FIG. 8.2 – Format d'une entrée de fichier de log des instants d'appel des opérations et de leurs réponses.

par le biais de l'opération *getAttribute* est la différence $T2 - T1$. Ce choix de format nous permet aussi de compter le débit en comptant le nombre de lignes de type *T1 T2* et le temps total du test. Chaque test débute par une période de 60 secondes qui représente la période d'amorçage (*warm-up*) qui élimine l'effet de démarrage à froid. Cette période est suivie d'une durée de test de 1200 secondes.

Les facteurs variés au cours d'une série de mesures de délai sont le nombre d'agents et l'intensité de surveillance d'une variable.

Dans un premier temps, nous avons évalué cette pratique avec un gestionnaire et un seul agent en variant seulement les périodes de surveillance d'une variable. L'intensité de scrutation de la variable est variée entre 1 à 1000 *getAttribute* par seconde. Dans un deuxième temps, nous avons évalué la même pratique en variant le nombre d'agents et en fixant la période de surveillance de la variable à une seconde. Nous avons varié le nombre d'agents entre 1 et 700 agents.

Description des environnements physiques et logiciels

Nos mesures sont effectuées sur une grappe de 100 machines connectées par un réseau Ethernet de capacité 1 Gbits/s. Il s'agit de la grappe *I-Cluster2*[82] faisant partie de la grille *Grid5000*. Chaque machine repose sur deux processeurs *Itanium* 2 d'architecture 64 bits cadencés chacun à une vitesse de 900 MHZ et doté d'une mémoire de capacité 3 Giga octets. Le système d'exploitation déployé sur ces machines est un Linux RedHat version AS 3, reposant sur un noyau version 2.4.21-32.0.1.EL. La JVM déployée est une BEA WebLogic JRockit reposant sur une JDK 1.4.2_04.

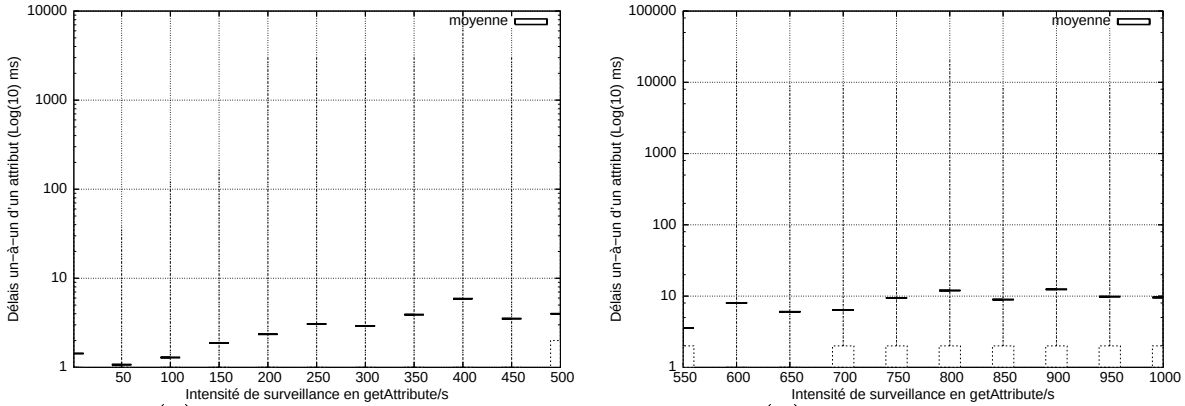
[82] http ://i-cluster2.inrialpes.fr/

Dans ce travail, nous avons utilisé l'implantation MX4J version 2.0 pour JMX étant donné quelle présente de meilleure performance que l'implantation de référence de SUN, comme il était indiqué dans la section 5.4.5.

8.3.3 Effet de l'intensité de surveillance sur le délai un-à-un d'une variable

Dans cette section, nous analysons l'effet de la variation de l'intensité de demande de surveillance en terme de *getAttribute*/s sur les délais que subit une variable par le biais de cette opération. Ici nous avons utilisé comme environnement physique la grappe de machines décrite dans la section 8.3.2 où le réseau est de très faible latence (moyenne 0,083 ms[83]).

Tests synthétiques



(a) Faible intensité **(b) Forte intensité**

FIG. 8.3 – Statistiques de délais un-à-un d'un attribut sous différentes intensités de supervision en terme de *getAttribute*/s.

La figure 8.3 représente le délai un-à-un d'un attribut surveillé sur un agent. Il s'agit d'une représentation sous forme de boîte à moustaches (*Box and Whisker plot*[84]) qui utilise 5 valeurs pour résumer les délais : le minimum, les 3 quartiles Q_{25}, $Q2_{50}$, Q_{75} et le maximum. Nous avons représenté aussi la moyenne sous la forme d'un trait horizontal. Ces délais sont mesurés sur le gestionnaire. On observe que ces délais sont en moyenne au dessous de 10 ms. En revanche, le maximum de ces délais dépasse 1 seconde dès que l'intensité de surveillance est au delà de 200 *getAttribute*/s. La figure 8.4 montre les fonctions de répartition empiriques de ces délais sous différentes intensités de demande de supervision d'un agent synthétique. On observe que même avec une intensité de 1000 *getAttribute*/s, 75% de ces délais sont de l'ordre de 1 ms.

Nous avons utilisé les outils d'analyse statistiques présentés dans la section 1.5.4 du chapitre 1 pour identifier une distribution statistique qui approche au mieux ces délais un-à-un sous différentes intensités de surveillance. Le tableau 1.5.4 récapitule les différentes distributions trouvées avec leurs valeurs d'adéquation λ^2. Les distributions empiriques de délais ont été vérifiées

[83] Cette valeur est obtenue par un simple Ping entre deux machines de test du cluster.

[84] La boîte à moustaches, une traduction de Box and Whiskers plots, est une invention de TUKEY en 1977 pour représenter schématiquement la distribution d'un jeu de données.

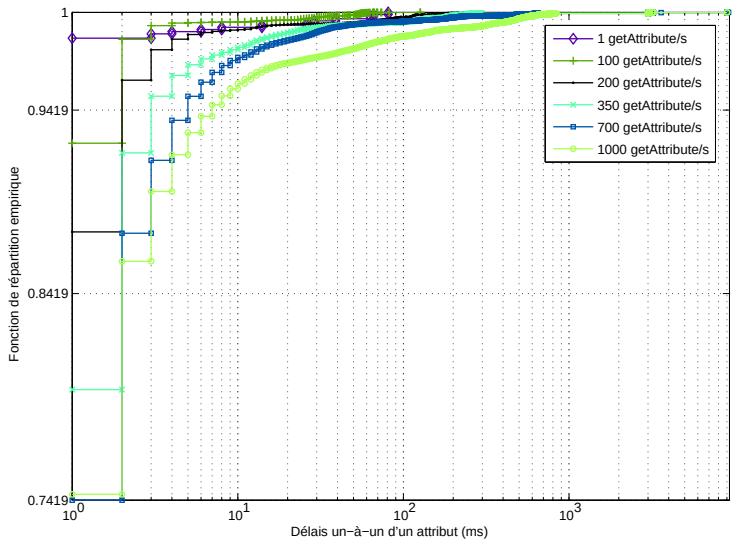


FIG. 8.4 – Fonctions de répartition empirique de délais un-à-un d'un attribut accédé depuis un agent synthétique.

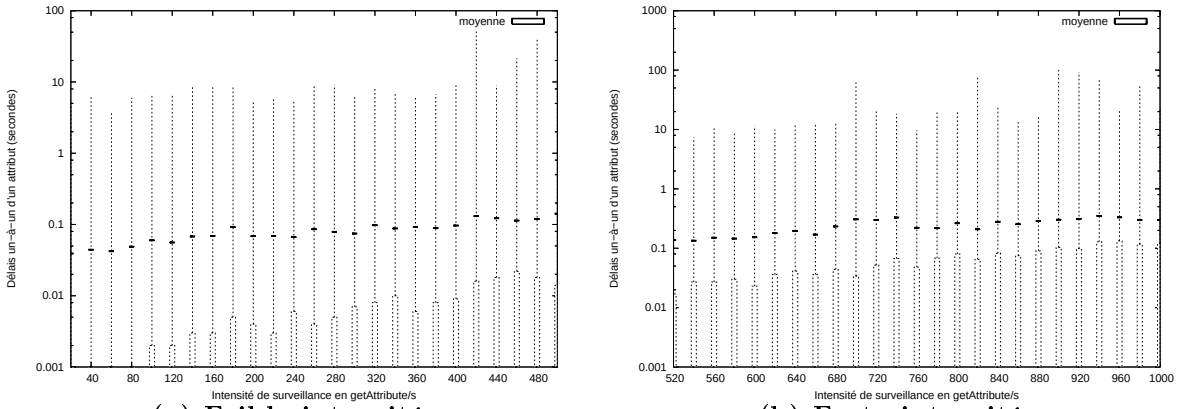
par rapport à l'ensemble des distributions classiques suivantes : normal, exponentiel, Weibull, Gamma, extrème, LogNormal, Rayleigh et uniforme. Ces distributions sont les bons candidats utilisés dans la littérature pour modéliser les délais des applications qui reposent sur le protocole TCP [60]. On observe que la distribution LogNormal est dominante et elle approche au mieux les délais un-à-un empiriques mesurés entre un gestionnaire et un agent sous différentes intensités de surveillance.

Intensité de surveillance	Distribution de délais un-à-un	Coefficient d'adéquation λ^2
1 getAttribute/s	LogNormal	0.0012
100 getAttribute/s	LogNormal	1.0231
200 getAttribute/s	LogNormal	0.0184
350 getAttribute/s	LogNormal	0.3534
700 getAttribute/s	Weibull	0.22
1000 getAttribute/s	LogNormal	0.3079

TAB. 8.1 – Distributions statistiques approchant les délais un-à-un d'un attribut sous différentes intensités de surveillance.

Tests réalistes

Dans la suite, nous avons analysé les délais un-à-un d'un attribut dans un test plus réaliste. En effet, nous avons collecté une trace de ces délais depuis les tests de mesure décrits dans le chapitre 7 où nous avons couplé à l'agent un serveur web à surveiller. L'objectif de cette analyse est d'avoir une idée de l'effet du temps d'accès à la valeur d'un attribut d'une application gérée (un serveur web dans notre cas), sur les délais un-à-un que subit cet attribut collecté par le gestionnaire . Évidement on observe sur la figure 8.5 que ces délais sont plus importants que ceux d'un agent synthétique sans aucune application gérée à superviser. Ainsi on s'aperçoit que



(a) Faible intensité **(b) Forte intensité**

FIG. 8.5 – Statistiques de délais un-à-un d'un attribut sous différentes intensités de supervision d'un agent JMX intégré comme composant dans un serveur web.

le temps d'accès de la valeur de l'attribut depuis l'application géré à un effet important sur les délais de bout-en-bout (gestionnaire, agent, application gérée). Sous une intensité de 100 *getAttribute*/s les délais depuis un agent couplé au serveur web sont en moyenne de l'ordre de 60 ms. Sous la même intensité les délais depuis un agent synthétique sont en moyenne de l'ordre de 1.29 ms.

La figure 8.6 représente les fonctions de répartition empiriques de ces délais un-à-un. On observe que si dans le premier cas d'un agent synthétique 75% de délais sont de l'ordre de 1 ms pour une intensité de 1000 getAttribute/s, ce pourcentage descend à 28% dans le cas d'un agent intégré dans un serveur web. D'après le tableau 8.2, la distribution empirique des délais un-à-un d'un attribut accédé depuis un agent supervisant un serveur web approche au mieux les distributions LogNormal et Weibull. Ce résultat est le même obtenu pour les délais d'un agent synthétique. Nous étendons cette caractérisation de délais un-à-un au cas d'un agent intégré

Intensité de surveillance	Distribution de délais un-à-un	Coefficient d'adéquation λ^2
100 getAttribute/s	LogNormal, Weibull	0.1365, 0.2664
200 getAttribute/s	LogNormal, Weibull	0.2504, 0.4663
700 getAttribute/s	LogNormal, Weibull	0.2456, 0.2844
1000 getAttribute/s	LogNormal, Weibull	0.0972, 0.1857

TAB. 8.2 – Distributions statistiques approchant les délais un-à-un d'un attribut soumis à différentes intensités de surveillance depuis un agent JMX intégré sous forme d'un composant dans un serveur web.

dans un serveur web qui est soumis à une charge fonctionnelle sous forme des transactions HTTP. L'agent est aussi soumis à une charge de supervision sous la forme d'une intensité de surveillance d'un attribut. Une description détaillée de l'environnement de mesure est présentée dans la section 7.4.2. Dans ce cas nous observons que les distributions empiriques de délais un-à-un s'approchent des distributions classiques de types *LogNormal* et *Weibull*.

En effet, dans un scénario de type 1 gestionnaire-1 agent, le couplage de l'application gérée à

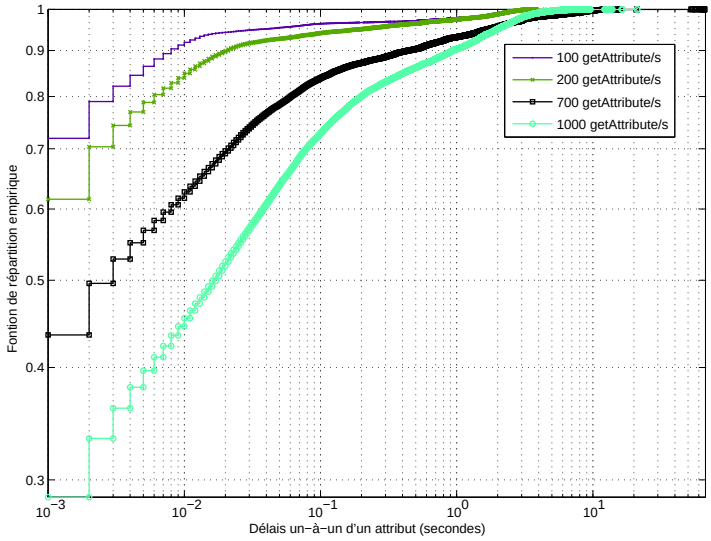


FIG. 8.6 – Fonctions de répartition empirique de délais un-à-un d'un attribut trouvé depuis un agent intégré comme composant dans un serveur web.

l'agent ne change pas la nature de la distribution des délais que subissent les attributs mais ne fait qu'augmenter leurs ordres de grandeur.

