UNIVERSITE SCIENCES BRETAGNE DE LA MER LOIRE ET DU LITTORAL

UBO
Université de Bretagne Occidentale

THESE DE DOCTORAT DE

L'UNIVERSITE
DE BRETAGNE OCCIDENTALE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 598
Sciences de la Mer et du littoral
Spécialité : Géomatique

Par
Loïc SALMON

Une approche holistique combinant flux temps-réel et données archivées pour la gestion et le traitement d'objets mobiles : application au trafic maritime.

Thèse présentée et soutenue à Lanvéoc, le 17 janvier 2019
Unité de Recherche : Institut de Recherche de l'Ecole Navale (EA3634)

Rapporteurs avant soutenance :

Alain BOUJU Maître de conférence HDR à l'Université de La Rochelle
Thomas DEVOGELE Professeur des universités à l'Université de Tours

Composition du Jury :

Rapporteurs
Alain BOUJU Maître de conférence HDR à l'Université de La Rochelle
Thomas DEVOGELE Professeur des universités, Université de Tours

Examinateurs
Karine ZEITOUNI Professeur des universités, Université de Versailles (USQV)
Jean-Philippe BABAU Professeur des universités, Université Bretagne Occidentale (UBO), Président du jury

Co-directeur de thèse
Cyril RAY Maître de conférences, IRENav Brest

Directeur de thèse
Christophe CLARAMUNT Professeur des universités, IRENav Brest

Table des matières

Table des figures

Liste des algorithmes

Liste des tableaux

Chapitre 1

INTRODUCTION

1.1 "Big Data" et objets mobiles

Depuis les dernières décennies le nombre de capteurs, collectant notamment des données géolocalisées n'a cessé d'augmenter. Ceci soulève de nouveaux enjeux en termes de stockage, de traitement et d'extraction de connaissance relatifs à ces données spatio-temporelles qu'il est nécessaire de plus en plus de gérer "à la volée", là où les systèmes de gestion de bases de données actuels ne suffisent plus. En effet, lorsque le volume de données à traiter est conséquent, les requêtes peuvent être très coûteuses en temps de calcul et en ressources alors même que les besoins tendent vers du temps-réel. De plus, l'ajout continu de données en base est limité dans un modèle relationnel, car usuellement l'arrivée de nouvelles données n'est faite que de manière périodique face à des systèmes qui doivent donner des réponses à la volée. Enfin, le schéma de donnée est fixe et inclut des données de même type, avec un format et une norme sur les valeurs de chacun des champs. Un des enjeux est de pouvoir stocker et traiter des données hétérogènes, parfois erronées ou incomplètes et de les fusionner pour extraire une information globale.

L'analyse de mobilité intervient dans de nombreuses disciplines scientifiques et applications telles que la surveillance du trafic, l'aménagement du territoire, la sociologie, la robotique ou la zoologie. Généralement l'une des problématiques principales associées est de trouver des motifs fréquents ou des données aberrantes, dans le but de découvrir une nouvelle connaissance concernant le phénomène étudié [Giannotti et al., 2007]. Par exemple, dans le contexte d'analyse du trafic maritime, des comportements normaux et des trajectoires anormales peuvent être identifiés [Etienne et al., 2012]. De manière similaire, en sociologie, l'émergence de motifs de mobilité peuvent être inférés et étudiés [Giannotti et al., 2011]. Cependant, la plupart des systèmes actuels ne fournissent pas d'outil de stockage et de traitement efficace pour les données d'objets mobiles. En effet, ces objets mobiles produisent de larges volumes de données de trajectoires, ce qui nécessite en plus que le système soit mis à jour régulièrement, au fur et à mesure que les données sont collectées.

L'émergence de nouveaux systèmes et paradigmes pour traiter de larges volumes de données à

l'image de MapReduce fournissent des solutions prometteuses, mais qui ne prennent pas en compte la spécificité des données spatiales. Ceci a motivé des travaux plus récents dérivés notamment de MapReduce [Dean and Ghemawat, 2004], plus spécifiquement orientés pour traiter des données spatiales [Eldawy and Mokbel, 2013], [Aji et al., 2013] ou plus particulièrement pour la gestion des objets mobiles [Nidzwetzki and Güting, 2015]. Mais ces systèmes sont toujours orientés pour du traitement et de l'analyse a posteriori des données et ne sont pas efficaces pour gérer les données en temps-réel. Cependant, dans un contexte de détection en temps-réel d'anomalies pour des objets mobiles, les données les plus récentes ne sont pas suffisantes pour différencier une situation "normale" d'une anomalie. C'est pourquoi il est nécessaire d'envisager une solution dite "hybride" combinant des résultats extraits de données anciennes avec les données arrivant "à la volée" dans le système [Salmon et al., 2015]. Cette thèse aborde donc la conception d'un système permettant le suivi, le traitement et l'analyse de données de mobilité en temps-réel combinant les informations extraites d'une base de données archivée et les flux de données temps-réel.

1.2 Objets mobiles : Enjeux de recherche

Les données de mobilité nécessitent un traitement particulier, ce qui impose certains prérequis et soulève des questions sur plusieurs éléments. Voici une liste non exhaustive des spécificités à prendre en compte pour notre architecture :

Partitionnement des données. La complexité des traitements et l'important volume de données à manipuler nous contraignent d'envisager une architecture distribuée et un paradigme de traitement distribué associé. Celui-ci doit être adapté au contexte spatial et permettre ainsi de traiter les données de mobilité de façon plus performante. Pour stocker et distribuer les données de manière efficace, les données spatiales et spatio-temporelles doivent être réparties équitablement sur l'ensemble des noeuds de l'architecture distribuée. Mais un partitionnement équilibré des données spatiales, et a fortiori d'objets mobiles est plus difficile, car les données associées sont réparties par rapport au temps et l'espace de manière inégale, comparé à des données usuelles pouvant être réparties aléatoirement pour avoir une distribution homogène des données. Ainsi, les données doivent être réparties au mieux par rapport à leur zone d'appartenance spatiale et spatio-temporelle, bien que les événements et phénomènes ne le soient pas. Ceci rend difficile l'utilisation d'une grille statique par exemple, tandis que placer dynamiquement les données sur les noeuds pose des problèmes de performance, car chaque nouvelle donnée entrante a besoin d'être déplacée de manière à préserver la répartition. Une répartition équilibrée des données sur l'architecture est d'autant plus difficile que les

données de positions sont reportées non uniformément par rapport au temps, l'espace et la sémantique.

Complexité des traitements. Le traitement de données spatiales et en particulier de données dérivant d'objets mobiles se révèle plus difficile à traiter. En effet, des opérations sur des données spatiales font intervenir de nombreux objets spatiaux et entités, chacune d'entre elles pouvant être composée d'un grand nombre de points, de lignes et de polygones. De plus, les opérations spatiales ne se limitent pas à des calculs de distance entre objets spatiaux ou des opérations topologiques sur des zones particulières, mais font intervenir de l'analyse complexe et de la fouille de données spatiales. L'analyse de mobilité augmente le spectre et la complexité des requêtes, le traitement d'objet mobile nécessitant de reconstruire chacune des trajectoires et parfois de comparer les positions successives de ces objets mobiles deux à deux ce qui est particulièrement coûteux.

Complexité des données. Les données à manipuler sont hétérogènes et peuvent varier en forme et en taille au cours du temps, un modèle de données et des index spécifiques peuvent sans doute permettre de faire face à cette difficulté et stocker au mieux les données de mobilité.. En effet, la représentation de l'espace comprend des positions, mais également des lignes et des polygones qui peuvent avoir des formes et des tailles très variables. Ceci pose un problème au niveau de l'indexation des objets spatiaux par rapport à des données non spatiales. De plus la composante temporelle rajoute en complexité, car en fonction de la méthode d'indexation choisie, certaines requêtes se verront favorisées et d'autres dépréciées (e.g. notamment pour une requête portant sur des trajectoires avec un index considérant uniquement les coordonnées géographiques). De nombreuses solutions d'index existent suivant la problématique ciblée, mais compte tenu de l'arrivée en continu de ces données qu'il faut stocker et traiter, il est difficile de garder une structure optimale (en lecture) sans avoir à le reconstruire au fur et à mesure des ajouts, ce qui peut affecter les performances.

Localité et granularité. Les données spatiales dépendent du référentiel et du niveau de détail utilisés pour stocker celles-ci. Par exemple une trajectoire peut être représentée par un ensemble de points ou de segments. La manipulation de ces données est donc limitée par le niveau de détail choisi. Suivant la localité et la granularité considérées, les résultats d'une requête peuvent totalement différer. La prise en compte de différents niveaux de représentations et granularités peut permettre de pallier en partie ce problème, mais complexifie les traitements. Ainsi, la granularité et la localité doivent être étudiées pour adopter des choix de plages, temporelles et spatiales permettant d'éviter les biais de localité et les phénomènes de lissage. Ceci soulève également le problème de pertinence des données et du volume de données nécessaires pour donner une réponse "satisfaisante" à une requête. En effet, le traitement de l'ensemble de toutes les données anciennes n'est peut-être pas

nécessaire et considérer uniquement un sous-ensemble est suffisant pour fournir la réponse avec un bien meilleur temps de réponse.

Dépendance spatiale et temporelle. Tous les événements considérés sont corrélés partiellement au temps et à l'espace. Ainsi, les méthodes d'échantillonnage ou d'agrégation utilisées ou d'analyses statistiques de données spatiales doivent être spécifiques pour prendre en compte la dimension spatiale. Dans le contexte d'objets mobiles, une méthode pour échantillonner les positions d'une trajectoire consiste à conserver toutes les positions qui suffisent à décrire le mouvement de l'objet et retrouver par interpolation la position de l'objet à un instant t choisi en minimisant la perte d'information occasionnée comparé à un échantillonnage qui peut être plus aléatoire pour des données usuelles.

Spécificité des objets mobiles et incertitude. Les objets mobiles génèrent un grand nombre de positions qu'il est nécessaire de traiter "à la volée". C'est pourquoi l'analyse de mobilité ou de trajectoire nécessite de considérer les notions de volume et de vélocité. De plus, alors que les mouvements sont stockés de manière discrète en base de données, il est nécessaire d'interpoler et prendre en considération l'incertitude résultant à la fois des données collectées et de l'erreur générée par l'interpolation. En effet, si nous souhaitons déterminer la position actuelle d'un objet mobile il est nécessaire de considérer la dernière position reçue et d'estimer la position actuelle à partir du cap et de la vitesse associée.

Conception d'un modèle hybride d'analyse des données. Pour comprendre le mouvement et le comportement des objets mobiles, il est nécessaire de coupler des informations issues de données anciennes avec les flux reçus en temps-réel. En effet, si l'étude se limite aux données reçues récemment il n'est pas possible de faire de la détection d'anomalies ou de juger de la normalité d'un comportement. Il est donc nécessaire de mettre en place des systèmes de résumés de données pouvant être comparés aux données aux flux reçus en temps-réel afin de comprendre le comportement des objets mobiles et pouvoir détecter d'éventuels comportements anormaux en temps-réel.

1.3 Objectifs de recherche

La recherche présentée dans cette thèse concerne la mise en place et l'étude d'une architecture hybride (i.e. requérant le couplage entre informations obtenues de données anciennes et flux continus) et distribuée pour le stockage et le traitement de données d'objets mobiles. Deux problématiques principales associées à cette architecture ont été retenues et sont d'importance pour la suite du travail.

1.3.1 Gestion et stockage des données

Le premier objectif à traiter est la gestion des données sur une architecture hybride et distribuée indispensable pour interroger et stocker des volumes grandissants de données. Celle-ci repose sur plusieurs éléments (i) d'abord la spécificité des données d'objets mobiles nécessitant un modèle de données particulier tandis que le grand volume de données impose d'étudier différentes distributions des données et l'utilisation d'index pour avoir un meilleur traitement (ii). Enfin, (iii) la structure hybride envisagée implique également de s'interroger sur la façon dont les données vont être transférées depuis la partie on-line et off-line et de la granularité choisie pour ces deux parties.

Le choix du modèle de données à adopter pour retrouver et analyser de manière la plus satisfaisante possible les données dépend en grande partie de la nature des données à manipuler. De nombreux travaux ont été réalisés pour mettre en avant certains modèles de données ou les comparer [Abadi et al., 2008], [Cattell, 2011]. Seulement, la spécificité des données spatiales [Anselin, 1989] nécessite d'être prise en compte pour traiter le déluge de données actuel [Shekhar et al., 2012]. C'est pourquoi diverses options ont été étudiées pour déterminer un modèle de données adapté aux données spatiales [de Souza Baptista et al., 2014], [Baas, 2012].

Ensuite, concernant la répartition des données sur l'architecture distribuée pour réduire le temps de traitement. En ce qui concerne les données spatiales des travaux comme SpatialHadoop [Eldawy and Mokbel, 2013] ou Hadoop-GIS [Aji et al., 2013] étendent le framework Hadoop en introduisant opérateurs, fonctions et index spatiaux, mais aussi en ajoutant la notion de répartition des données en fonction de la zone spatiale de couverture. D'autres travaux plus spécifiques s'intéressent à différentes répartitions de données pour l'étude de trajectoires [Sun et al., 2013], [Ma et al., 2009]. En complément de cette répartition sur les noeuds de l'architecture, diverses éventualités peuvent être considérées concernant l'utilisation d'index. Ce domaine a largement a été investigué [Nguyen-Dinh et al., 2010] pour prendre en compte la mobilité, la fréquence d'ajout en base de données et le type de requêtes préféré. Cette problématique reste vraie pour la partie on-line avec l'usage d'index dits *in-memory*.

Enfin le dernier volet relatif à la gestion et au stockage de données concerne l'acheminement des données de la partie on-line vers la partie off-line. En effet, la taille choisie pour les fenêtres glissantes, le moment auquel s'effectue la transition des données, le transfert des données de la partie on-line possiblement réparties différemment qu'en off-line et avec des index différents sont autant de sous-questions qu'il peut être intéressant d'étudier.

1.3.2 Processus de traitements et requêtes

Le second objectif concerne le traitement des données de trajectoires. Dans ce contexte il est nécessaire de coupler les informations relatives aux éléments passés et aux positions enregistrées en temps-réel. L'objectif est donc de concevoir un système qui permette ce traitement hybride des données de manière similaire à certains travaux concernant une architecture centralisée pour des données à caractère non spatial, [Chandrasekaran and Franklin, 2004] mais en considérant une architecture distribuée et la prise en compte la dimension spatiale et spatio-temporelle.

Au niveau off-line et on-line séparément, le paradigme de traitement distribué à utiliser peut faire l'objet de diverses questions. Le paradigme MapReduce [Dean and Ghemawat, 2004] fait l'objet d'un engouement malgré quelques défauts inhérents [Doulkeridis and Nørvåg, 2014]. Différents autres paradigmes de traitement peuvent être envisagés comme les approches de traitement à base de workflow [Isard et al., 2007] ou distribuant les données et communiquant pendant le processus [Malewicz et al., 2010].

La gestion de requêtes multiples et parfois concurrentes est aussi un des enjeux [Golab, 2006], certains travaux en font état concernant les données d'objets mobiles [Mokbel et al., 2004], [Mokbel et al., 2005]. La fusion des informations extraites de la partie offline avec les données reçues en temps-réel est également une problématique à étudier, qui nécessite de s'intéresser à des techniques de résumés de données s'accompagnant aussi de mécanismes de mise à jour au fur et à mesure que les données entrent dans le système.

Enfin, des techniques de résumés de données doivent être étudiées tant bien pour la partie online que offline. Pour la partie online, il pourra s'agir de techniques d'échantillonnage, de synopse ou bien des algorithmes spécifiques ne parcourant les données qu'un nombre limité de fois. Pour la partie offline, il pourra s'agir d'agrégat de données ou de techniques de résumés de données qui pourront être facilement croisées avec des flux de données reçus en temps-réel. Pour des données spatio-temporelles ces techniques peuvent être particulièrement spécifiques ou plus raffinées, par exemple pour échantillonner des données sans composante spatiale, il est tout à fait envisageable de

réaliser un échantillonnage aléatoire là où pour des trajectoires il sera nécessaire de conserver les positions les plus représentatives du mouvement de l'objet (là ou l'objet change radicalement de cap ou bien de vitesse) [Potamias et al., 2006]. De même il est possible d'envisager des vues spécifiques aux données spatiales étant plus efficaces que celles utilisées pour traiter des données "usuelles" [Claramunt, 1998].

1.4 Plan du mémoire

La structure de la thèse est définie comme suit :

Le chapitre 2 dresse un ensemble de généralités concernant la problématique des "Big Data", notamment en abordant les travaux en termes de stockage et de traitement concernant l'aspect de volumétrie, puis en examinant le paradigme de traitement temps-réel à son origine puis distribué permettant de répondre à la problématique de vélocité. Cependant, ces systèmes n'intègrent pas la dimension spatiale et spatio-temporelle des données, et même s'ils fournissent des solutions intéressants, celles-ci ne sont pas appropriées pour stocker et traiter des données d'objets mobiles.

Le chapitre 3 liste l'ensemble des solutions et travaux qui ont été réalisés concernant la gestion et le traitement d'objets mobiles dans un contexte "Big Data". Deux approches principales sont distinguées, une dite offline se focalisant sur le stockage et le traitement de données anciennes pour l'extraction d'information et l'autre dite online sur la gestion en temps-réel des données permettant une analyse au gré des flux entrants, sans parvenir toutefois séparément à traiter des données spatio-temporelles de manière pertinente tout en respectant des contraintes de faible latence.

Le chapitre 4 introduit un modèle à différents niveaux pour la gestion et le traitement d'objets mobiles dans ce même contexte, prenant en compte les défauts inhérents des deux approches précédentes offline et online. Une approche hybride mêlant ces deux visions différentes pour permettre une compréhension plus précise des événements et comportements associés aux objets mobiles est proposée.

Le chapitre 5 développe la notion de requêtes hybrides, avec la description et la méthode de résolution associées à plusieurs d'entre elles. Ce chapitre montre la pertinence et la façon dont le modèle proposé précédemment peut être mis en oeuvre pour prendre en compte des problèmes spécifiques aux objets mobiles qu'une approche offline ou online seulement n'aurait pu résoudre.

Le chapitre 6 est l'application de notre modèle présenté en chapitre 4 au contexte maritime et au déplacement de navires. Ce domaine maritime étant caractérisé par une multiplication des capteurs émettant des positions géolocalisées relatifs à des objets mobiles se déplaçant dans un milieu

ouvert ou semi-ouvert par rapport à d'autres domaines où les déplacements sont contraints(par exemple pour le réseau routier. Certains cas d'intérêt sont développés et discutés comme la détection des zones de pêches ou la découverte des voies les plus traversées.

Le chapitre 7 présente l'implémentation du modèle introduite au chapitre 4, de quelques requêtes hybrides introduites dans le chapitre 5 dans un contexte maritime décrit au chapitre 6. Les résultats sont analysés et montrent la pertinence de notre approche et son application concrète pour le suivi de navires.

Enfin la conclusion résume les différentes contributions et en définit les limites avant de donner un aperçu des perspectives de recherche catégorisées par type, en rapport direct avec la thèse émise et de problématiques plus larges.

Chapitre 2

Gestion et traitement de données massives

Sommaire

2.1 Introduction

L'explosion du nombre de données à traiter dans de nombreux secteurs est due en partie à la multiplication de capteurs et senseurs de toutes sortes. Ceci a provoqué de fait l'émergence de nouveaux systèmes et de solutions différentes des traditionnelles bases de données relationnelles utilisées jusqu'à lors. En effet, devant les différents "V" abordés et soulevés par différents travaux, il est nécessaire de développer de nouvelles solutions pour permettre de stocker ces données massives de manière répartie sur différents noeuds formant un cluster, pour permettre de les traiter de manière distribuée et donc plus efficace que sur un système centralisé et à l'aide du modèle relationnel traditionnellement usité. Ce modèle trouvant ses limites face aux problématiques suivantes :

— *Volume* : Lorsque le nombre de données augmente, celles-ci doivent être stockées sur des ordinateurs plus puissants. Seulement quand la taille des données à manipuler dépasse les capacités d'un ordinateur, il est nécessaire de distribuer les données et les traitements sur plusieurs machines. Le modèle relationnel n'a pas été conçu pour opérer sur des systèmes distribués et le fait de combiner plusieurs tables sur différentes machines s'accompagne d'une perte en performance. De plus, certaines requêtes suivant un modèle relationnel

deviennent très coûteuses en temps parcourant parfois des champs ou effectuant des traitements inutiles. Dès lors, il n'est plus possible d'obtenir des solutions en un temps admissible ;

— *Vélocité* : Lorsque les données arrivent de manière continue et en grande quantité le modèle relationnel trouve ses limites. Les bases de données relationnelles utilisées usuellement pour extraire de la connaissance sont optimisées pour des opérations nécessitant de lire et d'accéder aux éléments. Par exemple, pour optimiser la lecture des données, des index sont construits pour trier les éléments et y avoir accès plus rapidement. Dans une optique de lecture, ils sont très performants, mais si des données sont ajoutées ou modifiées trop souvent il faut alors réordonner ces éléments continuellement et ceci peut-être très coûteux ;

— *Variété* : Les bases de données relationnelles sont caractérisées par des schémas statiques définissant les données. Ainsi en présence de données hétérogènes comme des images, des vidéos ou des objets dont les champs ne sont pas parfaitement prédéfinis et des données peuvent être manquantes, le modèle relationnel peut trouver ses limites.

Une autre problématique a émergé par la suite face à l'accumulation de données à traiter, une partie de celles-ci étant impures, incomplètes ou de mauvaise qualité. Il s'agit dans ce cas de *Véracité* des données, qui vient en complément des trois autres "V" et qui prend de plus en plus de place. En effet, plutôt que de traiter toutes les données entrantes, il est plus efficace d'examiner celles qui sont d'importance ou de meilleure qualité. Enfin, un problème de qualité de données peut permettre de détecter une anomalie au niveau des capteurs ou être révélateur d'un comportement suspect.

Dans ce contexte d'importante volumétrie des données et de la rapidité avec laquelle elles entrent dans le système, il n'est plus envisageable de les stocker et les traiter sur une seule et même machine, mais d'utiliser des systèmes de stockage et de traitement distribués. La distribution interviendra donc à deux niveaux d'abord au niveau des données et ensuite au niveau des processus de traitement, les deux pouvant être liés.

Nous définissons les deux termes offline et online qui seront utilisés dans la suite du manuscrit :

— *Approche Offline.* Cette approche concerne le stockage et le traitement de données massives sur un SGDB (Système de Gestion de Base de Données), s'exécutant en différé et permettant de gérer essentiellement l'aspect de volume décrit précédemment. Il s'agit d'une approche où les données sont généralement stockées sur disque et où des index sont utilisés pour accéder plus rapidement aux données. L'utilisateur pose des questions au système à l'aide de requêtes ces dernières étant exécutées une fois et nécessitant de parcourir l'ensemble

des données présentes (en l'absence d'index), fournissant une réponse de bonne qualité mais nécessitant des temps de calcul parfois longs en comparaison de systèmes temps-réel.

— *Approche Online.* Cette approche concerne la gestion et le traitement de données "au fil de l'eau" sur un SGFD (Système de Gestion de Flux de Données) à l'image de Stream [Arasu et al., 2016] ou Aurora [Abadi et al., 2003]. Les requêtes s'exécutent en temps-réel de manière continue ce qui permet de répondre à la problématique de vélocité précédemment introduite. Il s'agit d'une approche où les données sont assimilée "à la volée", traitées en mémoire et délestées lorsqu'elles ne sont plus utiles pour permettre d'en ingérer d'autres mais également de pouvoir traiter plus efficacement les données présentes dans le système. Cependant, en examinant seulement une partie des données ou en les délestant, le résultat des requêtes peut parfois être approximatif ou de qualité moindre par rapport à une approche offline.

TABLE 2.1 : Différences entre un système offline et un système online

	Système offline (SGBD)	Système online (SGFD)
Requêtes	Exécutées une fois	Continues (exécutées plusieurs fois)
Stockage	Sur disque (illimité)	En mémoire (limité, données délestées)
Qualité de réponse	Exacte	Approximative (en l'absence de toutes les données)
Temps de réponse	Différé	En temps-réel
Nature des données	Données conservées avec un format de données précis	Flux de données volatiles & impures
Traitement et accès aux données	Accès par biais d'index	Au fil de l'eau, utilisation de méthodes de résumés de données

Dans ce chapitre nous abordons d'abord la notion de gestion et traitement de données à l'aide d'une approche offline (cf section 2.2) censée répondre à la problématique de volumétrie, avant de passer en revue l'existant concernant les approches online c'est-à-dire pour la gestion et le traitement de flux temps-réel (cf section 2.3) correspondant à l'aspect de vélocité.

2.2 Gestion et traitement de masses de données

2.2.1 Gestion et stockage de données massives

Depuis quelques années le modèle relationnel et une architecture centralisée ont trouvé leurs limites pour stocker et traiter de gros volumes de données de manière efficace. Des travaux sur des modèles de données différents et des systèmes de base de données distribués ont donc émergé à l'image du HDFS [Shvachko et al., 2010] (Hadoop File System) et le framework Hadoop associé (dérivé de MapReduce [Dean and Ghemawat, 2004]).Nous pouvons catégoriser les différents travaux concernant cette thématique de recherche suivant les différents points suivants (cf. Figure 2.1) :

— *Gestion des ressources et de l'architecture distribuée.*
A l'image de Yarn [Vavilapalli et al., 2013] ou Mesos [Hindman et al., 2011], il s'agit ici d'avoir des mécanismes permettant de savoir où sont stockées les données sur l'architecture distribuée, de répliquer celles-ci de défaillance d'un des noeuds, mais aussi de pouvoir être inter opérable avec les différents paradigmes de traitements qui seront ensuite exécutés sur l'architecture distribuée pour traiter des requêtes ou algorithmes.

— *Communication et synchronisation des noeuds du cluster.*
À l'instar de Zookeeper [Hunt et al., 2010] ou ZeroMQ [1] le but principal de ces systèmes est de permettre la communication entre les différents noeuds de l'architecture, de permettre la synchronisation des noeuds en cas de mise à jour et de gérer les différents processus de manière efficiente.

— *Stockage et distribution des données.*
Ces travaux concernent plus la manière dont sont stockées les données et le modèle de données associées. À l'origine il s'agissait de systèmes de fichiers partagés entre différents noeuds [Shvachko et al., 2010], [Ghemawat et al., 2003] cependant l'intérêt d'un tel système était efficace simplement lorsqu'il était nécessaire de traiter l'ensemble des données [Dean and Ghemawat, 2004] plutôt que d'exécuter des requêtes sélectives. C'est pourquoi différents types de bases de données distribuées ont été définis par la suite [Chang et al., 2006], [Lakshman and Malik, 2010], [DeCandia et al., 2007] et ont été qualifiés de NoSQL, [Cattell, 2011] car différents des bases de données relationnelles. Ces systèmes seront discutés dans la suite du manuscrit et notamment leurs avantages et inconvénients dans le cadre du stockage et traitement de données spatiales.

— *Paradigmes de traitement et algorithmes distribués.*
À l'image de MapReduce [Dean and Ghemawat, 2004], de Spark [Zaharia et al., 2010] ou de Flink [Alexandrov et al., 2014] il s'agit de paradigmes de traitement permettant une

1. http ://zeromq.org/

exécution distribuée des algorithmes et applications. Ce sont des plateformes génériques qui peuvent être enrichies et implémenter des fonctions utilisateurs servant à résoudre des problèmes spécifiques. Nombre de travaux concernent le développement de nouveaux types de données ou d'opérations (opérateurs) spécifiques exécutés en parallèle grâce à ces différents frameworks et dans le cadre de ce travail de thèse nous étudierons les différentes extensions concernant le traitement de données spatiales et spatio-temporelles.

— *Langage haut niveau, analyse et fouille de données.*
A l'image de Hive [Thusoo et al., 2009], Pig [Olston et al., 2008] ou plus récemment Piglet [Hagedorn and Sattler, 2016] il s'agit ici de s'astreindre de la couche "Paradigmes de traitement et algorithmes distribués" précédente et de pouvoir formuler des requêtes ou des instructions à un plus haut niveau (à l'aide de langages proches du SQL), plutôt que de développer manuellement tous les processus. Certains autres travaux à l'image de Mahout [Owen et al., 2011], Samoa [De Francisci Morales and Bifet, 2015] ont pour but non plus seulement l'exécution de requêtes, mais également de faire de la fouille de données et d'appliquer des algorithmes de machine learning de manière distribuée sur un cluster [Landset et al., 2015].

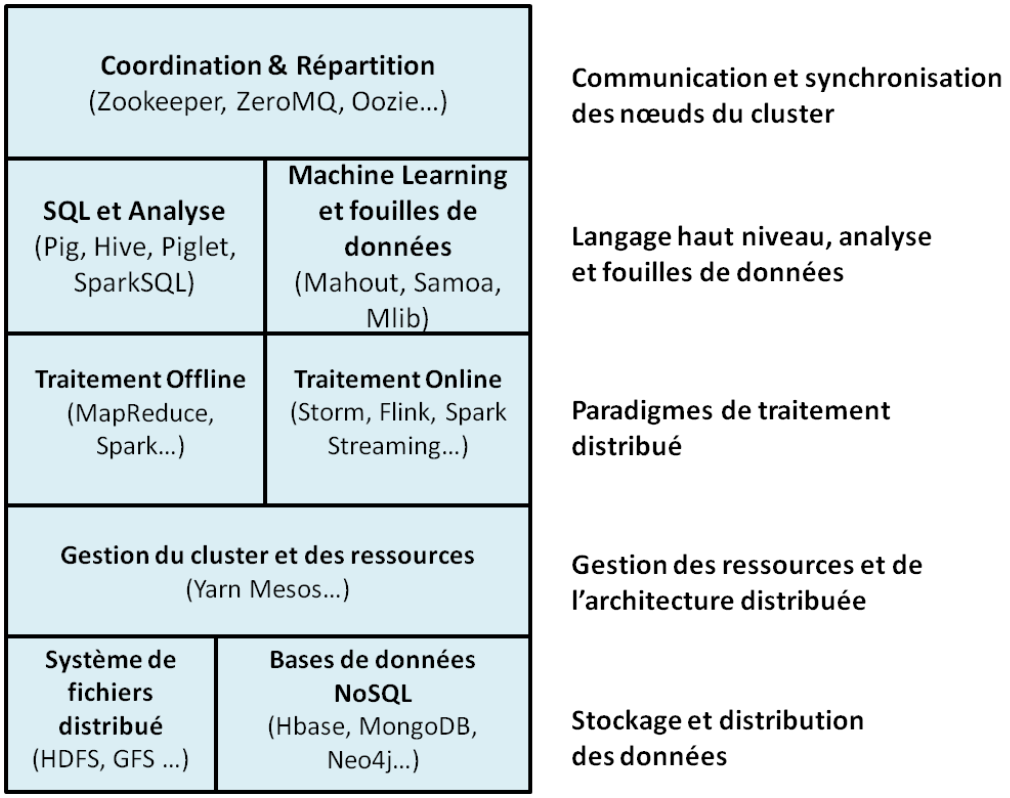

FIGURE 2.1 : Big Data Ecosystème et enjeux

Toute architecture dite "Big Data" intègre ces divers éléments et la plupart des travaux situent

leurs apports sur différents de ces aspects. Dans ce manuscrit, la réflexion et la contribution apportées concerneront principalement la couche relative à la gestion et au stockage (Storage Layer Figure 2.1) où nous nous intéressons notamment au modèle de données et partitionnement de celles-ci ; et celle relative au traitement distribué (Processing Layer 2.1) avec l'implémentation d'algorithmes spécifiques pour le traitement d'objets mobiles.

2.2.1.1 Le paradigme NoSQL

Le modèle relationnel proposé par Codd [Codd, 1970] donne un modèle clair pour stocker différents types d'information. Cependant un certain nombre de modèles non relationnels ont été développés, car le modèle relationnel n'était pas efficace face à certains problèmes spécifiques. Les limites du modèle relationnel évoquées précédemment conduisent donc à envisager une solution dite NoSQL pour stocker et traiter les données et prendre en compte la problématique de volume.