8.3.4 Effet du nombre d'agents sur le délai un-au-groupe d'une variable

Dans cette section, nous analysons l'effet du nombre d'agents sur les délais que subit une variable accédée depuis un groupe d'agents de taille spécifique par le biais de l'opération *getAttribute* sous une intensité de 1 *getAttribute*/s.

Environnement à faible latence réseau

Dans cette analyse nous nous reposons sur la trace collectée sur notre banc de mesure sous un scénario 1 gestionnaire et plusieurs agents avec une charge synthétique. Tout d'abord, nous analysons le délais un-à-un d'un attribut trouvé depuis un agent faisant parti de groupe d'agents. Cela nous permet de comparer ces délais avec ceux obtenus dans la section précédente où un seul agent soumis à une charge synthétique est attaché au gestionnaire. Comme l'indique le tableau 8.3, les délais un-à-un d'un attribut accédé depuis un agent faisant parti d'un groupe d'agents attaché au gestionnaire sont plus importants que ceux d'un gestionnaire et un seul agent. Même dans le cas où les intensités de surveillance sont importantes, les délais de scénario 1 gestionnaire, 1 agent sont nettement plus faibles que ceux présentés dans le tableau 8.3. D'un point de vue distribution statistique approchant la distribution empirique de ces délais, nous avons trouvé que la distribution LogNormal reste encore valable pour approcher ces délais.

Pour analyser les délais un-au-groupe d'un attribut accédé depuis un groupe d'agents, nous avons considéré la moyenne de chacun des échantillons de délais un-à-un d'un attribut accédé depuis chaque agent. La figure 8.7 montre la relation entre les délais un-à-un et ceux de un-au-groupe. Chaque unité de l'axe des abscisses représente l'index d'un agent (le groupe est de taille 140 agents) dans le groupe, sur l'axe des ordonnées est l'échantillon des délais un-à-un de l'attribut

Taille de groupe	Statistiques de délai un-à-un d'un attribut (ms)					
	Moyenne	Écart type	Médiane	IQR	Min	Max
70	1.02	0.20	1	0	1	4
140	4.45	46.34	1	0	1	973
210	68.47	432.37	1	0	1	3614
280	105.08	983.41	1	0	1	9960
350	392.73	2250.64	1	0	1	20269
420	675.24	3689.65	1	0	1	33365
490	561.55	4254.14	1	0	1	39816
560	1787.82	8179.39	1	0	1	50395
630	1916.86	8764.2	1	0	1	60646
700	2394.55	12073.7	1	0	1	78994

TAB. 8.3 – Statistiques des délais un-à-un d'un attribut accédé depuis un agent faisant parti d'un groupe d'agents de différentes tailles.

accédé depuis chacun des agents. La moyenne de chacun de ces échantillons par agent constitue un singleton de l'échantillon de délai un-au-groupe, présentée par une courbe sur la figure 8.7. Sur l'échantillon de délai un-au-groupe obtenue nous avons appliqué les statistiques usuelles. Les résultats sont présentés dans le tableau 8.4. On observe que les valeurs des estimateurs de délai

Taille de groupe	Statistiques de délai un-au-groupe d'un attribut (ms)					
	Moyenne	Écart type	Médiane	IQR	Min	Max
70	1.05	1.07	1	0	1	100
140	4.65	51.48	1	0	1	1988
210	53.77	410,351	1	0	1	7039
280	67.60	736.238	1	0	1	11141
350	495.38	2504.66	1	0	1	24225
420	1042.73	4363,95	1	0	1	34683
490	1776.89	6456,44	1	0	1	45182
560	2975.32	9630,81	1	0	1	56159
630	4244.80	12944,9	1	1	1	70543
700	6562.28	18678,6	1	1	1	103105

TAB. 8.4 – Statistique de délai un-au-groupe d'un attribut accédé depuis un groupe d'agents de différentes tailles, déployés sur un cluster.

un-au-groupe sont proches de celles de délai un-à-un d'un attribut lorsque la taille de groupe est inférieure à 280 agents. En revanche, dès que la taille de groupe dépasse cette limite, les estimateurs de la moyenne et de l'écart type deviennent différents de ceux de délai un-à-un d'un agent individuel. Comme l'indique la figure 8.8, pour des tailles de groupe entre 70 et 280 agents les distributions empiriques de délais un-au-groupe d'un attribut basées sur les moyennes de délais un-à-un semblent proches de celles de délai un-à-un d'un seul agent (voir figure 8.4). En revanche, au-delà de ces tailles les distributions deviennent plus plates puisque le délai moyen un-à-un de chacun de agents est différent de celui d'un autre agent dans le même groupe.

Nous allons maintenant vérifier si les distributions statistiques LogNormal et Weibull approchant le délai un-à-un d'un attribut sont encore valables pour le délai un-au-groupe. On observe

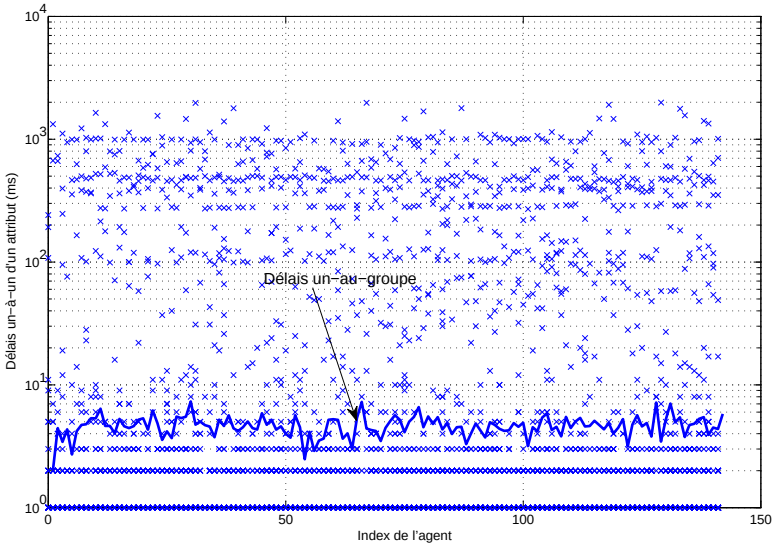
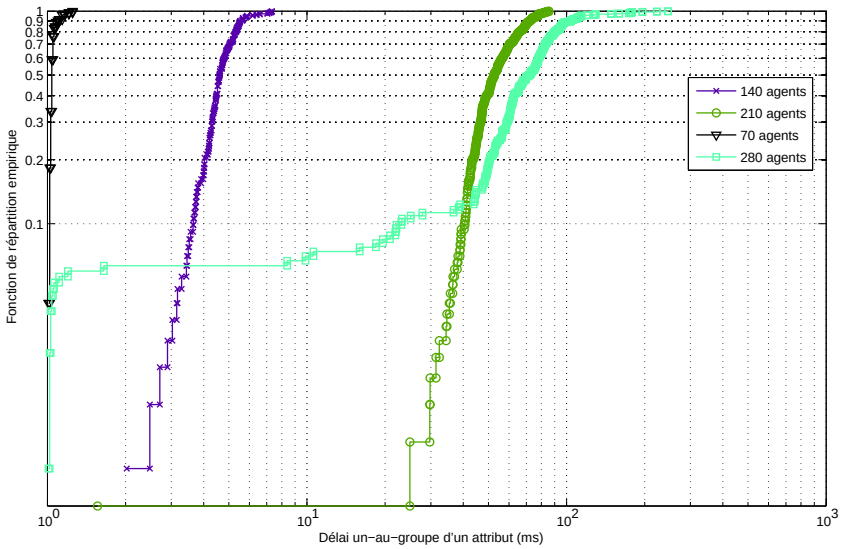


FIG. 8.7 – Relation entre les délais un-à-un d'un attribut accédé depuis un groupe de 140 agents et les délais moyens un-au-groupe.

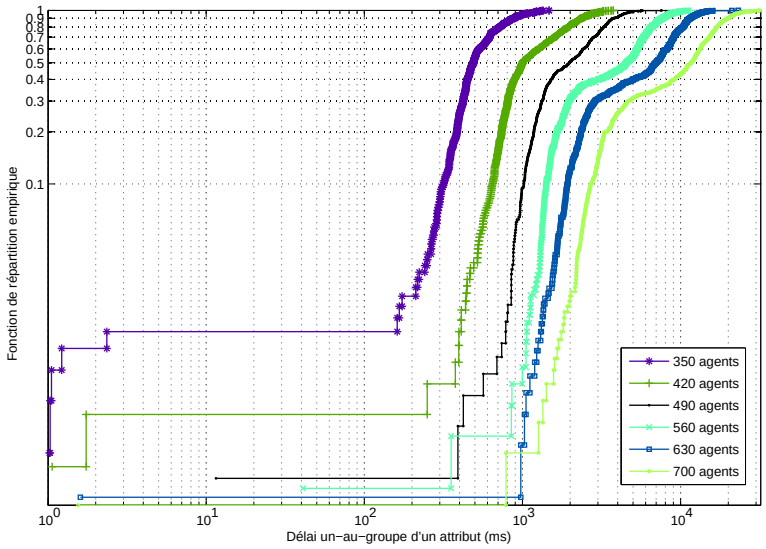
d'après le tableau 8.5 que la distribution approchant les délais un-au-groupe d'un attribut dépend de la taille de groupe. Dans un groupe de faible taille inférieure ou égale à 280 agents, ces délais suivent approximativement une distribution normale puisque les délais sont symétriques centrés autour de la moyenne. Dans le cas d'une taille de 70 agents, la distribution est exponentielle puisque la moyenne est égale à l'écart type comme l'indique le tableau 8.4. À partir d'une taille de groupe de 350 agents, la distribution empirique approche de plus en plus une distribution de Weibull. Cela indique que la distribution devient de plus en plus asymétrique et à queue lourde. Cela est dû au fait que les délais que subit un attribut depuis un agent sont différents de ceux d'un autre agent. Ainsi, les opérations transportant les valeurs d'un attribut depuis un groupe d'agents ne subissent pas les mêmes délais. En effet, le gestionnaire sature au delà de cette taille de 350 agents, et les lois individuelles de probabilités des délais un-à-un perdent la caractéristique d'être identiquement distribuées et indépendantes. Cela explique l'asymétrie des distributions de délai un-au-groupe lorsque les tailles de groupe sont supérieures à 350 agents.

Environnement à latence réseau moyenne

Les délais présentés précédemment sont obtenus dans un environnement clos à très faible latence. Nous avons étendu nos tests sur un environnement multi-sites où les agents sont distribués sur 4 clusters différents. La latence réseaux moyenne entre les différents site est de l'ordre de 17 ms. Ces clusters font partie du projet Grid5000 et sont ceux de Nancy, Nice, Bordeaux et Toulouse. Le gestionnaire est déployé à Bordeaux. Les agents sont déployés sur tous les sites y compris celui de Bordeaux. Nous avons varié le nombre d'agents déployés entre 6 à 60 agents. D'après le tableau 8.6, les statistiques de délai un-au-groupe sont quasiment stables en augmentant la taille de groupe d'agents. Les distributions des délais un-à-un que subit un attribut accédé depuis les différents agents de groupe suivent approximativement les distributions normale et Weibull. Les valeurs d'adéquations pour les deux distributions sont très proches. Les délais un-au-groupe basés sur les moyennes de délais un-à-un de chacun d'agents suivent plutôt

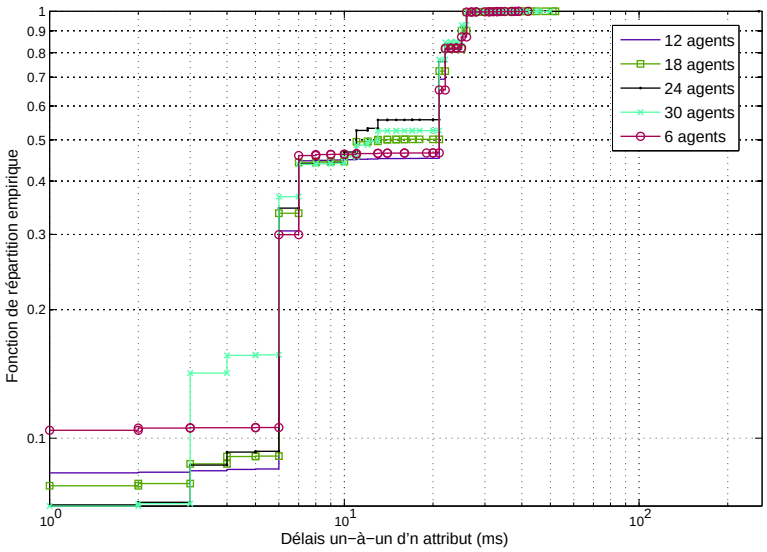


(a)

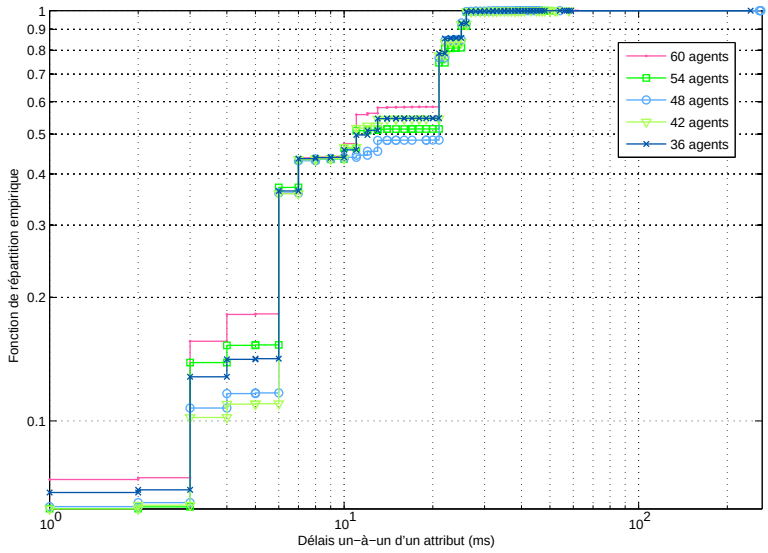


(b)

FIG. 8.8 – Distributions empiriques des délais un-au-groupe moyens d'un attribut accédé depuis un groupe d'agents déployés sur un cluster.