Le terme NoSQL utilisé aujourd'hui fait référence à une base de données distribuées avec un modèle de données différent du modèle relationnel. Approprié ou non, le terme englobe un nombre croissant de systèmes de bases de données distribuées non relationnelles. Le terme NoSQL reste toujours difficile à définir, car il ne caractérise pas des bases de données avec un même modèle de données ou les mêmes capacités. Les éléments de base de ces systèmes NoSQL peuvent être soit des documents, soit des colonnes, soit des clefs et des valeurs ... Plusieurs caractéristiques sont semblables entre tous ces systèmes NoSQL à savoir [Cattell, 2011] :

— *Absence de schéma prédéfini.*
Contrairement aux bases de données relationnelles, les bases de données NoSQL sont exemptes de schéma prédéfini, chacun des éléments peut avoir un nombre variable de n'importe quel type et des données avec des champs manquants peuvent être stockées de manière plus optimisée. Cette absence de schéma permet donc d'avoir un modèle de données plus flexible et d'éviter en partie un schéma prédéfini avec des champs pour lesquels des valeurs ne sont pas renseignées.

— *Plus performant sur des opérations d'agrégation et de lecture "simples".*
Ces systèmes ont des mécanismes d'interrogation des données assez sommaires et sont donc plus bas niveau par rapport un langage déclaratif proposé par les bases de données relationnelles (i.e. SQL). D'un côté cela permet d'effectuer les traitements uniquement sur les données concernées et de gagner en performance, mais nécessite le développement de techniques ou d'algorithmes spécifiques et plus bas niveau. De plus, il n'y a pas de jointures

nativement dans les systèmes NoSQL.

— *Scalabilité et cohérence partielle des données.*
Un enjeu important dans le cadre d'un système distribué est la notion de scalabilité sur plusieurs centaines ou milliers de machines au détriment de la cohérence des données assurée dans les bases de données relationnelles avec les règles ACID. À la place, ces systèmes proposent ce qu'ils appellent une cohérence éventuelle pour permettre de fournir une haute disponibilité des données. Ceci se fait de manière transparente grâce à la récupération de données et les processus qui continuent même en cas de panne d'un des nœuds. La cohérence des données n'est pas toujours préservée sur l'ensemble des nœuds du cluster. Cette caractéristique est la conséquence du CAP théorème défini par [Gilbert and Lynch, 2002], énonçant le fait que seulement deux des trois propriétés suivantes peuvent être respectées dans un système distribué : Cohérence, Disponibilité et Partitionnement (cf. Figure 2.2) :
  — *Cohérence.* Les transactions respectent les règles ACID, ce qui veut dire que tous les nœuds contiennent les données correspondant à l'état le plus récent.

  — *Disponibilité.* Toutes les données doivent être disponibles en lecture et écriture malgré les différents processus en cours sur le cluster ou les actions de plusieurs utilisateurs .

  — *Tolérance aux partitionnement.* Lorsque le nombre de nœuds présents sur une architecture distribuée augmente, les pannes ou les non-réceptions de messages ne sont pas rares. Il est donc nécessaire de mettre en place des techniques pour que le système et les processus engagés aboutissent même si des messages entre les composants sont perdus ou en cas de dysfonctionnement d'un des nœuds de l'architecture, pour que toutes les transactions puissent prendre fin.

Généralement, la cohérence est relaxée au profit de la disponibilité et la tolérance au partitionnement. Construire une base de données avec ces propriétés tout en respectant les règles ACID est difficile En effet, dans des systèmes distribués sur plusieurs centaines de machines les pannes et erreurs sont la norme et une cohérence éventuelle peut être suffisante. Les bases de données NoSQL fournissent certains avantages en termes de performance par rapport au modèle relationnel, mais en contrepartie perdent en robustesse. C'est pourquoi les règles ACID sont bien souvent assouplies pour avoir un système suivant une approche BASE [Pritchett, 2008], où Base signifie Basiquement Disponible, Soft State et cohérence éventuelle. Une base de données BASE est toujours disponible, mais pas toujours cohérente avec un état connu.

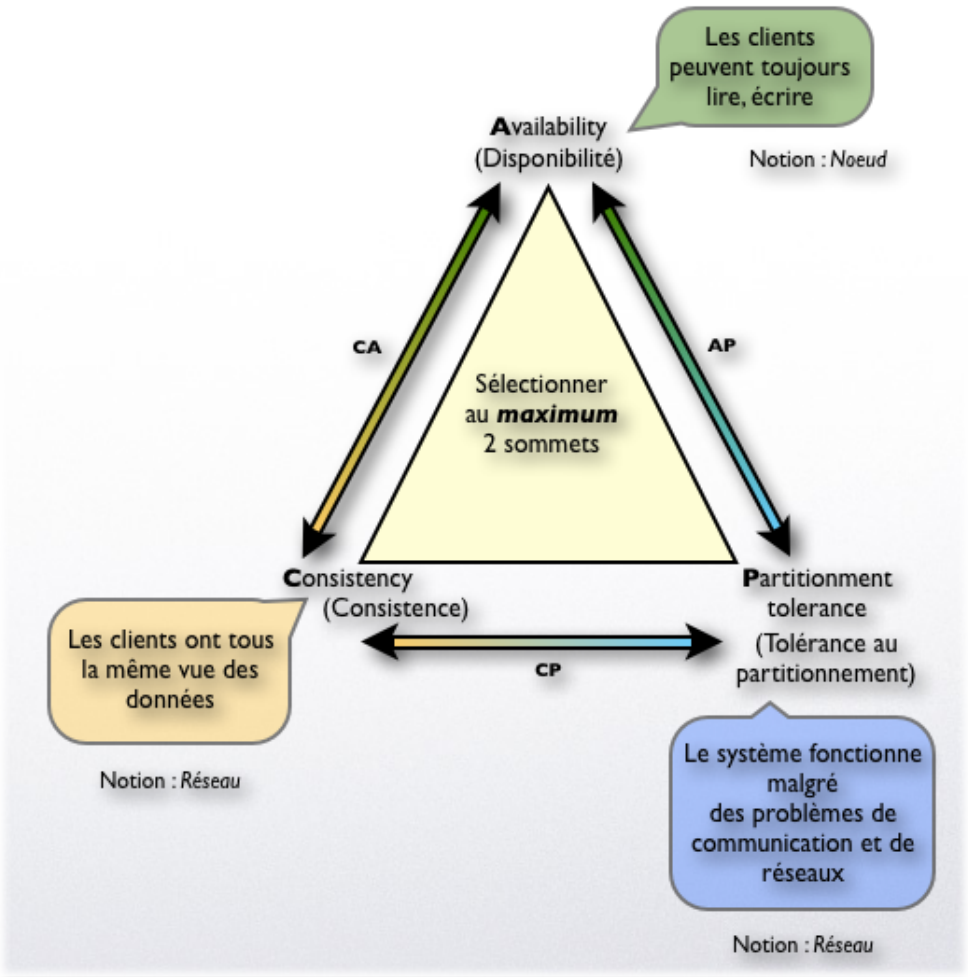

FIGURE 2.2 : Illustration du CAP Theorème

2.2.1.2 Modèle de données

Traditionnellement avec un modèle de données relationnel, il est nécessaire (en l'absence d'index existant) de parcourir chacune des lignes d'une table parfois pour une information concernant une ou seulement deux colonnes. Pour une base de données distribuée [Ozsu, 2007], les informations peuvent être réparties soit en affectant à chaque nœud une partie des lignes, soit une partie des colonnes (cf Figure 2.3). Dans le premier cas, il s'agit de partitionnement horizontal, dans le second de partitionnement vertical. Le premier cas est intéressant lorsqu'on a besoin d'informations nécessitant l'utilisation de différentes colonnes, le problème étant que s'il y a interdépendance des enregistrements (lignes) il faut distribuer ces lignes de manière à limiter le transit d'informations

entre les différents nœuds. Une répartition verticale quant à elle est efficace lorsqu'une requête repose sur une seule colonne, les difficultés intervenant quand il est nécessaire de combiner des informations sur différentes colonnes. Un découpage "hybride" est aussi souvent utilisé en assignant des "sous-lignes" (i.e. des regroupements de colonnes qui sont généralement interrogées ensemble lors de l'exécution de requêtes) aux différents nœuds. Les différents modèles de données utilisés par ces bases de données NoSQL reprennent en partie cette problématique introduite par l'étude des bases de données distribuées.

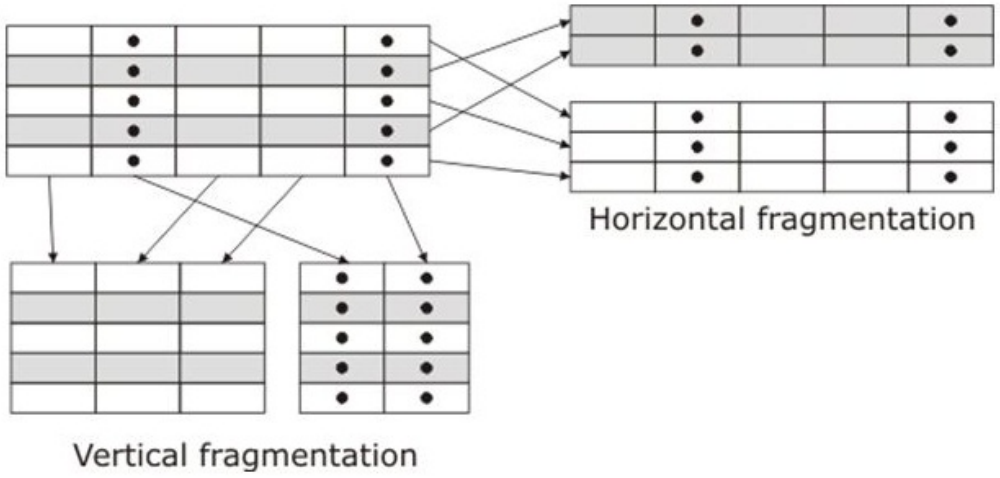

FIGURE 2.3 : Différents types de partitionnement [Worboys and Duckham, 2004]

Devant faire face aux limites du modèle relationnel, l'utilisation d'un modèle de données spécifique associé à certains traitements est nécessaire. Les différents modèles classés du plus basique au plus élaboré sont les suivants : modèle clef/valeur,modèle orienté document ,modèle orienté colonne, modèle de graphe (cf. Figure 2.4). Pour chaque modèle un ou plusieurs travaux sont discutés et étudiés pour montrer les tenants et aboutissants des solutions proposées. Certains de ces travaux présentent des extensions permettant d'effectuer quelques opérations spatiales à un niveau basique que nous décrivons donc dans le cadre de cette thèse abordant le traitement de données spatiales.

Modèle Clef/Valeur. Les bases de données clefs valeurs utilisent un modèle de données simple où celles-ci sont stockées suivant un système d'index clef-valeurs. Chaque clef est unique et les utilisateurs peuvent formuler des requêtes sur les données relatives aux clefs ou identifiants. Ce type de base de données doté d'une structure simple donne des réponses bien plus rapidement qu'une base de données relationnelle, car très performant sur des requêtes simples d'accès ou d'agrégation, mais

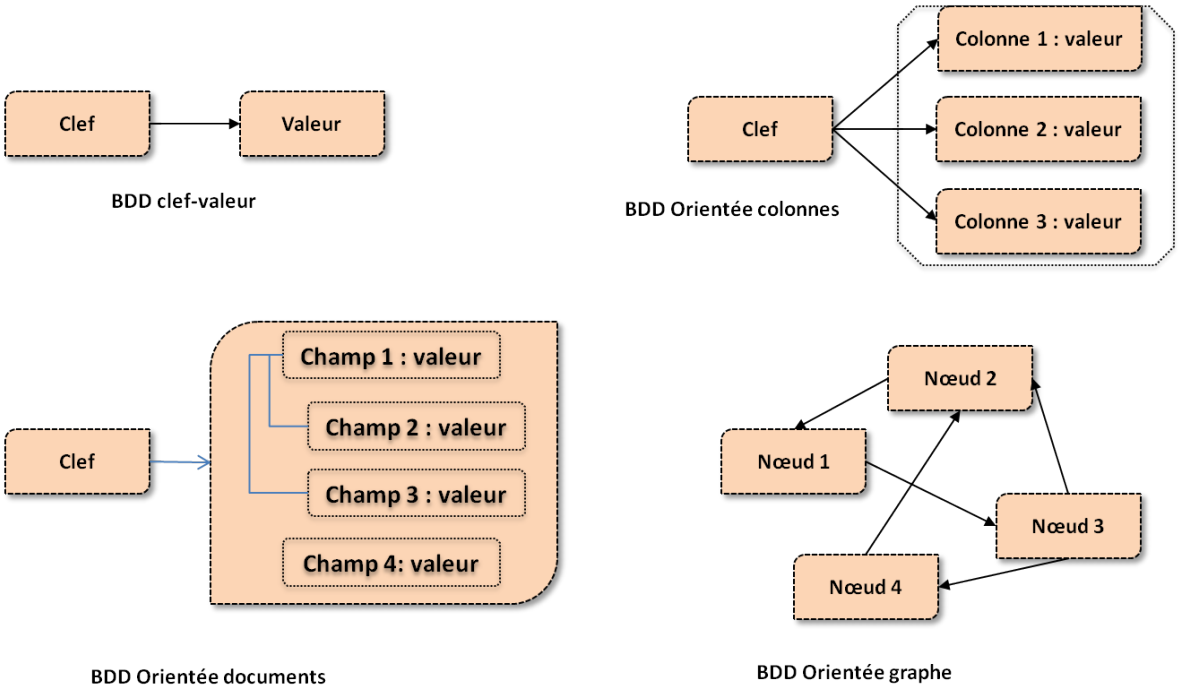

FIGURE 2.4 : Taxonomie des bases de données NoSQL

qui peut trouver ses limites lorsque les requêtes deviennent compliquées (i.e. lorsqu'il est nécessaire de requérir une partie des différentes valeurs correspondant à une clef et faire des opérations de jointure).

Depuis quelques années beaucoup de systèmes de base de données clefs valeurs ont été élaborés à l'image de Dynamo [DeCandia et al., 2007]. Dynamo est un système de base de données distribué clef-valeur utilisé principalement pour stocker et gérer les statuts de divers services vitaux pour la plateforme Amazon. Dans ce contexte, le modèle relationnel n'est pas suffisant, car limité en accessibilité des données et en scalabilité, Dynamo a été conçu pour résoudre ces problèmes avec de simples opérations de lecture et d'écriture. Dynamo assure son élasticité et une disponibilité des données par partitionnement et réplication des données. Dynamo assure une cohérence éventuelle du système avec des mises à jour faites et propagées de manière asynchrone sur l'ensemble des nœuds de l'architecture.

Modèle orienté document. Les bases de données orientées document permettent de stocker et d'organiser les données comme des collections de documents plutôt que comme des tables structurées avec des champs uniformes pour chaque enregistrement. Ce modèle peut être équivalent à celui clef valeur décrit précédemment si une donnée est stockée par document, ainsi la clef correspond au nom du document et les données stockées dans ce document aux valeurs. Mais le modèle orienté document est plus complet, car il permet de pouvoir mettre des données de même type dans un

document que l'on peut également caractériser par des identifiants, permettant d'avoir un "double index". Comparé au modèle clef valeur, les bases de données documents peuvent supporter des formes de données plus complexes.
Il s'agit d'un modèle semi-structuré à l'image de MongoDB qui permet de stocker des documents sous format d'objets BSON (pour binary JSON). Les documents sous MongoDB peuvent contenir plusieurs champs et valeurs où la valeur peut être un autre document, un vecteur de documents ou des types basiques tels que des double, des strings, ou des dates. La structure de données ne correspond pas schéma fixe (i.e problèmes de données manquantes, de champs en plus ou en moins) et les documents dans une collection peuvent être hétérogènes. MongoDB définit des index au niveau des collections. MongoDB utilise la réplication des données pour assurer la redondance et la haute disponibilité des données. MongoDB supporte le passage à l'échelle (i.e. sur plusieurs machines) par sharding automatique c'est-à-dire que les collections sont partitionnées et stockées sur différents nœuds de l'architecture tout en respectant une répartition équilibrée des données (ou aussi appelé "load balancing") sur l'ensemble des machines. Le système permet de formuler des requêtes sur tous les documents incluant les objets et vecteurs contenus dans ces documents et de créer des index sur les champs requêtables des documents. MongoDB peut être utilisé pour traiter des données spatiales notamment avec un format de fichier adapté (i.e. Geojson) et des index spatiaux 2D et 2D sphère. Ceci permet de traiter les requêtes spatiales suivantes : inclusion, intersection, et k plus proches voisins. Il n'y a par contre pas de support pour des opérateurs géométriques comme la génération d'enveloppe convexe ou d'opérations ensemblistes (union, différence etc ...). Enfin, MongoDB permet de gérer les types spatiaux suivants : Point, Polygone, Multipoint et collection de géométries.

Modèle orienté colonne. Le modèle orienté colonne, lui, permet à un niveau à peu près semblable que le modèle orienté documents de gérer les données. Dans un modèle relationnel toutes les données relatives à un objet et donc l'ensemble des champs affiliés sont stockés dans une même structure. En orienté colonne tous les éléments concernant un champ précis peuvent être conservés dans une même structure à la place, l'intérêt étant que sur une requête agrégative il sera nécessaire de parcourir un ensemble d'éléments plutôt que de parcourir un ensemble de lignes. En général les champs d'intérêt commun sont regroupés, car si chaque champ est stocké indépendamment dans une structure alors la situation devient analogue à celle rencontrée avec un modèle clef valeur.
Bigtable [Chang et al., 2006] est une base de données NoSQL orientée colonne distribuée et tolérante aux pannes ; Les données sont stockées dans des tables constituées de :

- Une clef associée à la ligne
- Une clef associée à la colonne ou à un ensemble de colonnes, groupées en familles de colonnes (i.e. des colonnes correspondant à des champs interdépendant ou souvent requêtés ensemble).
- Une estampille temporelle qui permet de stocker différentes copies de chaque cellule avec

TABLE 2.2 : Étude des différents modèles de données pour base de données NoSQL

	Exemples	Avantages	Inconvénients	Extensions Spatiales
Modèle orienté clef/valeur	Dynamo Ryak Voldemort	Rapide en insertion et lecture d'information "simples"	Ne permet pas de gérer des requêtes complexes (i.e. besoin d'informations sur plusieurs champs)	
Modèle orienté colonne	Bigtable Hbase Cassandra Accumulo	Partitionnement vertical et horizontal. Techniques de compression pour des champs redondants. Regroupement de champs souvent requêtés souvent en même temps.	Moins complet qu'un modèle orienté graphe. Champs "statiques" par rapport à un modèle orienté documents.	MD-Hbase Geomesa Geowave
Modèle orienté document	MongoDB CouchDB SimpleDB	Index à plusieurs niveaux (documents et clefs), permet de stocker des données hétérogènes plus facilement	Pas de format précis pour les documents.	MongoDB
Modèle orienté graphe	Neo4j Virtuoso	Gestion de relations, performant pour certains types de problèmes (réseaux sociaux, transport urbain) Permet de gérer l'hétérogénéité des données et de les lier, de gérer aussi la notion sémantique.	Partitionnement des données plus difficile. Temps de traitement plus long, malgré une meilleure qualité de réponse.	Neo4j Spatial

des estampilles distinctes et ainsi de conserver et maintenir les changements au cours du temps.

La scalabilité et la tolérance aux pannes sont permises par un partitionnement dynamique des données dans lequel chaque ligne est appelée un "tablet". Plusieurs copies de la même information sont conservées sur le GFS (Google File System)[Ghemawat et al., 2003] développé pour supporter de manière efficace le stockage des données sur une architecture distribuée. Le paradigme MapReduce est utilisé pour distribuer les différents traitements sur le cluster. D'autres systèmes ont été développés et sont semblables à BigTable, à l'image de Cassandra [Lakshman and Malik, 2010], de Hbase ou bien encore Accumulo. Ce type de système de base de données NoSQL est le plus couramment utilisé lorsqu'il s'agit de stocker de gros volumes de données tout en ayant un temps d'accès aux données qui soit "rapide".

Modèle de graphe. Le modèle de graphe semble le plus abouti, gardant la notion de lien entre éléments, il peut permettre de conserver les relations et d'introduire de la sémantique. Cependant, l'intégration des données est plus lente dans un modèle graphe, de plus distribuer les données et effectuer le traitement sur celles-ci n'est pas aussi évident que sur les modèles précédents.
Neo4j (Network-oriented database for java) est un système de gestion de base de données graphe développé en utilisant java. Le modèle de données utilisé est inspiré de la théorie des graphes [Diestel, 2012], composé de noeuds, de propriétés et de relations entre objets éventuellement connectés entre eux. Neo4j permet le stockage et la manipulation de données spatiales via la bibliothèque Neo4j spatial. Les types de données gérées sont les points, les polylignes, les polygones, ainsi que des ensembles de points, polylignes, et polygones. Neo4j spatial utilise des R-Tree (défini dans le chapitre suivant relatif au stockage et traitement de données spatiales) pour indexer les données, et fonctionne également avec des index 2D et 3D. Il implémente également les opérations topologiques suivantes contient, couvre, couvert par, croise, disjoint, intersection (spatiale), inclusion, touche et dispose des opérations géométriques suivantes distance, centroïde, enveloppe convexe ainsi que des opérations ensemblistes comme différence, intersection, union et différence symétrique.

Il convient dès lors d'adopter le modèle de données le plus performant pour retrouver et analyser de manière la plus satisfaisante possible les données (cf Table 2.2). En effet, le choix du modèle de données est un enjeu et dépend des données à manipuler. De nombreux travaux ont été réalisés pour mettre en avant certains modèles de données ou les comparer [Abadi et al., 2008], [Cattell, 2011]. Cependant, les extensions de ces travaux permettent de gérer la dimension spatiale des données à un niveau basique seulement (mis à part le modèle de graphe). C'est pourquoi d'autres travaux ont été réalisés pour étendre les systèmes précédents dits NoSQL pour la gestion et le stockage de

données spatiales, d'autres ont construit leur propre système de stockage et de traitement de données spatiales et cela sera abordé dans la suite du manuscrit (Section 3.3).

2.2.1.3 Paradigme de traitements pour un système fortement distribué

MapReduce [Dean and Ghemawat, 2004] a été proposé comme paradigme de traitement des données simple et flexible pour traiter les données sur des architectures distribuées. Le système permet de traiter des volumes massifs de données et fonctionne en divisant le processus de traitement en deux phases distinctes : map et reduce pour lesquelles l'utilisateur fournit deux fonctions nommées map et reduce associées. Chaque donnée est traitée de manière indépendante sur chacun des nœuds pendant la phase de map, puis avec la liste des clefs valeurs intermédiaires obtenues a lieu ensuite la réduction (ou phase de reduce) qui permet d'agréger les données sélectionnées ou modifiées pendant la phase de map. La suite décrit de manière plus précise le principe de fonctionnement de cette approche MapReduce composé des fonctions map et reduce dont voici les prototypes :

$map : (k1, v1) \leftarrow list(k2, v2)$

$reduce : (k2, list(v2)) \leftarrow list(k3, v3)$

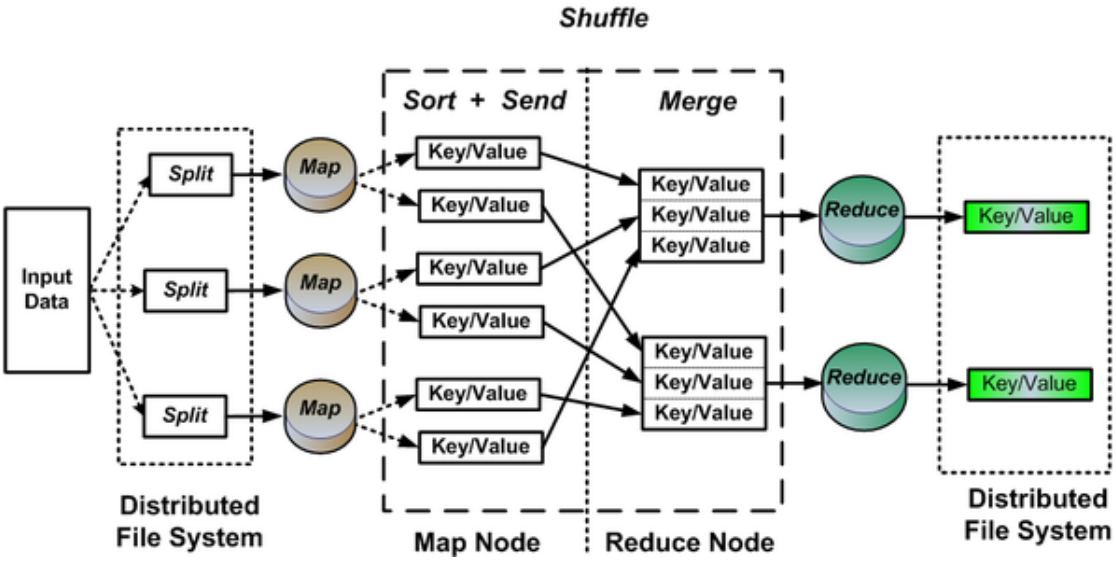

FIGURE 2.5 : Principe de fonctionnement de MapReduce (adapté de [Dean and Ghemawat, 2004])

Les types d'entrée et de sortie peuvent être de natures différentes avec la contrainte que le type d'entrée du reduce soit le même que celui de sortie du map. L'exécution d'un traitement MapReduce fonctionne de la manière suivante (cf Figure 2.5) :

— Le système divise le jeu de données d'entrée en différents sous jeux de données qui seront

traités de manière indépendante par différentes tâches de map de façon parallèle. Chaque appel du map produit une liste de clefs valeurs (k2,v2) associée au traitement de clefs valeurs (k1,v1) d'entrée.

— La sortie du map constitue une sortie intermédiaire, qui est transférée aux noeuds reduce par une redistribution qui assure que toutes les sorties du map associées à une clef soient redirigées sur un même noeud reduce.

— A chaque noeud reduce, tous les résultats intermédiaires reçus sont triés et regroupés par clef. Puis chaque reduce est exécuté simultanément sur chacun des noeuds reducer pour produire des résultats agrégés par clef.

Les données d'entrée et de sortie sont stockées sur le HDFS (Hadoop Distributed File System). Le système MapReduce s'occupe de la répartition des tâches, d'en surveiller l'évolution et de les exécuter de nouveau en cas de pannes ou dysfonctionnement.

Ainsi, tout traitement peut être modélisé sous la forme d'une suite de MapReduce successifs. Cela sans compter les défauts inhérents de MapReduce et du système de gestion de fichiers partagés associés à Hadoop. Un des manques du système de fichiers partagés GFS est le fait qu'il est privé d'index, ayant été prévu pour être parcouru entièrement, il est moins performant qu'une base de données classique pour la sélection et la lecture d'une donnée précise (i.e. qui serait repérée par un index). Un des problèmes majeur de MapReduce est que sous couvert de tolérance aux pannes, à chaque phase l'intégralité des données sont "lues" sur le disque, traitées puis réécrites, ce qui est particulièrement coûteux en termes de temps et peu approprié lorsqu'il s'agit de traiter des données en temps-réel ou d'exécuter des algorithmes itératifs.

Pour faire face aux défauts inhérents de MapReduce et notamment son système de lecture écriture sur disque, d'autres structures données ont été conçues à l'image des RDD (pour Resilient Distributed Datasets) une abstraction présente dans Spark permettant de traiter les données sur un système distribué de manière transparente pour un utilisateur, et ce en mémoire pour être plus performant que l'écosystème MapReduce et HDFS. Un RDD est une collection partitionnée d'enregistrements créée à partir d'opérateurs déterministes à partir de données stockées ou bien à partir d'autres RDD, à l'image de map, filter ou join. Les RDDs n'ont pas besoin d'être matérialisés tout du long puisqu'un RDD dispose de suffisamment d'informations sur la façon dont il a été produit à partir d'un autre ensemble de données pour pouvoir être recalculé. Ainsi le système de RDD est tolérant aux pannes. Ces RDD peuvent être conservés en mémoire avec la notion de persistance, il est possible de stocker et d'optimiser l'utilisation et les traitements sur ces flux de données RDD. Cette persistance peut être faite suivant différents vecteurs, soit en stockant en mémoire soit en stockant sur disque (via HDFS par exemple). Les données de RDD peuvent également être partitionnées entre les différentes machines de l'architecture distribuée en utilisant une clef. Pour effectuer un traitement

un RDD subit plusieurs transformations map, filter etc ... puis les données peuvent ensuite être exploitées ou "sorties de leur contexte" RDD pour obtenir un résultat (count, collect ...). Les RDD sont à préférer dans un contexte Batch où les opérations appliquées le sont sur chacun des éléments du flux RDD, et ainsi le traitement global peut être vu comme un graphe acyclique où chaque noeud correspond à un opérateur appliqué aux flux RDD.

D'autres paradigmes de traitements alternatifs à MapReduce ont vu le jour à l'instar de Pregel [Malewicz et al., 2010] qui utilise un paradigme de traitement BSP [Cheatham et al., 1995] pour le traitement des données. Pregel a été conçu plus spécifiquement pour traiter des données correspondant à de grands graphes, pour l'analyse de réseaux sociaux ou des algorithmes plus particuliers comme le pagerank de google. Une opération est exprimée sous forme de graphe acyclique orienté constitué de nœuds et d'arêtes. Chaque nœud est associé à une valeur et chaque arête orientée associant un nœud source à un nœud destination. Quand le graphe est construit, le programme produit une suite de traitements itératifs, exécutés de manière parallèle, chaque nœud traitant une fonction définie par l'utilisateur. Chaque nœud peut se modifier lui-même et le statut des arêtes suivantes, recevoir un message envoyé correspondant à la précédente phase itérative, envoyer le message au nœud suivant et ainsi modifier l'entièreté du graphe. Autrement dit, par rapport à l'approche MapReduce décrite précédemment, il y a interdépendance entre les calculs effectués et communication entre les nœuds en cours de processus.

Dryad [Isard et al., 2007] est un framework d'exécution distribuée pour effectuer des traitements parallèles de grandes masses de données. La structure de Dryad est un graphe acyclique orienté dans lequel chaque nœud représente un programme et chaque arête un flux de données. Dryad exécute les opérations sur les nœuds de l'architecture distribuée et transmet les données via les arêtes, incluant documents, TCP connexions et mémoire partagée. Durant l'opération, les ressources de l'opération logique du graphe sont affectées automatiquement à des ressources physiques. Avec Dryad, un programme central coordonne l'exécution de toute opération sur un cluster de nœuds en réseau. D'abord, un graphe décomposant les différentes tâches à effectuer est généré et est ensuite affecté à l'ensemble des nœuds sur l'architecture en fonction des ressources disponibles.

Enfin, d'autres systèmes permettent d'exécuter des tâches en parallèle à un niveau plus proche de la machine comme MPI (Message Passing Interface) [Foster and Karonis, 1998] usuellement utilisé pour le calcul scientifique et le HPC(High Performance Computing). Ceci en décomposant la tâche principale en différents sous-traitements indépendants, mais en laissant à l'utilisateur le soin de partitionner les données sur l'architecture et d'éventuellement déplacer les données, là où la communication entre les nœuds et la répartition des données et des traitements se fait de manière transparente pour l'utilisateur avec un Hadoop/MapReduce.

Dans cette section, nous avons présenté les différents enjeux associés au stockage distribué de données sur un cluster, pour le traitement de données massives, qu'il s'agisse des problématiques de cohérence intervenant avec des données répliquées, du choix du modèle de données essentiel pour les performances en termes de temps de réponse à une requête ou bien du paradigme de traitement utilisé pour traiter ces données de manière efficace sur une telle architecture dite NoSQL. Cependant, les solutions et points ayant été abordés concernent uniquement la gestion de l'aspect de volume précédemment introduit et ne permettent pas de répondre à l'aspect de vélocité.

2.3 Gestion et traitement de données en temps-réel

Dans la section précédente, nous avons vu un ensemble de solutions pour le stockage et le traitement de données massives, cependant ces systèmes sont orientés pour un traitement a posteriori et ne permettent pas d'ingérer et traiter des données en temps-réel. C'est pourquoi dans la partie suivante dans la section suivante, nous abordons la notion de système temps-réel traitant l'aspect vélocité introduit en début de chapitre.

2.3.1 Gestion de flux de données, modèle et définitions

Contrairement aux systèmes de gestion de bases de données (SGBD), les systèmes de gestion de flux de données (SGFD) (ou DSMS en anglais pour Data Stream Management System) introduits aux débuts des années 2000 ont pour but de traiter les données en temps-réel et de fournir un système réactif qui prévienne l'opérateur en cas de dysfonctionnement (cf Table 2.1). Ces systèmes se sont développés notamment avec la multiplication de capteurs de toute sorte et le besoin d'obtenir des informations en temps-réel, l'émergence des "Big Data" a suivi ce même essor, l'aspect volumétrie ayant été abordé maintes fois et en partie résolu les solutions mises en place et étudiées ont vite évolué pour adresser cette problématique de vélocité.

Dans [Golab and Özsu, 2003], les auteurs définissent un flux de données de cette manière "A data stream is a real-time, continuous, ordered (implictly by arrival time or explicitly by timestamp) sequence of items". Ces flux de données sont par nature volatiles (i.e. ils ne sont conservés que pour une période limitée de temps) et impurs (en présence de capteurs différents cas peuvent se produire des données non reçues, erronées ou reçues avec un décalage de temps). Dans le cas d'un grand volume de données à traiter en mémoire, celle-ci pouvant arriver à saturation le système doit être capable de délester une partie des données, permettant ainsi de réduire la charge de stockage et de traitement [Tatbul et al., 2003],[Babcock et al., 2004].

Ces données arrivent en continu et doivent être traitées au fil de l'eau suivant un modèle dit push-based [Arasu et al., 2004]. Le rôle des SGFD (systèmes de gestion de flux temps-réel) est d'exécuter des requêtes dites continues et de détecter des événements spécifiques pouvant se produire. Le système est réactif (on parle aussi de modèle DAHP database active, human passive), c'est-à-dire que le système de traitement est prépondérant et l'utilisateur observe les résultats émis au fur et à mesure.
Dans le cas d'un SGBD les données sont insérées et/ou mises à jour de façon périodique. La priorité étant de pouvoir lire et accéder aux données triées, il s'agit ici d'un modèle dit pull-based où c'est cette fois l'utilisateur qui questionne la base de données. Il s'agit d'un système HADP (human active database passive) où les données sont requêtées par l'utilisateur et ces requêtes ne sont souvent exécutées qu'une fois et non en continu comme dans le cadre d'un SGFD.

Les requêtes sur les flux de données s'exécutent de manière continue alors que les données sont assimilées continuellement par le système. Au vu de cette caractéristique, les données interrogées pour répondre à ces requêtes continues sont sélectionnées en fonction d'une période de temps considérée (i.e. traditionnellement en utilisant des fenêtres temporelles).

Comme les flux de données ne peuvent être stockés en mémoire, une alternative consiste à considérer uniquement les données les plus récentes en utilisant des "fenêtres" et l'exécution des requêtes portera sur ces fenêtres qui peuvent être de différentes natures.

La nature des fenêtres peut tenir dans un premier temps au fait qu'elles prennent en compte un aspect physique (essentiellement une plage temporelle) ou un aspect logique (i.e. le nombre d'éléments ou d'événements entrants dans le système).

Définition (Fenêtre physique). *Une fenêtre physique (ou temporelle) est définie sur un intervalle de temps allant de t_0 à t_n. Un élément e d'un flux de données appartient à la fenêtre si $\tau_e \in [t_0, t_n]$.*

Définition (Fenêtre logique). *Une fenêtre logique est définie par rapport à un nombre k d'éléments. Cette valeur k correspond à la taille de la fenêtre. Les k éléments les plus récents sont conservés et concernent l'état actuel du système. Le "slide" définit le nombre de nouveaux éléments reçus avant de générer de nouveau un résultat sur la fenêtre mise à jour.*

La nature de ces fenêtres peut aussi dépendre de la direction des mouvements de leurs points de début et de fin. Quand le début et la fin d'une fenêtre sont fixés, il s'agit d'une fenêtre fixe. Lorsque les points de début et de fin changent au fur et à mesure que les données arrivent dans le système le

terme de fenêtre glissante peut être employé. Lorsqu'un seul des points est fixé (généralement celui de début) il s'agit d'une fenêtre point de repère.

Définition (Fenêtre fixée). *Une fenêtre fixée est définie sur un intervalle de temps allant de t_d à t_f avec t_d et t_f fixés arbitrairement dans le temps. Un élément e d'un flux de données appartient à la fenêtre si $\tau_e \in [t_d, t_f]$.*

Définition (Fenêtre point de repère). *Une fenêtre point de repère est définie sur un intervalle de temps allant de t_d à t_f avec t_d fixé arbitrairement dans le temps et t_f qui correspond à la date courante. Un élément e d'un flux de données appartient à la fenêtre si $\tau_e \in [t_d, t_f]$.*

Définition (Fenêtre glissante). *Une fenêtre glissante est définie sur un intervalle de temps allant de t_d à t_f et réévaluée toutes les τ périodes. Un élément e d'un flux de données appartient à la fenêtre si $\tau_e \in [t_d, t_f]$.*

Enfin, ces fenêtres peuvent également être distinguées en fonction des périodes de mise à jour. En effet, la fenêtre peut être réévaluée à l'arrivée de chaque donnée entrant dans la fenêtre ou à l'inverse seulement lorsqu'une durée de temps a été dépassée.

Lorsque le paradigme de fenêtre n'est pas suffisant pour traiter les flux de données, d'autres techniques sont utilisées pour alléger le volume de données à traiter, à l'image du délestage de données, le résumé de données ou encore l'échantillonnage de données. Un résumé de flux permet de conserver une trace des événements apparus dans un flux de données. Par ailleurs, la conservation d'un résumé du flux permet de réaliser des avancées dans des domaines tels que la gestion des données en retard, les techniques de délestage de données ou encore l'approximation de certaines requêtes contenant des opérateurs bloquants. Les synopses sont des structures de résumé de flux de données ayant pour objectif de traiter une requête particulière sur les événements du flux. Ces requêtes sont définies avant l'arrivée du flux.

Dans un système de gestion des flux, une requête est la succession d'opérateurs organisés en un graphe acyclique orienté ou workflow. Chaque opérateur est connecté à un flux de données issu d'un opérateur précédent et chaque opérateur dispose d'une "mémoire" correspondant aux données dont il a besoin pour effectuer les traitements dont il est responsable. Le graphe orienté utilise un paradigme de traitement dans lequel chacun des résultats produits par un opérateur est transmis aux opérateurs suivants relatifs à l'exécution d'une requête continue. Toutes les données sont traitées

en mémoire dont certaines d'entre elles sont conservées ainsi que certains résultats ou agrégats de données et partagés entre différentes requêtes exécutées simultanément pour gagner en efficacité. Dans ce genre de système les requêtes sont effectuées de manière continue et lèvent des alertes lorsqu'un problème ou un événement se produit contrairement à un système offline où la requête est exécutée une seule fois.

En ce qui concerne les traitements cela pose problème à différents niveaux, d'abord concernant l'ordonnancement des opérateurs. En effet, certains opérateurs peuvent être interchangés et l'ordre dans lequel ils sont exécutés est déterminant pour améliorer les performances du système. Le système doit pouvoir construire en fonction de la requête entrant dans le système la succession d'opérateurs la plus appropriée pour répondre au mieux à la requête.

Ensuite, un enjeu intervient en termes de gestion des flux non uniformément répartis. Les requêtes sont exécutées en continu, et le plan de requête préalablement défini pour répondre à cette requête peut ne plus être optimisé lorsque le flux de données entrant est trop grand. Il est alors nécessaire de redistribuer ce flux sur deux opérateurs ou bien de ne pas traiter une partie des données qui seront délestées par le système.

Enfin, la gestion de requêtes multiples et/ou concurrentes, des requêtes qui sollicitent les mêmes données ou des parties de traitement en commun, l'enjeu est de repartir au mieux les traitements et de les factoriser au maximum lorsque cela est possible à l'image du "shared-execution paradigm" proposée dans Telegraphcq [Chandrasekaran and Franklin, 2003].

Une partie des systèmes de gestion de flux de données étendent et modifient les langages de requêtes comme SQL pour prendre en compte de manière efficace des requêtes continues sur des flux de données. Les requêtes sont évaluées de façon continue et produisent des résultats incrémentaux sur le temps. Les opérateurs sur des requêtes continues (sélection, projection, jointures, agrégation etc) s'exécutent sur les tuples rentrant dans le système au fur et à mesure et ne supposent pas un flux de données fini, ce qui a des implications négatives. Deux types de requêtes continues sont distinguées :

Définition (Requête monotone). *Une requête continue Q ou un opérateur est dit monotone si $Q(\tau) \subseteq Q(\tau')$ pour tout $\tau \leq \tau'$.*

Une simple sélection sur un flux de données est un exemple de requêtes monotones a chaque instant t quand un nouveau tuple arrive dans le système, soit il satisfait le prédicat de sélection soit il ne le satisfait pas, et tous les résultats (tuples) précédemment retournés appartiennent à Q(t).

Définition (Requête bloquante). *Un opérateur ou une requête continue Q est dite bloquante*

si $Q(\tau) \not\subseteq Q(\tau')$ *pour tout* $\tau \leq \tau'$.

Certains opérateurs (produit cartésien, jointure, union, agrégation spatiale) requièrent d'avoir l'entièreté des données pour être évaluées. Ces opérateurs dits bloquants ne vont produire aucun résultat avant que le flux de données ne se finisse, ce qui constitue évidemment une limitation sérieuse. Pour obtenir des résultats de manière continue et sans pour autant stopper le flux entrant de données, les opérateurs bloquants peuvent être considérés simplement sur un flux de données contraint (i.e. une fenêtre temporelle) et d'utiliser un processus incrémental permettant de mettre à jour le résultat de la requête au fur et à mesure que les flux rentrent dans le système.

2.3.2 Etat de l'art des systèmes de gestion de flux de données

Les premiers DSMS comme TelegraphCQ [Chandrasekaran et al., 2003], NiagaraCQ, Aurora [Abadi et al., 2003], Borealis [Cangialosi et al., 2005] et Stream [Arasu et al., 2016] ont été conçus spécialement pour l'exécution de requêtes avec une faible latence sur des flux de données. Ils fournissent un langage de requête déclaratif qui supporte des opérateurs relationnels (e.g selection, projection, jointure etc) sur des fenêtres de flux de données. Cependant, ils ont des capacités de scalabilité limitées et ne permettent pas de gérer des données anciennes ou archivées.

STREAM [Arasu et al., 2016] a été développé à l'université de Standford. Il a été conçu à l'origine pour gérer les environnements et flux de données dont le chargement est fluctuant et très changeant. C'est pourquoi il a été construit pour s'adapter en pleine exécution en cas de restrictions ou de contraintes sévères (augmentation du nombre de données à traiter, requêtes entrantes et concurrentes). Stream fournit un langage déclaratif haut niveau appelé CQL pour définir des requêtes continues sur les flux de données entrants dans le système. Le système temps-réel fournit un ensemble de techniques pour optimiser les performances du système par exemple en partageant les synopses pour des requêtes qui sont proches, l'affectation globale des opérateurs qui réduit l'utilisation de la mémoire dans le cas d'un pic de flux, une surveillance et un contrôle du traitement des requêtes dont le processus est évalué et modifié en cours d'exécution en fonction des performances du systèmes.

TelegraphCQ [Chandrasekaran et al., 2003] est l'un des premiers systèmes de gestions de flux temps-réel développé à l'université de Berkeley. Son implémentation a été faite "au-dessus" de de PostgreSQL et Teleghraph un système pour la gestion de flux et processus adaptatifs. TelegraphCQ a été conçu pour supporter le traitement de requêtes continues aussi bien sur des flux que les tables stockées en base de données. Il fournit un langage déclaratif StreaQuel qui supporte des requêtes formulées à la manière du SQL en utilisant la notion de fenêtres. Les requêtes sont ainsi sur

l'environnement de Telegraph doté d'une logique adaptative utilisée pour "router" efficacement les données aux opérateurs.

Aurora [Abadi et al., 2003] a été conçu comme un système centralisé pour de flux temps-réel et est inefficient sur les aspects de scalabilité et de tolérance aux pannes. Ce système peut exécuter des requêtes sur des flux infinis à l'aide d'une algèbre nommée SQuAl définie pour le traitement de flux. L'algèbre peut permettre de traiter aussi bien les données relatives aux flux que des données qui seraient stockées dans des tables. Enfin Aurora supporte le délestage de données en supprimant du flux des tuples de manière aléatoire quand le système est surchargé. Plus tard, Borealis [Cangialosi et al., 2005] a été introduit pour remplacer Aurora combinant le coeur de traitement temps-réel d'Aurora avec les fonctionnalités de distribution de Medusa. "Au-dessus" des fonctionnalités d'Aurora, Borealis fournit plusieurs améliorations d'importance comme un traitement distribué via un parallélisme inter opérateur, un mécanisme de tolérance aux pannes pour fournir un accès aux données et l'intégrité de celles-ci en cas de pannes, un mécanisme de "révision" permettant de modifier une requête à la volée.

Des plateformes de traitement temps-réel ont émergé plus récemment avec la problématique dite de "Big Data". Là où les systèmes présentés précédemment avaient pour but de traiter des requêtes et trouver des plans de requêtes efficients pour pouvoir y répondre le plus rapidement et efficacement possible, les plateformes temps-réel "Big Data" fournissent un environnement permettant de développer et exécuter des applications sur des flux temps-réel tout en supportant les contraintes de scalabilité et de tolérance aux pannes abordées précédemment. Une abstraction naturelle pour le traitement de flux temps-réel est un modèle de graphe (ou graphe direct acyclique) des opérateurs s'enchaînant les uns à la suite des autres plutôt qu'une succession d'opérations MapReduce. Le graphe contient les données sources qui émettent de manière continue des flux de données consommés et traités par les noeuds en charge des flux entrants. De nombreux systèmes temps-réel ont ainsi vu le jour à l'image de Storm, S4, Héron ou IBM Infosphere Stream, dans la suite nous présentons quelques-uns d'entre eux qui sont représentatifs où servirons à la compréhension de ce manuscrit.

Twitter Storm [Toshniwal et al., 2014] est une plateforme distribuée de traitement de flux temps-réel qui facilite le développement d'applications distribuées temps-réel. Avec Storm l'ensemble du processus de traitement est transformé ou traduit en une topologie logique (cf Figure 2.6). Une topologie est un flux de données modélisé par un graphe direct acyclique qui représente le traitement requis. Ceci permet au développeur de se concentrer seulement sur l'enchaînement des processus de traitement et non sur les aspects distributifs.Les noeuds de la topologie représentent les éléments de traitements tandis que les arêtes représentent les flux de données entre chacun des traitements particuliers sur ces flux de données. Comme les topologies sont utilisées comme des graphes acycliques,

les flux de données depuis les "spouts" vers les "bolts" et flux inverses ne sont pas possibles. Un "spout" joue le rôle de source d'entrée pour la topologie. Storm permet une bonne scalabilité avec un partitionnement horizontal et utilise ZeroMQ pour la communication entre les noeuds ce qui assure une faible latence et garantit le traitement des messages. Dans le cas d'une panne du système, les messages sont automatiquement réassignés en "rejouant" le flux de données.

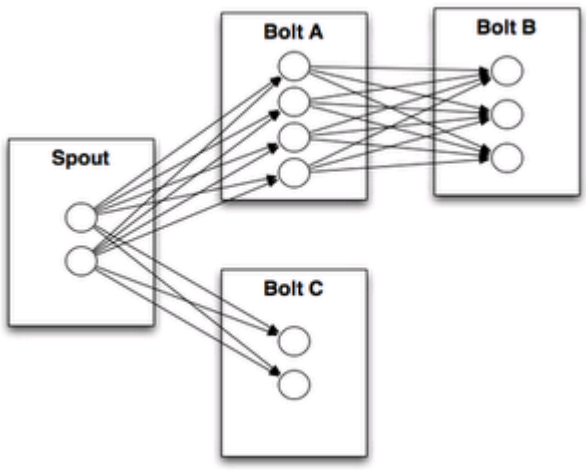

FIGURE 2.6 : Fonctionnement d'une topologie via Storm [Toshniwal et al., 2014]

Heron [Kulkarni et al., 2015] a été développé pour remplacer Storm et reste compatible avec les topologies précédemment présentées. Heron a été conçu dans le but de contrebalancer certains défauts inhérents de Storm notamment des goulets d'étranglement en termes de performance. Des enjeux de performances critiques interviennent durant le traitement des tuples du fait d'un "mauvais routage" des flux de données qui doivent être multiplexés et démultiplexés via multiples files d'attentes et processus sur différentes couches. Ceci provoque des conflits et rend difficile l'affectation des tâches sur les différents noeuds et processus. Heron contrevient à cela en utilisant une structure hiérarchique entre les noeuds pour la communication.

Millwheel [Akidau et al., 2013] est un framework pour le traitement de flux temps-réel et le développement d'applications nécessitant une faible latence. Il définit un modèle de programmation qui permet aux développeurs d'écrire des applications logiques représentées par des graphes directs acycliques où les enregistrements sont délivrés de manière continue via les arrêtes du graphe.

S4 (simple scalable streaming system) [Neumeyer et al., 2010] est une plateforme distribuée qui permet le traitement continu de flux de données non bornés. Son modèle de traitement dérive du système de clef utilisé avec MapReduce pour le partitionnement et la répartition des flux de données à traiter. Un traitement est construit comme un graphe de PE(Processing Element) interconnecté par des flux de données, dérivant du modèle d'acteur [Hewitt et al., 1973]. S4 est basé sur un modèle

"push-based" où les événements sont transférés aux Processing Elements (c'est à dire des éléments chargés du traitement des données reçues), chacun des PE étant défini par l'utilisateur. Les messages considérés comme événements sont transmis d'un PE à un autre. Les événements ou résultats découlant de traitements précédents sont consommés par chacun des PE. Les PE interagissent les uns avec les autres aussi bien en étant considérées comme des sources que comme des sorties.

IBM InfoSphere Stream [Biem et al., 2010] est une plateforme conçue pour le traitement de flux de données. InfoSphere fournit un langage déclaratif (SPL) pour le développement d'applications temps-réel. SPL permet aux développeurs de construire des applications sans se soucier des problématiques associées à la distribution des traitements. Des opérateurs performants peuvent également être ajoutés en c++ ou java. Une tâche dans une application est ainsi constituée d'un ou plusieurs PE (Procesing Elements) comme c'est le cas pour S4, ces derniers communiquant via "message passing".

Spark Streaming [Zaharia et al., 2013] suit une approche micro-batch où les flux de données sont divisés en petits ensembles de données (mini batches) et considérés comme Resilient Distributed Datasets. Comme nous l'avons vu précédemment, les RDD sont une abstraction permettant le traitement en mémoire de manière distribuée tout en étant tolérant aux pannes [Zaharia et al., 2012] et les résultats des traitements sont conservés en mémoire pour répondre aux problématiques temps-réel. Spark Streaming utilise la notion de DStream (pour Discretize Streams) [Zaharia et al., 2013] représentant un flux continu de données, l'intérêt de ces DStream est qu'ils peuvent être obtenus à partir de données anciennes et déjà stockées comme ils peuvent être le résultat d'opérateurs s'effectuant sur des flux de données entrants.

Kafka [Kreps et al., 2011] est un système qui permet la gestion et l'échange des messages et flux temps-réel. Utilisé par Linkedin, il s'interface avec le reste de l'architecture développée (i.e. Samza , il permet de faire circuler les flux et conserver les données pour faire du "rejeu"). Kafka est un système dit *publish and suscribe* qui permet de lire et écrire des flux de données comme un système de messages. Il permet aussi de traiter les données en temps-réel et d'écrire des applications adaptées à ce contexte à un niveau "basique", son rôle principalement étant l'affectation des messages et la gestion de ces derniers. Enfin il permet de stocker les données sous forme de flux de messages de manière sûre sur un cluster en répliquant les données et avec de la tolérance aux pannes. Un message est composé d'une valeur, d'une clef et d'une estampille temporelle. Kafka organise les messages en catégories appelées topics, concrètement des séquences ordonnées et nommées de messages. Les topics ne sont pas modifiables, le système permettant seulement l'ajout de messages. Un client Kafka ne peut pas modifier ou supprimer un message, ne peut pas modifier l'ordre des messages ou insérer un message dans un topic ailleurs qu'à la fin. Il ne peut pas non plus créer ou supprimer un topic. Un émetteur ajoute des messages à la fin des topics de son choix. De son côté, le consommateur lit des

topics à partir d'un index ou d'une estampille temporelle donnés, dans l'ordre d'arrivée des messages, c'est-à-dire du plus ancien message au plus récent, et sans s'arrêter. Si un consommateur a traité tous les messages d'un topic, il y reste connecté pour recevoir les nouveaux messages insérés par les émetteurs. Les messages sont lus et non consommés, les messages sont donc conservés. Comme les topics sont immuables, ils peuvent être lus simultanément par de multiples consommateurs. Chacun d'eux en est à un index différent et lit à une vitesse différente. Kafka peut donc être utilisé à la fois comme source et stockage de messages et données. Samza [Noghabi et al., 2017] a été développé dans la lignée de Kafka pour gérer la partie traitement temps-réel et fouille de données, tandis que Kafka à un rôle d'affection et de stockage des messages par topics sur différents noeuds pouvant être sources et/ou consommateur. Liquid [Fernandez et al., 2015] utilise Samza pour la partie traitement et la partie distribution et affectation des messages grâce à Kafka , permettant de gérer à la fois le stockage et le traitement des flux de données entrants dans un système.

2.4 Conclusion

Ce chapitre aborde les différentes problématiques associées au stockage et au traitement de volumes massifs de données, s'accompagnant des problématiques relatives à la distribution des données (problème de cohérence, réplication de données pour être tolérant aux pannes) et la distribution des traitements (paradigme de traitement distribué associé et dépendant du type de données et de leur répartition sur l'architecture). Cependant ces systèmes ne permettent pas de gérer convenablement des données à caractère spatial ou spatio-temporel (i.e. pas de mécanismes de stockage spécifique, pas d'index ni d'opérateurs topologiques ...). Ensuite, nous avons rappelé les éléments relatifs à une approche online pour la gestion de l'aspect de vélocité, et passé en revue l'ensemble des solutions pour traiter des flux de données de manière efficace et notamment de manière distribuée. Là encore, ces systèmes et solutions ne permettent pas de traiter des données d'objets mobiles nécessitant des processus différents pour gagner en efficacité. Dans la suite, nous nous intéresserons à l'étude des systèmes spécifiquement conçus pour le stockage et le traitement de données spatiales et spatio-temporelles. Certains hériteront des concepts vus dans ce chapitre permettant le stockage et le traitement massif de données et d'autres permettront de prendre en considération la dimension spatio-temporelle sans pour autant permettre de gérer l'arrivée massive de données pouvant éventuellement avoir lieu en temps-réel. Pour mettre en place une solution pratique il s'agira donc soit de prendre un système permettant de gérer des forts de volumes de données arrivant en temps-réel et de l'étendre pour gérer les dimensions spatiale et temporelle, soit de choisir un système spécifique au stockage et au traitement de données spatio-temporelles et faire en sorte qu'il puisse être efficace en cas de grande volumétrie de données devant être ingérées et

traitées en temps-réel.

Chapitre 3

Gestion et traitement massifs de données spatio-temporelles

Sommaire

3.1 Introduction

Nous avons vu dans la partie précédente les problématiques relatives à l'utilisation d'un système dit "Big Data" pour gérer aussi bien l'aspect volumétrie et que l'aspect vélocité. La gestion et le traitement d'objets mobiles n'ont pas échappé à cet essor de données. En effet les techniques utilisées traditionnellement pour la collecte, le stockage et l'interrogation de trajectoires héritent des travaux sur les bases de données pour objets mobiles (Moving object Databases ;MOD). Ces travaux sont presque exclusivement basés sur un modèle relationnel et intègrent ou exploitent des extensions pour la gestion d'objets mobiles (types et opérateurs spatiaux, composante temporelle, index associés aux objets mobiles) comme Hermes [Pelekis et al., 2006] ou Secondo [de Almeida et al., 2006]. Ces données d'objets mobiles stockées et archivées peuvent être exploitées à l'aide de différentes techniques de fouille de données extraction, agrégation, clustering, fusion [Giannotti et al., 2007]. Cependant, ces approches sont limitées dès lors qu'il s'agit de stocker de forts volumes de données, de les traiter de manière efficace ou lorsque le système doit répondre en temps-réel [Salmon et al., 2014], [Eldawy and Mokbel, 2015], [Vatsavai et al., 2012].

Après avoir vu dans un premier temps l'ensemble des généralités sur les données spatiales et spatio-temporelles (Section 3.2), nous abordons et décrivons l'ensemble des systèmes dédiés au traitement de données spatiales ou spatio-temporelles soit dans un contexte de traitement massif de données (Section 3.3), soit dans un contexte de traitement temps-réel (Section 3.4).

3.2 Généralités sur le stockage et le traitement de données à caractère spatial et spatio-temporel

Avant l'émergence de cette problématique de gros volumes de données, déjà il existait un besoin de stocker et traiter les données spatiales et d'objets mobiles d'une manière différente. Initiée entre autres par Guting, la notion de Moving Object Databases (MOD) a été introduite, la recherche sur les bases de données d'objets mobiles pouvant être classifiée suivant deux catégories principales :

— Une première catégorie concernant la gestion de positions [Sistla et al., 1997], [Wolfson et al., 1999] pour laquelle le déplacement actuel et le futur des entités mouvantes sont représentés. Un modèle de données appelé MOST a été défini avec des attributs dynamiques dans le but de représenter les propriétés changeantes des objets mobiles à l'aide par exemple de vecteurs de mouvement plutôt que des points mobiles. Enfin, le langage de requêtes FTL (Future Temporal Logical) a été proposé pour pouvoir exprimer des requêtes continues en spécifiant des relations temporelles entre objets qui sont d'intérêt pour les requêtes actuelles.

— La seconde approche pour la gestion de bases de données d'objets mobiles concerne une perspective spatio-temporelle [Güting et al., 2000], [de Almeida et al., 2006]. Cette approche gère des objets géométriques dépendants du temps (i.e. des points, des lignes et des régions qui se déplacent de manière continue) dans une base de données pour conserver l'historique des mouvements d'entités d'intérêt. Ce modèle de données abstrait comprend des types spatio-temporels et un ensemble d'opérations spatio-temporelles qui forment une algèbre pour la représentation d'objets mobiles avec un langage de requête et de manipulation des données qui a été par la suite développé [de Almeida et al., 2006]. Une base de données open source appelée Secondo a été développée comme prototype de recherche pour l'étude de bases de données spatiales et spatio-temporelles et des techniques d'optimisation.

3.2.1 Différents index pour la gestion de données spatiales et spatio-temporelles

Un des enjeux principaux concernant la gestion de données d'objets mobiles est d'indexer de manière appropriée et pertinente les données dans le système de gestion de base de données. Avec la

multiplication des systèmes de positionnements et de capteurs, de nombreux travaux ont été faits pour étendre les méthodes traditionnelles d'indexation comme le R-tree [Guttman, 1984], R+-tree [Sellis et al., 1987], R*-tree [Beckmann et al., 1990] dans le but de supporter des ajouts fréquents de données. Nous décrivons d'abord ces différents index qui serviront à comprendre leurs extensions pour la gestion et le stockage sur une architecture distribuée.

Le R-tree peut être considéré comme une extension du B-tree pour la gestion de données spatiales. De manière analogue au B-tree, le R-tree est un arbre équilibré en hauteur avec les enregistrements de l'index présents dans les nœuds feuilles contenant des pointeurs vers les objets concernés. Les nœuds feuilles sont de la forme (id,MBB) avec id l'identifiant associé à l'objet et MBB pour minimum bounding box correspond à l'espace minimum occupé par cet objet. Les nœuds non-feuilles sont de la forme (ptr,mbb) où ptr est un pointeur vers le nœud fils et mbb correspond à la plus petite surface englobant l'ensemble des nœuds fils. Pour chaque nœud, le nombre d'objets associés est compris entre une valeur max et une valeur min pour s'assurer d'avoir un arbre équilibré, ainsi si la valeur max du nœud est dépassée, alors un autre nœud feuille est créé et l'arbre est «reconstruit». Dans le domaine de l'indexation de données spatio-temporelles, des variantes du R-tree incluant entre autres un R-tree à trois dimensions [Vazirgiannis et al., 1998], Tb-Trees et Str-trees [Pfoser et al., 2000] tandis que SETI [Chakka et al., 2003] constitue une indexation hybride entre R-tree et partitionnement.

Pour l'indexation de trajectoires d'objets mobiles en espace non restreint, le R-tree à trois dimensions [Vazirgiannis et al., 1998] a été proposé comme une extension évidente du R-Tree pour gérer des données en trois dimensions (i.e. 2 dimensions pour l'espace et une pour le temps). Il traite le temps comme une dimension supplémentaire et est capable de répondre à des requêtes basées sur les coordonnées. Bien que proposé initialement pour des données multimédia, le TB-Tree [Pfoser et al., 2000] permet également de traiter des trajectoires. Évidemment, le R-tree à 3 dimensions indexe des collections de segments dans l'espace à 3 dimensions, seulement en ce qui concerne les traditionnelles requêtes portant sur les coordonnées tout en étant inefficace sur les requêtes portant sur les trajectoires. Le tb-tree essaie de combler cette inefficacité. Le tb-tree [Pfoser et al., 2000] est un arbre équilibré en hauteur avec les enregistrements de l'index présents dans les nœuds feuilles de manière analogue au R-Tree. Cependant, il est fondamentalement différent des autres index spatio-temporels principalement en raison de ces stratégies d'insertion et de répartition des données. Son algorithme d'insertion n'est pas basé sur les aspects spatiaux et temporels des objets mobiles, mais il prend en compte l'identifiant de l'objet mobile à la place.

Quand un nouveau segment est inséré, l'algorithme recherche le nœud feuille contenant la dernière entrée de la même trajectoire et simplement écrit le nouveau segment dans celui-ci, en formant ainsi un nœud feuilles contenant les segments d'une même trajectoire. Si le nœud feuille

est plein alors un nouveau nœud est créé et inséré à la droite de l'arbre. Pour chaque trajectoire, une liste connecte les nœuds qui contiennent ses portions ensemble, formant une structure qui peut facilement et efficacement répondre à des requêtes portant sur des trajectoires (plutôt que seulement sur les coordonnées).

Les auteurs présentent également le STR-tree dont le but est de combiner les propriétés à la fois du TB-tree et du R-tree à trois dimensions. Malheureusement, il semble peu efficace pour les requêtes portant sur les trajectoires comme le R-tree à trois dimensions conformément aux résultats expérimentaux. En dépit de ces avantages pour traiter les requêtes relatives aux trajectoires le TB-tree présente un désavantage majeur : à cause de sa stratégie d'insertion, les nouvelles données de trajectoires sont toujours insérées à droite de l'arbre, les performances de l'index dépendant donc fortement de l'ordre d'arrivée des données dans le système. Cette stratégie d'insertion peut très bien fonctionner dès lors que les données sont insérées dans l'arbre dans l'ordre chronologique : l'insertion va organiser temporellement les lignes proches spatialement temporellement dans l'index.

SETI [Chakka et al., 2003] est une structure hybride indexant les trajectoires à deux niveaux pour séparer l'indexation spatiale de celle temporelle. Partant du principe que les données de trajectoires s'étendent principalement sur la composante temporelle alors que l'emprise spatiale reste statique ou change très peu, SETI partitionne l'espace en cellules hexagonales disjointes qui restent statiques pendant la durée de vie de la structure. Chaque cellule contient un segment de trajectoire si celui-ci est contenu entièrement dans la cellule, tandis qu'un segment de trajectoire à cheval entre deux cellules est séparé en deux et chaque sous segment associé aux deux cellules. Les segments sont insérés dans un fichier de données, chaque fichier contenant l'ensemble des segments associés à une cellule. Ensuite, un index temporel (i.e. un R-tree à une dimension) indexant les intervalles de temps de chaque cellule particulière du fichier de données. Les algorithmes d'insertion et de recherche associés à cet index suivent une approche en plusieurs étapes composée de filtrage spatial, temporel et ensuite de raffinement. En particulier, pour chaque insertion, l'algorithme localise la cellule dans laquelle le segment doit être inséré (en considérant aussi un éventuel découpage du segment sur deux cellules) et l'ajoute ensuite au fichier de données correspondant en mettant à jour par la même occasion l'index temporel associé à la cellule. Comme présenté dans [Chakka et al., 2003] SETI donne de bien meilleurs résultats que le R-tree à trois dimensions et le TB-tree aussi bien pour des requêtes portant sur des intervalles de temps que sur des instants précis. Cependant, il ne peut pas être utilisé facilement pour le traitement de requêtes de trajectoires à cause de sa stratégie d'insertion et de la répartition des segments de trajectoires dans des fichiers associés aux cellules, ce qui peut amener dans le pire cas à parcourir l'ensemble des fichiers de chacune des cellules pour reconstituer une trajectoire.

Pour le lecteur intéressé, d'autres nombreux travaux ont été réalisés sur les index (cf Figure

3.1), certains portant plutôt sur des données anciennes (avec des arbres fournis et denses sur lesquels la capacité de lecture est privilégiée), d'autres sur les données présentes (avec un accent mis sur les capacités d'insertion pour que l'index puisse être manipulé en quasi temps-réel), d'autres pour de la prédiction. Ensuite, la qualité de ces index dépend aussi de la nature des objets mobiles, certains se déplaçant en milieu contraint (automobiles sur des routes) et d'autres en milieu ouvert (des oiseaux dont le déplacement est libre). Des travaux concernant des index d'objets mobiles sur des données distribuées et des architectures "Big Data" ont plus récemment été étudiés. Certains seront discutés par la suite, à l'image de MD-Hbase [Nishimura et al., 2011], de Geomesa [Fox et al., 2013] ou Geowave [Whitby et al., 2017] implémentant des index distribués.