(a)



(b)

FIG. 8.9 – Distributions empiriques de délais un-au-groupe d'un attribut accédé depuis un groupe d'agents dans un environnement multi-sites (*Grid5000*).

Taille de groupe d'agents	Distribution de délais un-au-groupe	Coefficient d'adéquation λ^2
70	Normal, Exponentiel	0.33,1.6
140	Normal, Weibull	0.0414, 0.19
210	Gamma, Normal, Weibull	0.03, 0.04, 0.07
280	Normal, Weibull	0.21, 0.37
350	Gamma, Weibull, Normal	0.15, 0.16, 0.27
420	Gamma, Weibull	0.21, 0.3
490	Gamma, Weibull	0.16, 0.2
560	Gamma, Weibull	0.42, 0.44
630	Weibull, Gamma	0.36, 0.37
700	Weibull, Gamma	0.21, 0.23

TAB. 8.5 – Distributions statistiques approchant les délais un-au-groupe d'un attribut accédé depuis un groupe d'agents de différentes tailles.

Taille de groupe	Statistiques de délai un-au-groupe d'un attribut (ms)					
	Moyenne	Écart type	Médiane	IQR	Min	Max
6	14.72	9.13	21	16	1	42
12	14.99	9.18	21	16	1	259
18	14.4	8.86	13	16	1	52
24	13.78	8.71	11	15	1	255
30	13.87	8.81	13	15	1	50
36	13.68	8.73	12	15	1	240
42	13.85	8.64	11	15	1	58
48	14.47	9.03	21	15	1	260
54	14.09	8.97	11	16	1	50
60	13.29	8.85	11	15	1	62

TAB. 8.6 – Statistiques de délais un-au-groupe d'un attribut accédé depuis un groupe d'agents de différentes tailles, distribués sur plusieurs sites de Grid5000.

une distribution uniforme. Les résultats obtenus pour cette expérimentation d'une supervision multi-sites avec une latence réseau moyenne de l'ordre de dizaine de milli-secondes, ne peuvent pas être généralisés puisque nous avons considéré qu'un nombre d'agents déployés limité (entre 6 et 60). En revanche, on s'aperçoit que ces latences amortissent les délais un-à-un des attributs surveillés par le gestionnaire. Cela évite l'écroulement de ce dernier à cause de faible latence du réseau où les réponses aux requêtes de supervision reviennent rapidement au gestionnaire. En effet, une latence réseau moyenne entre les différents sites supervisés *dilue* les rafales (*burst*) des réponses qui arrivent sur le gestionnaire. Cette technique d'introduction d'une latence dans les fonctions de gestion pour optimiser leurs performances a été proposée par David Breight et al dans [49] et par Ehab Al Shaer et al dans [44].

À travers cette analyse, on s'aperçoit qu'il est possible d'inférer des délais un-au-groupe depuis des délais de type un-à-un dans le cas où la taille de groupe d'agents est faible de l'ordre de 280 agents dans notre cas. En revanche, si le nombre d'agents augmente considérablement les délais un-au-groupe deviennent plus importants que ceux de un-au-un. Cela montre que les résultats de mesure de performances d'un gestionnaire et d'un seul agent n'aident pas à la compréhension complète de la performance globale de cette approche dans des environnements à grande échelle.

8.4 Estimation de la précision temporelle

La métrique que nous analysons dans cette section est la précision temporelle. Elle est en lien directe avec les délais des attributs collectés par le biais d'une opération. Cette précision temporelle se présente sous la forme de la divergence de délais que subit l'attribut par rapport à un délai de tolérance (voir section 4.2.4 du chapitre 4). Nous considérons 1 seconde comme délai de tolérance, noté τ. Ce choix repose sur le fait que les *threads* d'injection que nous avons utilisés sur le banc de mesure possède comme intervalle de scrutation de 1 seconde. Ainsi, cela nous permet entre autre d'identifier la limite du gestionnaire en terme de nombre de *threads* concurrents.

Dans la section 4.2.4, nous avons défini la précision temporelle $\theta(V_i)$ d'une variable V_i par une fonction binaire qui renvoie 1 si un singleton $D_j(V_i)$, d'un échantillon de N mesures des délais $\{D_j(V_i), j = 1..N\}$, est supérieur à τ sinon la fonction renvoie 0. Cette fonction de précision temporelle représente la proportion de l'échantillon des délais qui sont inférieurs au délai de tolérance. Statistiquement, cette proportion est calculée en utilisant la fonction de répartition empirique ($ECDF$) sur l'échantillon des délais $\{D_j(V_i)\}$ de la manière suivante :

$$\theta(V_i) = ECDF(\tau) \tag{8.2}$$

En effet, $\theta(V_i)$ est un indicateur de la précision temporelle de l'attribut collecté. Dans le cas d'un échantillon de mesure de la métrique un-au-groupe déduite à partir des délais un-à-un moyens de chacun de agents dans le groupe, la valeur de θ représente le nombre d'agents qui répondent avec un délai inférieur au délai de tolérance. Le tableau 8.7 présente les valeurs de la précision

Taille de groupe d'agents	Précision temporelle θ
70	1
140	1
210	1
280	1
350	0.95
420	0.49
490	0.09
560	0.01
630	0.01
700	0.01

TAB. 8.7 – Précision temporelle θ inférée de délais un-au-groupe d'un attribut accédé depuis un groupe d'agents de différentes tailles.

temporelle θ obtenue d'une manière statistique depuis la fonction empirique de répartition. On observe que cette précision commence à décroître au delà d'un groupe de taille 280 agents. Cette valeur limite de taille de groupe est la même que celle pour laquelle les délais un-au-groupe se rapprochent des délais un-à-un. C'est la limite deçà de laquelle les délais de différents agents sont semblables. Dans la section précédente, nous avons identifié la loi de Weibull comme loi à laquelle obéissent les délais un-au-groupe dans le cas d'un groupe d'agents de taille supérieure à 280 agents. Cette distribution nous permet d'estimer la précision temporelle d'une manière analytique en se reposant sur sa fonction de répartition présentée dans la section 1.5.4. La figure 8.10 représente la prédiction de la précision temporelle d'un attribut trouvé depuis un groupe de 420 agents en se basant sur ces délais un-au-groupe.

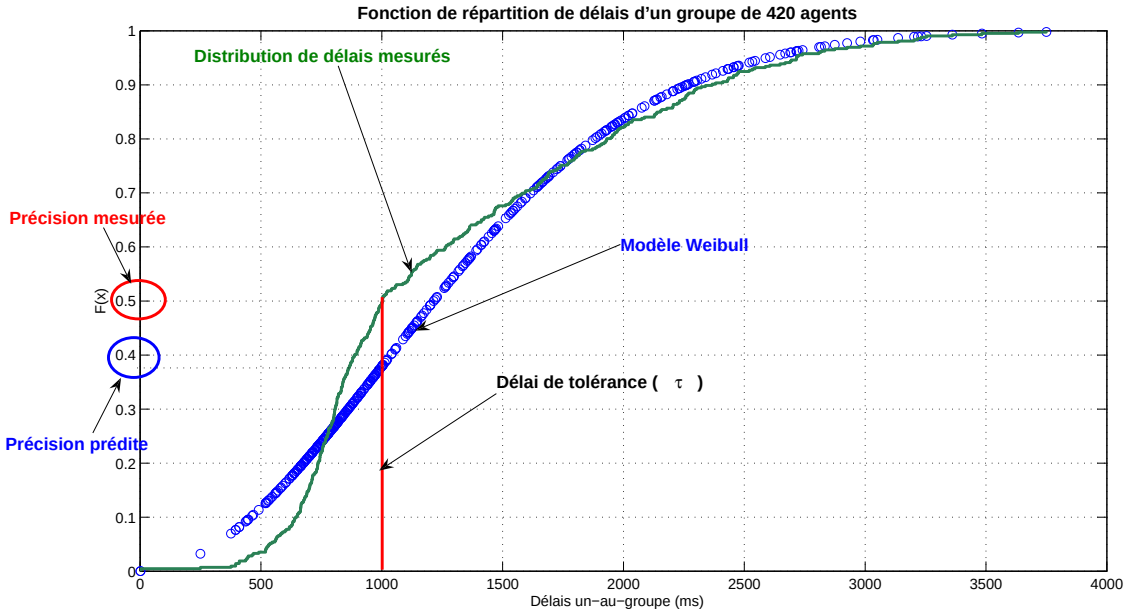


FIG. 8.10 – Estimation de la précision temporelle à partir d'un modèle Weibull de délais un-au-groupe d'un groupe d'agents de taille 420.

8.5 Effet des délais sur le comportement d'un algorithme de supervision

Dans la section 8.4, nous avons utilisé la distribution de Weibull comme moyen pour estimer la précision temporelle d'une variable accédée depuis un groupe d'agents. Dans la suite nous allons utiliser cette distribution qui approche les délais un-au-groupe pour comprendre l'effet de ces délais sur le comportement d'un algorithme de supervision.

8.5.1 Simulation de comportement des algorithmes de gestion

Dans la section 5.3.1 nous avons identifié certains comportements de valeurs des objets réels d'un système géré. Ces objets réels sont représentés sur l'agent par des variables de gestion observées depuis un gestionnaire par un algorithme de supervision. Pour mieux comprendre cette observation par l'algorithme de supervision des valeurs de ces variables, nous allons introduire des délais générés d'une manière synthétique à partir d'une distribution statistique parmi celles identifiées dans la section 8.3.4. Nous avons pris l'exemple d'un algorithme de supervision *SimpleRate*, proposé dans [124]. Son pseudo code est présenté par l'algorithme 2. Ce qui nous intéresse dans cet algorithme est l'opération $POLLALL(x_i)$ qui permet de scruter les valeurs d'une variable depuis un groupe de n agents. Le résultat de cette opération est une fonction d'agrégation f de l'ensemble des valeurs de la variable accédée depuis le groupe d'agents. Cette fonction peut être définie sous plusieurs formes : somme, minimum, maximum, moyenne. Dans notre cas d'étude nous considérons une fonction d'agrégation définie de la façon suivante : $f = \sum_{i=1}^{n} x_i$. D'après ce qui précède, nous avons identifié la distribution de Weibull comme la distribution approchant au mieux les délais un-au-groupe d'un attribut accédé depuis un groupe d'agents. En effet, on peut générer un échantillon synthétique de ces délais en s'appuyant sur les deux

Algorithme 2 Extrait de [124]. Pseudo code d'un algorithme de supervision.

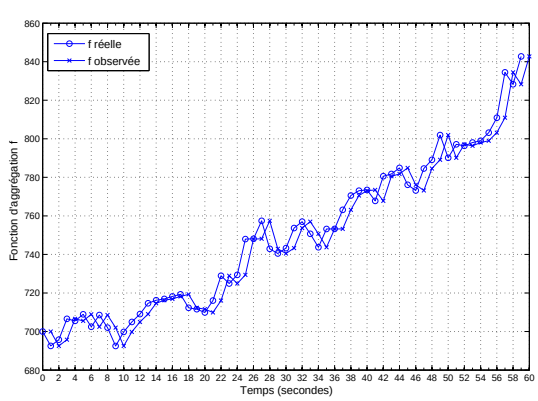
Require: $t_m \leftarrow 0\,;\ f = 0$
  while TRUE **do**
    if $t \geq t_m$ OR report RECEIVED **then**
      $f \leftarrow \text{POLLALL}(x_i)$
      if $f > T$ **then**
        report ALARM
      else
        $t_m \leftarrow t + \frac{T - \sum_{i=1}^{n} x_{it}}{\delta_n}$
      end if
    end if
  end while

paramètres de la distribution qui sont la forme et l'échelle, pour analyser le comportement de cet algorithme en présence de ces délais sous différentes tailles de groupe d'agents.

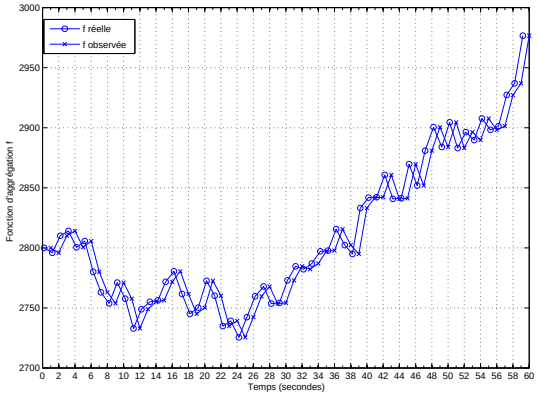
8.5.2 Résultats numériques

Nous avons développé des scripts Matlab pour simuler le comportement de l'algorithme 2 en introduisant des délais un-au-groupe générés d'une manière synthétique à partir d'une distribution de Weibull. Les paramètres de la distribution sont obtenus depuis les échantillons de mesures présentés dans la section 8.3.4. Ainsi la taille de groupe d'agents varie entre 70 et 700 avec un palier de 70. Nous avons considéré une variable de gestion avec changement de ses valeurs réelles sur l'agent selon une loi normale $N(0, 1)$, comme celle décrite dans la section 5.3.1. Ainsi, la fonction d'agrégation f calculée sur le gestionnaire est la somme de valeurs observées depuis les n agents. Chacun des agents généré synthétiquement une variable aléatoire suivant une loi normale. La période de mise à jour AUP de la variable sur chaque agent est de 1 seconde. Son intervalle de surveillance MPP sur le gestionnaire est aussi de 1 seconde. La figure 8.11 représente le comportement temporel de la fonction d'agrégation d'une variable obtenue depuis un groupe d'agents de différentes tailles. Pour chaque taille de groupe d'agents, nous présentons les valeurs réelles de la fonction d'agrégation calculées sur les agents, ainsi que les valeurs observées de la fonction d'agrégation sur le gestionnaire. Ces valeurs sont observées après l'introduction d'un délai un-au-groupe qui représente le maximum d'un échantillon de délais moyens un-à-un de différents agents. L'échantillon de délais un-au-groupe est généré à partir d'une distribution de Weibull. On aperçoit que tant que la taille de groupe d'agents est inférieure à 280, les deux fonctions d'agrégations (calculée et observée) ont la même allure. Il existe juste un décalage temporel de l'ordre de 1 seconde entre elles qui représente l'intervalle de scrutation. Au-delà de cette taille de 280 agents, la fonction d'agrégation réelle et celle observée sur un gestionnaire commencent à être différentes. D'ailleurs, le décalage temporel entre les deux fonctions est proche de 30 secondes pour un groupe de 700 agents.