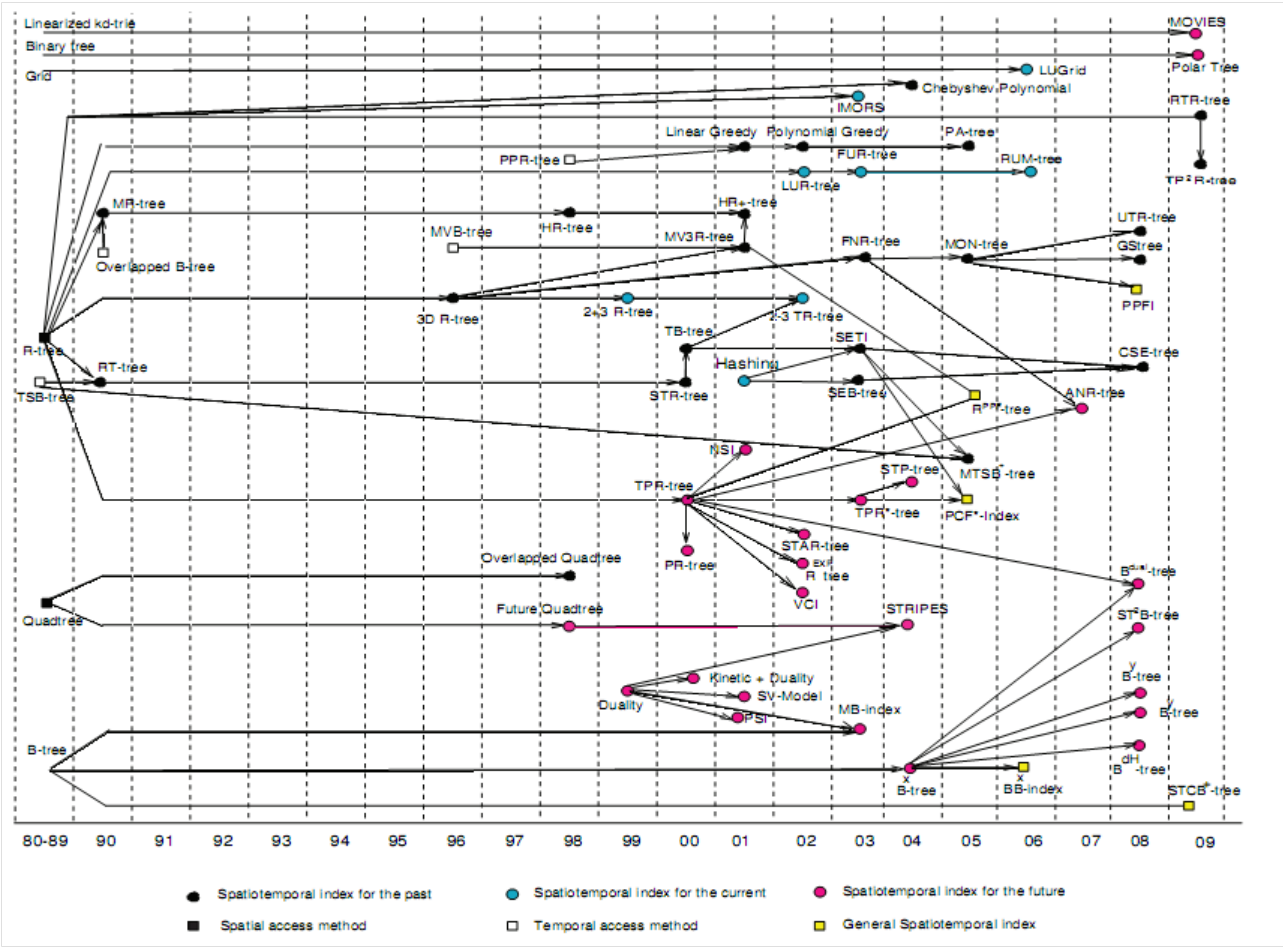

FIGURE 3.1 : Etat de l'art des différents index pour la gestion et le stockage de donnes d'objets mobiles [Nguyen-Dinh et al., 2010]

3.2.2 Opérations topologiques et requêtes pour la gestion de données spatiales et spatio-temporelles

Différentes opérations topologiques dérivant de RCC-8 [Egenhofer, 1991], [Cohn et al., 1997] et des standards OGC prenant en considération la nature spatiale des objets peuvent être définies et implémentées dans un système gérant des données spatiales (cf Figure 3.2). Dans la suite nous en définissons quelques-unes, nécessaires à l'évaluation des systèmes de stockage et traitement massif de données spatiales :

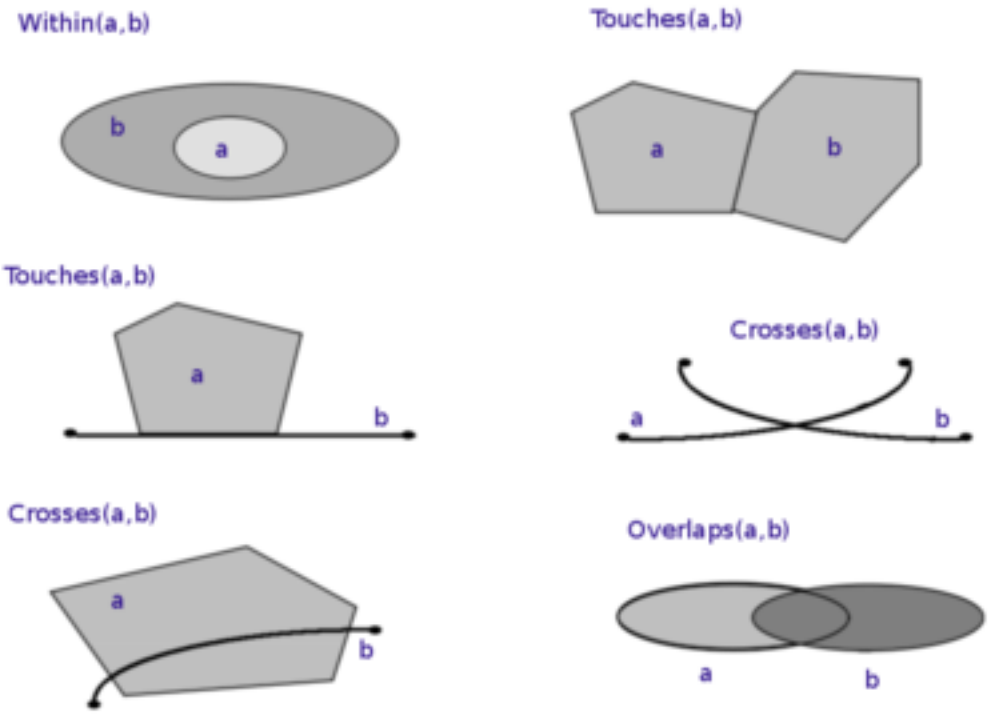

FIGURE 3.2 : Exemple de différentes opérations topologiques

— Intersection(a,b) renvoie l'intersection entre les deux objets spatiaux a et b.
— Contient(a,b) renvoie vrai si l'objet spatial a contient entièrement l'objet spatial b.
— Distance(a,b) fournit la distance entre les deux objets spatiaux a et b.
— Chevauche(a,b) renvoie vrai si les deux objets spatiaux a et b se chevauchent.
— Touche(a,b) renvoie vrai si les objets spatiaux a et b ont un point en commun.
— Disjoint(a,b) renvoie vrai si les objets spatiaux a et b n'ont aucun point en commun.

Ces différents opérateurs spatiaux peuvent être étendus pour prendre en compte la dimension temporelle avec des éléments équivalents entre intervalles temporels définis par Allen [Allen, 1983]. Dans le cadre de l'étude des systèmes pour la gestion de données massives à caractère spatial que

nous faisons par la suite, la plupart des travaux prennent seulement en considération l'aspect spatial et non temporel.

Accompagnés de ces opérateurs spatiaux différentes requêtes ont été définies et étudiées à maintes reprises, celles-ci prenant en compte (grandement) la notion d'index abordée dans ce manuscrit (cf Section 3.2.1). Ici nous définissons différentes requêtes implémentées dans quelques-uns des systèmes de gestion massive de données spatiales.

Parmi les requêtes classiques les range query [Kalashnikov et al., 2002] permettent d'obtenir tous les objets contenus dans une zone spatiale spécifiée par l'utilisateur, en utilisant l'opérateur topologique contient. Les requêtes k plus proches voisins [Xiong et al., 2005] fournissent les k voisins les plus proches spatialement en utilisant la notion de distance. Enfin, une jointure spatiale [Jacox and Samet, 2007] permet de relier des éléments entre deux tables en fonction de leur emprise spatiale. Ces trois requêtes sont implémentées dans la plupart des systèmes de gestion massive de données spatiales que nous verrons par la suite.

Certaines requêtes plus particulières concernent des opérations plus spécifiques ou plus difficiles. La requête enveloppe convexe [Barber et al., 1996], prend un ensemble d'objets spatiaux et renvoie la plus petite zone spatiale contenant l'ensemble de ces éléments. Une requête skyline [Sharifzadeh and Shahabi, 2006] est une dérivée d'une requête plus proche voisin, permettant d'obtenir les points plus proches remplissant certaines conditions supplémentaires (minimisant le prix par exemple) en prenant en compte les autres champs (i.e. non spatiaux).

3.3 Bases de données distribuées pour la gestion et le traitement de données spatiales et spatio-temporelles

Les travaux présentés dans le chapitre précédent pour la gestion et le traitement de données massives ne permettent pas la gestion du caractère spatial ou spatio-temporel de données. C'est pourquoi des solutions et des systèmes spécifiques ont été étudiés et développés pour intégrer et gérer des données de nature spatiale. La plupart des contributions s'articulent autour des différentes problématiques et sont décrites dans la suite dans la Table 3.1 tandis que la Table 3.2 présente les avantages et inconvénients de ces systèmes :

— L'implémentation de types spatiaux
— La mise en place d'index spécifiques pour gérer les données spatiales et de mobilité.
— L'implémentation d'opérateurs spatiaux et de requêtes «basiques» (k-plus proche voisin, range query).

— L'implémentation d'un langage de requête haut niveau permettant de questionner la base de données plus facilement.

Après avoir vu les différents éléments spécifiques et nécessaires au traitement des données spatiales et spatio-temporelles (i.e. index, partitionnements, requêtes, opérateurs spatiaux abordés dans la section précédente), nous faisons dans cette partie le résumé des différents systèmes distribués pour la gestion et le traitement massifs de données spatiales et spatio-temporelles que nous distinguons suivant différents critères. La grande majorité des travaux étudiés dérivent de systèmes utilisés pour traiter ou stocker de gros volumes de données à caractère non spatial, mais qui ont été étendus (en partie) pour prendre en considération la composante spatiale. Le tableau 3.1 référence donc pour chaque travail de recherche le système de gestion massive de données qui a été étendu (avec son langage de requête associé, s'il existe), les différents types spatiaux qu'il prend en considération, les opérateurs spatiaux et requêtes implémentés et en effet l'index et le partitionnement des données sur l'architecture distribuée pour le stockage et le traitement des requêtes.

Spatial Hadoop [Eldawy and Mokbel, 2013] est l'un des premiers travaux à implémenter des opérations spatiales comme extensions de MapReduce. Ce travail fournit des opérateurs spatiaux pour traiter différentes requêtes range query, knn et jointures. SpatialHadoop utilise un index à deux niveaux, un index global concerne la répartition des données sur l'ensemble des nœuds de l'architecture, tandis que sur chaque nœud un index local est utilisé pour repérer les données. Les index supportés étant les suivants : grille, R-tree et R+ tree. Diverses extensions à SpatialHadoop ont été faites pour traiter des requêtes supplémentaires (convex hull etc ...) [Eldawy et al., 2013], pour étudier les diverses méthodes de partitionnement [Eldawy et al., 2015] ainsi que pour implémenter un langage haut niveau PIGEON [Eldawy and Mokbel, 2014] dérivé de Pig [Olston et al., 2008] pour requêter les données d'une manière similaire à du SQL classique. SpatialHadoop gère l'ensemble des types spatiaux (i.e. points, polylignes, polygones, géométrie) et dispose d'un ensemble de fonctions spatiales de base calcul d'enveloppe, génération d'un buffer autour d'un point, création d'un rectangle. Il implémente également des prédicats spatiaux cross, contain, intersect ou overlap, mais aussi contrairement à d'autres travaux des opérateurs d'analyse spatiale permettant de générer l'enveloppe convexe de plusieurs points ou de récupérer l'intersection ou l'union de plusieurs polygones.

Hadoop-GIS [Aji et al., 2013] s'inscrit dans une démarche similaire à SpatialHadoop, au-dessus de Hive [Thusoo et al., 2009] et fournit un langage déclaratif semblable au SQL pour exécuter des requêtes. De la même façon que Spatial Hadoop, HadoopGIS utilise un index à deux niveaux, un index global de partitionnement et un index spatial local à chaque nœud. La partie responsable du traitement des requêtes utilise ce double niveau d'index pour identifier les partitions à «charger en mémoire» et réduire le temps de traitement. Des opérateurs spatiaux basiques sont implémentés

TABLE 3.1 : État de l'art des systèmes de gestion et traitements de données spatiales

	origine & langage	type de données	Opérateurs	Requêtes	Partitionnement & Index
Spatial Hadoop	Hadoop Pigeon (dérivé de Pig)	Point, Rectangle, Geometry, Polygone, Polyligne	intersect contain distance cross overlap touch	Range query Knn query Jointure Convex Hull Skyline Polygon Union Farthest/Closest Pair	Grille uniforme, Quad-Tree, STR, Kd-Tree Z-courbe, courbe d'Hilbert, R-Tree, R+-Tree
HadoopGIS	Hadoop/Hive HiveQL (SQL-like)	Point,MultiPoint, LineString, MultiLineString, Polygon,MultiPolygon	intersect contain distance overlap within disjoint	Range query Knn query Jointure	Grille uniforme, Quad-Tree, BSP,courbe d'hilbert, par bande, STR
Geomesa	Accumulo	Spatio-temporelle			Geohash (Z-courbes)
MD-HBase	HBase	Points		Range query Knn Query	Z courbes, Kd-Tree, Quad-Tree
Geospark	Spark-RDD java	Point, Cercle, polygones, rectangles, polylignes	intersect contain	Knn query, Range query, Jointure	R-Tree, Quad-Tree Voronoi,Hilbert(expérimental), Fixed grid(expérimental) R-Tree and Quad-Tree on RDD
Spatial Spark	Spark CPU-GPU java	Point		1nn query	Fixed grid partitionning, BSP, Tile No live index
Simba	Spark-SparkSQL SQL-like	points, rectangles, LineSegment, cercles		Box-Circle range query Knn query Distance/knn join	Hash, Kd-Tree, Quad-Tree, Voronoi, STR index sur RDD
Stark	Spark Piglet (Pig-like)	Spatio-temporelle	contain intersect within distance	knn query jointure DBscan	fixed grid, cost BSP live index

notamment intersects, contains, distance. Les deux types de requêtes gérées par HadoopGIS sont des box range query et les jointures utilisant les prédicats ou opérateurs spatiaux cités précédemment. Des techniques de partitionnement des données ont également été étudiées [Vo et al., 2014] notamment grille fixe, partitionnement selon des courbes d'Hilbert, du partitionnement par bandes, Boundary Optimized Strip Partitioning, Sort-Tile-Recursive (STR) Partitioning.

MD-Hbase [Nishimura et al., 2011], [Shoji, 2014] utilise des Z courbes pour construire des index au-dessus de Hbase une base de données orientée colonne dérivée de Bigtable [Chang et al., 2006]. Ainsi, des points de l'espace peuvent être stockés dans Hbase de manière efficace. De plus, le critère de séparation pour les index sur l'ensemble des nœuds est modifié pour s'aligner sur la structure d'un kd-tree ou d'un Quad-tree. Enfin, range et knn query sont traitées à l'aide de ces index de manière plus efficace par rapport à un «Hbase natif», de plusieurs ordres de magnitude plus rapide. Cependant, ces index se limitent à des points à deux dimensions et ne peuvent traiter des formes géométriques tels que des rectangles ou des polygones.

Geomesa [Fox et al., 2013] est un travail qui étend la base de données NoSQL Accumulo et dont le but est d'être un équivalent de PostGis pour Posgresql. Avec Geomesa les clefs sont créées en combinant la valeur temporelle avec le geohash pour la partie spatiale de l'objet considéré. Geomesa a été principalement développé pour traiter des points et les données spatiales qui n'en sont pas doivent être décomposées en multiples géohash disjoints, produisant des entrées dupliquées dans l'index. Les données sont toujours à la fois spatiales et temporelles (i.e. contrairement à d'autres systèmes Geomesa gère la temporalité). Quand les données sont requêtées, sont sélectionnées uniquement celles qui sont en intersection avec la région spatiale requêtée, ceci basé sur le calcul des géohashs. L'intérêt de ce travail est son interopérabilité avec de nombreuses bases de données NoSQL Hbase, Cassandra [Lakshman and Malik, 2010] ou Bigtable [Chang et al., 2006] en plus d'Accumulo pour lequel il avait été conçu à la base, les interactions possibles avec Spark en utilisant des opérations spatiales sont également un plus.

Geowave [Whitby et al., 2017] est un système également dérivant et étendant les bases de données accumulo ou hbase en fournissant un index spatial. Comme Geomesa, des «space filling curves» sont utilisées pour représenter les objets multidimensionnels comme des clefs concernant une dimension.

L'exécution «en mémoire» de spark est devenue populaire jusqu'à remplacer MapReduce dont les temps d'exécution étaient dramatiquement plus élevés. Quelques systèmes ont vu le jour implémentant des opérateurs spatiaux pour Spark à l'image de Geospark [Yu et al., 2015a] et SpatialSpark.

Geospark [Yu et al., 2015a] [Yu et al., 2016] fournit des SRDD (des resilient distributed datasets

spatiaux) pour l'exécution de requêtes spatiales comme les knn, range query et jointures spatiales. Les types d'index supportés sont quad-tree et R-tree. Cependant, Geospark ne prend en compte que les données spatiales (i.e. pas d'intégration de la notion de temps, ni d'attributs additionnels) ce qui reste limité pour traiter des données d'objets mobiles. Un index persistant ne semble pas possible en l'absence de fonctionnalité pour «sauvegarder» et «charger» un index. Geospark utilise la librairie JTS fournissant les index mentionnés précédemment et les étend pour traiter d'autres requêtes. Geospark fournit également divers techniques de partitionnement comme du R-Tree Voronoi, du Hilbert ou une grille de taille fixe. Un des principaux défauts de Geospark est le fait qu'il se restreint à certains types d'objets spatiaux pour lesquels des RDD ont été créés (i.e. Point, Cercle, Rectangle, Polygone) et que ceux-ci ne peuvent pas être traités en même temps. Enfin, Spark ne permet pas l'utilisation d'ID sur les objets spatiaux en entrée ni ne prend en compte les aspects temporels.

Le principal but de Spatial Spark est de fournir une technique de jointure parallèle pour des grandes masses de données en utilisant aussi bien GPU que CPU pour parvenir à mettre cela en place. Spatial Spark implémente un ensemble d'opérateurs spatiaux avec Spark permettant de faire du range query et des jointures spatiales utilisant des conditions telles que intersect ou within. Utilisation d'une grille fixe ou de kd-tree sur le HDFS pour accélérer les opérations, pas d'index natifs en RDD et ne supporte que les données à deux dimensions (i.e. pas de notion de temporalité).

Simba [Xie et al., 2016] est une extension de SparkSQL [Armbrust et al., 2015] pour le traitement en mémoire de données spatiales fournissant une interface semblable à celle du SQL. Il fournit des index sur les RDD dans le but de travailler avec des grandes masses de données spatiales et réaliser des opérations spatiales plus complexes grâce à l'interface SQL. Simba implémente un composant chargé de l'optimisation des requêtes prenant en compte ces index et la particularité des données spatiales pour arriver à des contraintes de faible latence.

Stark [Hagedorn et al., 2017] fournit des partitionneurs spatiaux, différents index aussi bien que des opérations de filtre, de jointure et des opérateurs de clustering. Pour les différents partitionnements disponibles, les auteurs proposent une grille fixe, un «binary space partitioner» et partitionnement par polygone. Les opérateurs implémentés par Stark sont les suivants : intersection, contenance et distance, ainsi que les opérations de jointure. Contrairement à d'autres travaux, Stark prend en considération la notion de temps.

Secondo [de Almeida et al., 2006] est un système de base de données fournissant un langage de requête spécifique, un modèle de données et une architecture spécialement conçus pour la gestion et le traitement de données d'objets mobiles. Il s'agit d'un système adaptable et implémentant différents types d'index (Mon-Tree, bounding boxes et différents types de R-Tree). Plus récemment, pour faire face aux volumes grandissants de données et exploiter le stockage et le traitement distribué

de données, des travaux ont été réalisés à l'image de Parallel secondo [Lu and Güting, 2013] qui intègre le système de stockage hdfs [Shvachko et al., 2010] de manière totalement transparente pour l'utilisateur. Il implémente les types et opérateurs spatiaux inhérents à Secondo et intègre des index spécifiques pour le stockage et le «requêtage» des données d'objets mobiles. Plus récemment, Distributed Secondo [Nidzwetzki and Güting, 2015] a été développé par les auteurs, se basant sur la base de données orientée colonne Cassandra [Lakshman and Malik, 2010].

Cependant ces systèmes sont orientés pour un traitement à posteriori et ne permettent pas de gérer le traitement de données spatiales en temps-réel. C'est pourquoi dans la partie suivante de ce chapitre, nous abordons la notion de système temps-réel traitant l'aspect vélocité introduit en début de chapitre.

3.4 Traitement temps-réel de données spatiales et spatio-temporelles

De nombreux travaux ont exploré l'utilisation et le développement de système temps-réels pour la gestion et le traitement de données d'objets mobiles (cf Table 3.3), ceux-ci permettant de traiter certains problèmes spécifiques avec leurs avantages et inconvénients inhérents (cf Table 3.4).

À notre connaissance, le premier travail de recherche abordant la problématique d'objets mobiles dans le cadre de gestion de flux de données temps-réel est introduit dans [Patroumpas and Sellis, 2004]. Dans cet article, l'auteur définit un modèle de données pour la gestion de données d'objets mobiles en tant que flux de données, mais semble trouver ses limites dans certains cas (i.e. seulement les points mobiles sont gérés par ce modèle, pas de régions mobiles). Le système développé propose des algorithmes d'échantillonnage spécifiques aux objets mobiles dans le but de réduire le volume de données présent en mémoire [Potamias et al., 2006], utilise des fenêtres glissantes à différents niveaux de granularité [Patroumpas, 2013] ou bien encore des synopses [Potamias et al., 2007] formant un agrégat de données permettant de répondre plus rapidement aux requêtes tout en limitant l'espace mémoire. Cependant, et malgré les faibles volumes de données traitées en mémoire, la gestion de requêtes multiples n'est pas prise en compte et le couplage entre des données plus anciennes avec les flux entrants dans le système n'est pas géré.

D'autres travaux explorent plutôt les problématiques de scalabilité par exemple [Chandrasekaran and Franklin, 2003] où les auteurs introduisent la notion de paradigme d'exécution partagée. En d'autres termes une jointure est effectuée entre les données et les requêtes afin d'affecter les données aux requêtes dans lesquelles elles interviennent en fonction de leurs emprises spatiales [Mokbel et al., 2005]. L'accent est mis également sur le traitement incrémental des

TABLE 3.2 : Avantage et inconvénients des systèmes de gestion massive de données spatiales

Travaux	Proposition	Avantages	Inconvénients
Spatial Hadoop	Extension d'Hadoop et Pig pour la gestion de données spatiales	Grand nombre de types et de requêtes implémentées Langage haut niveau Interface de visualisation	Pas de gestion temporelle Utilisation d'Hadoop là ou d'autres alternatives sont possibles (e.g. Spark)
Hadoop-GIS	Extension d'Hadoop et Hive pour la gestion de données spatiales	Grand nombre de requêtes implémentées Langage haut niveau	Pas de gestion temporelle Utilisation d'Hadoop
Geomesa	Equivalent de PostGIS pour des systèmes NoSQL et gestion de la temporalité	Types spatiaux Index spatio-temporels Interopérabilité avec différents types de bases de données et Spark	Gère plutôt le côté stockage
MD-HBase	Extension de HBase pour la gestion d'objets mobiles	Implémentation de k plus proches voisins et range query utilisant des index spécifiques	Principalement notion d'index mais peu de traitement associé Orienté pour le traitement des points mobiles seulement
Geospark	Implémentation de «RDD spatiaux» et de requêtes spatiales	Types et index spatiaux en natif en utilisant Spark	Pas de gestion temporelle Opérateurs spatiaux limités (par rapport à Spatial Hadoop)
Spatial Spark	Utilisation de Spark et GPU pour le traitement de données spatiales	Implémentation de range query et de jointures spatiales	Pas de gestion du temps Pas d'index RDD
Simba	Extension de SparkSQL pour la gestion de données spatiales	Langage haut niveau pour l'exécution de requêtes spatiales Travail sur l'ordonnancement et l'exécution des requêtes Index et opérations spatiales	Pas de gestion temporelle
Secondo	Couplage de Cassandra et Secondo pour la gestion massive de données d'objets mobiles	Utilisation de MapReduce en natif sous secondo et stockage distribué des données	Pas d'utilisation de Spark qui serait sans doute plus approprié
Stark	Utilisation et extension de spark pour la gestion de données spatio temporelles	Index RDD pouvant être persistant Gestion spatiale et temporelle Langage haut niveau (Piglet)	En cours de développement

TABLE 3.3 : Résumé des différents systèmes temps-réels pour la gestion d'objets mobiles [Salmon and Ray, 2017]

Research works	Origin system	Management and query repartition	Semantic & Evaluation	Distribution
PLACE [Mokbel et al., 2005]	Nile [Hammad et al., 2004] (online) Predator (offline)	-Scheduling [Hammad et al., 2003] -Pipeline -Multi-join [Hammad et al., 2008]	-Predicate windows [Ghanem et al., 2006] -Incremental evaluation [Ghanem et al., 2007] [Xiong et al., 2005] -Views handling [Ghanem et al., 2010] -Spatio-temporal histogram [Elmongui et al., 2005]	Place * QTP model [Xiong et al., 2007]
StreamSecondo [Zhang et al., 2009]	Secondo (offline) [de Almeida et al., 2006] Possible link with Parallel-Secondo [Lu and Güting, 2013]		-Stream Algebra [Huang and Zhang, 2008] -Windows implementation [Huang and Zhang, 2009] -Stream types	
Kostas Patroumpas works	TelegraphCQ (online) [Chandrasekaran et al., 2003] Possible link for offline with Hermes(PostGis) [Pelekis et al., 2008]	-Sharing paradigm (Psoup) [Chandrasekaran and Franklin, 2003] -Adaptive processing (eddy)[Avnur and Hellerstein, 2000] -Dynamic scheduling [Urhan and Franklin, 2001]	-ST Sampling [Potamias et al., 2006] -Windows aggregation [Patroumpas and Sellis, 2011] -Windows at different granularity level [Patroumpas and Sellis, 2010] -Index at different granularity level [Potamias et al., 2007]	Flux [Shah et al., 2003]
SCUBA [Nehme and Rundensteiner, 2006]	CAPE [Rundensteiner et al., 2004]	-Plan migration [Zhu et al., 2004] -Adaptive scheduling [Sutherland et al., 2005b] -Cluster sharing paradigm [Nehme and Rundensteiner, 2006]	-Clustersheddy [Nehme and Rundensteiner, 2007] -Aggregation	D-CAPE [Sutherland et al., 2005a]
GeoInsight [Kazemitabar et al., 2010]	Microsoft StreamInsight [Ali et al., 2009]	-Fusing horizontal & vertical [Ali et al., 2009] -Stream partitionning knn range queries [Miller et al., 2011]	-Event-based [Ali et al., 2009] -Views derived from archive in-memory [Kazemitabar et al., 2010] -Native support for ST stream [Ali et al., 2010]	
Infosphere Streams ITS [Biem et al., 2010]	SPADE [Ali et al., 2009]	-scheduling component [Wolf et al., 2008] -operator fusing [Khandekar et al., 2009] -basic PE (Processing Element)	-map-matching [Ali et al., 2010] -shortest path	-Datafaflow [Biem et al., 2010]
Zaghreb laboratory works	-TelegraphCQ -Implementation in java	-General framework for MO [Galic et al., 2014]	-Uncertainty handling -Trajectory buffering [Meskovic et al., 2014]	

requêtes pour éviter une réévaluation des requêtes continues dans le système. En effet, afin d'éviter la réévaluation d'une requête donnée, une différence est faite par rapport aux résultats précédents, prenant en compte les mises à jour positives ou négatives suivant que les données traitées par le système soient «entrantes» ou «sortantes» sur des fenêtres dites de prédicats [Ghanem et al., 2006]. Dans [Xiong et al., 2007], l'auteur propose la gestion à la fois de requêtes multiples et de processus incrémental, mais les résultats intermédiaires et vues sont stockées sur disque et ralentissent donc l'efficience du système. Avec SOLE [Mokbel and Aref, 2008] les mêmes auteurs proposent de traiter les données d'objets mobiles entièrement en mémoire, mais ceci sans gérer les aspects de distribution (i.e. sur une architecture distribuée).

Le travail introduit dans SCUBA [Nehme and Rundensteiner, 2006] utilise des techniques de fouille de données comme du «clustering» afin de réduire le nombre de données en mémoire et réduire le temps des traitements, en utilisant un paradigme d'exécution partagée sur ces clusters de données et délestant une partie des données dans un contexte d'objets mobiles [Nehme and Rundensteiner, 2007] pour conserver au maximum des données «utiles» à la résolution des requêtes. Malgré son intérêt, une des limites de ce travail est que malgré son efficience sur les données d'objets mobiles, les auteurs présupposent que le mouvement de ces objets et leur comportement sont relativement prévisibles comme cela peut-être le cas sur un réseau contraint, par exemple le réseau urbain.

StreamSecondo [Zhang et al., 2009] étend le système Secondo [de Almeida et al., 2006] en fournissant une algèbre pour la gestion des flux de données spatiales. Cependant, la gestion des flux et la répartition des requêtes relatives aux objets mobiles ne semblent pas être les problématiques principales traitées par ce travail se concentrant plutôt sur les objets spatiaux que sur des objets mobiles. Un système de fenêtres glissantes et d'opérateurs spatiaux a été implémenté dans ce cadre, mais ne semble pas suffire pour notre domaine d'application.

Dans [Galic et al., 2014] les auteurs proposent un framework général pour la gestion des objets mobiles s'inspirant de l'algèbre définie dans [Zhang et al., 2009]. Ce modèle peut servir de base au développement d'un système de gestion de flux spécifique aux objets mobiles, mais ne prend pas en compte les notions de répartition des requêtes et de gestion incrémentale.

Dans [Kazemitabar et al., 2010] les auteurs abordent la conception du système pour traiter les données de mobilité étendant le système de gestion des événements complexes suivant [Ali et al., 2009]. Les auteurs proposent de fusionner des données anciennes dérivées en vues avec des flux entrants d'objets mobiles. De plus, les flux de données spatio-temporelles sont gérés et les requêtes de k plus proche voisin et spatial range query ont été implémentées. Cependant, ce système semble plutôt dédié à l'étude de réseaux contraints de déplacement et ne semble pas aborder les aspects de distribution.

IBM Infosphere streams ITS [Biem et al., 2010] fournit un module distribué pour traiter des

données de position. Ce travail se concentre principalement sur la gestion et la répartition des requêtes en réordonnant les requêtes [Wolf et al., 2008] et fusionnant ou regroupant les opérateurs en cours d'exécution [Khandekar et al., 2009]. Cependant, bien que les données spatiales soient gérées et que quelques opérateurs soient implémentés, l'intérêt principal de ce travail se focalise plus sur des problématiques de map-matching ou de recherche de plus court chemin plutôt que l'analyse de mobilité.

D'autres travaux ont émergé avec la prolifération de systèmes orientés temps-réel à l'instar de Storm ou S4 (vu dans la partie précédente Section 2.3.2) [Neumeyer et al., 2010] les étendant pour prendre en compte la dimension spatiale et traiter des données de mobilité [Yu et al., 2015b], [Garzó et al., 2013]. Cependant, ces systèmes bien que permettant de répartir les traitements ne semblent pas fournir d'optimisation pour les requêtes ni de possibilités d'échantillonner ou de délester des données.