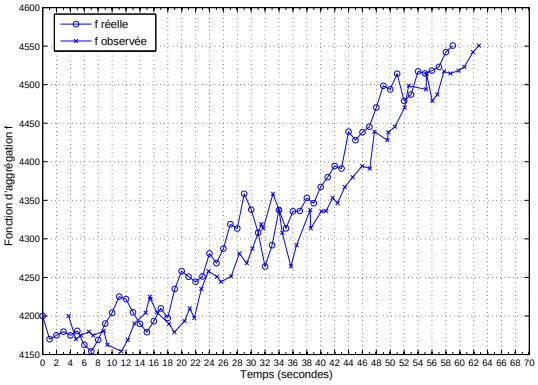
Nous observons aussi que la fonction d'agrégation observée subit une distorsion temporelle due au délai important que subissent certaines valeurs de variables accédées sur les agents. Cette distorsion est nettement visible dans le cas d'un groupe de 700 agents. Dans ce cas, la fonction agrégée observée est une aberration de la fonction réelle. Les conséquences de cette aberration sont potentiellement dramatiques pour l'identification des problèmes en s'appuyant sur une telle fonction d'agrégation.



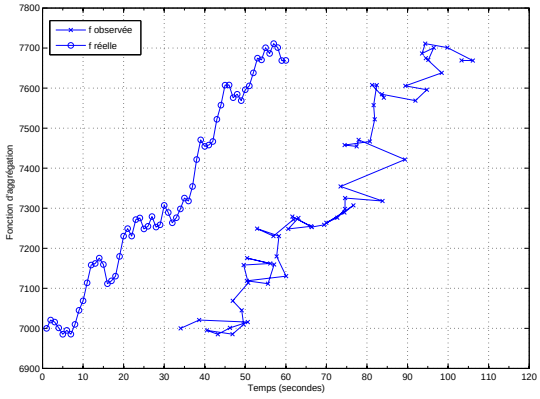
(a) 70 agents



(b) 280 agents



(c) 350 agents



(d) 700 agents

FIG. 8.11 – Effet des délais un-au-groupe sur le comportement temporel d'une fonction d'agrégation d'un algorithme de supervision basé sur un mécanisme de scrutation.

8.6 Synthèse

Dans ce chapitre nous avons caractérisé les délais que subissent les attributs d'une gestion basée sur le *framework* JMX. Ces délais sont mesurés dans différents environnements et sous différents scénarios. Tout d'abord, nous avons analysé les délais dans un environnement à faible latence réseau. Les délais un-à-un d'un attribut dans ces environnements suivent approximativement des distributions LogNormal et Weibull. Dans un scénario de tests synthétiques, ces délais sont stables et ne dépassent pas l'intervalle de scrutation. Dans un environnement plus réaliste où une application gérée est couplée à un agent, ces délais gardent la même distribution mais augmentent d'ordre de grandeur. Les délais un-au-groupe gardent le même ordre de grandeur que les délais un-à-un jusqu'à une certaine taille de groupe d'agents. Au-delà de cette taille ils deviennent plus importants à cause de l'instabilité et de l'augmentation de délais un-à-un de certains agents. Dans un environnement à latence moyenne comme une grille multi-sites, nous avons observé que les délais sont plus stables et sont uniformément distribués. Cela est dû à la latence réseau qui amortit les délais que subissent les messages de réponses de gestion avant d'atteindre le gestionnaire, ce qui évite un goulot d'étranglement à son niveau. Ce phénomène de retard des opérations a été analysé par plusieurs travaux afin d'optimiser le coût de gestion en terme de trafic introduit sur le réseau. Dans la dernière partie de ce chapitre, nous avons montré l'effet des délais dans un contexte large échelle sur les valeurs d'une fonction d'agrégation observée depuis un ensemble de variables accédées sur un groupe d'agents. Nous avons observé que ces délais engendrent une aberration entre la fonction d'agrégation réelle et celle observée. Ainsi, il est indispensable de contrôler ses délais en s'appuyant sur des techniques d'optimisation pour aboutir à des algorithmes de supervision plus efficaces en terme de précision temporelle.

Chapitre 9

Conclusions et perspectives

Nous nous sommes intéressé dans cette thèse à la problématique des performances des approches de gestion de réseaux et de services. Cette problématique est introduite à cause des défis auxquels sont confrontées ces approches en terme des dimensions, de l'intégration de leur plan dans le plan fonctionnel et de la précision qu'elles doivent offrir afin de garantir son contrôle sur les systèmes gérés. Comme nous l'avons vu dans la première partie de ce manuscrit, soulever ces verrous passe par la compréhension et la maîtrise de l'efficacité des activités de gestion. Malgré l'existence de plusieurs travaux portant sur la performance de la gestion, ces travaux manquent de méthodologie et de métriques communes pour quantifier l'efficacité de la gestion d'une manière précise et claire. En effet, le travail effectué durant cette thèse n'a donc pas consisté à l'amélioration des approches de gestion existantes mais bien à proposer une méthodologie de mesure de leurs performances qui permette une meilleure compréhension afin d'initier des travaux d'évaluation reposant sur un cadre commun de mesures.

Nos travaux de recherches ont porté sur deux axes principaux : (1) les métriques de performances et leur méthodologie de mesure et (2) la quantification de l'efficacité d'une approche de gestion en terme de son passage à l'échelle, son incidence sur le système géré et la caractérisation des délais de ses opérations.

9.1 Résumé des contributions

9.1.1 Métriques de performances pour la gestion

Le premier axe de notre travail a consisté à définir un ensemble de métriques pour capturer les performances d'une approche de gestion. Ces métriques doivent suivent les standards IPPM afin qu'elles soient déterministes. Nous avons identifié trois grandes familles de métriques : des métriques de rapidité, des métriques de coût et des métriques de qualité. La première famille mesure la capacité de la gestion en terme de son débit en variables par seconde et des délais que subissent ses variables par les biais des opérations de gestion. La deuxième famille mesure le coût introduit par une activité de gestion en termes de coût de traitement imposé aux niveaux des entités de gestion, de coût de communication introduit sur le réseau et de coût de stockage au niveau de la mémoire. Ces différents coûts sont relatifs à une variable de gestion spécifique. Enfin la troisième famille regroupe des métriques de qualité des activités de gestion. Ces métriques mesurent notamment la précision des données de gestion en terme de l'exactitude de la valeur de la variable de gestion par rapport à celle de l'objet réel qu'elle représente, la précision temporelle en terme de la déviation de ses délais par rapport à un délai de tolérance au-delà de lequel sa valeur devient obsolète et inutile.

9.1.2 Banc de mesure dédié aux approches de gestion

Après la définition des métriques pour la mesure de la performance d'une approche de gestion, nous avons présenté une méthodologie pour leur mesure. La mise en œuvre d'une méthodologie nécessite la conception et le développement d'un banc de mesure dédié à la gestion. Nous avons donc développé un banc de mesure générique et flexible reposant sur le modèle gestionnaire-agent largement utilisé dans les approches de gestion traditionnelles comme SNMP, NetConf ou JMX. Notre banc de mesure est conçu pour qu'il soit extensible à différentes technologie des gestion et la réalisation de campagnes de mesure large échelle en s'appuyant sur une plate-forme physique. En effet, nous avons introduit la notion de pratique de gestion pour décomposer une approche de gestion en sous-ensembles de paramètres à valeurs fixées au cours d'une mesure et en variant un de ses facteurs pour voir son effet sur sa performance. Cette décomposition nous permet d'assurer la continuité, la représentativité et la reproductibilité des tests de mesure comme il est recommandé dans la plupart des documents standardisés d'évaluation de performances. Nous avons développé une instance dédiée à la technologie JMX pour la gestion des applications Java. Cette instance nous a permis de mener des expérimentations de mesure de ses performances dans différents environnements réseau et sous différents scénarios, notamment les deux scénarios types : 1 gestionnaire, vers 1 agent et 1 gestionnaire vers plusieurs d'agents.

9.1.3 Efficacité d'une approche de gestion

Les trois familles de métriques qui nous avons définies sont de nature primaires puisqu'elles sont mesurables directement sur le banc de mesure. Néanmoins, la quantification de l'efficacité d'une approche de gestion requiert la mise en relation de ses métriques afin de capturer le *trade-off* qui existe entre elles. Ainsi, nous avons défini l'efficacité d'une approche de gestion comme étant le rapport des métriques de rapidité et des métriques de qualité aux métriques de coût. Ceci permet de capturer les différents aspects de la performance d'une pratique de gestion sous une métrique unique.

Passage à l'échelle de la gestion

En s'appuyant sur cette métrique d'efficacité, nous avons proposé une nouvelle métrique qui capture le degré de passage à l'échelle d'une approche de gestion. En effet, ce degré de passage à l'échelle est défini par le rapport des efficacités sous différentes valeur d'un facteur d'échelle spécifié. Ce rapport capture la dégradation de cette efficacité en augmentant la valeur du facteur d'échelle. En identifiant un point de rupture de ce degré, il devient indispensable d'introduire un facteur d'amélioration pour stabiliser ce degré ou l'augmenter. Évidement ces facteurs d'amélioration sont identifiés d'une manière intuitive ou quantitative en se basant sur la notion de pratique de gestion et notre banc de mesure. Nous avons illustré cette métrique sur une approche centralisée de gestion dans un scénario un gestionnaire et un groupe d'agents. Grâce à la métrique de passage à l'échelle, nous avons montré que le délai que subit une variable de gestion lors de transfert de son valeur entre deux entités est un facteur de limitation de degré de passage à l'échelle de l'approche de gestion centralisée. En effet, nous avons validé des formulations analytiques proposées par Chen et al dans [108] et nous avons montré que ces formulations sous estiment le degré de passage à l'échelle de cette approche.

Incidence de la gestion

L'efficacité d'une approche de gestion ne signifie pas seulement sa performance propre mais aussi son incidence sur le rendement du système géré et sur la performance de ses fonctions primaires. Nous avons proposé une métrique unique d'incidence qui capture la dégradation du rendement du système géré en fonction des activités de gestion auxquelles il est soumis. Dans un premier temps, nous avons présenté l'incidence de l'intensité d'une supervision basée sur JMX sur la performance d'un serveur applicatif. Cette expérimentation nous a montré qu'une intensité de supervision non contrôlée engendre un dégradation conséquente de performances perçues par les utilisateurs de serveur géré. Cette dégradation est due essentiellement à la forte intégration du modèle d'agent JMX au sein de serveur applicatif où il représente le noyau du système géré. Nous avons conclu que le modèle d'intégration de l'agent par rapport au plan fonctionnel à un effet important sur la performance du plan de gestion ainsi que celle du plan fonctionnel. Nous avons étendu notre banc de mesure avec des agents qui s'intègrent sous les trois modèles d'intégration que sont le modèle démon, le modèle composant et le modèle pilote. Nous avons implanté ces trois différents modèles pour superviser un serveur web. Nos mesures ont montré que le modèle démon affecte la performance du plan de gestion essentiellement en terme de débit et de délais, plus que le plan fonctionnel. Les deux autres modèles affectent eux plus la performance du plan fonctionnel. Ceci est dû à leur plus forte intégration dans le système géré où ils sont en concurrence avec les requêtes utilisateurs.

Il faut noter que la métrique d'incidence que nous avons définie est générique et elle permet de mesurer l'impact d'une fonction secondaire sur une fonction primaire (par exemple, l'impact de la sécurité sur un système fonctionnel, l'impact d'une attaque sur un système, etc). Autrement, cette métrique met en relation les travaux portant sur la performance dans deux communautés réseaux différentes qui sont la communauté réseaux et services qui possède ses propres modèles de performances et la communauté de leurs gestion qui possède ses modèles de performances de systèmes de gestion. En effet, la métrique d'incidence permet de combiner ces différents modèles en une seule métrique pour identifier l'effet de la gestion en terme de performances sur les services gérés.

Le tableau 9.1 récapitules les différentes métriques que nous proposons pour évaluer les performances de la gestion.

	Familles	Métriques primaires	Métriques secondaires
Métriques	Rapidité	– Débit – Délais – Pertes	– Efficacité – Degré de passage à l'échelle – Métrique d'incidence
	Coût	– Traitement – Communication – Stockage	
	Qualité	– Sensibilité – Exactitude – Précision temporelle	

TAB. 9.1 – Métriques primaires et secondaires pour évaluer les performances de la gestion.

Délais de la gestion

Le dernier axe de notre travail s'intéressait à la caractérisation de délais que subit une variable par le biais d'une opération de gestion. Cette caractérisation repose sur l'analyse statistique

des mesures obtenues pour les deux métriques de délais que nous avons identifiées. La première métrique mesure les délais que subit une variable entre un seul gestionnaire et un seul agent. Dans un scénario avec un agent synthétique où aucune application gérée ne lui est couplée, ces délais sont très stables et ne dépassent pas une seconde, même sous des fortes intensités de surveillance. Ces délais suivent approximativement des distributions classiques de types LogNormal et Weibull. En couplant une application gérée à l'agent, la nature des distributions statistiques ne change pas mais les délais deviennent plus importants. Dans la suite, nous nous sommes intéressés aux délais de type un-au-groupe qui représentent les délais qui subit une variable accédée depuis un groupe d'agents. Ces délais sont très proches de ceux un-à-un tant que le groupe d'agents est de faible taille. Dès que la taille devient plus importante, les délais un-au-groupe deviennent plus importants à cause de l'asymétrie entre les délais un-à-un de différents agents. Néanmoins, la distribution statistique de ces délais s'approche de plus en plus d'une Weibull connue comme une distribution à queue lourde. Ces différents mesures sont effectuées dans un réseau à très faible latence. Dans un réseau à latence moyenne comme celui de Grid5000, nous avons observé que les délais un-au-groupe sont plus stables et suivent approximativement une distribution uniforme. Il faut noter que cette caractérisation statistique est de très grande utilité pour de travaux futurs reposant sur des techniques de simulations.

9.2 Perspectives

Les travaux de recherche réalisés au cours de cette thèse ouvrent différentes perspectives scientifiques à moyen et long terme. Ils contribuent à une meilleure compréhension de la performance de la gestion de réseaux et de services. Ils ouvrent également la voie, de manière plus générale, à de nouveaux axes de recherche dans la communauté de gestion.

9.2.1 Construire un cadre complet de métriques

Dans cette thèse nous avons proposé différentes métriques pour la mesure de performances de plan de gestion. Ces métriques se présentent comme les premières en la matière. Une première perspective de recherche vise à construire un cadre commun et complet de métriques de performances de la supervision. Il permettra de comparer les différentes approches de gestion sur une même base de référence. Une seconde perspective porte plus particulièrement sur le raffinement des métriques que nous avons proposées dans le cadre de la gestion de configuration. En effet, nos métriques, malgré leur généricité ont été validées uniquement sur une seule fonction de surveillance. Ainsi, il sera intéressant de les valider sur une fonction de contrôle comme la configuration. Au niveau temporel, une fonction de configuration est moins récurrente que celle de supervision. Ainsi, son intensité de demande est plus faible. En revanche, ses activités sont plus critiques que celles de la fonction de supervision. Une action de configuration est généralement requise pour ajuster une variable de système. Si le système de gestion ne parvient pas à effectuer une telle action rapidement, le résultat peut être fatidique pour le système géré. Il est en effet intéressant de raffiner les métriques de qualité pour qualifier ces aspects critiques de la fonction de la gestion.

9.2.2 Étendre le banc de mesure aux autres protocoles de gestion

Nous avons par ailleurs conçu un banc de mesure dédié à l'injection de la charge et la génération de trafic de gestion. Actuellement, seule l'instance JMX est développée sur ce banc. Grâce à sa flexibilité et sa modularité où chaque technologie de gestion possède son propre module de

déploiement de ses processus, ce banc peut naturellement être étendu aux autres protocoles de gestion notamment SNMP où plusieurs de nos partenaires dans le projet Européen *Emanics*[85] collectent de traces trafic afin de dégager les pratiques de gestion SNMP les plus couramment utilisées. Une perspective consistera à effectuer des mesures de performances de JMX et SNMP sur notre banc de mesure avec les paramètres communs de pratiques observées dans un réseau réel déduites à partir de ces traces. Une autre perspective de notre banc de mesure porte sur l'instanciation d'autres services gérés afin d'identifier l'impact de la supervision sur leurs plans fonctionnel. Parmi ces services on peut citer la voix sur IP qui est un service critique exigeant en terme de performances et de qualité. Ainsi, il sera intéressant d'identifier à quel point une intensité d'une charge de supervision peut dégrader ce service.

9.2.3 Validation par d'autres techniques d'évaluation

Dans ce travail nous avons utilisé une technique de mesure. Ce choix est dû au manque des modèles de performances pour la gestion à partir desquels nous avons pu initier nos travaux. En revanche, à la suite de ce travail, nous avons acquis une meilleure compréhension de la performance de la gestion qui nous permettra d'effectuer des simulations pour vérifier les résultats obtenus, notamment ceux portant sur la caractérisation des délais. Par ailleurs, nous avons entamé des modélisations à base de files d'attentes et des chaînes de Markov portant sur les activités de gestion et leurs liaisons avec des modèles de disponibilité d'un service afin d'identifier le rôle que joue ces activités pour maintenir cette disponibilité.

[85] http ://www.emanics.org

Annexe A

Compléments sur les méthodologies existantes d'évaluation de performances des approches de gestion

Dans cette annexe, nous présentons un tableau comparatif des méthodologies existantes dans la littérature portant sur l'évaluation de performances des approches de gestion. Ces méthodologies sont déduites des travaux que nous avons recensé dans ce domaine sur une dizaine d'années entre 1996 et 2006. Dans notre recherche bibliographique, nous nous sommes basé sur deux bases électroniques : *ACM* [86] et *IEEE* [87]. Ces études sont celles publiées dans des conférences consacrées à la gestion de réseaux et de services comme : *IM*, *NOMS*, *DSOM*, *MMNS* ou *IPOM* ; ou des conférences de la communauté réseau comme *SIGCOMM*, *INFOCOM*, *GLOBECOM* ou *ICC* qui incluent généralement des sessions dédiées à la gestion. D'autres études ont été publiées dans des journaux, notamment *TNSM*, *JNSM*, *IEEE Communication Magazine series on Network and Service Management*, *IEEE Journal On Selected Areas In Communications* and A*CM transactions on Networking*.

[86] http ://portal.acm.org
[87] http ://ieeexplorer.ieee.org

TAB. A.1: Éléments de base retenus par les études de performances des approches de gestion de réseaux

Paramètres de base				
Étude	Technique	Approche de gestion	Hétérogeneité	Métriques de performances
Katchabaw et al. [72]	mesure	gestion OSI : gestionnaire/agent	non	– métriques de réponse : temps de réponse d'une requête, temps d'un cycle (start-to-completion), temps d'initialisation. – métriques de productivité : volume du travail par unité de temps. – métriques d'utilisation : ressources consommées. – métriques de pannes et d'erreurs : taux d'erreur, taux de panne.
Sheikh et al. [144]	analytique (LQM) et simulation	gestion basée sur CORBA	non	– temps de réponse (sec) – fraction de l'utilisation réseau
Strasser and Schwehm [145]	analytique et mesure	gestion basée sur les agents mobiles (MOLE) avec une faible mobilité	non	– trafic généré (kilo octets) – temps d'exécution d'un appel RPC (sec)
Luderer et al. [135]	mesure	gestion basée sur des agents intelligents Java comparé à SNMPv2	agents Java, agents CMU-SNMP	temps de réponse mesuré au niveau de gestionnaire
Baldi and Picco [120]	analytique et mesure	gestion basée sur des agents avec faible mobilité comparé à SNMPv1	non	trafic de gestion généré
Fuggetta et al. [127]	analytique	gestion basée sur des agents avec faible et forte mobilité comparé à SNMP	non	trafic généré
Zapf et al. [150]	mesure	gestion basée sur des agents avec faible mobilité : AMETAS comparé à SNMPv1	non	trafic de gestion
Liotta et al. [134]	analytique	gestion basée sur des agents avec faible mobilité	non	– trafic de gestion généré – délais de supervision : nombre d'unités de temps nécessaires pour réaliser une opération de supervision – passage à l'échelle en fonction d'entités gérées, fréquence de scrutation et diamètre de réseau géré
Subramanyan et al. [146]	mesure	gestion hiérarchique basée sur SNMP	non	– délai de supervision – consommation CPU – bande passante utilisée
Chen and Liu [108]	analytique	gestion traditionnelle : centralisée, hiérarchique statique, mobilité forte, mobilité faible	non	– temps moyen de détection de changement d'une variable – trafic généré (bits) – temps cpu – consommation mémoire (bits)

Suite du tableau page suivante...

Étude	Technique	Approche de gestion	Hétérogeneité	Métriques de performances
Lim and Stadler [133]	analytique et mesure	gestion basée sur les patterns de navigation	non	– complexité de trafic : volume de trafic introduit sur le réseau pour réaliser une opération de gestion – complexité de temps : temps nécessaire pour réaliser une opération de gestion
Pattinson [139]	analytique et simulation	gestion basée sur SNMPv1	non	– trafic généré – temps de réponse
Rubinstein et al. [143]	mesure et simulation	gestion basée sur des agents avec faible mobilité (MOLE) comparée au SNMPv1	adventNet SNMPv1 version 2.2 et linux UCD-SNMP	– bande passante moyenne consommée par le gestionnaire – temps de réponse moyen pour récupérer la valeur d'une variable depuis 1 à 250 agents
Bohoris et al. [107]	mesure	gestion basée sur des agents avec faible mobilité (Grasshopper) reposant sur CORBA comparée au SNMPv1	AdventNet SNMPv1 version 2.0	– trafic de gestion généré au niveau TCP – temps de réponse d'aller retour – consommation mémoire
Gu and Marshall [130]	mesure	gestion basée sur CORBA comparée au SNMPv1/v2	UCD-SNMP et plateforme TAO pour CORBA	– volume de trafic de gestion – temps de réponse
Neisse et al. [138]	mesure	gestion basée sur les services web comparée au SNMPv1	non	volume de trafic de gestion
Pavlou et al. [140]	mesure	gestion basée sur les services web comparée au SNMPv1/v2 et CORBA	– 3 platesformes de service web : WASP, AXIS, gSOAP – 2 platesformes SNMP : NET-SNMP et AdventNet – plateforme CORBA : orbacus	– volume de trafic de gestion – temps de réponse applicatif mesuré sur le gestionnaire
Pras et al. [142]	mesure et analytique	gestion basée sur les services web comparée au SNMPv2	Net-SNMP et plateforme gSOAP pour les web services	– utilisation réseau au niveau IP – temps CPU pour codage et décodage XML et BER (ms) – temps d'aller-retour d'une requête/réponse – utilisation mémoire
de Lima et al. [123]	mesure et analytique	gestion basée sur les services web comparée au SNMPv1,v2c avec un mode de notification	Net-SNMP et gSOAP	– utilisation réseau au niveau IP – temps pour délivrer une notification mesurée au niveau réseau

Fin du tableau

Tab. A.2: Facteurs de performances retenus par les études d'évaluation de performances des approches de gestion de réseaux

Facteurs de performances								
Étude	Scénario	Objets de gestion		Workload		Communication	fonction de gestion	mode d'intéraction
		#objets	#objets/opération	fréquence	type d'opération			
Katchabaw et al. [72]	1 gestionnaire-12 agents	1 objet par processus géré	N/A	N/A	singuliaire	2 LAN	surveilance	scrutation
Sheikh et al. [144]	1-5 gestionnaires, 1-3 agents, 2 domaines	5000 à 50000 objets	N/A	N/A	opérations asynchrones	WAN avec une latence de 20 ms	surveillance	scrutation
Strasser and Schwehm [145]	1 gestionnaire, 5 agents, 2 domaines	N/A	N/A	N/A	singuliaire	LAN Ethernet : 10 Mbits/s, WAN entre USA et allemagne	surveillance	scrutation
Luderer et al. [135]	1 gestionnaire, 4 agents	system et ip MIB	1 OID/requête	N/A	singuliaire	LAN Ethernet : 10 Mbits/s	surveillance	scrutation
Baldi and Picco [120]	1 gestionnaire, 50 agents*30 interfaces	5 objets du if-MIB pour calculer le niveau de la charge d'une interface	5 OID/requête	N/A	singuliaire	LAN Ethernet, gestion à distance d'un LAN, interconenction de L LANs	surveillance	scrutation
Fuggetta et al. [127]	1 gestionnaire, N agents	charge réseau sur chaque interface	1 OID/requête	N/A	singuliaire	réseau uniforme avec une latence constante	surveillance	scrutation
Zapf et al. [150]	1 gestionnaire, 100 agents	4 variables du MIB HP-UNIX	1 OID/requête	5 secondes	singuliaire : Get-request	10 Mbits Ethernet	surveillance	scrutation
Liotta et al. [134]	1 gestionnaire, N agents	N variables	N/A	1 à 5 requêtes/s	singuliaire	réseau uniforme	surveillance	scrutation
Subramanyan et al. [146]	1 gestionnaie, 2 à 64 sous gestionnaires, 1024 agents (128 machines)	une variable de la MIB : temps CPU consommé par un processus	1 OID/requêtes	2 à 3 secondes	singuliaire	LAN Ethernet	surveillance	scrutation
Chen and Liu [108]	1 gestionnaire, N agents	1 variable	1 variable/requête	dynamique	singuliaire	réseau non iniforme	surveillane et tracking	scrutation
Lim and Stadler [133]	1 gestionnaire, N agents	N/A	N/A	N/A	logique	réseau sous forme de graphe, LAN Ethernet 100 Mbits, Internet avec une latence de 120 ms et 34 hops	surveillance	scrutation, aggregation

Suite du tableau page suivante...