3.5 Discussion et nécessité d'un système hybride

Ce chapitre a abordé les différentes problématiques associées au traitement d'objets mobiles dans un contexte dit "Big Data", centrées principalement autour de gros volumes de données et a fortiori intégrant une composante spatiale (Section 3.3) et du traitement de flux de données en général et plus spécifiquement pour des objets mobiles (Section 3.4). Cependant, ces systèmes ne permettent pas à la fois d'être réactif et de donner une réponse de bonne qualité ; l'évaluation des requêtes étant un compromis entre temps d'exécution et précision ou qualité de la réponse. L'approche base de données historiques (ou offline) a donc pour précepte de préférer la qualité au temps de calcul et inversement en ce qui concerne les systèmes temps-réels (ou approche online). En combinant les deux, nous souhaitons donc obtenir une réponse de bonne qualité en un temps raisonnable. Dans cette thèse nous étudions donc les mécanismes de fusion entre ces deux sources d'informations différentes pour l'étude de mobilité. Nous postulons que l'apport de l'information tirée des données historiques donne un gain suffisant en qualité par rapport au temps d'exécution nécessaire pour compléter l'information temps-réel et permet d'obtenir une réponse qui n'aurait pas pu être trouvée sans l'usage combiné des deux approches. Dans la suite, nous définissons un modèle générique pour la gestion et le traitement de données d'objets mobiles s'appuyant sur un principe architectural, un modèle de données et des processus de traitements spécifiques. En effet, une approche hybride nécessite d'abord la conception d'une architecture spécifique permettant un traitement distribué et mettant en jeu des interactions entre différents composants (i.e. notamment un relatif à la partie temps-réel et l'autre à la partie offline). Il s'agit dans ce cas de définir les rôles de chacun de ces composants, la répartition des données et des traitements pour stocker au mieux

TABLE 3.4 : Avantages et inconvénients des systèmes temps-réel pour le traitement de données d'objets mobiles [Salmon and Ray, 2017]

Research works	Proposition	Advantages	Drawbacks
PLACE	-Stream processing of moving objects	-sharing paradigm	-Views stored on disk
SEA-CNN	-Incremental processing	-incremental evaluation	-Unavailable
SINA	-Multiple concurrent queries handling	-predicate windows	-Not distributed
StreamSecondo	-Algebra for real-time spatial processing	-Definition of spatial stream objects -Definition of spatial and windows operators	-Basic query sharing -Basic operator ordering -Need for more operators -Not distributed
Kostas Patroumpas works	-Sampling -Windows at different granularity levels -Specific synopses	-Granularity handling -Amount of data in memory reduce -Quick time response	-Limited sharing & ordering abilities -Only moving point are handled -Not distributed -Works not implemented in only one system
SCUBA	-Shared cluster paradigm -Load shedding and compression	-Better sharing than PLACE -Less data and in memory process	-Network based -Relevant for MO that moves together -Accuracy of the response (aggregation & deletion)
IBM Infosphere	ITS	-Distributed -Dataflows processing -Windows handling	-Basic use case -Network based a priori -Incremental processing ? -Spatial types ?
GeoInsight	Coupling archived & real-time data	-Views of archived data in main memory (sketch) -Spatial operators	-Network-based a priori -Incremental processing ? -Only few spatial queries (knn, range)
Zaghreb laboratory works	General framework for MO Algrebra for stream MO	-Uncertainty handling -Trajectory buffering	-Preliminary works -Not implemented on only one system

ces données et pouvoir exécuter les requêtes de manière efficace. Ceci s'accompagne également d'un modèle de données et de processus de traitement spécifiques permettant de fusionner facilement des informations extraites de données anciennes avec des flux entrant dans le système. Enfin, des techniques spécifiques doivent être mises en place et être étudiées pour tirer parti de la particularité des données relatives à des objets mobiles.

Chapitre 4

Vers une approche hybride

Sommaire

4.1 Introduction

Nous avons vu dans la partie précédente les différentes approches pour le stockage et le traitement d'objets mobiles, qu'elles soient orientées temps-réel ou relatives au stockage et traitement de données anciennes. Le besoin d'un système hybride permettant de combler les défauts inhérents de l'une et l'autre des approches (temps d'exécution trop long pour une requête portant sur des données archivées, réponses approximatives pour un système temps-réel) est une nécessité que nous explorons dans ce chapitre. Dans un premier temps, les différents travaux réalisés pour un traitement hybride de données (non spécifiques à des données spatiales) sont présentés et discutés, puis les principes architecturaux d'un système hybride pour la gestion d'objets mobiles est présenté

(Section 4.2). Dans un second temps, nous identifions les différents enjeux relatifs à un système temps-réel, notre approche hybride reposant principalement sur la partie online dont nous présentons l'architecture et les principes de fonctionnement (Section 4.3). Enfin, nous proposons un modèle de données adapté aux traitements d'objets mobiles prenant en compte les prérequis architecturaux définis précédemment et explicitons le mode de fonctionnement de notre système en caractérisant les différentes requêtes pouvant être traitées (Section 4.4).

4.2 Modèle d'architecture d'un système hybride de gestion et de traitement d'objets mobiles

4.2.1 Existant et état de l'art sur une approche hybride temps-réel/données archivées

Dans cette partie, différents travaux concernant un traitement combiné de flux temps-réel et données archivées sont présentés. Ceux-ci ne sont pas dédiées au traitement et stockage de données spatiales et spatio-temporelles, mais peuvent être des alternatives intéressantes pour notre système hybride de gestion et de traitement de données d'objets mobiles à condition d'incorporer une dimension spatiale et temporelle (Type de données spatiales, index, opérateurs).

Les motivations principales et les généralités pour une approche hybride ont été décrites dans [Chandrasekaran and Franklin, 2004] où les auteurs distinguent trois types de requêtes : celles concernant les données archivées ; celles relatives aux données reçues en temps-réel et les dernières dites hybrides qui requièrent la combinaison des données temps-réel et des résultats dérivés de l'analyse des données anciennes. À notre connaissance, il s'agit du premier travail considérant la fusion de données temps-réel et les résultats issus de données archivées pour traiter et gérer une base de données. La solution proposée par les auteurs est de réduire le volume de données archivé à évaluer lorsque le système est saturé. Le volume de données requis est réduit par échantillonnage ou par des méthodes d'agrégation pour répondre aux requêtes hybrides. Fastbit [Reiss et al., 2007] étend ces travaux en construisant un index spécifique pour pouvoir faciliter l'accès aux données archivées et ainsi répondre aux contraintes temps-réel.

Les systèmes hybrides existants peuvent être classifiés comme suit : les systèmes orientés offline, les systèmes orientés MapReduce, et les systèmes orientés temps-réel [Golab and Johnson, 2013]. Les systèmes orientés offline permettent de traiter des requêtes "haut niveau" avec des processus de requêtes optimisés tandis que les systèmes dérivés de MapReduce répartissent les traitements sur une architecture distribuée avec une tolérance aux pannes pour gérer de plus grands volumes de

données. Cependant, ni les systèmes orientés offline, ni ceux issus de MapReduce ne sont appropriés pour traiter les données en temps-réel. Les systèmes dérivés de systèmes temps-réel sont les seuls respectant et répondant aux attentes de faible latence. Ils peuvent répondre en temps-réel, avec un recul ou un contexte fourni par l'analyse des données historiques. En revanche, ces systèmes sont difficiles à mettre en place, car ils nécessitent de coupler l'arrivée massive de données et la fusion avec les données plus anciennes. Enfin plus récemment, les systèmes dérivants de MapReduce ont évolué, déviant de l'aspect purement volume à l'aspect vélocité pour répondre aux attentes des systèmes temps-réel. Par conséquent, ils peuvent être considérés comme des candidats sérieux pour la gestion, le stockage et le traitement de données d'objets mobiles.

Notre propre classification fait état de différentes approches suivies pour le traitement hybride de données (cf Tableau 4.1). Nous distinguons les systèmes utilisant un paradigme de "complex event processing" pour permettre de faire le lien entre des événements passés et les données entrant dans le système (ces systèmes peuvent être associés aux systèmes orientés temps-réel de la classification précédente [Golab and Johnson, 2013]). Les systèmes dérivés de l'approche MapReduce utilisant principalement une approche incrémentale pour traiter les données et réduire les temps de calcul (ces systèmes peuvent être associés aux systèmes orientés MapReduce de la classification précédente [Golab and Johnson, 2013]). Les systèmes inspirés de l'architecture Lambda avec des parties distinctes temps-réel et traitement massif de données, dont les résultats conjoints sont fusionnés. Des travaux reposant sur la notion d'entrepôts de données ont été développés, et intègrent pleinement la partie stockage, mais peuvent être moins performants sur les traitements(au niveau des temps de calcul) (ces systèmes peuvent être associés aux systèmes orientés offline de la classification précédente [Golab and Johnson, 2013]). Enfin, des systèmes dédiés au traitement si bien temps-réel que traitement massif de données ont émergé à l'image de Spark et Flink et permettant la gestion de ces deux aspects de manière efficace.

Systèmes intégrant la notion de complex event processing

Certains travaux utilisent le concept d'événement pour un traitement hybride des données, en essayant de trouver des corrélations entre données temps-réel et archivées pour identifier des événements spécifiques tandis que les données sont reçues et traitées par le système. Pour ce faire, les résultats des requêtes sont mis en mémoire tout en traitant les données temps-réel et identifier des motifs spécifiques. Ainsi, les auteurs ont développé leur concept d'événement pour construire le système de traitement d'événements complexes (CEP) agissant de manière hybride avec comme cas d'étude le contexte de la finance [Dindar et al., 2009]. D'autres travaux proches concernent l'identification de motifs sur des données temps-réels et archivées dans un contexte de traitement d'événements complexes [Balazinska et al., 2007]. Moirae est une plateforme spécialement conçue pour des applications de surveillance intégrant la notion d'événements [Balazinska et al., 2007].

TABLE 4.1 : tableau récapitulatif des différentes approches hybrides pour le traitement de données

Type de système hybride	Travaux correspondants	Proposition et Avantages	Défauts ou difficultés de mise en oeuvre
Complex Event Processing	Dejavu[Dindar et al., 2009] MOIRAE [Balazinska et al., 2007] HYPE[Peng et al., 2011]	-Détection de motifs et détection d'événements grâce à un modèle de données d'événement	Limité en termes de traitement de données (peu de scalabilité)
MapReduce	HOP[Condie et al., 2010] Muppet[Lam et al., 2012] Scalla[Li et al., 2012] Nova[Olston et al., 2011]	-Traitement incrémental via MapReduce -Traitement massif de données (scalable)	-Absence d'un schéma de données pour la détection d'anomalies -Traitement pouvant être lourd, car dérivé de MapReduce
Lambda	Summingbird[Boykin et al., 2014] GoogleDataFlow[Akidau et al., 2015] Radstack[Yang et al., 2017]	-Traitement à plusieurs niveaux avec un système temps-réel et un système traitement massif de données	-Maintenance de deux différents codes pour les parties offline et online
Entrepot de données et OLAP	Polystore[Duggan et al., 2015] Quill[Chandramouli et al., 2016] Tidalrace[Johnson and Shkapenyuk, 2015]	-Stockage et vues des résultats à plusieurs niveaux de granularité -Interopérable avec différents systèmes et SGBD	-Orienté offline et moins performant pour les aspects temps-réel
Nouveaux paradigmes de traitement	Spark[Zaharia et al., 2013] Flink[Alexandrov et al., 2014]	-Approche permettant de gérer aussi bien temps-réel que traitement massif de données -Librairies pour le traitement de graphes, machine learning ou notion d'événements	-Dépendant d'autres systèmes pour le stockage de données et leur assimilation (Hbase, Kafka, Cassandra)

Moirae supporte quatre types de requêtes : les requêtes d'événements, les requêtes hybrides standard, les requêtes contextuelles et les requêtes hybrides contextuelles. Avec Moirae, les événements sont observés et détectés à l'aide de l'information extraite des données historiques et les événements similaires apparus dans le passé. À cause des forts volumes de données, les résultats des requêtes sont obtenues de manière incrémentale et peuvent être approximatifs de manière volontaire pour simplifier les traitements. Concernant l'analyse et la détection de motifs hybrides entre temps-réel et données archivées HYPE fournit un modèle plus générique considérant non plus des motifs contigus, mais discrets [Peng et al., 2011]. De plus, un mécanisme de mise en mémoire de motifs issus de la partie historique aussi bien que temps-réel est utilisé pour faire de la reconnaissance de motif partiel et ainsi accélérer le processus d'identification de motifs.

Systèmes dérivant de MapReduce

À l'origine, MapReduce [Dean and Ghemawat, 2004] et Hadoop [Shvachko et al., 2010] ont été développés spécifiquement pour le traitement de larges volumes de données de manière distribuée. Seulement, ces types de systèmes ne sont pas appropriés pour gérer l'aspect vélocité et les traitements itératifs. D'autres solutions ont donc été développées pour régler en partie ces manquements à l'instar d'un MapReduce online [Condie et al., 2010] qui fournit un MapReduce plus interactif en entamant la phase de reduce alors même que le map n'est pas fini pour des soucis de performance. D'autres systèmes incrémentaux à l'image de Muppet ont vu le jour [Lam et al., 2012] où des vues sont mises à jour au fur et à mesure que les données arrivent dans le système. Dans Nova [Olston et al., 2011] les auteurs proposent un système construit "au-dessus» de Hadoop et Pig permettant l'utilisation inhérente d'un workflow. Nova propage ainsi les delta mises à jour en transformant les données ressources en vues dérivées plutôt qu'en retraitant l'entièreté des données sources. Enfin, Scalla [Li et al., 2012] est un système dérivé de MapReduce basé sur un processus de traitement limitant le nombre de fois où les données sont évaluées en utilisant une approche incrémentale. Ce système utilise du "hashing" pour faciliter un traitement rapide en mémoire et introduit une répartition des tuples sur le disque et en mémoire en cas de surcharge de cette dernière. Néanmoins ces systèmes restent orientés traitement "batch" et ne fournissent pas les performances suffisantes pour gérer les aspects de vélocité et des traitements itératifs comme pour les algorithmes de machine learning ou de fouille de données.

Systèmes dérivant de l'architecture Lambda

Plus récemment avec l'émergence des problématiques conjointes de volume et de vélocité, un nouveau modèle d'architecture a été proposé. L'"architecture Lambda" (cf figure 4.1) est un système de gestion des données qui prend en compte à la fois les aspects vélocité et volume avec la contrainte de répondre en quasi-temps réel. Cette architecture peut être décomposée en trois parties distinctes,

une couche correspondant aux données stockées dans un système de base de données NoSQL et des pré calculs ou vues relatifs aux requêtes fréquemment formulées (Batch Layer), une autre couche correspond au traitement temps-réel des données et les vues ou résultats associés (Speed Layer), tandis qu'une couche intermédiaire est chargée de fusionner les résultats des deux parties précédentes (Serving Layer).

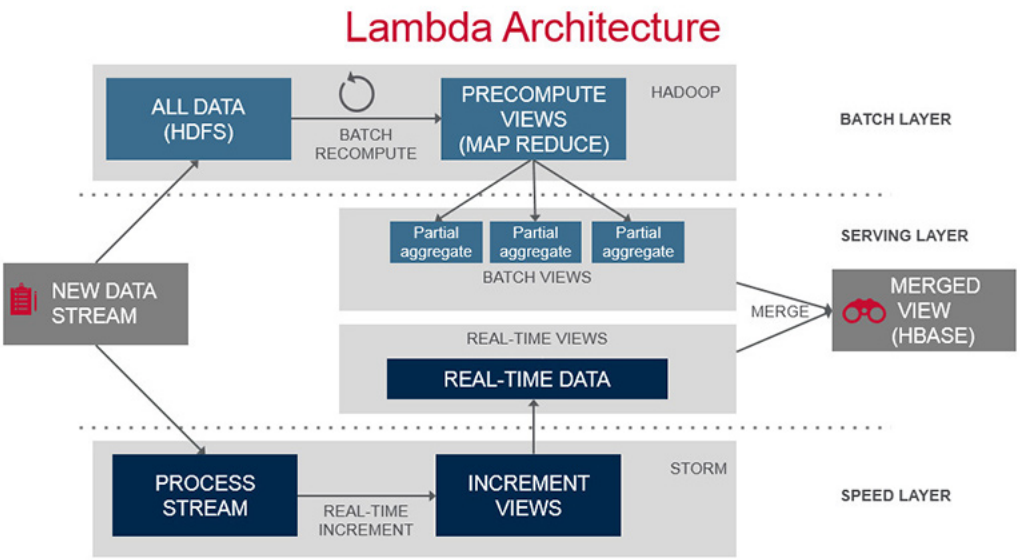

FIGURE 4.1 : Illustration d'architecture lambda [Marz, 2013]

Summingbird [Boykin et al., 2014] est l'illustration de l'architecture Lambda précédemment décrite, la couche offline fonctionnant avec un paradigme de traitement MapReduce pour traiter les larges volumes de données, tandis que Storm est utilisé pour la partie temps-réel. Ce système permet d'écrire le code une seule fois, celui-ci étant traduit différemment suivant qu'il s'exécute sur la partie offline ou online à l'aide d'une algèbre.

GoogleDataflow [Akidau et al., 2015] est un équivalent de Summingbird utilisant Millwheel [Akidau et al., 2013] pour la partie temps-réel et Flume et MapReduce pour la partie offline, mais n'est pas pourvu d'une algèbre comme Summingbird.

Druid [Yang et al., 2014] est une base de données orientée colonne distribuée disposant de noeuds temps-réels, de noeuds dédiés aux données historiques et des noeuds gérant la communication et les processus. Il est principalement orienté pour l'assimilation des données temps-réel et utilise des éléments agrégés (OLAP) pour la partie offline dans le but de faire de l'analyse complexe de données. Dans [Yang et al., 2017] avec Radstack les auteurs développent un modèle d'architecture lambda en utilisant une combinaison de Samza [Noghabi et al., 2017] pour la partie temps-réel, Hadoop pour la partie offline, Druid [Yang et al., 2014] pour la partie serveur/communication et

Kafka [Kreps et al., 2011] pour l'absorption des flux et la redirection vers Hadoop et Samza.

L'intérêt de cette approche hybride et qu'un système dédié et efficace est utilisé pour chaque partie du traitement (un système orienté temps-réel pour les données traitées à la volée et un système de traitement de données massives, ce qui évite les écueils d'une approche uniquement MapReduce par exemple qui bien que générique ne soit pas performant sur ces deux aspects). Cependant, cette architecture lambda n'est pas sans défaut nécessitant entre autres de maintenir deux codes distincts pour les parties online et offline. L'architecture kappa (cf figure 4.2) se présente comme une alternative pour le traitement de données orienté temps-réel. Le principe de fonctionnement est plus "simple" que celui de l'architecture lambda, comme pour les anciens systèmes temps-réel du début des années 2000 il s'agit de conserver une archive de données facilement accessible et d'en faire le rejeu lorsque des données plus anciennes ou ayant été délestées sont nécessaires. Traditionnellement avec cette architecture Kappa, Kafka [Kreps et al., 2011] est utilisé comme "base de données" étant rejouée comme un flux pour compléter les informations issues des données temps-réel. L'avantage de cette approche Kappa est qu'elle permet lorsque le processus de traitement est le même de traiter à la fois des données plus anciennes avec des données récentes à la volée. Cependant, Kafka n'étant pas dédié explicitement au stockage de données, il ne dispose pas d'index performants et ne permet pas de stocker un volume conséquent de données. L'architecture lambda a l'avantage qu'elle possède un système de traitement efficace pour la partie offline sur de gros volumes de données et que le traitement pour les parties offline et temps-réels ne sont pas les mêmes ce qui permet une combinaison des informations à ces deux niveaux qui est différente.

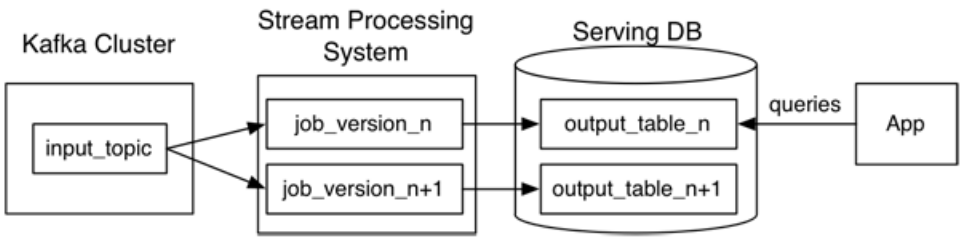

FIGURE 4.2 : Illustration d'architecture Kappa [Kreps, 2014]

Systèmes exploitant la notion d'entrepôt de données et des technologies dites "Big Data"

Polystore [Duggan et al., 2015] est un système hybride utilisant S-Store [Meehan et al., 2015] pour la partie temps-réel (dérivant de H-Store [Kallman et al., 2008]) et différentes sortes de bases de données pour la partie offline (Postgres, Accumulo, SCiDB). S-Store permettait principalement de gérer des opérations "basiques" en temps-réel et n'était pas suffisant pour fournir des résultats

analytiques en temps-réel. L'accent de ces travaux est mis sur l'interopérabilité des systèmes plus que sur la fusion et le traitement hybride (online/offline) des données et la connexion avec S-store pour la partie temps-réel n'est pas totalement opérationnelle.

Tidalrace [Johnson and Shkapenyuk, 2015] est un Data Stream Warehouse, c'est à dire un système ingérant continuellement des flux de données, calculant des vues dérivées des tables et données et stocke des données archivées.

Trill [Chandramouli et al., 2014] est une librairie permettant de faire un traitement hybride sur des données offline/online. Quill [Chandramouli et al., 2016] est une extension de Trill avec un modèle de données sharded stream générique et peut être exporté sur une plateforme cloud (notamment Microsoft Azure). Trill et sa version parallèle, Quill, sont des plateformes pour le traitement incrémental et l'analyse de données basées sur un modèle tempo-relationnel. Ils fournissent un ensemble riche de types de données et d'opérateurs pour un traitement efficace des flux de données et des requêtes relationnelles. Enfin, ils exploitent des approches bas niveau orientées colonnes qui améliorent de manière significative la faible latence et les lectures/écritures.

Systèmes utilisant de nouveaux paradigmes de traitement

Plus récemment, Spark [Zaharia et al., 2013] a émergé comme le successeur d'Hadoop, supposé être 100 fois plus rapide pour le traitement des données. Spark est basé sur la notion de RDD (Resilient Distributed Datasets) [Zaharia et al., 2012] qui permettent de conserver en mémoire une partie des résultats intermédiaires sans écrire sur disque et tout en restant tolérant aux pannes, ce qui est important pour gérer des processus itératifs. De plus des opérateurs primaires comme les filtres, les jointures ou partitionnements ont été ajoutés aux RDD à la base MapReduce déjà présente pour répartir et traiter aux mieux les données suivant un paradigme DAG (Direct Acyclic Graph) [Arasu et al., 2004]. Avec son extension Spark Streaming [Zaharia et al., 2013], Spark permet de traiter les données de manière hybride comme nous le souhaitons en traitant aussi bien les données plus anciennes comme celles arrivant en temps-réel. Pour agir comme un système temps-réel, Spark traite les données en considérant le flux temps-réel comme un mini-batch de données. Donc malgré l'efficacité reconnue de Spark, ce système reste avant tout et peut trouver ses limites, car la granularité des réponses est bornée à la taille accordée au mini-batch. En effet, si durant une période de temps peu de données arrivent dans le système celles-ci peuvent être non traitées, car n'étant pas présentes en quantité suffisante pour atteindre le seuil maximum pour le mini batch.

Apache Flink dérivé des travaux initiaux sur Stratosphère [Alexandrov et al., 2014] est un système distribué orienté temps-réel. Il dispose d'un ensemble d'opérateurs primaires plus riches et génériques que MapReduce, et agit comme un système de gestion de flux pour faire des traitements itératifs [Ewen et al., 2013] [Ewen et al., 2012]. En effet, la notion de delta itération permet de

traiter les données et la phase itérative de manière incrémentale. Le système inclut un optimiseur de requêtes qui parallélises et optimise le plan de requête temps-réel même pour des fonctions définies par l'utilisateur (udf) [Hueske et al., 2012] et réordonner la succession des opérateurs si nécessaire [Hueske et al., 2013]. De plus, différentes sorties de fenêtres et opérateurs sont disponibles à l'image des "triggers" (déclencheurs) qui sont appelés lorsqu'une valeur spécifique est reçue par le système. On peut dès lors mettre en place des triggers, déclenchant d'autres triggers soit lorsque certains événements sont détectés. Enfin, par rapport à Spark, avec Flink les données sont traitées "à la volée" passant directement par l'ensemble des opérateurs en jeu. Avec ce système, le traitement offline est simplement un traitement des flux borne aux données plus anciennes jusque celles actuelles. C'est pourquoi Flink semble plus orienté temps-réel et donc plus approprié pour ce que nous souhaitons développer ici.

Cependant ces travaux ne prennent pas en considération la dimension spatiale ou spatio-temporelle et de données et à notre connaissance aucun travail de recherche n'a concerné l'usage conjoint de flux temps-réel et données archivées pour le traitement et l'extraction d'information concernant des objets mobiles. Dans la suite, nous dressons un panel d'attentes dont une partie est générique à tout système hybride et d'autres plus spécifiques à ce contexte spatio-temporel.

4.2.2 Caractéristiques d'une architecture hybride pour la gestion et le traitement d'objets mobiles

Dans le but de traiter en temps-réel des données d'objets mobiles et de détecter des anomalies ou d'extraire des informations, certains prérequis sont nécessaires. Ici, nous définissons certaines de ces nécessités s'accompagnant de problématiques que nous souhaitons résoudre (au moins en partie) dans le cadre de cette thèse, parmi lesquelles l'utilisation de résumés de données pour une analyse en temps-réel, l'adaptabilité du système, une définition plus générique concernant les requêtes d'objets mobiles ou encore l'autonomie et la scalabilité des traitements et des données (cf Table 4.2).

Prise en compte de la dimension spatiale et spatio-temporelle.
Contrairement à une grande partie des systèmes développés pour répondre aux problématiques de volumétrie et de vélocité ne gérant pas la spécificité des données à caractère spatial et spatio-temporel (cf Chapitre 2) et face à l'inefficacité d'une approche seulement online ou offline dans ce contexte (Chapitre 3), il est nécessaire de définir un système hybride implémentant des types spatiaux et spatio-temporels, ainsi que des index et requêtes associées. Le système que nous voulons concevoir doit être suffisamment générique pour traiter l'ensemble des requêtes relatives aux objets mobiles, là où les travaux observés concernent plus la résolution d'une requête spécifique. Une (re)définition ou une catégorisation des requêtes concernant les objets mobiles peut donc être une clef d'amélioration

pour un traitement distribué, notamment en identifiant des sous-opérations communes à différentes requêtes. Pour pouvoir traiter une large variété de requêtes, il est nécessaire de décomposer les requêtes en sous-requêtes et d'analyser éventuellement des similarités entre requêtes et utiliser d'anciens résultats de requêtes ou vues et fusionnants celles-ci avec les données entrant dans le système. Il y a toujours un besoin concernant l'analyse et l'étude des différents types de requêtes relatives aux objets mobiles et l'intérêt de trouver des similarités et des traitements communs entre-elles [Sakr and Güting, 2014].

Utilisation de résumés de données pour une analyse en temps-réel.
La motivation principale pour la conception d'un système hybride est de contrevenir aux défauts inhérents d'une base de données conséquente avec des temps de réponse pouvant être longs (de l'ordre de plusieurs minutes à plusieurs heures suivant la requête) et d'un système temps-réel dont les réponses peuvent être approximatives ou inexactes (des données anciennes ayant été délestées pour pouvoir répondre plus rapidement aux requêtes). Idéalement, nous souhaitons construire un système permettant de répondre aux requêtes avec un délai proche des systèmes temps-réel classiques, mais avec une précision ou une qualité de réponse se rapprochant de ce qui est obtenu pour une approche offline. Ceci nécessite donc l'usage de vues ou synopses qui permettront de synthétiser l'information pour fournir une réponse avec un délai de temps relativement court comparativement à un système purement offline et de meilleure qualité par rapport à un système uniquement basé sur du temps-réel. Une mesure de la qualité de réponse doit être évaluée en complément pour estimer si le système est capable d'exécuter des requêtes "compliquées" et fournir une analyse de trajectoire en temps-réel. Pour prendre en compte ces besoins temps-réel, un tel système doit stocker des vues agrégées et synopses en mémoire vive qu'il s'agisse de résultats extraits de la partie offline ou de résumés de flux de données pour la partie temps-réel. Le système doit être capable de prendre, "traduire" et fusionner ces vues synthétiques avec les données entrant dans le système [Ghanem et al., 2010]. Les vues doivent être la pierre angulaire de notre architecture pour permettre l'exécution partagée des requêtes et ainsi réduire les temps de réponse.

Adaptabilité du système et gestion temps-réel.
Comme c'est le cas pour les systèmes temps-réel, les flux de données et le nombre de traitements à effectuer peuvent varier grandement et ceux-ci doivent pouvoir être assimilés et traités par le système au delà des fluctuations au niveau du nombre de données entrantes. Ceci nécessite donc que le système puisse s'adapter à la volée pour être le plus performant à chaque instant et rester en mesure de répondre aux requêtes. Le système doit être réactif et s'adapter lui-même graduellement alors que les données et requêtes sont reçues par le système pour rester le plus efficace possible et traiter les requêtes continues [Deshpande et al., 2007]. Pour ce faire, le système peut par exemple à l'aide de métriques pouvoir mesurer ses propres performances et allouer si nécessaire des ressources

supplémentaires par exemple ou bien modifier l'affectation des tâches aux différents noeuds de l'architecture afin de s'adapter au mieux aux requêtes actuellement exécutées. Les données qui ne sont plus utiles doivent également être agrégées ou transférées vers la partie offline pour réduire le volume de données à traiter.

Scalabilité et répartition des données et traitement.
Le système étant distribué, pour gagner en efficacité il est nécessaire de définir des méthodes spécifiques pour d'une part répartir et stocker les données (idéalement de manière adaptée à la dimension spatio-temporelle) en ce qui concerne la partie offline et d'autre part de répartir et limiter le nombre de traitements pour les parties offline et online dans un contexte de requêtes multiples. En ce qui concerne le stockage et la distribution des données, celle-ci peut intervenir à plusieurs niveaux, d'abord sur l'ensemble du cluster et sur chacun des noeuds, tout en utilisant des index spécifiques. Pour répartir les traitements, idéalement le système doit ne pas évaluer plusieurs fois les mêmes traitements dans un contexte de gestion de requêtes multiples (prenant avantage à la fois des vues et synopses extraits et de la spécificité des données spatio-temporelles et des similarités entre certaines des requêtes).

Enfin, des notions supplémentaires peuvent être d'importance dans un contexte de détection d'anomalies et d'obtenir un système autonome qui prévienne l'utilisateur. Celles-ci pourront faire l'objet de travaux ultérieurs, la priorité ayant été mise sur les différents enjeux identifiés précédemment, ces derniers seulement ayant été retenus pour le choix d'un système hybride (cf Table 4.2)

Un système de traitement autonome et réactif.
S'inspirant des prérogatives des systèmes temps-réel, l'accent peut être mis sur l'autonomie du système qui doit être capable par exemple d'identifier des requêtes récurrentes ou réexécuter des calculs pour alerter l'utilisateur en cas de changement notable ou de problèmes. Le système doit prendre en main le traitement et la maintenance des vues en accord par exemple avec les requêtes fréquemment formulées par les utilisateurs. Il devrait détecter l'émergence de nouveaux événements et prévenir l'utilisateur lorsque ces changements interviennent. Il devrait aussi réduire l'intervention de personnes et traiter les données lui-même et fournir des résultats si nécessaire suivant un système DAHP (Data Active Human Passive) [Abadi et al., 2003].