Étude	Scénario	Objets de gestion		Workload		Communication	fonction de gestion	mode d'intéraction
		#objets	#objets/opération	fréquence	type d'opération			
Pattinson [139]	1 gestionnaire, 120 agent (5 machines)	utilisation d'une interface réseau d'un agent, débit IP, paquets perdus et taux d'erreurs au niveau IP et TCP	1 OID/ requête	60-450 secondes	singuliaire	4 LAN Ethernet interconnecté avec un lien de 1.544 Mbits	surveillance, détection de fautes	scrutation
Rubinstein et al. [143]	1 gestionnaire, 250 agents	ifnErrors du MIB-II	1 OID/requête	N/A	singuliaire : Get	LAN Ethernet : 10 Mbits, topologie Internet : 2 Mbits	surveillance	scrutation
Bohoris et al. [107]	1 gestionnaire, 1 agent	1 table d'objets synthétique contenant 25, 50, 75 et 100 scalaire	1 table/requête	N/A	singuliaire	LAN Ethernet : 100 Mbits faiblement chargé	surveillance	scrutation
Gu and Marshall [130]	1 gestionnaire, 1 agent	1 objet, 3 objets, table BGP synthétique de 1 à 2000 lignes	1 OID/requête, 3 OID/requête, 25 OID/requêtes	N/A	singuliaire : Get, Get-Next, Get-Bulk	Ethernet LAN	surveilance	scrutation
Neisse et al. [138]	1 WS gestionnaire, 1 WS gateway, 1 SNMP agent	MIB synthétique d'une colonne et plusieurs lignes	1 OID/ requête	N/A	singuliaire : Get-Next et logique : GetWSColumn	LAN Ethernet	surveillance	scrutation
Pavlou et al. [140]	1 gestionnaire, 1 agent	MIB TCP (connections TCP) avec 40 lignes et 8 colonnes	1 OID/requête, 8 OID/requête, 1 MO/requête, N MO/requêtes	N/A	singuliaire : Get, Get-next, Get-Bulk	LAN Ethernet	surveillance	scrutation
Pras et al. [142]	1 gestionnaire, 23 agents	MIB-II (ifTable) avec 10 lignes (entre 1 et 270 objets)	1 OID/requête, 22, 66 OID/requête, N MO/requête	N/A	singuliaire et logique (getifTable)	LAN Ethernet	surveillance	scrutation
de Lima et al. [123]	1 gestionnaire/1 WS gateway/1 agent SNMP	linkdown traps (ifIndex, IfAdminStatus, ifOperSTatus), MIB de table de routage (ipCidrRouteDest, ipCidrRouteMask, ipCidrRouteIfIndex, ipCidrRounteNexTHop)	N objets/trap, $N \in [1:100:600]$, 1 object/trap et m objects/requête $m \in [5:25:200]$	N/A	singuliaire et logique (1 trap vers plusieurs requêtes SNMP)	Ethernet LAN 100Mbits	surveillance	notification

Fin du tableau

Glossaire

ADSL	Asymmetric Digital Subscriber Line	1
API	Application Programming Interface	35
ARM	Application Response Measurement	12
BGP	Border Gateway Protocol	45
CBR	Constant Bit rate	14
CORBA	Common Object Request Broker Architecture	51
CPU	Central Processing Unit	48
EJB	Entreprise Java Beans	117
EMANICS	European Network of Excellence for the Management of Internet Technologies and Complex Services	3
GC	Garbage Collector	87
HTTP	Hypertext Transfer Protocol	119
IETF	Internet Engieering Task Force	55
IP	Internet Protocol	1
IPPM	IP Performance Metrics	52
IPTV	Internet Protocol Television	1
IQR	Interquartile Range	19
J2EE	Java 2 Enterprise Edition	117
JBoss	JBoss Application Server	116
JDK	Java Developement Kit	94
JMX	Java Management eXtension	35
LAN	Local Area Network	12
MADYNES	Management of Dynamic Networks and Services	3
MBean	Managed Bean	35
MBeanServer	Serveur des MBean	35
MIB	Management Information Base	42
MLE	Maximum Likelihood Estimation	22
MVA	Mean Value Analysis	15

NS	Network Simulator	14
OID	Object Identifier Descriptor	42
OSI	Organisation internationale de normalisation	26
PERL	Practical Extraction and Report Language	101
QoS	Quality of Service	1
RAM	Random Access Memory	94
RDTSC	Real Time Stamp Counter	49
RMI	Remote Method Invocation	35
SAR	System Activity Reporter	86
SIP	Session Initiation Protocol	129
SLA	Service Level Agreement	91
SMI	Structure of Management Information	29
SNMP	Simple Network Management Protocol	35
SOA	Service Oriented Architecture	41
SOAP	Simple Object Access Protocol	2
SSH	Secure Shell	48
TCP	Transmission Control Protocol	44
TJWS	Tiny Java Web Server	119
TLS	Transport Layer Security	48
ToIP	Voix sur réseau IP	1
UDP	User Datagram Protocol	44
UMTS	Universal Mobile Telecommunications System	1
XML	eXtensible Markup Language	27

Publications relatives

Conférences internationales avec comité de lecture et actes

[1] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. On the impact of management on the performance of a managed system : a jmx-based management case study. In Jüregen Schönwalder and Joan Serrat, editors, *16th IFIP/IEEE Distributed Systems : Operations and Management, DSOM 2005, Barcelona, Spain*, volume LNCS 3775, pages 24–37. Springer-Verlag's Lecture Notes in Computer Science (LNCS), 24-26 October 2005.

[2] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. On delays in management frameworks : Metrics, models and analysis. In Declan O'Sullian Radu State, Sven van der Meer, editor, *17th IFIP/IEEE Distributed Systems : Operations and Management, DSOM 2006, Dublin, Ireland*, volume LNCS. Springer-Verlag's Lecture Notes in Computer Science (LNCS), 24-26 October 2006.

[3] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. Modeling and performance evaluation of the network and service management plane. In *AIMS, Inter-Domain Management, First International Conference on Autonomous Infrastructure, Management and Security, AIMS 2007, Oslo, Norway, June 21-22, 2007, Proceedings*, volume 4543 of *Lecture Notes in Computer Science*, pages 156–159. Springer, 2007.

[4] Abdelkader Lahmadi, Anca Ghitescu, Laurent Andrey, and Olivier Festor. On the impact of management instrumentation models on web server performance : A jmx case study. In *AIMS, Inter-Domain Management, First International Conference on Autonomous Infrastructure, Management and Security, AIMS 2007, Oslo, Norway, June 21-22, 2007, Proceedings*, volume 4543 of *Lecture Notes in Computer Science*, pages 1–12. Springer.

Conférences francophones avec comité de lecture et actes

[5] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. Performances et résistance au facteur déchelle d'un agent de supervision basé sur JMX : Méthodologie et premiers résultats. In *Colloque GRES 2005 : Gestion de REseaux et de Services, Luchon, France*, volume 6, pages 269–282, Mar 2005. ISBN : 2-9520326-5-3.

[6] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. évaluation du passage à l'échelle des systèmes de gestion : métriques et modèles. In *Colloque CFIP 2006 : Colloque Francophone sur l'Ingéniereie des Protocoles, Tozeur, Tunisie*, volume 6, Octobre 2006.

[7] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. Une approche de benchmarking des pratiques de gestion basées sur un middleware jmx pour les services et les applications.

In *Colloque NOTERE 2006 : Nouvelles Technologies de la Répartition, Toulouse, France*, volume 6, pages 145–151, Juin 2006.

Rapports de recherche

[8] Laurent Andrey, Abdelkader Lahmadi, and Julien Delove. A jmx benchmark. Technical report, Loria-INRIA Lorraine, Juin 2005. Livré dans le projet RNRT Amarillo (en cours d'enregistrement).

[9] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. Évaluation de performance des architectures de gestion de réseaux : état de l'art et perspectives. Rapport de recherche RR-5598, Loria-INRIA Lorraine, Juin 2005.

Bibliographie

La dernièe visite des *URL* présents dans cette bibiliographie date de septembre 2007.

Évaluation de performances et analyse statistique

[10] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. Jhon Wiley & Sons, New York, 1998. ISBN 978-0-471-56525-3.

[11] Anne-Louise Burness, Richard Titmuss, Caroline Lebre, Katie Brown, and Alan Brookland. Scalability evaluation of a distributed agent system. *Distributed Systems Engineering*, 6 (4) :129–134, 1999.

[12] Emmanuel Cecchet, Julie Marguerite, and Willy Zwaenepoel. Performance and scalability of ejb applications. *SIGPLAN Not.*, 37(11) :246–261, 2002. ISSN 0362-1340.

[13] Allen B. Downey. An empirical model of tcp performance. In *MASCOTS*, pages 45–54. IEEE Computer Society, 2005. ISBN 0-7695-2458-3.

[14] Kevin Fall and Kannan Varadhan. Ns notes and documentation. Technical report, The VINT Project, January 1999.

[15] Greg Franks and Murray Woodside. Performance of multi-level client-server systems with parallel service operations. In *WOSP98 : Proceedings of the 1st international workshop on Software and performance*, pages 120–130, Santa Fe, New Mexico, United States, 1998. ACM Press. ISBN 1-58113-060-0.

[16] Michele Garetto and Emilio Leonardi. Analysis of random mobility models with pde's. In *MobiHoc '06 : Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, pages 73–84, Florence, Italy, 2006. ACM Press. ISBN 1-59593-368-9.

[17] Ananth Grama, Anshul Gupta, and Vipin Kuma. Isoefficiency function : A scalability metric for parallel algorithms and architectures. *IEEE PDT : Parallel and Distributed Technology*, 1 :12–21, 1993.

[18] The Open Group. Application Response Measurement. http ://www.opengroup.org/tech/management/arm/.

[19] Marjan Hericko, Matjaz B. Juric, Ivan Rozman, Simon Beloglavec, and Ales Zivkovic. Object serialization analysis and comparison in java and .net. *SIGPLAN Notices*, 38(8) : 44–54, 2003. ISSN 0362-1340.

[20] David C. Hoaglin, Frederick Mosteller, and John W. Tukey. *Understanding Robust and Exploratory Data Analysis.* John Wiley & Sons, 1983.

[21] Raj Jain. *The art of Computer Systems Performance Analysis.* John Wiley & Sons, Inc, 1991. ISBN : 0-471-50336-3.

[22] P Jardin. Supporting scalability and flexibility in a distributed management platform. *Distributed Systems Engineering*, 3(2) :115–123, 1996.

[23] Prasad Jogalekar and Murray Woodside. Evaluating the scalability of distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 11(6) :589–603, june 2000.

[24] Leonard Kleinrock. *Queueing Systems Theory, Volume 1.* Wiley-Interscience, 1975. ISBN 0471491101.

[25] Kevin Lai and Mary Baker. Measuring bandwidth. In *IEEE INFOCOM*, volume 1, pages 235–245, New York, USA, March 1999.

[26] Lyndon Lee, Hyacinth S. Nwana, Divine T. Ndumu, and Philipe De Wilde. The stability, scalability and performance of multi-agent systems. *BT Technology Journal*, 16(3) :94–103, 1998. ISSN 1358-3948.

[27] Bruce Lindsay. A conversation with Bruce Lindsay. *ACM Queue*, 2(8), November 2004.

[28] Gilberto Flores Lucio, Marcos Paredes-Farrera, Emmanuel Jammeh, Martin Fleury, and Martin J. Reed. Opnet modeler and ns-2 : comparing the accuracy of network simulators for packet-level analysis using a network testbed. *WSEAS Transactions on Computers*, 2 (3), July 2003.

[29] SUN microsystems. The netbeans profiler project. http ://profiler.netbeans.org/.

[30] ObjectWeb. Rubis : Rice university bidding system. http ://rubis.objectweb.org, 2002.

[31] Krzysztof Pawlikowski, H-D. Joshua Jeong, and J-S. Ruth Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, pages 132–139, January 2002.

[32] The GNU project. The GNU profiler, gprof
. http ://www.gnu.org/software/binutils/manual/gprof-2.9.1/gprof.html.

[33] Vladimir Robutsov. Do you know your data size ? http ://www.javaworld.com/javaworld/javatips/jw-javatip130-p2.html, august 2002.

[34] Gilbert Saporta. *Probabilités, analyse des données et statistique.* Technip, Juin 2006. ISBN : 978-2-7108-0814-5.

[35] OPNET Technologies. The OPNET modeler. http ://www.opnet.com/products/modeler.

[36] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. An analytical model for multi-tier internet services and its applications. In *SIGMETRICS '05 : Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 291–302, Banff, Alberta, Canada, 2005. ACM Press. ISBN 1-59593-022-1.

[37] Pere vilà, José L.Marzo, Antonio Bueno, Eusebi Calle, and Lluis Fàbrega. Distributed network resource management using a multi-agent system : scalability evaluation. In *SPECTS (International Symposium on Performance Evaluation of Computer and Telecommunication Systems)*, San José California, July 2004.

[38] C.Murray Woodside, John E. Neilson, Dorina C. Petriu, and Shikharesh Majumdar. The stochastic rendez-vous network model for performance of synchronous client-server-like distributed software. *IEEE Transactions on Computer*, 44(1) :20–34, January 1995.

Gestion et supervision de réseaux et de services

[39] Hasina Abdu, Hanan Lutfiyya, and Michael A. Bauer. Monitoring overhead in distributed systems : visualization and estimation techniques. In *CASCON '96 : Proceedings of the 1996 conference of the Centre for Advanced Studies on Collaborative research*, page 1, Toronto, Ontario, Canada, 1996. IBM Press.

[40] Hasina Abdu, Hanan Lutfiyya, and Michael A. Bauer. A model for adaptive monitoring configurations. In *IEEE/IFIP International Symposium on Integrated Network Managemen (IM)*, pages 371–384, Boston, MA, USA, 1999. ISBN 0-7803-5748-5.

[41] Hasina Abdu, Hanan Lutfiyya, and Michael A. Bauer. A framework for determining efficient management configurations. *Computer Networks*, 46(4) :437–463, 2004.

[42] Arup Acharya, Xiping Wang, Charles Wright, Nilanjan Banerjee, and Bikram Sengupta. Real-time monitoring of sip infrastructure using message classification. In *MineNet '07 : Proceedings of the 3rd annual ACM workshop on Mining network data*, pages 45–50, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-792-6.

[43] Nazime Agoulmine and Omar Cherkaoui. *Pratique de la gestion de réseau : Solution de contrôle et de supervision d'équipements réseau pour les entreprises et les opérateurs télécoms*. Eyrolles, 1ère édition edition, Mars 2003. ISBN 2-212-11259-9.

[44] Ehab Al-Shaer. Active management framework for distributed multimedia systems. *Journal of Network Systems Management*, 8(1) :49–72, 2000. ISSN 1064-7570.

[45] Alessio Bechini, Pierfrancesco Foglia, and Cosimo Antonio Prete. Use of a corba/rmi gateway : characterization of communication overhead. In *Proceedings of the third international workshop on Software and performance*, pages 150–157, Rome, Italy, 2002. ACM Press. ISBN 1-58113-563-7.

[46] Robert Beverly. Rtg : A scalable snmp statistics architecture for service providers. In *Proceedings of the 6th Systems Administration Conference (LISA 2002)*, pages 167–174. USENIX, november 2002. ISBN 1-931971-03-X.