Identification de motifs spatio-temporels
Pour identifier d'éventuelles anomalies, le système doit permettre de déterminer des motifs intervenant régulièrement dans le déplacements des objets mobiles, les considérant comme étant normaux pour la partie offline, et de pouvoir synthétiser ces comportements ou motifs sous forme de résumés de données facilement exploitables. Ces résumés de données pourront être utilisés pour mettre en contexte les flux temps-réel et détecter ainsi les anomalies ou comportements suspects parmi les

TABLE 4.2 : tableau récapitulatif des systèmes hybrides et leurs performances en termes d'enjeux pour la gestion d'objets mobiles

Type de système hybride	scalabilité et répartition des données et traitement	Prise en compte de la dimension spatiale et spatio-temporelle	Adaptivité du système et gestion temps-réel	Utilisation de résumés de données
Complex Event Processing	-Peu performant pour la distribution des données et traitement	-Nécessite l'implémentation de types spatiaux et spatio-temporels et opérateurs associés. Difficile	-Très performant	-Géré par la notion d'événement
MapReduce	-Très performant pour le stockage de données -Peu efficace pour la répartition et traitement simultané des requêtes	-Peu performant pour les opérations de selection et necessité d'index spécifiques -Necessite beaucoup de développement	-Traitement pouvant être lourd, car dérivé de MapReduce	-Peut être géré par traitement incrémental (delta iteration, pipeline)
Lambda	Très performant pour la scalabilité	Nécessite l'implémentation de types spatiaux et spatio-temporels et opérateurs associés à deux niveaux offline et online	-Performant -Nécessite des mécanismes de transfert de données et traitement online vers offline	"Vues" online et offline
Entrepot de données et OLAP	Scalable, mais devant gérer différents types de stockage	Nécessite l'implémentation de types spatiaux et spatio-temporels et opérateurs associés potentiellement à plusieurs niveaux	-Orienté offline et moins performant pour les aspects temps-réel	-Stockage et vues des résultats à plusieurs niveaux de granularité -Interopérable avec différents systèmes et SGBD
Nouveaux paradigmes de traitement	Très performant pour la scalabilité	Implémentation de types spatiaux et spatio-temporels et opérateurs associés peut se faire facilement	-Pouvant stocker en mémoire ou sur disque en fonction des besoins.	-Structures de données abstraites pour la gestion et le traitement. -Mécanismes itératifs

différents objets mobiles.

Compte tenu de ces différentes prérogatives, dans la suite nous proposons un modèle architectural pour la gestion des objets mobiles suffisamment générique afin de prendre en compte chacun de ces aspects.

4.2.3 Modèle d'architecture hybride pour la gestion et le traitement d'objets mobiles

Dans le but de traiter de grandes masses de données d'objets mobiles, et ceci en temps-réel, nous proposons une architecture hybride fusionnant traitement temps-réel et informations extraites de données anciennes (cf figure 4.3). Celle-ci est distribuée pour gagner en efficacité et améliorer les temps de réponse de manière conséquente.

Au niveau de la gestion des données, différents éléments peuvent être distingués : d'abord le composant relatif au traitement offline (Offline part), celui relatif au traitement online (Online part) et une partie faisant le lien à la fois entre l'utilisateur et les deux parties online et offline précédentes que nous appellerons la partie hybride (Mediator et Evaluator).

Dans notre système les reports de positions s'effectuent via différents flux de données qui sont gérés sur un système temps-réel distribué, soit ici via le composant dédié au traitement online. La gestion des traitements en mémoire est faite sur une fenêtre glissante distribuée dont la taille est modifiée selon le nombre de données collectées en temps-réel sur la zone de couverture concernée. Des vues online sur les requêtes continues sont mises à jour et incrémentées au gré des flux entrants de données. Si l'utilisateur exprime une requête portant sur des données n'ayant pas produit de résultats intermédiaires, les données sont accessibles via la fenêtre glissante. Une fois que la période temporelle dédiée à la fenêtre glissante est dépassée, les données sont déplacées vers la base de données historiques distribuée afin d'effectuer les traitements offline.

La partie offline est composée d'un système de gestion de données distribué dit "Big Data" et est associé à un système de traitement distribué pour gagner en efficacité. Pour avoir un système réactif, des pré calculs sont effectués sur les données historiques et mis à jour au fur et à mesure des arrivées en base de données. Ces pré calculs sont utilisés dans le cadre de la résolution de requêtes hybrides le but étant de limiter les accès directs à la base de données. Ils sont régulièrement remis à jour alors que les données sont ajoutées en base de données et doivent servir de complément aux traitements effectués pour la partie temps-réel.

Au niveau des requêtes deux entités, appartenant à la partie hybride, sont utilisées pour identifier les données à extraire et traiter, ainsi que pour gérer les interactions entre la base de données historiques et le système de traitement temps-réel. Une de ces entités est le médiateur dont le rôle est de gérer les flux entre les composants online et offline, de conserver et stocker les vues associées et de pouvoir les fusionner pour permettre de répondre aux requêtes hybrides. L'évaluateur analyse la requête en entrée et essaie d'inférer le type de la requête, à savoir online, offline ou ou hybride pour orienter, en fonction du type de requête identifiée, la récupération des données et des

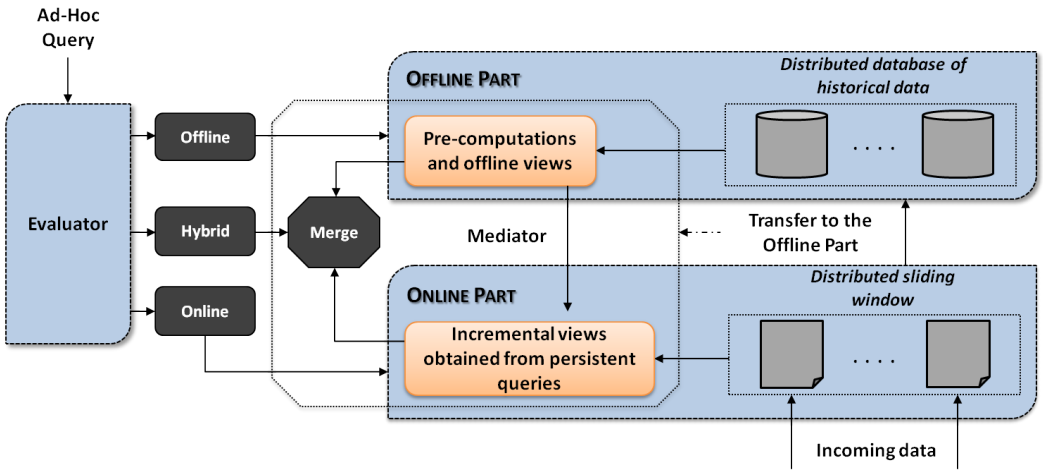

FIGURE 4.3 : Schéma de fonctionnement d'un système hybride pour le traitement d'objets mobiles

informations nécessaires dans notre architecture. Il transmet au médiateur les données désirées à traiter et ce dernier se charge de prendre, combiner, ou d'effectuer des traitements sur la fenêtre glissante ou l'archive suivant la demande de l'évaluateur.

L'approche défendue ici, est principalement orientée temps-réel et est donc différente d'une architecture lambda classique, ce qui devrait induire une architecture kappa (cf section 4.2.1). Cependant, une base de données est utilisée pour générer des vues à différents niveaux de granularité pour la partie offline, ce qui nécessite l'usage d'algorithmes de traitement différents pour les parties offline et online, ce qui exclut donc l'usage d'une architecture dite Kappa. L'intérêt de notre approche est donc de définir des traitements différents pour les parties temps-réel et offline pour obtenir un meilleur résultat, les informations extraites de la base de données archivées devant servir de complément aux traitements temps-réel.

4.3 Modèle de traitement pour la gestion et l'analyse d'objets mobiles

Comme nous l'avons vu précédemment, différents types de systèmes peuvent être identifiés (cf Table 4.1), les nécessités de notre système hybride concernent principalement la prise en compte de la composante spatiale et spatio-temporelle et la gestion de l'aspect temps-réel (cf Table 4.2). C'est pourquoi dans la suite, nous définissons les enjeux génériques d'un système temps-réel (Section 4.3.1) avant de poser les éléments de principe de la partie online de notre système hybride pour la gestion d'objets mobiles (Section 4.3.2).

TABLE 4.3 : Enjeux pour un système temps-réel distribué

Enjeux principaux	Ordonnancement et Répartition des traitements	Sémantique, Evalutation et Approximations	Distribution des traitements et des données
Enjeux secondaires associés	-Scalabilite et optimisation -Gestion de requêtes mulitples -Ordonnancement des opérateurs	-Echantillonage et approximation -Evaluation incrémentale des données -Algorithmes approximatifs ou simplifiés -Fenêtres et utilisation de vues	-Répartition et propagation des données -Répartition des requêtes -Réduire les communications et échanges -Maintenance de la cohérence des données

4.3.1 Enjeux d'un système temps-réel

Les enjeux intervenant pour le traitement en temps-réel de données peuvent être catégorisés en trois parties, une relative à l'ordonnancement et la répartition des requêtes, une autre à la gestion et la limitation du nombre de données à traiter et enfin une découlant de la distribution à la fois des données et de requêtes (cf Table 4.3)

Dans un système de gestion de flux de données, une requête est composée d'opérateurs sur ces flux organisés en un graphe acyclique ou workflow [Arasu et al., 2004]. Les opérateurs sont connectés via des "queues" et chaque opérateur dispose en mémoire des tuples requis pour répondre à l'opération dont il a la charge. Le workflow fonctionne grâce à un paradigme de "pipeline" où chaque résultat produit par un opérateur est transmis aux opérateurs suivant dans la chaîne de traitement. Toutes les données sont traitées en mémoire vive et certains tuples ou résultats sont gardés en mémoire et partagés entre les différentes requêtes qui sont exécutées simultanément et de manière continue.

4.3.1.1 Ordonnancement et répartition des traitements

Ce système peut être être modélisé par un workflow où les flux de données sont traités en continu et les requêtes réévaluées en accord avec ces arrivées de données [Deshpande et al., 2007]. Ceci nécessite des techniques spécifiques pour la gestion, l'ordonnancement et la répartition des requêtes. En effet, un tel système doit être capable de créer et supprimer de nouveaux processus

lorsque cela est nécessaire pour rester efficace alors que l'état du système évolue. De plus, les systèmes temps-réels doivent aussi traiter des requêtes multiples de manière parallèle, ce qui nécessite d'associer les données entrantes aux différentes requêtes et de définir des opérateurs communs pour réduire le nombre d'opérateurs. Enfin, le système doit pouvoir réordonner les opérateurs et traiter les données pour choisir le meilleur plan de requête, lorsque les flux de données entrants et les requêtes à traiter varient.

Scalabilité et optimisation.
L'arrivée des données et le nombre de traitements intervenant dans le système étant fluctuant, il est nécessaire que les requêtes soient adaptatives c'est-à-dire que le système s'adapte en fonction de l'arrivée des données et des traitements à faire pour ne pas être saturé en dédoublant un opérateur, en modifiant le plan de requête ou en supprimant une partie des données.

Gestion de requêtes multiples et/ou concurrentes.
Dans un système temps-réel, plusieurs requêtes s'exécutent en continu et peuvent solliciter les mêmes données ou avoir des parties de traitement en commun. Dans ce cadre, l'enjeu est de répartir au mieux les traitements et de les factoriser au maximum lorsque cela est possible à l'image du "shared-execution paradigm" proposé dans telegraphcq [Chandrasekaran and Franklin, 2003].

Ordonnancement des opérateurs.
La construction du schéma d'exécution des requêtes est également un levier. Le système doit être capable d'ordonner les différentes sous-tâches pour limiter les traitements. Le traitement de requêtes dans un système de base de données est plus aisé, car les exécutions sont formulées par l'utilisateur et les mises a jours de la base de données sont peu fréquentes, ainsi l'ordonnancement des tâches et opérations répondant à la requête est plus simple à optimiser. À l'inverse pour un système de gestion de flux en temps-réel, l'environnement est dynamique et au fur et à mesure que les données arrivent dans le système de nouvelles requêtes peuvent être exécutées et certains opérateurs peuvent être ajoutés, divisés, réduits ou supprimés pour gérer les variations de flux de données.

4.3.1.2 Sémantique, évaluation et approximation

Une autre problématique importante concernant les systèmes temps-réel est que les données sont traitées en mémoire. C'est pourquoi, le volume de données à manipuler et à traiter doit être réduit, notamment en utilisant des techniques de compression, de délestage de données ou d'échantillonnage

[Golab and Özsu, 2003]. Une autre approche consiste à élaborer des algorithmes optimisés qui ne traitent qu'un nombre de fois limité les données en fournissant une réponse approximative. Enfin, des résumés de données peuvent être utilisés dans le but de traiter les données de manière incrémentale [Ghanem et al., 2010] et éviter ainsi de réévaluer l'entièreté de la requête de manière systématique.

Techniques d'échantillonnage et d'agrégation.
Toutes les données et les traitements s'effectuant en mémoire, il est nécessaire d'utiliser des techniques permettant de réduire le nombre de données présentes en mémoire par exemple en filtrant, compressant ou échantillonnant les données

Évaluation incrémentale des données.
La notion d'évaluation incrémentale, en effet les requêtes s'effectuent en continu et réévaluer systématiquement l'ensemble des données en mémoire ou sur la fenêtre temporelle considérée est coûteux. Une des techniques possibles est donc sur les opérateurs non bloquants (comme pour l'opérateur somme par exemple) d'agréger les données systématiquement au fur et à mesure. Il s'agit de traiter les données de manière incrémentale plutôt que de tout réévaluer, cela implique de conserver et stocker les résultats intermédiaires et éventuellement les mettre dans un index pour y accéder plus vite.

One pass algorithm.
Pour les opérateurs dits bloquants (qui ne peuvent s'effectuer au fil de l'eau et nécessitant de traiter l'ensemble des dernières données entrées dans le système), il faut étudier des algorithmes approximatifs qui permettent de donner des réponses parfois de moins bonnes qualités, mais qui répondent rapidement sans être obligés d'examiner l'entièreté du flux par exemple en considérant les flux relatifs une fenêtre temporelle et de préférence en limitant le nombre de fois où les données sont "lues" par le système, ces techniques et algorithmes ne parcourant qu'une seule fois les flux de données sont également appelés *one pass algorithm*.

Fenêtrage et vues partielles.
Pour la gestion et le traitement de flux de données, la notion de granularité et le type de fenêtrage sont d'importance. Il est en effet nécessaire de développer des techniques pour choisir un volume de données ou un laps de temps optimal pour répondre à une requête de manière satisfaisante. Une des méthodes répondant à cette problématique est l'usage de fenêtres spécifiques, et de nombreux travaux ont été développés en rapport avec différents types de fenêtres qu'elles soient landmark, glissantes, ou dites de prédicat (tant que la donnée respecte certaines conditions, elle est conservée dans la fenêtre) ... Il existe des opérateurs pour fusionner les résultats sur différentes fenêtres et donc

avec un niveau de granularité plus "grossier" (s'il s'agit de données plus anciennes, une information partielle est suffisante, en partant du postulat que les données les plus pertinentes sont les plus récentes).

4.3.1.3 Distribution des données et des traitements

À cela s'ajoutent les difficultés inhérentes à la distribution des données et des traitements. En effet, les deux enjeux principaux (et ceux secondaires associés) peuvent être étendus dans notre contexte de système distribué. Une telle distribution ajoute de nouvelles problématiques comme la tolérance aux pannes et la distribution des données. Dans ce contexte l'endroit où sont stockés les synopses, l'allocation dynamique des opérateurs, le transfert de données d'un noeud à un autre doivent aussi être pris en compte [Shah et al., 2004]. Ceci implique de distribuer les données aussi bien les données que les traitements sur les noeuds de l'architecture, la distribution des données pouvant se faire aussi bien par couverture spatiale, par période de temps que par groupe de trajectoires (regroupées par exemple sémantiquement) [Sun et al., 2013].

Répartition et propagation des données.
Par défaut, chaque nœud traitera un ensemble de données relatif à la couverture des données (spatiale par exemple), cependant cela nécessite de mettre à jour les états des entités considérées au fur et à mesure qu'elles se déplacent.

Répartition des requêtes.
Cet enjeu déjà rencontré dans le cas d'un système centralisé est d'autant plus important sur une architecture distribuée. L'exécution des requêtes et l'ordonnancement des opérateurs doit être réparti sur l'ensemble des nœuds ce qui est autrement plus complexe notamment avec une contrainte de communication entre les noeuds et d'éventuelles pannes.

Réduire les coûts de communication et les échanges.
Sur une architecture distribuée, le plus coûteux en termes de performances est la transmission d'informations entre les nœuds, il convient donc de mettre en place des mécanismes pour limiter les échanges, par exemple en faisant effectuer les traitements par le nœud possédant les données, en ne faisant transiter que des résultats intermédiaires ...

Enfin, un des derniers enjeux et il est la base de la thèse avancée dans ce manuscrit est la

nécessité d'un "couplage" avec des informations anciennes. Typiquement, pour un système souhaitant détecter une anomalie, si la période de temps considérée est trop courte (ce qui est le cas avec un système temps-réel, puisque nous nous limitons au maximum pour ne conserver que les données les plus récentes) comment distinguer quelque chose de "normal" d'une anomalie ? Dans ce travail, nous avons choisi de prendre un système orienté temps-réel et de nous servir de données archivées pour mettre en contexte et "donner du sens" aux flux de données traités en temps-réel. Nous avons ici défini les bases d'un système hybride orienté temps-réel et dont la partie online doit être prépondérante.

4.3.2 Architecture d'un système temps-réel pour la gestion d'objets mobiles

Compte tenu des enjeux associés à la gestion d'objets mobiles en temps-réel définis dans la partie précédente, l'architecture suivante est proposée pour répondre au mieux à ces défis (cf figure 4.4).

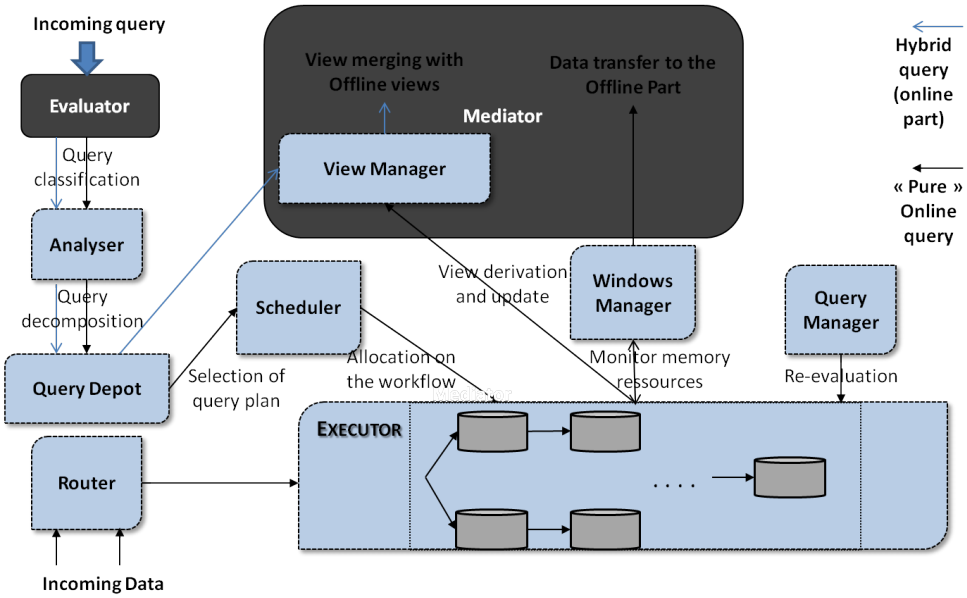

FIGURE 4.4 : Schéma explicatif d'un système temps-réel au sein d'un système hybride

Deux différentes entrées sont considérées, une relative au système qui collecte les données et les traite "à la volée" (incoming data) et une relative aux requêtes formulées par l'utilisateur qui sont ajoutées à l'ensemble des requêtes continues exécutées de manière périodique dans le système (incoming query). Le lien entre ces deux composants est fait par l'exécuteur qui est le workflow à proprement parlé, responsable d'associer les objets mobiles et les requêtes concernant les zones considérées.

Les flux entrants sont reçus, prétraités et soumis aux opérateurs via le routeur tandis que les nouvelles requêtes sont analysées et traduites sous forme d'un plan de requête par l'analyseur. Lorsqu' une nouvelle requête est formulée, deux alternatives sont possibles en fonction du résultat donné par l'évaluateur sur la partie hybride. Si la requête entrante dépend uniquement des données temps-réel, alors elle est ajoutée au système et s'exécute de manière continue tandis qu'une sous requête temps-réel extraite de la requête hybride n'est exécutée qu'une fois. Les plans de requêtes associés à chacune des requêtes sont stockés dans le dépôt de requêtes et gérés par le "query manager" qui s'occupe d'indexer les requêtes et de procéder à la réévaluation régulière des requêtes continues. L'Exécuteur fait le lien entre les données envoyées par le routeur et les requêtes qui ont été traduites et décomposées en opérations successives de manière optimisée par l'ordonnanceur. Le gestionnaire de mémoire est responsable des fenêtres glissantes et du transfert des données depuis la partie online vers la partie offline via le Médiateur (cf Figure 4.3) lorsque les données deviennent obsolètes (i.e. excèdent la durée de la fenêtre temporelle considérée). Enfin, le gestionnaire de vues est en charge du maintien des vues obtenues et mise à jour par exécutions successives des requêtes continues.

Si la requête entrante est hybride, alors le système vérifie si la sous-requête online associée est déjà présente dans l'ensemble des requêtes continues exécutées par le système. Si la requête est déjà présente alors le système, plutôt que de réévaluer la requête, considère la vue qu'il obtient à partir du gestionnaire de vues. Si la requête est absente dans le dépôt de requêtes le résultat est obtenu à partir de la fenêtre temporelle distribuée en générant une vue comme c'est le cas pour une requête dépendant uniquement des données temps-réel.

4.4 Modèle de données et paradigme d'événements pour le traitement d'objets mobiles

4.4.1 Définitions et modèle de données pour le stockage et le traitement d'objets mobiles

Nous considérons un grand flux de positions, ces positions représentant un ensemble d'objets mobiles dans une région S $\subset \mathbb{R}^2$. Nous représentons ces objets mobiles en tant que points sous la forme de tuples *(x,y,t)* où le couple (x,y) représente une coordonnée p (cf. Définition 1) et $t \in T$ (cf. Définition 2) correspond au temps à laquelle la position a été enregistrée. Dans le cadre de notre étude, nous considérons une sous-région de $\mathbb{R}^2$ telle que S $\subset$ P. Ainsi un segment sg est défini par une paire (a,b) $\in$ S. De manière analogue pour la dimension temporelle, un intervalle de temps $[t_1, t_2] \subset$ T est défini par l'ensemble des instants de T contenus entre t_1 et t_2.

Définition 1 : Domaine spatial *Soit P le domaine spatial qui contient l'ensemble des paires de valeurs <x,y> coordonnées contenues dans le plan* $x,y \in \mathbb{R}^2$.

Définition 2 : Domaine temporel *Soit T le domaine temporel définit comme un ensemble infini d'instants* t_i *ordonnés dans le temps T.*

Définition 3 : Trajectoire *Une trajectoire Traj est définie comme une liste de positions où les positions sont définies comme des points issues du domaine Spatial P et les temps auxquels elles ont été enregistrées comme des instants appartenant au domaine temporel T. Une trajectoire est notée de la manière suivante Traj* : $Id \in \mathbb{N} \rightarrow List(p_i \in S, t_i \in T)$ *où* p_i *est la position de l'objet mobile à l'instant* t_i *avec* $i \in \mathbb{N}$ *et Id l'identifiant de la trajectoire. Chaque trajectoire peut être modélisée également comme une polyligne donnée par l'ensemble des segments dérivés de la liste des positions successives reçues correspondant à l'objet mobile. Plus formellement une trajectoire peut s'écrire Traj* : $Id \in \mathbb{N} \rightarrow List(p_i \in S, t_i \in T)$ *et peut être "traduite" de la manière suivante Poly* : $Id \in \mathbb{N} \rightarrow List(s_i \in S^2, \tau_i \subset T)$ *où chaque* s_i *est une paire de deux positions successives et* τ_i *est l'intervalle de temps correspondant au temps écoulé entre l'enregistrement des deux positions successives.*

Nous introduisons ensuite la notion de grille et cela pour plusieurs raisons. La première est de pouvoir considérer les trajectoires à un niveau macroscopique, ce qui est utile notamment pour comparer plusieurs trajectoires entre elles. Le second intérêt d'utiliser une grille est de disposer par ailleurs d'un index pour l'étude de trajectoire. Enfin, l'intérêt premier dans le cadre d'une architecture hybride pour la gestion et le traitement d'objets mobiles est de pouvoir fusionner les flux temps-réels avec des données archivées ou résumés associés de manière simple (i.e. il est plus facile de fusionner ou de réaliser un traitement combiné si la structure spatiale est la même pour les parties temps-réel et données anciennes).

Une approche à base de cellules décompose l'espace physique en un ensemble fini de zones disjointes, construisant un partitionnement qui couvre l'entièreté de l'espace. Cette approche fournit la capacité implicite de trouver des adjacences entre cellules voisines. Deux principaux types de maillages peuvent être distingués : soit en utilisant un maillage régulier où l'espace est divisé en cellules qui ont exactement la même forme et la même taille, soit en utilisant un maillage irrégulier dont le but est de fournir une décomposition de l'espace qui soit adaptée et plus pertinente pour représenter la complexité de l'environnement étudié. Les cellules formant un partitionnement irrégulier peuvent être de formes et de tailles différentes à l'image d'un découpage en cellules de

Voronoi.

Parmi les maillages réguliers, le plus commun est l'utilisation d'un modèle de grille. En effet, les techniques utilisant une grille s'implémentent facilement et peuvent permettre une représentation de l'espace pertinente. Une approche grille fournit un modèle spatial avec des propriétés géométriques continues favorisant aussi bien l'usage de requêtes géométriques que des interactions au niveau des cellules. L'étendue de la surface étudiée et le niveau de granularité choisis sont deux paramètres essentiels qui doivent être déterminés à l'avance pour l'obtention de la grille. La précision de la grille ainsi obtenue dépend de la résolution choisie pour les cellules. Ainsi, un des compromis principaux concerne la préservation d'un haut niveau pour la recherche d'information et son impact sur la consommation en mémoire et temps de calcul, plus spécifiquement lorsque la surface d'étude est grande. Une grille composée de cellules fines fournit des résultats plus précis et pertinents en termes de locations, mais peut introduire de lourds traitements. Faire avec un grand nombre de cellules peu augmenter le temps de traitement des requêtes de manière exponentielle, ceci menant à des problèmes de performance et de scalabilité. De plus, un maillage régulier ne représente pas les objets précisément avec des formes arbitraires.

D'autres structures de données ont donc été développées pour contrebalancer les inconvénients d'une approche à base de cellules statiques, à l'image des quadtree [Samet, 1984] pour un espace à deux dimensions. Par exemple, une région structurée en termes de quadtree est utilisée communément quand moins de détails sont requis pour représenter l'espace d'étude. Cela permet aussi de modéliser l'espace de manière adaptée aux besoins en jeu en reconstruisant de manière récursive de plus petits carrés pour avoir un niveau de détail suffisant lorsque cela est nécessaire. Cependant, le désavantage de cette approche tient au fait que cela manque de flexibilité spécifiquement lorsqu'il faut faire face à un environnement très dynamique. En effet, un environnement dynamique comprend des utilisateurs se déplaçant, des capteurs et des éléments qui peuvent être distribués de manières diverses dans l'espace. Cela signifie que quand la distribution des objets mobiles change, une mise à jour significative peut affecter l'entièreté du quadtree en temps-réel.

Définition 4 : Grille. *G est définie comme une grille régulière telle que G=Set($C_{i,j}$) et chaque cellule disjointe $C_{i,j}$ est de taille fixe $size_{cell}$, définie par $f : p \in S \rightarrow C_{i,j}$ / i = ($x/size_{cell}$) et j=($y/size_{cell}$), avec $C_{i,j}$ la cellule correspondant à la i^e abscisse et la j^e ordonnée pour $i \in 1..n$, $j \in 1..n$.*

Différentes notations additionnelles relatives à cette grille sont développées dans la suite. Pour chaque cellule $C_{i,j}$ relative à la i^e abscisse et la j^e ordonnée pour i $\in 1..n$, j $\in 1..n$, les indices de mesure suivants sont introduits :

— $Npos_{i,j}$ τ correspond au nombre de positions qui ont été enregistrées dans la cellule $C_{i,j}$ pendant l'intervalle de temps $\tau \subset T$, plus formellement nous avons $Npos_{i,j}$ $(\tau) = \sum_{Id=1}^{nbpositions} t_{Id}$ during τ /

$$t_{Id} = \begin{cases} 1, & \text{if} p_k \in C_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

où nbpositions est le nombre de positions qui ont été enregistrées pendant l'intervalle de temps τ.

— $Nobj_{i,j}$ τ donne le nombre d'objets mobiles distincts ayant enregistré au moins une position dans la cellule $C_{i,j}$ pendant la période de temps $\tau \subset T$, plus formellement $Nobj_{i,j}$ $(\tau) = \sum_{Id=1}^{nbobj} t_{Id}$ pendant τ /

$$t_{Id} = \begin{cases} 1, & \text{if } \exists (p_k, t_k) \in Traj(Id)/p_k \in C_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

où nbobj est le nombre d'objets mobiles distincts dans la position a été enregistrée sur la période de temps τ.

— $Ncross_{i,j}(\tau)$ donne le nombre d'objets mobiles distincts ayant intersecté la cellule $C_{i,j}$ pendant l'intervalle de temps $\tau \subset T$, ce qui signifie ici le nombre d'objets mobiles dont la trajectoire (modélisée en tant que polyligne) a intersecté le cellule correspondante $C_{i,j}$, plus formellement $Ncross_{i,j}(\tau) = \sum_{Id=1}^{nbobj} t_{Id}$ durant l'intervalle de temps τ /

$$t_{Id} = \begin{cases} 1, & \text{if } Poly(Id) \cap C_{i,j} \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

où nbobj est le nombre d'objets mobiles distincts dont la position a été enregistrée sur la période de temps τ.

Il est immédiat de noter que $Ncross_{i,j}(\tau) > Nobj_{i,j}(\tau)$.

Compte tenu de la grille ainsi construite et des définitions précédentes, nous pouvons étendre la notion de trajectoire. En effet, celles-ci peuvent être considérées non plus comme une succession de points repérés dans le temps ou une liste de segments associés à des intervalles de temps, mais comme à une succession de cellules $C_{i,j}$ et à chaque cellule est affecté un intervalle de temps $[t_a, t_s] \subset$ T où t_a est le temps d'arrivée de l'objet mobile dans la cellule et t_s celui de sortie. Cette structure de données nous permet de comparer des trajectoires de manière plus "grossière", mais également de réduire les coûts de traitement pour certains cas, en ne comparant que la succession des cellules deux à deux, plutôt que chaque position deux à deux. Ceci illustre et montre l'intérêt de notre système permettant de stocker des "vues" ou

synopses à plusieurs niveaux, le principe étant de filtrer la plupart des cas "normaux" et pouvoir "zoomer" lorsqu'une anomalie se produit au niveau du trafic, ce qui ici correspond à considérer la trajectoire d'une manière différente suivant la précision et la nature de l'analyse que nous voulons faire.

Concernant les notations associées au temps, $t_{current}$ est défini comme l'estampille relative au temps courant tandis que t_{old} correspond à l'estampille temporelle du plus vieil enregistrement en base de données. τ_{Past} lui définit l'intervalle de temps concernant les données historiques depuis la première estampille temporelle de la base de données jusqu'à la dernière de la base de données historique (l'intervalle de temps correspond en fait à l'ensemble des données privées de celles manipulées dans la fenêtre temporelle actuelle de range ω et de slide β), plus formellement $\tau_{Past} = [\ t_{old}, t_{current-\omega}\ ]$. $\tau_{current}$ correspond à l'intervalle de temps pour l'ensemble de la fenêtre glissante, autrement dit nous avons $\tau_{current} = [\ t_{current-\omega}, t_{current}\ ]$. Finalement, nous avons τ_β qui correspond à l'intervalle de temps entre $t_{current}$ et $t_{current-\beta}$, plus formellement $\tau_\beta = [\ t_{current-\beta}, t_{current}\ ]$.

Le déplacement d'objets mobiles peut être modélisé en termes d'événements. La notion d'événements pour traiter des données de trajectoires a été explorée récemment dans [Patroumpas et al., 2015] où un système de reconnaissance d'événements complexes est utilisé pour identifier quelques comportements typiques dans le domaine du trafic maritime. La notion d'événement et de recherche hybride de motifs (i.e. hybride signifie sur les données temps-réels et archivées conjointement) développée dans [Balazinska et al., 2007] peut être étendue à notre contexte d'objet mobile.

Nous considérons les définitions suivantes :

Définition 5 : Événement basique. *Un événement basique concernant un objet mobile peut être noté comme suit E=E(EventId,TypeId,time,<a_1,...,a_n>), où EventId identifie l'événement correspondant à l'objet mobile, Typeid correspond au type d'événement considéré, la valeur "time" définit le moment auquel l'événement se produit et <a_1,...,a_n> correspond aux attributs spécifiques à l'événement.*

Parmi ces événements basiques, nous pouvons en décrire quelques-uns. Soit o un objet mobile se déplaçant à une vitesse v et un angle θ donnés, les différents événements basiques associés peuvent être décrits comme suit :

— A l'aide la grille construite (cf Définition 4), l'événement l'objet mobile o a enregistré sa position dans la cellule $C_{i,j}$ peut être considéré.

— Un événement à considérer est "l'objet mobile a accéléré/décéléré" cet événement est mesuré entre deux positions successives reçues et grâce à une valeur seuil δ_v.

— Un autre événement est "l'objet mobile a changé de direction", aussi mesuré entre deux positions successives à l'aide là encore d'une valeur seuil δ_θ.

Un objet mobile peut être considéré de différentes manières, la façon la plus naturelle est comme une succession de positions dans le temps. Seulement, cette représentation ne permet pas de caractériser le déplacement ou le comportement d'un objet en particulier de manière satisfaisante. Dans la suite, nous proposons un niveau d'abstraction supplémentaire permettant de comparer les comportements des objets mobiles et discriminer plus efficacement ceux qui seraient "anormaux". La notion d'événement est un élément clef de notre système et l'ensemble des requêtes peuvent être modélisées à l'aide de cette notion. Un événement s'inscrit dans le temps et l'espace, les événements dits basiques ont lieu à un instant t et un endroit repéré par une cellule C (i,j), tandis que les événements complexes ou composites (qui peuvent être vus comme une succession d'événements basiques) concernent un intervalle de temps allant du premier événement au dernier enregistré dans un ensemble de cellules C (i,j). Un événement basique est "l'objet mobile a enregistré sa position dans la cellule C (i,j) à l'instant t", l'événement complexe direct associé est "l'objet mobile est passé dans la cellule C (i,j) entre l'instant t1 d'entrée avec la position p1 et l'instant de sortie t2 avec la position p2". La notion d'événements a deux avantages suivant notre approche, le premier est comme nous l'avons dit précédemment de caractériser le comportement d'un objet mobile et d'introduire une notion de "sémantique" plus haut niveau par rapport à une représentation "naturelle" du mouvement. Le second avantage est de fournir un niveau d'abstraction qui peut être agrégés facilement et permettre de comparer des flux temps-réel avec des connaissances synthétisées extraites d'une base de données archivées, la plupart des travaux sur de l'hybride concernant plus la combinaison a un niveau architectural et pas a un niveau "modèle de données".

Définition 6 : Événement complexe pour un objet mobile. *Un événement complexe concernant un objet mobile est défini comme suit E=E(EventId, C=<e_1 Opr e_2 Opr... e_n>, ts, te, <a_1,...,a_n>), où C est un ensemble d'événements complexes ou basiques qui sont liés les uns aux autres par des opérateurs opr qui peuvent correspondre par exemple à une disjonction ou une conjonction. Enfin, ts et te correspondent respectivement aux moments de début et de fin de l'événement.*

Les différents événements complexes peuvent être construits à partir des événements basiques

définis précédemment :

— Un événement complexe peut concerner le temps d'arrivée et de départ d'un objet mobile dans une zone (i.e. une celle de la grille G définie précédemment cf Définition 4) et la succession des positions associées au sein de cette zone.

— Un événement traditionnellement intéressant ou étudié dans le cadre d'objets mobiles est la détection de "move and stop" couvert par notre définition, si la succession de décélérations est progressive jusqu'à rester constante à 0 pendant un moment, un stop est identifié.

— Un événement pouvant être intéressant dans le cadre de l'étude d'objets mobiles est la détection d'un changement de direction brutal couvert par notre définition, si la succession d'événements basiques de changement de direction est progressive et significative jusqu'à dépasser une valeur seuil, un changement de direction brutal est détecté.

Une trajectoire peut donc être traduite ou décrite en fonction d'une suite d'événements et à l'aide la grille définie précédemment. Ainsi une trajectoire s'exprime aussi bien de la manière suivante : TrajCell : Id $\in \mathbb{N} \rightarrow$ List($C_{i,j} \in G$, $\tau_i \subset$ T) que par la succession de l'événement basique l'objet mobile o a enregistré sa position dans les cellules $C_{i,j}$. Cependant en se restreignant seulement à l'enchaînement de ces événements, de l'information sur le comportement de l'objet mobile est perdue. Par conséquent, nous utilisons en complément les notions de vitesse et de cap subsistent pour mieux caractériser le comportement de l'objet mobile. autrement dit on peut décrire une trajectoire comme l'ensemble des cellules par lesquelles l'objet mobile ainsi que les intervalles de temps correspondant à la traversée de chaque cellule. Si on considère l'événement l'objet mobile o a enregistré sa position dans la cellule $C_{i,j}$, la trajectoire ou le comportement de cet objet mobile peut être traduit à un niveau "basique" comme la succession de ces événements. Pour compléter ou enrichir cette définition de comportement, nous décrivons des événements particuliers aux objets mobiles que nous espérons les plus génériques possible sans prétendre à l'exhaustivité. Nous considérons un objet mobile se déplaçant à une vitesse v et un angle θ donnés, un événement associé est "l'objet mobile a accéléré/décéléré" cet événement est mesuré entre deux positions successives reçues et grâce à une valeur seuil δ_v. Un événement traditionnellement intéressant ou étudié dans le cadre d'objets mobiles et la détection de "move and stop" qui peut être couvert par notre définition, si la succession de décélérations est progressive jusqu'à rester constante à 0 pendant un moment, un stop est identifié. Un autre événement associé est "l'objet mobile a changé de direction", aussi mesuré entre deux positions successives à l'aide là encore d'une valeur seuil δ_θ.

D'autres types d'événements faisant intervenir plusieurs objets mobiles peuvent avoir lieu, ce

qui nécessite de considérer non pas seulement le comportement d'un individu, mais la concomitance d'événements arrivant pour différents objets mobiles.

Définition 7 : Événement complexe pour un groupe d'objets mobiles. *Un événement complexe concernant un groupe objets mobiles est défini comme suit E=E(EventId, C=<*e_1, MO_{Idi} *Opr* e_2, MO_{Idj} *Opr... *e_n*>, ts, te, <*a_1*,...,*a_n*>), où C est un ensemble d'événements complexes ou basiques qui sont liés les uns aux autres par des opérateurs opr qui peuvent correspondre à une disjonction, une conjonction ... concernant différents objets mobiles (MO) d'identifiant Idi, Idj tandis que ts et te correspondent respectivement aux moments de début et de fin de l'événement.*

L'exemple de tels événements complexes peut concerner les interactions entre objets mobiles notamment avec la présence de prédateur et de proie en zoologie ou bien des phénomènes d'évitement pour le domaine maritime [Pallotta et al., 2013] pour ne citer que ces exemples . Cela permet également de gérer des éléments faisant intervenir plus d'éléments pour identifier des essaims ou groupes d'objets mobiles se délaçant ensemble [Giannotti et al., 2011].

Ces définitions d'événements doivent permettre de répondre plus facilement et rapidement à des requêtes faisant intervenir des motifs spatio-temporels, mais aussi de caractériser de manière plus précise les déplacements d'objets mobiles. Les motifs spatio-temporels exprimés comme séquence d'événements peuvent fournir des vues à un autre niveau pour filtrer et comparer des éléments avec les flux entrants dans le système. Pour ce faire, il est nécessaire d'utiliser conjointement l'exécution de requêtes en temps-réel définie dans la section précédente 4.3.1 avec des motifs spécifiques conservés et considérés comme des vues ou des résumés de flux agrégés, ces motifs pouvant être stockés en mémoire ou sur disque en fonction de l'espace disponible. Différents types de motifs peuvent être distingués [Li, 2014], d'abord les motifs individuels sont relatifs principalement au comportement d'un objet mobile particulier dans le but d'identifier des éléments ou des motifs observés de manière périodique et peuvent être identifiés à partir des définitions d'événements basiques et complexes spécifiques à un objet mobile. Ensuite, les motifs relatifs à des paires d'objets concernent le traitement de couples d'objets mobiles pour détecter des phénomènes de collision ou de fuite par exemple traité à l'aide de définition qui suivront (i.e pouvant être modélisé par la notion d'événements relatif à un groupe d'objets mobiles). Enfin, les motifs d'agrégation sont extraits pour identifier des comportements spécifiques sur un ensemble de trajectoires et notamment des clusters d'objet mobiles et dérivés (convois ...), des motifs de trajectoires fréquentes ou des zones denses. Ces motifs doivent évoluer et être mis à jour de manière incrémentale pour la partie temps-réel afin de réduire le traitement engendré par une réévaluation complète.

L'intérêt de notre approche combinant l'usage d'une grille servant d'index et de base pour le partitionnement avec la notion d'événements est qu'il permet une comparaison rapide entre différentes trajectoires ou de détecter des anomalies grossières en considérant uniquement la succession des cellules intersectées par l'objet mobile. Puis s'il y a lieu (i.e. si une anomalie est repérée) alors il est possible de "récupérer" et examiner la succession d'événements s'étant produits à l'intérieur de la cellule (changement de direction, de vitesse). C'est cette approche à plusieurs niveaux de précision et granularité qui fait la spécificité de notre traitement, d'autres travaux ayant étant faits, mais décorrélant les données de positions elles-mêmes de la notion d'événement et ne proposant pas de méthode d'index sur ces événements [Patroumpas et al., 2015].

Dans cette partie nous avons décrit l'ensemble des éléments basiques que nous considérons pour le traitement d'objets et avons introduit la notion d'événements pour mieux les caractériser. Ce modèle de données proposé permet de répondre aussi bien à des requêtes traditionnellement formulées pour les objets mobiles (k-plus proche voisin ou range query en offline ou online) qu'à des requêtes hybrides nécessitant l'usage conjoint des données anciennes et temps-réel. Dans la suite, nous définissons l'ensemble des requêtes traitées par notre système et proposons une taxonomie des requêtes de mobilité.

4.4.2 Types de requêtes pour le traitement d'objets mobiles

Depuis le développement de base de données spécifiques aux objets mobiles ou MOD (Moving Object Databases), différents travaux ont été réalisés pour traiter souvent une requête en particulier comme la recherche continue de k plus proches voisins [Xiong et al., 2005] ou range query [Kalashnikov et al., 2002]. A la place, nous souhaitons fournir un modèle de traitement capable de gérer différents types de requêtes spécifiques aux objets mobiles qui pourront être formulées par un utilisateur et exécutées de manière générique comme dans [Mokbel and Aref, 2005]. Le tableau 4.4 résume une taxonomie des requêtes qui doivent être prises en charge par notre système.

Certains travaux définissent une typologie de requêtes pour les objets mobiles en deux catégories celles qui sont basées sur les points et celles qui sont basées sur des trajectoires. Cela dépend du niveau de granularité nécessaire pour le traitement de la requête (i.e. est ce que la requête concerne un instant t et donc des points associés ou un intervalle de temps donc une sous trajectoire associée). Nous proposons une classification différentes pour la distinction d'objets mobiles.

Les différentes requêtes relatives aux objets mobiles peuvent être classifiées en trois catégories [Deng et al., 2011] à savoir P-requêtes, R-requêtes et T-requêtes. D'abord, les requêtes peuvent concerner la comparaison de trajectoires avec des localisations spécifiques ou POI (point d'intérêt).

Un exemple de telles requêtes aussi appelées P-requête est "Quels sont les objets mobiles qui ont été proches de ce point spécifique ?". Ensuite, pour certaines requêtes appelées R-requêtes les trajectoires et les régions sont considérées, par exemple un problème intéressant peut être de déterminer les trajectoires qui traversent une région spatio-temporelle ou bien d'identifier les régions les plus empruntées par des objets mobiles. L'illustration d'une telle requête est par exemple "Quelles sont les régions les plus traversées durant la période temps τ ?".Enfin les T-requêtes sont des requêtes relatives à l'analyse de similarité entre trajectoires. Ceci peut être intéressant pour identifier des phénomènes de collision, des couloirs usuellement utilisés ou des motifs/comportements spécifiques. Voici un exemple illustratif de cette catégorie de requêtes "Quels sont les objets mobiles qui pourraient rentrer en collision avec cet objet mobile spécifique d'ici les cinq prochaines minutes ?". Voilà les différentes requêtes génériques qui doivent être prises en charge par un système hybride de gestion d'objets mobiles.

Cette catégorisation des requêtes en fonction de la nature des objets spatiaux considérés bien que profitable n'est pas exhaustive et mérite d'être mise en perspective au vu de la période de temps concernant la résolution de la requête. Nous distinguons donc les requêtes également par rapport aux périodes temporelles sur lesquelles elles portent comme le font les auteurs dans [Nguyen-Dinh et al., 2010] pour la construction d'index en considérant les requêtes portant sur ce qui s'est passé (past query), sur ce qui se passe actuellement (present query) et sur ce qui se passera (predictive query). Les requêtes "passées" concernent des données "anciennes" (i.e. stockées en base de données) tandis que les requêtes "présentes" nécessitent l'utilisation de données reçues au temps courant et plutôt récentes, requérant parfois d'être complétées par l'information extraite des données archivées. Les requêtes prédictives font intervenir aussi bien des données récentes qu'anciennes ceci en fonction de la précision de la requête et de la période de temps séparant l'instant actuel de l'instant de la prédiction. En effet, s'il faut estimer la position d'un objet mobile dans deux minutes, alors cette position peut être calculée en prenant la position précédente seulement, en considérant cap et vitesse précédente. L'erreur sera ainsi minime, tandis que si la position à estimer concerne la position de l'objet dans dix minutes, ceci peut nécessiter l'utilisation conjointe de données récentes (cap, vitesse) qui sont comparées avec des données plus anciennes ou des résumés de données pour avoir une meilleure précision.

Enfin, dans notre système hybride orienté temps-réel les requêtes peuvent être distinguées suivant qu'elles sont exécutées une fois par le système ou si elles sont continues (i.e. elles sont réévaluées régulièrement alors que les données entrent dans le système). Les requêtes continues ou persistantes pourront permettre d'extraire des connaissances ou de former des résumé de données, ceci dans le but de réduire les temps de calculs pour des requêtes exécutées une fois mais aussi de répondre à des requêtes de "plus haut niveau" qui pourraient s'apparenter à de la fouille de données.

TABLE 4.4 : table représentative de différentes requêtes pour objets mobiles

Type de requêtes	Requêtes historiques	Requêtes présentes	Requêtes prédictives
P-Requêtes	"Quels sont les objets mobiles qui se sont approchés de ce point spécifique ?"	"Quels sont les points spécifiques proches de cet objet mobile ?"	"Quels sont les objets mobiles qui seront proches de ce point d'ici les 5 prochaines minutes ?" "Quelle sera la position de cet objet mobile au temps t ?"
R-requêtes	"Quels sont les objets mobiles qui ont traverse cette zone durant la période de temps τ ?" "Quelles sont les régions qui sont le plus traversées ? "	"Quels sont les objets mobiles présents à l'intérieur de cette zone ? " "Quels sont les objets mobiles qui sont en train de quitter la zone ? " **"Quelles sont les zones de couverture ? "**	"Quels sont les objets mobiles qui vont entrer dans la zone ? "
T-Requêtes	"Quelle est la trajectoire représentative de ce type d'objet mobile ?"	"Quelle est la vitesse moyenne des objets mobiles dans cette zone spécifique ? "	"Quels sont les objets mobiles qui devraient se croiser d'ici les 5 prochaines minutes ?"

Un exemple de ce type de requêtes est présenté dans le tableau 4.4 et écrite en gras, "Quelles sont les zones de couverture ? ", cette requête qui fera l'objet d'un chapitre, est un cas courant de ce qui peut se produire lorsqu'il s'agit de suivi d'objets mobiles, à savoir la perte de signal de ces objets en cas de défaillance, d'extinction ou tout simplement de non-couverture d'une zone. Ce genre de requêtes nécessite le concours de données anciennes et des flux reçus en temps-réel, les données anciennes permettant de savoir quelles sont les zones usuellement couvertes, alors que les flux permettent d'avoir une vision plus claire des phénomènes actuels (la réception du signal n'étant pas absolue, mais pouvant varier notamment à cause des conditions climatiques ou d'un défaut du récepteur/émetteur).

Concernant le modèle d'événement proposé dans la partie précédente, celui-ci peut être exporté chacune des requêtes précédentes peut s'exprimer en termes d'événements. Pour chaque cellule, nous stockons l'ensemble des événements associés (et donc les objets mobiles qui ont pénétré dans la cellule concernée). Ces différentes requêtes peuvent être formulées et évaluées à l'aide de la notion d'événements décrite dans la partie précédente. Prenons l'exemple d'une range query "quels sont les objets mobiles qui sont passés dans la zone Z pendant l'intervalle de temps I ?". Ces ranges queries ont pour but de trouver et identifier certains objets ou endroits d'intérêt dans un intervalle ou une aire définie par l'utilisateur [Kalashnikov et al., 2002]. Ces requêtes dans un contexte de requête continue doivent mettre à jour les informations pertinentes relatives à l'évolution des objets mobiles. Les intervalles de recherche peuvent correspondre à des formes circulaires ou rectangulaires dans lesquelles chaque objet d'intérêt peut être localise. Pour ce faire, nous définissons la zone Z comme un ensemble de cellules C (i,j) et extrayons l'ensemble des objets mobiles ayant enregistré leur position dans chacune des cellules durant l'intervalle de temps I. L'extraction des données peut s'effectuer de manière distribuée en affectant le traitement d'un groupe de cellules à chaque noeud de l'architecture. De même une requête k plus proches voisins "Quels sont les n objets mobiles les plus proches de cet objet mobile spécifique à l'instant t ?" peut être traité en considérant seulement les objets mobiles associés à la cellule dans lequel l'objet de référence est situé. Les requêtes de recherche de K plus proches voisins sont là encore des requêtes classiques ayant pour but d'identifier les k objets les plus proches d'un objet mobile en accord avec sa position actuelle [Yu et al., 2005]. Contrairement aux range queries abordées précédemment, les requêtes knn sont indépendantes de la notion d'intervalle. L'utilisateur initie une requête en spécifiant quelques caractéristiques à propos des objets d'intérêts pour que les k objets les plus proches à ces spécifications soient retrouvés. L'intérêt d'un modèle d'événement est qu'il offre un niveau de représentation plus abstrait permettant de caractériser plus sûrement le déplacement des objets tout en donnant la possibilité de comparer plus simplement des motifs récurrents (extraits de données anciennes) avec des flux reçus en temps-réel par exemple pour détecter des anomalies. Mais également que l'on peut observer plus facilement les interactions entre les différents objets mobiles (évitement, similarité etc...).

4.5 Conclusion

Ce chapitre a étudié l'ensemble des solutions existantes pour la conception d'un système dit hybride (i.e., combinant des informations extraites de données anciennes avec des flux temps-réel). A partir de l'identification de différents types de systèmes hybrides, il apparaît en l'état qu'aucun ne permet la gestion de données à caractère spatial ou spatio-temporel. Sur la base des approches existantes nous avons pu définir notre propre architecture hybride et distribuée pour le traitement d'objets mobiles. Parmi les spécificités développées, un point critique pour la gestion d'objets mobiles est le besoin d'avoir un système réactif capable d'assimiler et de traiter des données temps-réel tout en les couplant avec des informations issues de données plus anciennes. Nous avons dans un deuxième temps défini un modèle de données associé permettant de décrire de manière plus précise le mouvement d'objets mobiles permettant ainsi une meilleure compréhension des dynamiques dans ce contexte, tout en permettant également de comparer plus facilement données temps-réel et plus anciennes. Enfin, nous avons décrit l'ensemble des requêtes pris en charge par notre système et défini la notion de requêtes hybrides, parmi lesquelles une requête associée à la couverture et la perte de signal est énoncée et nécessite le recours à cette approche hybride. Dans la suite, nous développons les processus liés à la résolution spécifique de cette requête (Chapitre 5), avant de mettre en oeuvre de ce modèle hybride sur un cas d'application mettant en exergue la pertinence de notre approche.

Chapitre 5

Requêtes hybrides pour la gestion d'objets mobiles

Sommaire

5.1 Introduction

Dans la partie précédente nous avons défini un modèle générique pour la gestion et l'analyse de données d'objets mobiles, ce modèle s'articule autour de trois axes différents il est d'abord architectural (avec la fusion entre les flux de données temps-réel et données archivées), ensuite en termes de modèle (avec des abstractions supplémentaires qui sont la notion d'événements et de grille) et enfin la définition de requêtes hybrides et les premiers mécanismes de fonctionnement de ce système et les traitements associés. Il s'agit dans cette partie d'illustrer le fonctionnement de notre système hybride par la résolution d'une requête spécifique concernant des objets mobiles nécessitant à la fois d'analyser les données reçues en temps-réel par des capteurs au regard de données plus anciennes stockées.

Il s'agit ici de détecter ce que nous avons appelé Black Holes, c'est à dire des zones de "vide" pour lesquelles aucune position d'objets mobiles n'est reçue. Ceci est principalement la cause de capteurs n'émettant ou ne recevant pas de manière effective les messages. Nous nous plaçons dans un contexte où plusieurs stations centralisées récupèrent les informations de positions émises par des capteurs en déplacement comme cela peut être le cas avec des antennes téléphoniques. La

défaillance de réception peut être due notamment à des problèmes de propagation de signal, celle-ci dépendant des conditions météorologiques par exemple, c'est pourquoi certaines zones peuvent être non couvertes pendant un certain laps de temps. Tout l'enjeu ici est de déterminer pour chaque période de temps une cartographie des zones couvertes et non couvertes, alors que les données de positions entrent effectivement dans le système. Pour gérer en temps-réel le déplacement et le suivi d'objets mobiles, il est nécessaire d'avoir une connaissance des zones couvertes ou non, notamment pour estimer et suivre certains objets dont aucune position n'a été reçue pendant un certain laps de temps. Ceci est également formalisé et discuté, mettant en avant les mécanismes de l'approche que nous proposons pour l'identification de "disparition suspecte" d'objets mobiles (i.e. qui devraient émettre, étant en zone couverte).

5.2 Notion de Black Holes et définitions associées

Cette partie introduit le concept de *Black Holes*, ainsi qu'un ensemble de définitions qui permettent d'appréhender cette notion de manière plus formelle ainsi que les implémentations préliminaires pour la détection de ces zones. Le but de la méthode explorée ici est de caractériser certaines zones spatiales comme couvertes ou non couvertes avec une précision dépendant du volume de données reçu pour les zones considérées et ceci sans prise en compte des problématiques de performances. Le problème de couverture considéré démontre la nécessité d'une approche hybride, ce qui signifie l'utilisation conjointe de connaissances extraites de données stockées en base de données et des flux reçus en temps-réel pour fournir une réponse dans un contexte fluctuant et dynamique (i.e. la couverture actuelle dépend de la propagation du signal). Ce problème de couverture n'a pas encore été étudié à notre connaissance, autrement qu'avec des techniques d'analyse de densité sur les positions [Ristic et al., 2008] et l'identification de zones non couvertes n'a pas été abordé dans un contexte temps-réel.

Pour modéliser le concept de *Black Hole*, une grille uniforme est construite, séparant l'espace en cellules disjointes régulières $C_{i,j}$ de taille fixée $size_{cell}$ pour déterminer par exemple le nombre de positions par cellule et finalement identifier celles qui ne sont pas couvertes conformément à celle décrite dans la partie précédente. La taille des cellules est évaluée en considérant la durée entre deux enregistrements provenant d'un même objet mobile considérant sa position actuelle, son cap et sa vitesse. La taille nécessaire des cellules est choisie de manière à empêcher les objets mobiles d'émettre deux positions successives dans deux cellules non adjacentes (i.e. à l'extérieur de zones non couvertes). Pour ce faire, le pire cas est considéré, soit celui d'un objet mobile situé aux frontières d'une cellule et avançant selon un cap droit et avec une vitesse max notée V_{max} et $T_{t->t+1}$ l'intervalle de temps entre deux positions enregistrées. Dans ce cas, il est nécessaire d'envisager une

taille fixe plus grande pour éviter l'enregistrement de deux positions successives dans deux cases non adjacentes. Pour minimiser le problème d'"answer loss" décrit dans [Jensen et al., 2006] qui peut arriver dans le cas d'une grille régulière, il est nécessaire de choisir les cellules les plus petites possible. En effet, en prendre des cellules plus fines augmente les temps de calcul tandis que prendre des cellules plus larges réduit les coûts, mais l'identification de certains événements peut être entachée. C'est pourquoi G est définie avec une taille de cellule $size_{cell}$ telle que $size_{cell} = V_{max} * T_{t->t+1}$.

Nous présentons dans la suite, un ensemble de définitions permettant de définir une taxonomie des cellules de notre grille, et ce dans le but de discriminer les cellules couvertes de celles qui ne le sont pas.

Definition 5 : Ensemble de cellules vides. *Un ensemble de cellules vides sur un intervalle de temps τ est composé de toutes les cellules qui n'ont été traversées par aucune trajectoire. L'ensemble des cellules vides noté E(τ) est défini par E(τ)=$\{C_{i,j} \in G \mid Ncross_{i,j}(\tau) = 0\}$.*

Definition 6 : Ensemble des cellules traversées. *L'ensemble des cellules traversées pendant un intervalle de temps τ est composé des cellules qui ont été traversées par un ou plusieurs objets mobiles. Plus formellement, l'ensemble des cellules traversées noté $\bar{E}(\tau)$ est défini de la manière suivante $\bar{E}(\tau)$=$\{C_{i,j} \in G \mid Ncross_{i,j}(\tau) > 0\}$. Nous avons G=$E(\tau) + \bar{E}(\tau)$*

Definition 7 : Ensemble des cellules Black Hole. *L'ensemble des cellules Black Hole noté Set(BH)(τ) est composé des cellules qui ne sont pas couvertes pour l'intervalle de temps τ considéré. Il est composé de deux sous ensembles l'ensemble des cellules formellement identifiées Black Hole et l'ensemble de celles supposées Black Hole.*

Ces deux sous-ensembles sont définis formellement comme suit :

Définition 8 : Ensemble des cellules formellement identifiées Black Hole. *Pour un intervalle de temps donné τ, les cellules Black Hole peuvent être définies comme toutes les cellules $C_{i,j}/Nvessel_{i,j}(\tau) = 0$ et $Ncross_{i,j}(\tau) > 0$ (i.e., l'ensemble des cellules dans lesquelles aucune position n'a été enregistrée, mais qui ont été traversées un nombre significatif de fois c'est-à-dire plus qu'une valeur seuil θ durant l'intervalle de temps τ.)*

Définition 9 : Ensemble des cellules supposées Black Holes. *Étant donné un intervalle*

de temps τ, l'ensemble des cellules supposées Black Hole est donné par les cellules qui appartiennent à l'ensemble des cellules vides et qui ne sont pas couvertes pour l'intervalle de temps τ. (i.e toutes les cellules $C_{i,j} \in Set(BH)(\tau) \cap E(\tau)$)

Définition 10 : Ensemble des cellules couvertes. *L'ensemble des cellules couvertes est donné par les cellules dont la couverture est assurée sur l'intervalle de temps τ considéré. L'ensemble des cellules couvertes noté $\bar{BH}(\tau)$ est le complément de $BH(\tau)$ / G=$BH(\tau) + \bar{BH}(\tau)$. Réciproquement à $BH(\tau)$, $\bar{BH}$ (τ) est composé de deux sous-ensembles celui correspondant aux cellules formellement identifiées comme couvertes et celles supposées couvertes.*

Plus formellement nous pouvons définir ces deux sous-ensembles comme suit :

Définition 11 : Ensemble des cellules couvertes formellement. *Pour un intervalle de temps donné τ et une valeur seuil θ, l'ensemble des cellules couvertes est composé de toutes les cellules $C_{i,j}/Nvessel_{i,j}(\tau) > \theta$ (i.e., l'ensemble des cellules qui ont été traversées plus de θ fois durant l'intervalle de temps τ.)*

Définition 12 : Ensemble des cellules supposées couvertes. *Considérant un intervalle de temps τ, l'ensemble des cellules supposées couvertes correspond aux cellules qui appartiennent à l'ensemble des cellules vides parce qu'aucun objet mobile n'a traversé la zone pendant la période de temps τ considérée, mais dont il est raisonnable de penser qu'elles sont couvertes. (i.e toutes les cellules $C_{i,j} \in Set(\bar{BH})(\tau) \cap E(\tau)$)*

Considérant la période de temps actuelle, la table suivante couvre l'ensemble des cas possibles pouvant se produire en lien avec les définitions précédemment fournies :

	Cellules traversées	Cellules vides
Cellules couvertes	Cellules formellement couvertes	Cellules supposées couvertes
Cellules Black Hole	Cellules Black Holes formellement identifiées	Cellules supposées Black Hole

TABLE 5.1 : Taxonomie des cellules

Pour expliciter ce concept de *Black Hole*, une distinction est faite entre les cellules où aucune

position n'a été enregistrée, car aucun objet mobile ne pouvait depuis signaler sa position depuis ces zones et les cellules n'ayant tout simplement pas été traversées durant un intervalle de temps τ. Dans les deux cas, aucune position n'a été reçue depuis ces régions (i.e. $C_{i,j} \in E(\tau)$). En effet, ces cellules vides peuvent correspondre aussi bien à des cellules qui n'ont pas été traversées durant la période de temps considérée aussi bien qu'une zone non couverte pour ce même intervalle de temps. Cependant, notre objectif est d'extraire les *Black Hole*s comme les zones dans lesquelles aucune position ne peut être enregistrée, car elles ne sont pas couvertes durant la période temporelle considérée. Le défi dans le cas présent est de dériver et d'inférer des comportements ou des événements à partir d'un manque d'information (i.e. principalement pour les cellules vides). Ceci justifie le besoin de données historiques ou d'informations extraites de données plus anciennes pour compléter les informations fournies par les flux de positions reçues en temps-réel, même si ce n'est parfois pas suffisant pour donner une réponse satisfaisante et classer une cellule spécifique dans l'une des définitions précédemment énoncées.

L'identification des zones les moins denses est d'un intérêt important pendant la phase de traitement offline. Aussi, en considérant les données historiques, le volume de données est relativement élevé, avec certaines régions ayant pu être tour à tour couvertes et non couvertes. Dans certains cas, certaines régions souvent non couvertes peuvent tout de même avoir enregistré des positions à des moments où la propagation du signal était favorisée. À partir du traitement offline, nous extrayons un ensemble de cellules nommé candidat *Black Holes* noté Set_{off}(cdtBH) correspondant aux cellules les moins denses.

Comme défini dans [Jensen et al., 2006] nous maintenons une structure de données appelée histogramme de densité et notée DH(τ) qui compte pour chaque cellule $C_{i,j}$ le nombre de positions $Npos_{i,j}(\tau)$ enregistrées durant la période de temps $\tau \subset T$. Finalement, nous avons un histogramme de densité que l'on peut représenter de la manière suivante DH(τ)=[$C_{i,j}$,$Npos_{i,j}(\tau)$] trié par ordre ascendant de $Npos_{i,j}(\tau)$.

Étant donné le nombre de positions $Npos_{i,j}(\tau)$ enregistrées dans une cellule, les notations suivantes sont définies Set($C(N < k,\tau)$) (respectivement Set($C(N < k,\tau)$)) représente l'ensemble des cellules qui ont enregistré moins (respectivement plus) de k positions pendant un intervalle de temps donné $\tau \subset T$, plus formellement :

— Set($C(N < k,\tau)$) = $\{C_{i,j} \in G \ / \ Npos_{i,j}(\tau) < k\}$.
— Set($C(N > k,\tau)$) = $\{C_{i,j} \in G \ / \ Npos_{i,j}(\tau) > k\}$.
— Set($C(N = k,\tau)$)= $\{C_{i,j} \in G \ / \ Npos_{i,j}(\tau) = k\}$.

L'ensemble des cellules candidates *Black Holes* ou Set_{off}(cdtBH), est obtenu pendant le

traitement offline. Étant donné un pourcentage p et une période de temps τ, Set_{off}(p,τ)= Set(C (N<q, τ)) où q est le plus grand nombre de positions pour les p pourcents cellules les moins peuplées.

Ces régions sont les plus susceptibles d'être des *Black Holes* et doivent être examinées tandis que les positions reçues en temps-réel sont traitées à la volée par la partie online.

Pour chaque cellule $C_{i,j}$,, nous définissons une métrique associée appelée valeur de densité et notée $d(i,j)(\tau)$. La fonction de densité d est définie de la manière suivante : $Npos_{i,j}(\tau) \in \mathbb{N} \rightarrow v \in [0..1]$ où v est le q quantile auquel $C_{i,j}$ appartient considérant la liste ordonnée des $Npos_{i,j}(\tau)$. Ainsi, si $C_{i,j} \in Set_{off}$(p,τ) alors v = 0.