[47] Jean-Chrysotome Bolot. End-to-end packet delay and loss behavior in the internet. In *SIGCOMM : Conference proceedings on Communications architectures, protocols and applications*, pages 289–298, San Francisco, California, United States, 1993. ACM Press. ISBN 0-89791-619-0.

[48] David Breitgand. *On Generic Scalability Problems in Monitoring of Data Communication Networks*. PhD thesis, Institute of Computer Science ; The Hebrew University of Jerusalem, Israel, 2003.

[49] David Breitgand, Danny Raz, and Yuval Shavitt. The traveling miser problem. *IEEE/ACM Trans. Netw.*, 14(4) :711–724, 2006. ISSN 1063-6692.

[50] Aaron B. Brown and Joseph L. Hellerstein. An approach to benchmarking configuration complexity. In *EW11 : Proceedings of the 11th workshop on ACM SIGOPS European workshop : beyond the PC*, page 18, Leuven, Belgium, 2004. ACM Press.

[51] Adam Buble, Lubomír Bulej, and Petr Tuma. Corba benchmarking : A course with hidden obstacles. In *IPDPS '03 : Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 279.1, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1926-1.

[52] Prasad Calyam, Chang-Gun Lee, Phani Kumar Arava, Dima Krymskiy, and David Lee. Ontimemeasure : a scalable framework for scheduling active measurements. In *E2EMON Workshop on End-to-End Monitoring Techniques and Services, Nice, France*, pages 86–100, 15 May 2005.

[53] Jeffrey D. Case, Mark Fedor, Martin Lee Schoffstall, and James R. Davin. Simple Network Management Protocol (SNMP). RFC 1157 (Historic), may 1990.

[54] Jeffrey D. Case, Keith McCloghrie, Marshall Rose, and Steven Waldbusser. Introduction to Community-based SNMPv2. RFC 1901 (Historic), january 1996.

[55] Yan Chen, David Bindel, Hanhee Song, and Randy H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *SIGCOMM '04 : Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 55–66, Portland, Oregon, USA, 2004. ACM Press. ISBN 1-58113-862-8.

[56] Mathieu Colonna and Laurent Andrey. Rmi dump. Technical report, INRIA Lorraine - équipe Madynes, Août 2006.

[57] Vincent Cridlig, Olivier Festor, and Radu State. Role-based access control for xml enabled management gateways. In *Utility Computing : 15th IFIP/IEEE International Workshop on Distributed Systems : Operations and Management (DSOM)*, volume 3278 of *Lecture Notes in Computer Science*, pages 183–195. Springer, 15-17 November 2004. ISBN 3-540-23631-7.

[58] Mike Daniele, Bert Wijnen, Mark Ellison, and Dale Francisco. Agent Extensibility (AgentX) Protocol Version 1. RFC 2741 (Draft Standard), january 2000.

[59] Mikael Desertot, Clement Escoffier, and Didier Donsez. Autonomic management of j2ee edge servers. In *MGC '05 : Proceedings of the 3rd international workshop on Middleware for grid computing*, pages 1–6, Grenoble, France, 2005. ACM Press. ISBN 1-59593-269-0.

[60] Anja Feldmann. *Self-similar Network Traffic and Performance Evaluation*, chapter Characteristics of TCP connections, pages 367–399. Johon Wiley & Sons, 2000. ISBN 0471319740.

[61] Olivier Festor. *Ingéniérie de la gestion de réseaux et de services : du modèle OSI à la technogie active.* PhD thesis, Habilitation à Diriger des Recherches, 2001.

[62] Olivier Festor and André Schaff. *Standards pour la gestion des réseaux et des services.* Hermès, 2003.

[63] Marc Fleury and Juha Lindfors. *JMX : Managing J2EE Applications with Java Management Extensions*. Sams, Indianapolis, IN, USA, 2001. ISBN 0672322889.

[64] François Fluckiger. *Understanding networked multimedia : applications and technology*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, 1995. ISBN 0-13-190992-4.

[65] Germán Goldszmidt. On distributed system management. In *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research*, pages 637–647, Toronto, Ontario, Canada, 1993. IBM Press.

[66] William Grosso. *Java RMI*. O'Reilly, October 2001. ISBN : 565924525.

[67] David Harrington, Randy Presuhn, and Bert Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (Standard), december 2002.

[68] Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M. Tilbury. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004. ISBN 047126637X.

[69] Marjan Hericko, Matjaz B. Juric, Ivan Rozman, Simon Beloglavec, and Ales Zivkovic. Object serialization analysis and comparison in java and .net. *SIGPLAN Not.*, 38(8) : 44–54, 2003. ISSN 0362-1340.

[70] Matjaz B. Juric, Ivan Rozman, and Simon Nash. Java 2 distributed object middleware performance analysis and optimization. *SIGPLAN Not.*, 35(8) :31–40, 2000. ISSN 0362-1340.

[71] Mohsen Kahani and H. W. Peter Beadle. Decentralised approaches for network management. *SIGCOMM Comput. Commun. Rev.*, 27(3) :36–47, 1997. ISSN 0146-4833.

[72] Michael J. Katchabaw, Stephen L. Howard, Andrew D. Marshall, and Michael A. Bauer. Evaluating the costs of management : a distributed applications management testbed. In *Proceedings of the 1996 conference of the Centre for Advanced Studies on Collaborative research*, page 18, Toronto, Ontario, Canada, 1996. IBM Press.

[73] Heather Kreger, Ward K. Harold, and Leigh Williamson. *Java and JMX : Building Manageable Systems*. Addison-Wesley, 2003. ISBN : 0672324083.

[74] David B. Levi, Paul Meyer, and Bob Stewart. Simple Network Management Protocol (SNMP) Applications. RFC 3413 (Standard), december 2002.

[75] Mo Li and Kumbesan Sandrasegaran. Network management challenges for next generation networks. In *LCN '05 : Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*, pages 593–598, Washington, DC, USA, November 2005. IEEE Computer Society. ISBN 0-7695-2421-4.

[76] Antonio Liotta, Graham Knight, and George Pavlou. Modelling network and system monitoring over the internet with mobile agents. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 303–312, New Orleans, LA, USA, February 1998. ISBN 0-7803-4351-4.

[77] Jean-Philippe Martin-Flatin, Simon Znaty, and Jean-Pierre Hubaux. A survey of distributed enterprise network and systems management paradigms. *Journal of Network and Systems Management*, 7(1) :9–26, 1999. ISSN 1064-7570.

[78] Keith McCloghrie and Marshall Rose. Management Information Base for Network Management of TCP/IP-based internets :MIB-II. RFC 1213 (Standard), march 1991.

[79] Pratyush Moghe and Michael H. Evengelista. Rap-rate adaptive polling for network management applications. In *NOMS 98 : Network Operations and Management Symposium*, volume 2, pages 395–399, New Orleans, LA, USA, 15-20 February 1998. ISBN 0-7803-4351-1.

[80] Robert Morris and Dong Lin. Variance of aggregated web traffic. In *IEEE INFOCOM*, pages 360–366, Tel Aviv, Israel, March 2000. ISBN 0-7803-5880-5.

[81] Dennis M.Sosnoski. Java programming dynamics,part 2. In *Java technology*. IBM developerWorks, April 2004.

[82] MX4J. Open source jmx for entreprise computing. http ://www.mx4j.org, April 2004. Release 2.0.

[83] Konstantina Papagiannaki, Rene Cruz, and Christophe Diot. Network performance monitoring at small time scales. In *IMC '03 : Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 295–300, Miami Beach, FL, USA, 2003. ACM Press. ISBN 1-58113-773-7.

[84] Konstantina Papagiannaki, Nina Taft, and Anukool Lakhina. A distributed approach to measure ip traffic matrices. In *IMC '04 : Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 161–174, Taormina, Sicily, Italy, 2004. ACM Press. ISBN 1-58113-821-0.

[85] KyoungSoo Park and Vivek S. Pai. Comon : a mostly-scalable monitoring system for planetlab. *SIGOPS Oper. Syst. Rev.*, 40(1) :65–74, 2006. ISSN 0163-5980.

[86] Aiko Pras. *Network Management Architectures*. PhD thesis, University of Twente, Netherlands, 1995. ISBN : 90-365-0728-6.

[87] Marshall T. Rose, Keith McCloghrie, and James R. Davin. Bulk table retrieval with the snmp. RFC 1187 (Experimental), october 1990.

[88] Jurgen Schoenwaelder. Simple Network Management Protocol Over Transmission Control Protocol Transport Mapping. RFC 3430 (Experimental), december 2002.

[89] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :379–423, July 1948.

[90] A. Siddiqui, D. Romascanu, and E. Golovinsky. Real-time Application Quality-of-Service Monitoring (RAQMON) Framework. RFC 4710 (Proposed Standard), october 2006.

[91] Paulo Simoes, Joao Rodriguez, Luis Silva, and Fernando Boavida. Distributed retrieval of management information : Is it about mobility, locality or distribution ? In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 79–94, 2002. ISBN : 0-7803-7382-0.

[92] Morris Sloman, editor. *Network and distributed systems management*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994. ISBN 0-201-62745-0.

[93] William Stalling. *SNMP, SNMPv2, and CMIP : The Practical Guide to Network-Management Standards*. Addison-Wesley, July 1993. ISBN : 978-0201633313.

[94] Fotis Stamatelopoulos and Basil Maglaris. Performance and efficiency in distributed enterprise management. *Journal of Network and Systems Management*, 7(1) :47–71, March 1999.

[95] Frank Strauß. Distribution models for network management functions. Technical report, TU Braunschweig, June 2000. http ://www.ibr.cs.tu-bs.de/papers/jasmin-dismod.ps.

[96] Rajesh Subramanyan. *Scalable SNMP-based monitoring systems for network computing*. PhD thesis, Purdue University, 2002.

[97] SUN. Java^{TM} management extensions, instrumentation and agent specification, v1.2. http ://jcp.org/en/jsr/detail ?id=3, october 2002. Maintenance Release 2.

[98] SUN. Java management extension (jmx)
. http ://java.sun.com/products/JavaManagement/index.jsp, August 2003. Release 1.2.1.

[99] SUN. Java^{TM} management extensions (jmx) remote api 1.0 specification. http ://www.jcp.org/en/jsr/detail ?id=160, october 2003. Final Release.

[100] SUN. Jsr 174 : Monitoring and management specification for the java^{TM} virtual machine. http ://www.jcp.org/en/jsr/detail ?id=174, september 2004. Final release.

[101] SUN. J2ee^{TM} management. http ://www.jcp.org/en/jsr/detail ?id=77, Jun 2006. Maintenance release.

[102] SUN. Jsr 262 : Web services connector for java^{TM} management extensions (jmx) agents. http ://www.jcp.org/en/jsr/detail ?id=262, april 2007. In Progress.

[103] Kornel Terplan. *Benchmarking for Effective Network Management*. McGraw-Hill, Inc., New York, NY, USA, 1995. ISBN 0070636389.

[104] Kornel Terplan. *Telecom Operations Management Solutions with NetExpert*. CRC Press, 1998.

[105] Kiyohito Yoshihara, Keizo Sugiyama, Hiroki Horiuchi, and Sadao Obana. Dynamic polling scheme based on time variation of network management information values. In *IEEE/IFIP International Symposium on Integrated Network Managemen (IM)*, pages 141–154, Boston, MA, USA, 25 May 1999. ISBN 0-7803-5748-5.

[106] Qi Zhao, Zihui Ge, Jia Wang, and Jun Xu. Robust traffic matrix estimation with imperfect information : making use of multiple data sources. *SIGMETRICS Perform. Eval. Rev.*, 34 (1) :133–144, 2006. ISSN 0163-5999.

Modélisation de la gestion

[107] Christos Bohoris, George Pavlou, and Haitham Cruickshank. Using mobile agents for network performance management. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 637–652, Hawaii, USA, April 2000. ISBN 0-7803-5928-3.

[108] Thomas M. Chen and Stephan S. Liu. A model and evaluation of distributed network managemetn approaches. *IEE Journal on selected areas in communications*, 20(2) :850–857, may 2002.

[109] German Goldszmidt, Yechiam Yemini, and Shaula Yemini. Network management by delegation : the mad approach. In *CASCON '91 : Proceedings of the 1991 conference of the Centre for Advanced Studies on Collaborative research*, pages 347–361, Toronto, Ontario, Canada, 1991. IBM Press.

[110] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5) :41–54, 2005. ISSN 0146-4833.

[111] Jefrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 1(36) :41–50, January 2003.

[112] Alberto Gonzalez Prieto and Rolf Stadler. Adaptive distributed monitoring with accuracy objectives. In *SIGCOMM workshop on Internet network management (INM)*, pages 65–70, Pisa, Italy, 2006. ACM Press. ISBN 1-59593-570-3.

[113] Akhil Sahai and Christine Morin. Towards distributed and dynamic network management. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 455–464, New Orleans, USA, 15-20 February 1998. ISBN 0-7803-4351-4.

[114] Jurgen Schoenwalder. Snmp traffic measurements and trace exchange formats. draft-irtf-nmrg-snmp-measure-01.txt, January 2007.

[115] Jurgen Schoewaelder. Characterization of snmp mib modules. In *IEEE/IFIP International Symposium on Integrated Network Management (IM)*, pages 615–628, Nice, France, 2005. IEEE. ISBN 0-7803-9087-3.

[116] Jurgen Schönwälder, Aiko Pras, Harvan Matus, Jorrit Schippers, and Remco van de Meent. Snmp traffic analysis : Approaches, tools, and first results. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 323–332, Munich, Germany, May 2007. ISBN 1-4244-0799-0.

[117] Daniel R. Seligman. How not to build a performance monitoring agent. *IEEE Communications Magazine*, 45(4) :138–142, April 2007. ISSN 0163-6804.

[118] Fetahi Wuhib, Mads Dam, Rolf Stadler, and Alexander Clemm. Decentralized computation of threshold crossing alerts. In *Ambient Networks, 16th IFIP/IEEE International Workshop on Distributed Systems : Operations and Management, DSOM 2005, Barcelona, Spain, October 24-26, 2005, Proceedings*, volume 3775 of *Lecture Notes in Computer Science*, pages 220–232. Springer, 2005. ISBN 3-540-29388-4.

[119] Martin Zach, Daryl Parker, Johan Nielsen, Claire Fahy, Ray Carroll, Elyes Lehtihet, Nektarios Georgalas, Ricardo Marin, and Joan Serrat. Towards a framework for network management applications based on peer-to-peer paradigms. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 1–4, Vancouver, Canada, 3-7 April 2006. ISBN 1-4244-0142-9.

Évaluation de performances de la supervision

[120] Mario Baldi and Gian Pietro Picco. Evaluating the tradeoffs of mobile code design paradigms in network management applications. In *20th International Conference on Software engineering*, pages 146–155, Kyoto, Japan, 1998. IEEE Computer Society. ISBN 0-8186-8368-6.

[121] Alan Bivens, Rashim Gupta, Ingo McLean, Boleslaw Szymanski, and Jerome White. Scalability and performance of an agent-based network management middleware. *International Journal of Network Management*, 14(2) :131–146, 2004. ISSN 1099-1190.

[122] Alessandro Corrente and Luigi Tura. Security performance analysis of snmpv3 with respect to snmpv2c. In *Network Operations and Management Symposium (NOMS)*, pages 729–742, Seoul, Korea, 2004. ISBN 0-7803-8230-7.

[123] Weldson Queiroz de Lima, Rodrigo Sanger Alves, Ricardo Lemos Vianna, Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco, and Lisandro Zambenedetti Granville. Evaluating the performance of snmp and web services notifications. In *Network Operations and Management Symposium (NOMS)*, pages 546–556, Montreal, Canada, April 2006. ISBN 1-4244-0143-7.

[124] Mark Dilman and Danny Raz. Efficient reactive monitoring. In *INFOCOM : 9th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1012–1019, 2001.

[125] James Du, Mark Shayman, and M. Rozenblit. Implementation and performances analysis of snmp on a tls/tcp/ base. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 453–466, Seattle, WA, USA, may 2001. IEEE. ISBN 0-7803-6719-7.

[126] Tiago Fioreze, Lisandro Zambenedetti Granville, Maria Janilce Almeida, and Liane Tarouco. Comparing web services with snmp in a management by delegation environment. In *International Symposium on Integrated Network Management (IM)*, volume 9, pages 601–614, Nice, France, May 2005.

[127] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5) :342–361, 1998.

[128] Damianos Gavalas, Dominic Greenwood, Mohammed Ghanbari, and Mike O'Mahony. Advanced network monitoring applications based on mobile/intelligent agent technology. *Computer Communications, Elsevier Science*, 23(8) :720–730, April 2000. ISSN 0140-3664.

[129] Germán S. Goldszmidt and Yechiam Yemini. Evaluating management decisions via delegation. In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services*, pages 247–257, Amsterdam, The Netherlands, The Netherlands, 1993. North-Holland Publishing Co. ISBN 0-444-89982-0.

[130] Qiang Gu and Alan Marshall. Network management performance analysis and scalability tests : Snmp vs. corba. In *Network Operations and Management Symposium (NOMS)*, pages 701–714, Seoul, Korea, 2004. ISBN 0-7803-8230-7.

[131] J. Kantorovitch and P. Mahonen. Case studies and experiments of snmp in wireless networks. In *IEEE Workshop on IP Operations and Management (IPOM)*, pages 179–183, Dallas, Texas, USA, October 2002. ISBN 0-7803-7658-7.

[132] Alva L.Couch, Ning Wu, and Hengky Susanto. Toward a cost model for system administration. In *19th Large Installation System Administration Conference (LISA)*, pages 125–141. USENIX, 2005.

[133] Koon-Seng Lim and Rolf Stadler. A navigation pattern for scalable internet management. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 405–420, Seatle, Washington, May 2001.

[134] Antonio Liotta, Graham Knight, and George Pavlou. On the performance and scalability of decentralized monitoring using mobile agents. In *IFIP/IEEE International Workshop on Distributed Systems : Operations and Management (DSOM)*, pages 3–18, Zurich, Switzerland, 1999. Springer-Verlag. ISBN 3-540-66598-6.

[135] Gottfried Luderer, Hosoon Ku, Baranitharan Subbiah, and Anand Narayanan. Network management agents supported by a java environment. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, volume 86, pages 790–790, San Diego, CA, USA, May 1997. ISBN : 0-412-80960-5.

[136] Vladislav Marinov and Jürgen Schönwälder. Performance analysis of snmp over ssh. In *Large Scale Management of Distributed Systems, 17th IFIP/IEEE International Workshop on Distributed Systems : Operations and Management (DSOM)*, pages 25–36, Dublin, Ireland, 23-25 October 2006.

[137] Giovane Cesar Moreira Moura, Giancarlo Silvestrin, Ricardo Nabinger Sanchez, Luciano Paschoal Gaspary, and Lisandro Zambenedetti Granville. On the performance of web services management standards - an evaluation of muws and ws-management for network management. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 459–468, Munich, Germany, 21-25 May 2007. IEEE.

[138] Ricardo Neisse, Ricardo Vianna, Lisandro Zambenedetti Granville, Maria Janilce Bosquiroli Almeida, and Liane Margarida Rockenback Tarouco. Implementation and bandwidth consumption evaluation of snmp to web services gateways. In *Network Operations and Management Symposium (NOMS)*, pages 715–728, Seoul, Korea, 2004. ISBN 0-7803-8230-7.

[139] Collin Pattinson. A study of the behaviour of the simple network management protocol. In *International Workshop on Distributed Systems : Operations & Management (DSOM)*, Nancy, France, 15-17 October 2001.

[140] Georga Pavlou, Paris Flegkas, Stelios Gouveris, and Antonio Liotta. On management technologies and the potential of web services. *IEE, Communications Magazine*, 42 :58–66, July 2004. ISSN 0163-6804.

[141] Ray Pradeep. Evaluation methodology for network management systems. In *Network Operations and Management Symposium (NOMS)*, pages 590–599, New Orleans, Lousiana, February 1998. ISBN 0-7803-4351-4.

[142] Aiko Pras, Thomas Drevers, Remco van de Meent, and Dick Quartel. Comparing the performance of snmp and web services-based management. *IEEE eTNSM (Transactions on Network and Service Management)*, 1(2), december 2004.

[143] Marcelo G. Rubinstein, Otto Carlos Muniz Bandeira Duarte, and Guy Pujolle. Reducing the response time in network management by using multiple mobile agents. In *MMNS, Third International Conference on Management of Multimedia Networks and Services, Fortaleza, Ceará, Brazil*, pages 253–265, 2000.

[144] Fahim Sheikh, Jerome Rolia, Panka Garg, Svend Frolund, and Allan Shephered. Layered performance modeling of a corba-based distributed application design. In *First World Congress on Computer Simulation*, pages 247–254, Singapore, September 1997.

[145] Markus Strasser and Markus Schwehm. A performance model for mobile agent systems. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, volume 2, pages 1132–1140, 1997.

[146] Rajesh Subramanyan, José Miguel-Alonso, and José A. B. Fortes. A scalable snmp-based distibuted monitoring system for heterogeneous network computing. In *Supercomputing : ACM/IEEE conference on Supercomputing*, page 14, Dallas, Texas, United States, 2000. IEEE Computer Society. ISBN 0-7803-9802-5.

[147] Remco van de Meent, Aiko Pras, Michel Mandjes, Hans van den Berg, and Lambert J. M. Nieuwenhuis. Traffic measurements for link dimensioning - a case study. In *Self-Managing Distributed Systems, 14th IFIP/IEEE International Workshop on Distributed Systems : Operations and Management (DSOM)*, volume 2867 of *Lecture Notes in Computer Science*, pages 106–117, Heidelberg, Germany, 20-22 October 2003. Springer.

[148] Ricardo Lemos Vianna, Everton Rafael Polina, Clarissa Cassales Marquezan, Leandro Bertholdo, Liane Margarida Rockenbach Tarouco, Maria Janilce Bosquiroli Almeida, and Lisandro Zambenedetti Granville. An evaluation of service composition technologies applied to network management. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 420–428, Munich, Germany, 21-25 May 2007. IEEE.

[149] Fetahi Wuhib, Mads Dam, Rolf Stadler, and Alexander Clemm. Decentralized computation of threshold crossing alerts. In *Ambient Networks, 16th IFIP/IEEE International Workshop on Distributed Systems : Operations and Management (DSOM)*, volume 3775 of *Lecture Notes in Computer Science*, pages 220–232, Barcelona, Spain, October 24-26 2005. Springer.

[150] Michael Zapf, Klaus Herrmann, and Kurt Geihs. Decentralized SNMP management with mobile agents. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, page 623, Boston, MA, USA, 1999. ISBN 0-7803-5748-5.

Documents IPPM

[151] Guy Almes, Sunil Kalidindi, and Mattew J. Zekauskas. A One-way Packet Loss Metric for IPPM. RFC 2680 (Proposed Standard), september 1999.

[152] Guy Almes, Sunil Kalidindi, and Matthew J. Zekauskas. A One-way Delay Metric for IPPM. RFC 2679 (Proposed Standard), september 1999.

[153] Guy Almes, Sunsil Kalidindi, and Matthew J. Zekauskas. A Round-trip Delay Metric for IPPM. RFC 2681 (Proposed Standard), september 1999.

[154] Carlo Demichelis and Philip Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393 (Proposed Standard), november 2002.

[155] Stephan Emile, Lei Liang, and Al Morton. IP Performance Metrics (IPPM) for spatial and multicast, July 2007.

[156] Rajeev Koodli and Rayadurgam Ravikanth. One-way Loss Pattern Sample Metrics. RFC 3357 (Informational), august 2002.

[157] Jamshid Mahdavi and Vern Paxson. IPPM Metrics for Measuring Connectivity. RFC 2678 (Proposed Standard), september 1999.

[158] Matt Mathis and Mark Allman. A Framework for Defining Empirical Bulk Transfer Capacity Metrics. RFC 3148 (Informational), july 2001.

[159] Al Morton, Len Ciavattone, Gomathi Ramachandran, Stanislav Shalunov, and Jerry Perser. Packet Reordering Metrics. RFC 4737 (Proposed Standard), november 2006.

[160] Vern Paxson, Guy Almes, Jamshid Mahdavi, and Matt Mathis. Framework for IP Performance Metrics. RFC 2330 (Informational), may 1998.

[161] Vern Edward Paxson. *Measurements and analysis of end-to-end Internet dynamics*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1998.

[162] Jon Postel. Internet Protocol. RFC 791 (Standard), september 1981.

Résumé

La performance et l'efficacité de la gestion sont devenues une préoccupation au sein de la communauté de gestion de réseaux et de services. Cette préoccupation est due essentiellement aux dimensions grandissantes des réseaux et des services et l'intégration de plan de gestion dans leurs plans fonctionnels. La précision avec laquelle elle effectue ses tâches est aussi une préoccupation majeure. Les études existantes qui ont tenté de quantifier la performance de la gestion présentent plusieurs limites au niveau des métriques et des méthodes employées. En effet, les principales lacunes de ces études sont l'absence de métriques standards et les caractères non comparables, non reproductibles et non représentatives de leurs méthodologies de mesure.

En nous basant sur ce constat, nous avons travaillé sur la définition d'un ensemble de métriques primaires et secondaires pour mesurer la performance d'une approche de gestion. Les métriques primaires sont regroupées en trois familles : rapidité, coût et qualité. Les métriques secondaires proposées reposent sur ces dernières et permettent de quantifier l'efficacité, le passage à l'échelle et l'incidence de la gestion. Nous avons élaboré une méthodologie pour mesurer ces métriques primaires. Afin de valider cette proposition, nous avons conçu et implanté un banc de mesure dédié à l'évaluation de performances de l'approche de gestion JMX.

La seconde partie de notre travail a porté sur l'élaboration de fines campagnes de mesures de performances de JMX. Les résultats de ces mesures, nous ont permis de caractériser le passage à l'échelle, l'incidence de la gestion sur la performance d'un système géré et les délais de l'approche JMX sous différents scénarios. Nous avons trouvé que les délais que subissent les opérations JMX suivent approximativement une distribution statistique de Weibull. Grâce à ce rapprochement, nous avons pu identifier l'effet des délais sur le comportement d'un algorithme de supervision, notamment la distorsion temporelle de la vue observée par le gestionnaire par rapport à la vue réelle du système géré.

Mots-clés: Gestion de réseaux et de services, JMX, évaluation de performances, métriques de mesure, méthodologie de mesure, efficacité de la gestion, passage à l'échelle, incidence de la gestion

Abstract

The efficiency and the performance of management systems is becoming a hot research topic within the networks and services management community. This concern is due to the new challenges of large scale managed systems, where the management plane is integrated within the functional plane and where management activities have to carry accurate and up-to-date information. Existing studies that have tried to assess the performance of management shows many limits in terms of used metrics and methods. These studies are burdened with the lack on common metrics and usually their results are not *comparable*, their experiments are not *reproducible* and they are not *representative*.

We defined a set of primary and secondary metrics to measure the performance of a management approach. We grouped the primary metrics into three categories : speed, quality and cost. Secondary metrics are derived from the primary ones and quantifies mainly the efficiency, the scalability and the impact of management activities. We have developed a measurement

methodology to measure the primary metrics. To validate our propositions, we have designed and developed a benchmarking platform dedicated to the measurement of the performance of a JMX manager-agent pattern based management systems.

The second part of our work deals with the collection of measurement data sets from our JMX benchmarking platform. We mainly studied the effect of management load and the number of agents on the scalability, the impact of management activities on the user perceived performance of a managed server and the delays of JMX operations when carrying variables values. We find that most of these delays follow a Weibull statistical distribution. We used this statistical model to study the behavior of a monitoring algorithm proposed in the literature, under heavy tail delays distribution. In this case, the view of the managed system on the manager side becomes noisy and out of date.

Keywords: Network and service management, JMX, performance evaluation, measurement metrics, measurement methodology, management efficiency, management scalability, management delays, management impact