Réciproquement, nous introduisons le concept de *Black Hole* intervenant pour la partie temps-réel :

L'ensemble des cellules candidates *Black Hole* ou Set_{on}(cdtBH), est obtenu pendant le traitement online. Set_{on}(cdtBH)= $\{C_{i,j} \in G \mid Nvessel_{i,j}(\tau)=0 \}$ où τ correspond à l'intervalle le plus récent ou l'intervalle courant.

De manière analogue à la partie offline, une structure de données est maintenue concernant les flux de données. Cette structure de données appelée histogramme de couverture et notée de la manière suivante $Cover(G,\tau)$, contient le nombre d'objets mobiles distincts $Nobjetmobile_{i,j}(\tau)$ ayant enregistré une position dans $C_{i,j}$ pendant la période de temps τ et $Ncross_{i,j}(\tau)$ le nombre d'objets mobiles ayant traversé la zone ceci associé à chacune des cellules $C_{i,j}$ de la grille G pendant l'intervalle de temps τ considéré. Plus formellement nous avons : Cover($C_{i,j}\tau$)=($Nobjetmobile_{i,j}(\tau)$,$Ncross_{i,j}(\tau)$).

La table 5.2 donne un récapitulatif des différentes notations définies.

5.3 Processus de traitement pour l'identification de Black Holes

Nous souhaitons donc identifier ces *Black Holes* à l'aide de l'approche hybride décrite dans la section précédente (cf Section 4.2.3). Concernant la partie offline les cellules les moins denses sont identifiées en utilisant un processus similaire à celui décrit dans [Hadjieleftheriou et al., 2003] où les auteurs cherchent à identifier eux, les zones les plus denses en objets mobiles. Nous maintenons un histogramme de densité noté DH(τ_{Past}) qui compte le nombre de positions enregistrées pour chaque cellule pour l'ensemble des données historiques c'est-à-dire concernant la période de temps τ_{Past},

TABLE 5.2 : Tableau Notations

Notations	Descriptif
τ_{Past}	Période de temps concernant la partie offline
$\tau_{current}$	Période de temps concernant la partie online (i.e la fenêtre glissante)
$C_{i,j}$	Cellule de la grille G de coordonnées i,j
$Ncross_{i,j}(\tau)$	Nombre d'objets mobiles dont la trajectoire a intersecté $C_{i,j}$ pendant τ
$Npos_{i,j}(\tau)$	Nombre de positions ayant été enregistrées dans $C_{i,j}$ pendant τ
$Nobjetmobile_{i,j}(\tau)$	Nombre d'objets mobiles distincts ayant enregistré une position dans $C_{i,j}$ pendant τ
Set_{on}(cdtBH)	L'ensemble des cellules candidates *Black Hole* (partie online)
DH(τ_{Past})	Histogramme de densité des cellules (partie offline)

ceci permettant de filtrer et extraire plus facilement l'ensemble des candidats *Black Holes* pour la partie offline (ou Set_{off}(cdtBH)). Pendant le traitement temps-réel, des synopses temps-réels sont construits pour chaque période de temps $\tau_{Current}$ et combinés avec les résultats extraits de la partie offline de façon à donner plus de poids aux données récentes tout en conservant les données plus anciennes comme garde-fou, ceci dans le but de finalement extraire l'ensemble courant des cellules *Black Holes* (cf. figure 5.1).

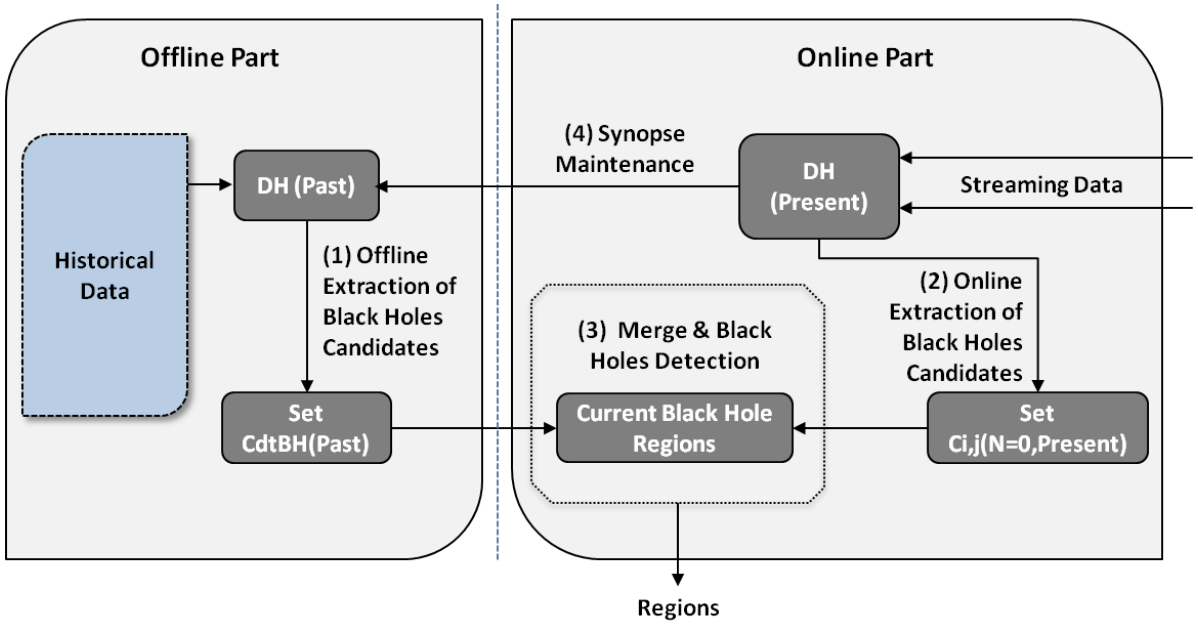

FIGURE 5.1 : Processus de traitement pour l'identification de Black Holes

Nous décrivons ensuite les différentes étapes intervenant dans le processus d'identification de

Black Holes :

- *Premère Etape : Extraction de candidats Black Hole pour la partie offline* Le traitement pour identifier les cellules les moins denses est appliqué aux données historiques. Une méthode similaire à celle décrite dans [Hadjieleftheriou et al., 2003] est utilisée, mais non pas pour identifier les zones les plus denses comme les auteurs, mais au contraire pour déterminer celles qui sont les moins denses. Cependant, des problématiques de perte d'information ou d'ambiguïté peuvent apparaître en considérant des cellules de taille fixe pour définir et observer les densités [Ni and Ravishankar, 2007]. Ces enjeux ne sont pas discutés dans le cadre de ce cas d'étude où nous nous intéressons principalement à montrer la pertinence d'une approche hybride de traitement de données dans le cadre d'objets mobiles.
- *Seconde Etape : Extraction de Black Holes pour la partie temps-réel* Pendant cette étape, l'objectif est de catégoriser les cellules de la grille et d'identifier l'ensemble des cellules formellement identifiées *Black Holes*, l'ensemble des cellules formellement identifiées comme étant couvertes et enfin celles qui sont vides (cf. Table 5.1). Ceci donne un aperçu des candidats *Black Hole* concernant la partie temps-réel qu'il sera ensuite nécessaire de croiser avec les données extraites de la première étape. À cette étape, un des enjeux concerne les aspects de granularité découlant de l'usage d'une fenêtre glissante (slide et range considérés) ainsi que la structure de données relative à la densité des cellules (ou leur couverture) [Loglisci and Malerba, 2014].
- *Troisième Etape : Fusion et identification des Black Holes* La fusion et l'identification de l'ensemble des cellules *Black Hole* pour la période de temps courante nécessite une technique spécifique pour croiser les informations dérivées de la partie historique et les données reçues en temps-réel [Chandrasekaran and Franklin, 2004]. Dans notre contexte de *Black Hole*, l'un des principaux enjeux est d'inférer une information concernant les cellules vides en combinant les données reçues en temps-réel avec les informations de densité sur les données historiques. (Pour rappel, une cellule vide sur la période de temps courante peut être vide soit par absence de couverture, soit, car aucun trafic maritime n'a cours dans la zone).
- *Quatrième Etape : Maintenance des synopses* La mise à jour continue des cellules *Black Holes* candidats dérivées de la partie offline et la maintenance des synopses relatives aux différents intervalles de temps et qui ont besoin d'être mis à jour continuellement alors que les flux anciens de données sont transférés vers la partie offline [Salmon et al., 2015].

Pour détecter ces *Black Hole*, une grille uniforme G est utilisée. Celle-ci est divisée en cellules $C_{i,j}$ de taille fixe et pour chacune d'elle, les différentes mesures introduites dans la section 4.4 sont évaluées ; c'est à dire ici, le nombre de positions qui ont été enregistrées à l'intérieur d'une cellule pour la partie offline, le nombre d'objets mobiles ayant enregistré une position à l'intérieur de la cellule et

enfin le nombre d'objets mobiles qui ont traversé la cellule (en considérant l'intersection entre la polyligne et la cellule concernée) pour la partie online dans le but de déterminer l'ensemble courant de cellules *Black Holes*. Nous avons choisi une taille de cellule fixe malgré les limitations d'une telle méthode [Ni and Ravishankar, 2007], car d'un point de vue pratique il est plus facile et approprié de comparer des structures similaires pour les parties offline et online (i.e. comparer l'histogramme de densité qui conserve les données relatives aux cellules de manière agrégée avec les données reçues en temps-réel, plutôt qu'avec un index). En effet, l'utilisation d'index nécessiterait l'usage de deux index différents pour les parties offline et online et vraisemblablement avec un index très fourni pour la partie offline et déséquilibré pour la partie temps-réel qui n'auraient pas été comparables facilement. Enfin, pour certaines cellules, une différenciation exacte entre cellules couvertes et non couvertes reste difficile, et ceci pour plusieurs raisons. Premièrement, car au-delà des problèmes de propagation, il y a une incertitude concernant la réception des données du fait que certains capteurs puissent être devenus hors service et défaillant . Deuxièmement, parce que nous avons à différencier cellules couvertes et non couvertes concernant des cellules vides (i.e. nous avons à inférer une connaissance à partir d'un manque d'information). Sans l'ajout de données supplémentaires comme des données météorologiques, il est difficile de déterminer avec exactitude les changements de couverture pour certaines régions. Dans certains cas, certaines cellules ne peuvent être identifiées comme couvertes ou non, ceci illustrant une incertitude sur la délimitation de ces zones que nous cherchons à identifier.

5.3.1 Traitement offline pour la détection de Black Holes

Le concept de *Black Holes* nécessite pour son implémentation l'extraction de connaissance à partir de données historiques ainsi que son croisement avec les données reçues en temps-réel. L'extraction à partir des données historiques a déjà été traitée dans des travaux précédents pour l'identification de régions denses pour des objets mobiles [Jensen et al., 2006], mais notre problème est celui inverse d'identification des zones les moins denses. D'abord, un résumé des cellules qui sont candidates à être *Black Holes* est construit pour la partie offline (ou Set_{off}(cdtBH)) et un histogramme de densité DHτ_{Past} est maintenu pour être fusionné avec les données de la partie temps-réel (cf. section merge). En fait, ces *Black Holes* sont des régions desquelles aucun signal émis par les capteurs ne peut être reçu et correspondent aux régions potentiellement les moins denses lorsqu'on considère l'ensemble des données historiques. Autrement dit, une partie des cellules *Black Holes* auront peu de reports de positions enregistrés (car très souvent non couvertes).

Nous considérons à nouveau une grille uniforme G séparant l'espace en cellules régulières $C_{i,j}$ de taille fixe. Pour chaque cellule le nombre de positions $Npos_{i,j}(\tau_{Past})$ pour la partie offline est mesuré. Ainsi, une cellule $C_{i,j}$ appartient à l'ensemble des cellules candidates à être *Black Hole* si

et seulement si le nombre de positions enregistrées dans $C_{i,j} < \theta$ où θ est une valeur seuil. Dans le but de déterminer cette valeur seuil, $Npos_{i,j}(\tau_{Past})$ est calculé pour chaque cellule $C_{i,j} \in$ G et θ correspond à la valeur haute du plus percentile q sur tous les $N_{i,j})(\tau_{Past})$ avec q un pourcentage donné par l'utilisateur (cf. algorithm 1). Une valeur de densité $d_{i,j}(\tau_{Past})$ est définie pour chaque cellule, cette valeur est utilisée pendant la phase de fusion pour identifier les *Black Holes* pour la période courante.

Algorithm 1 : Dérivation de l'ensemble des candidats *Black Holes* et calcul de l'histogramme de densité

Data : [p] Positions, Grid G [$C_{i,j}$], Int θ, Float q
Result : $Setoff$(cdtBH)
initialization ;
foreach *grid cell* $C_{i,j}$ **do**
  $Npos_{i,j}(\tau_{Past}) \Leftarrow \sum_{k=1}^{nbpositions}(v_k)$
  where v_k=1 if $p_k \in C_{i,j}$ otherwise 0
  DH((τ_{Past})) $\Leftarrow$ DH((τ_{Past})) : : ($C_{i,j}$, $Npos_{i,j}(\tau_{Past})$)
end
$\theta \Leftarrow Npos_{i,j}$ such that $Npos_{i,j} = DH\tau_{Past}[q * Length(G)]$
`// θ is the upper Npos value of the first quantile q ;`
foreach *grid cell* $C_{i,j}$ **do**
  compute $d_{i,j}(\tau_{Past})$
end
foreach *grid cell* $C_{i,j}$ **do**
  if $Npos_{i,j}(\tau_{Past}) < \theta$ **then**
    Set_{off} (cdtBH) $\Leftarrow Set_{off}(cdtBH) \cup C_{i,j}$
end

Considérons dans l'ensemble des cellules candidates à être *Black Hole*, soit l'ensemble Set_{off}(cdtBH) et dans un second temps la valeur de densité associée à chaque cellule $C_{i,j}$ de la grille. Cet ensemble candidat et les valeurs de densité pourront être utilisées alors que les flux de données de positions arrivent dans le système sur la période de temps $\tau_{current}$.

5.3.2 Traitement temps-réel pour l'identification de Black Holes

Dans cette partie, nous étudions la façon dont est généré l'histogramme de couverture introduit en section 5.2 et noté Cover(G,$\tau_{current}$) qui est continuellement mis à jour par l'arrivée de données entrantes.

Pour chaque cellule de la grille, les valeurs $Nobjetmobile_{i,j}(\tau_{current})$ et $Ncross_{i,j}(\tau_{current})$ associées sont calculées. Ainsi nous obtenons Cover($C_{i,j}$,$\tau_{current}$)=($Nobjetmobile_{i,j}$,$Ncross_{i,j}$). Nous considérons une fenêtre glissante avec un range ω et un slide β. Nous déterminons un histogramme de couverture pour chaque période de temps (ω/β), puis l'ensemble des cellules candidates à être *Black Holes* sont extraites à chaque période de temps β considérant une période globale ω en agrégeant les différents histogrammes de couverture relatifs à chaque période de temps correspondant à un "pane" (ω/β) en prenant en compte le fait qu'un objet mobile n'est compté qu'une et une seule fois. Pour déterminer l'ensemble des cellules candidates *Black Holes* ou Set_{on}(cdtBH) l'ensemble set($C(Nobjetmobile_{i,j}$ $(\tau_{current})< \theta$) est calculé. Les cellules ayant enregistré peu de passage d'objets mobiles différents (i.e. moins que la valeur seuil θ) sont considérées comme potentiellement non couvertes.

L'algorithme 2 génère l'ensemble des candidats *Black Holes* concernant la partie temps-réel. Pour obtenir la représentation des *Black Holes* durant la dernière période de temps ω, il est nécessaire de considérer les informations additionnelles du traitement temps-réel et aussi des données historiques pour distinguer cellules couvertes des cellules *Black Hole* parmi l'ensemble des cellules vides.

5.3.3 Identification de Black Holes à l'aide d'un traitement combiné offline online

À chaque période de temps β les cellules candidates *Black Hole* sont extraites. En fait, la taille ω de la fenêtre peut générer dans certains cas de candidats *Black Hole* (i.e. cellules qui). À cause de la taille des cellules, si elles sont trop petites notamment, il est possible qu'aucune position ne soit enregistrée sur ces cellules sur une courte période de temps donnée. Dans le but de résoudre ce problème, nous considérons d'abord le nombre de trajectoires dont la polyligne a une intersection avec une cellule donnée $C_{i,j}$. Ainsi nous comparons le nombre d'objets mobiles qui ont enregistré leur position dans une cellule $Nobjetmobile_{i,j}$ avec le nombre d'objets mobiles $Ncross_{i,j}$ qui sont passés dans la cellule (i.e. dont la polyligne passe dans la cellule). Nous listons ensuite les différents cas pouvant intervenir et les règles associées appliquées dans notre algorithme (cf. Algorithm 3, Figure 5.2) :

— *Cellules couvertes (i.e. $C_{i,j} \in \bar{BH}(\tau_{current})$).* Quand le nombre de bateaux enregistrés dans la cellule est proche du nombre de trajectoires qui ont croisé la cellule avec $Ncross_{i,j} > 0$

Algorithm 2 : Dérivation de l'ensemble des candidats *Black Holes* pour la partie online

Data : [v] Vessels, Cover($\tau_{current}$), Int θ, Float q
Result : Set_{on}(cdtBH)

initialization ;
foreach *each pane* **do**
  foreach *new vessel position* **do**
    if $v \in C_{i,j}$ *for the first time* **then**
      $Nvessel_{i,j}(\tau_\beta) \Leftarrow Nvessel_{i,j}(\tau_\beta) + 1$ $Ncross_{i,j}\ (\tau_\beta) \Leftarrow Ncross_{i,j}(\tau_\beta) + 1$
  end
end
Cover($\tau_{current}$) $\Leftarrow \sum_{i=0}^{\omega/\beta} Cover[t_{current-i*\beta}, t_{current-(i+1)*\beta}]$
`// we sum every cover histogram considering vessel and cross numbers ;`
Sort Cover(G,$\tau_{current}$) by $Nvessel_{i,j}$ $\tau_{current}$
`// we sort the Cover Histogram by vessel number ;`
$\theta \Leftarrow Nvessel_{i,j}/Nvessel_{i,j} = Cover(\tau_{current})[q * Length(G)]$ `// θ is the upper Nvessel value of the first quantile q ;`

foreach β *time period* **do**
  foreach *grid cell* $C_{i,j}$ **do**
    if $Nvessel_{i,j} < \theta$ **then**
      Set_{on}(cdtBH) $\Leftarrow Set_{on}$(cdtBH) $\cup C_{i,j}$
  end
end

alors nous considérons que cette cellule est couverte (i.e. quand $Nvessel_{i,j}(\tau_{current})$ / $Ncross_{i,j}\ (\tau_{current}) \simeq 1$ and $Ncross_{i,j} > 0$). La différence entre les deux nombres est la conséquence de certains capteurs étant devenus hors service. Pour la partie suivante et les expérimentations, nous considérons qu'une cellule $C_{i,j}$ appartient à cette catégorie si $Nvessel_{i,j}(\tau_{current})$ / $Ncross_{i,j}\ (\tau_{current}) >= \rho$.

— *Cellules Black Holes (i.e. $C_{i,j} \in BH(\tau_{current})$).* Quand le nombre d'objets mobiles ayant été enregistrés dans la cellule est bien inférieur à ceux qui l'ont traversée avec $Nvessel_{i,j} > 0$ nous considérons alors que la cellule est non couverte (i.e. quand $Nvessel_{i,j}\ (\tau_{current})$ / $Ncross_{i,j}\ (\tau_{current}) \simeq 0$ with $Ncross_{i,j}\ (\tau_{current}) > 0$). Dans ce cas, la différence entre les deux nombres peut s'expliquer par la non-couverture des cellules. En effet, quand les objets mobiles sont dans une cellule, les positions émises par ces objets mobiles ne sont pas reçues

par les antennes ou récepteurs les plus proches. Pour la suite et les expérimentations, une cellule appartient à cette catégorie si $Nvessel_{i,j}$ ($\tau_{current}$) / $Ncross_{i,j}$ ($\tau_{current}$) $< \rho$.

— *Cellules vides (i.e. $C_{i,j} \in E(\tau_{current})$).* Quand le nombre d'objets mobiles ayant enregistré leur position dans la cellule et le nombre de trajectoires ayant traversé la cellule sont égaux à 0 l'analyse précédente faite sur les données anciennes est nécessaire pour déterminer si la cellule est couverte ou non (i.e. quand $Nvessel_{i,j}$ ($\tau_{current}$) = $Ncross_{i,j}$ ($\tau_{current}$) = 0).

— Le dernier cas, $Nvessel_{i,j} > Ncross_{i,j}$ ne peut par définition pas être réalisable, car si un objet mobile enregistre sa position dans une cellule $C_{i,j}$ alors $Ncross_{i,j}$ est incrémenté, donc $\forall C_{i,j}, \tau timeperiod Ncross_{i,j} >= Nvessel_{i,j}$

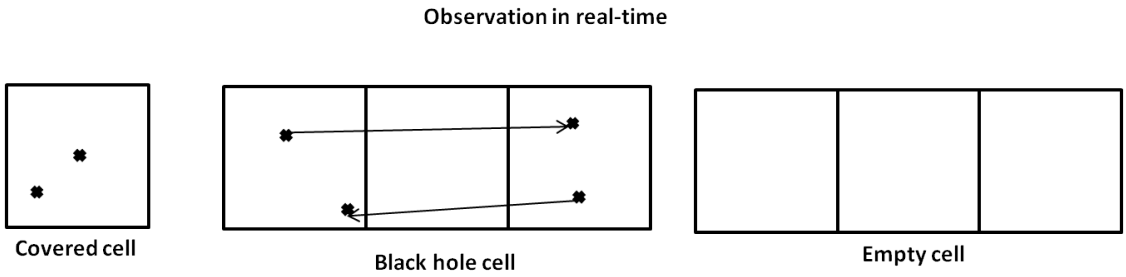

FIGURE 5.2 : Résultat de la partie online [Salmon et al., 2016]

Au regard de cellules spécifiques qui n'ont ni été traversées ni n'ont enregistré de positions en leur sein considérant uniquement la partie temps-réel (i.e. les cellules vides), elles nécessitent toujours d'être identifiées comme couvertes ou non (cf. Figure 5.3).

Dans le but de traiter ce cas, nous considérons l'apport de la partie offline comme suit. Nous considérons la valeur de densité $d_{i,j}(\tau_{Past})$ associée aux cellules $C_{i,j}$ pour identifier les cellules *Black Holes* parmi l'ensemble des cellules vides noté E($\tau_{current}$). Comme mentionné précédemment et à cause des aspects de granularité, nous pouvons avoir des cellules pour lesquelles aucun bateau n'a traversé la zone alors que ce sont des régions souvent couvertes (i.e. ayant une valeur de densité assez élevée, supérieure à une valeur seuil D_{sup}). De la même manière nous pouvons avoir des cellules qui ne sont généralement jamais couvertes (i.e. ayant une valeur de densité très faible, inférieure à une valeur seuil D_{min}) et qui n'ont enregistré aucune position sur la dernière période de temps considérée, celles-ci peuvent être considérées comme supposées non couvertes. Étant donné une valeur seuil D :

— Si $C_{i,j} \notin Setoff(cdtBH)$ and $d_{i,j}(\tau) > D_{sup}$, nous pouvons raisonnablement en déduire que $C_{i,j} \in E(\tau_{current}) \cap \bar{BH}(\tau_{current})$ (i.e. $C_{i,j}$ (i.e. $C_{i,j}$ est dans l'ensemble des cellules supposées couvertes).

— Si $C_{i,j} \notin Setoff(cdtBH)$ and $d_{i,j}(\tau) < D_{min}$, nous pouvons raisonnablement en déduire que $C_{i,j} \in E(\tau_{current}) \cap BH(\tau_{current})$ (i.e. $C_{i,j}$ (i.e. $C_{i,j}$ est dans l'ensemble des cellules

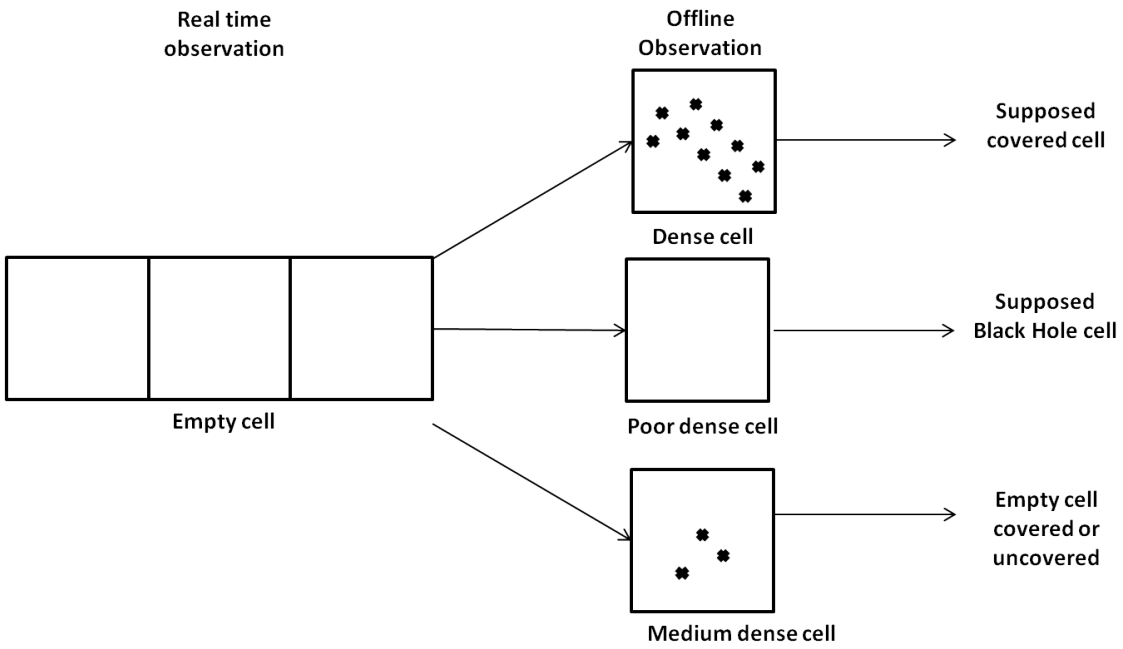

FIGURE 5.3 : Couplage des observations online et offline [Salmon et al., 2016]

supposées non couvertes).

— Autrement, la cellule $C_{i,j}$ est vide et ne peut être déterminée comme couverte ou non couverte et nécessite d'autres moyens d'investigations pour déterminer sa couverture.

Si $d_{i,j}(\tau_{Past})$ est comprise entre les valeurs D_{sup} et D_{min}, nous considérons que nous ne pouvons inférer aucune information concernant la cellule spécifique examinée. Des travaux futurs pourront permettre par exemple de tenir compte de la position des antennes ou de la nature des cellules adjacentes pour déterminer si ces cellules tendent à être couvertes ou non.

Tandis que les nouveaux flux de données entrent dans le système, les résumés de données aussi bien online que offline nécessitent d'être maintenus et mis à jour. La maintenance de synopses pour la partie temps-réel ont été étudiées dans la partie précédente où un synopsis est conservé pour chaque pane et fusionné à chaque intervalle de temps β avec l'entièreté de la fenêtre glissante de période ω pour calculer l'ensemble des cellules *Black Holes* correspondant à la période de temps actuelle. Une problématique survient lorsque la taille de la fenêtre est atteinte pour la partie temps-réel, ainsi les flux de données les plus anciens doivent être transférés vers la partie offline.

Algorithm 3 : Extraction et fusion des *Black Holes* par couplage avec la composante offline

Data : [v] Vessels, Cover($t_{current}-\beta$, $t_{current}$), Set_{on}(cdtBH), Float quantile

Result : Set_{on}(cdtBH)

initialization ;
foreach *β time period* **do**
  foreach *grid cell* $C_{i,j} \in Set_{on}(cdtBH)$ **do**
    if $Nvessel_{i,j}(\tau_{current}) = Ncross_{i,j}(\tau_{current}) = 0$ **then**
      $E(\tau_{current}) \Leftarrow E(\tau_{current}) \cup C_{i,j}$
    else
      if $Nvessel_{i,j}(\tau_{current}) / Ncross_{i,j}(\tau_{current}) < \rho$ **then**
        $BH(\tau_{current}) \Leftarrow BH(\tau_{current}) \cup C_{i,j}$
      else
        $Nvessel_{i,j}(\tau_{current}) / Ncross_{i,j}(\tau_{current}) >= \rho$
      $C_{i,j} \in \bar{BH}(\tau_{current})$
  end
  foreach *grid cell* $C_{i,j} \in E(\tau_{current})$ **do**
    if $d_{i,j}(\tau_{Past}) > D_{sup}$ **then**
      $C_{i,j} \in$ Supposed covered cells
    if $d_{i,j}(\tau_{Past}) < D_{min}$ **then**
      $C_{i,j} \in$ Supposed uncovered cells
    else
      $C_{i,j} \in$ is undefined (supposed covered or uncovered cell)
  end
end

5.4 Identification d'extinction volontaire ou de capteurs défaillants

Dans cette partie, nous considérons un ensemble de cellules identifiées comme *Black Holes*, celles-ci vont nous permettre de faire la distinction entre les objets mobiles n'émettant plus de signal à cause d'un passage en zones non couvertes de ceux dont le signal a été coupé volontairement ou dont le capteur est défaillant. Cette partie est décorrélée de la précédente (i.e. dans la partie expérimentation nous utiliserons des *Black Holes* artificiels, car utiliser ceux obtenus en temps-réel biaiserait les résultats). Pour ce faire nous avons besoin de faire de la prédiction à court et moyen terme, ceci nécessitant encore l'utilisation de données temps-réel et archivées.

Nous définissons un ensemble de notations pour la suite qui seront relatives à l'identification

d'objets mobiles dont le signal a été coupé :

- Soit LastPos la fenêtre contenant la dernière position (id,x,y,t,v,c) reçue pour chaque objet mobile où id est l'identifiant de l'objet mobile, x et y correspondent à sa géolocalisation, t l'instant auquel la position a été émise et respectivement v la vitesse et c le cap. Cette structure est d'intérêt pour savoir quel est le dernier instant auquel chacun des objets mobiles a émis une position et ainsi détecter ceux qui n'ont pas émis depuis très longtemps leur position.
- Soit NextPos la fenêtre contenant la prochaine position estimée en fonction de la dernière position reçue pour chacun des objets mobiles et de la forme (id,x,y) où id est l'identifiant de l'objet mobile tandis que x et y correspondent à sa géolocalisation.
- Soit BH($\tau_{current}$) l'ensemble des cellules identifiées comme *Black Holes* pour l'intervalle de temps courant. Pour rappel, il s'agit de cellules desquelles aucun signal émis ne peut être reçu pour la période de temps $\tau_{current}$ et donc pour lesquelles aucune position n'est enregistrée.
- Soit MissMO correspond à l'ensemble des objets mobiles n'ayant pas émis de signal depuis longtemps (i.e. supérieur à une valeur seuil δ_t). Cette structure est obtenue par observation périodique des dernières positions enregistrées dans la fenêtre LastPos.

La figure 5.4 montre le schéma de principe de notre détection de disparition d'objets et d'estimation de positions sous incertitude.

Les flux de données de positions sont reçus, dans le but de détecter des disparitions d'objets mobiles (MO) différents algorithmes de veille sont mis en place et les différentes tables définies précédemment sont utilisées.

Pour chaque nouvelle position reçue, on vérifie si l'identifiant de l'objet mobile est contenu dans la table MissMo des objets mobiles manquants, s'il n'apparaît pas une comparaison est effectuée entre l'estimation de la position contenue dans la table NextPos et celle effectivement obtenue pour détecter toute anomalie. Si l'écart entre position estimée et position reçue est inférieure à une valeur seuil θ alors la position reçue à l'instant t remplace la position reçue à l'instant t-1 dans la table LastPos des dernières positions reçues. Puis à partir de cette position, du cap et de la vitesse nous estimons la prochaine position de l'objet mobile que nous stockons dans dans la table NextPos qui devrait correspondre à la position de l'objet mobile à l'instant t+1. Si l'identifiant de l'objet mobile enregistrant sa position est contenu dans la table des objets mobiles manquants, alors on retire cet identifiant de la table MissMO en comparant la position enregistrée et la dernière position de cet objet mobile dans la table LastPos pour essayer de reconstituer le trajet effectué par l'objet mobile durant sa "disparition" (si l'écart entre la position enregistrée et celle estimée dans NextPos est inférieur à une valeur seuil θ_d). Ces données (dernière position avant disparition et position

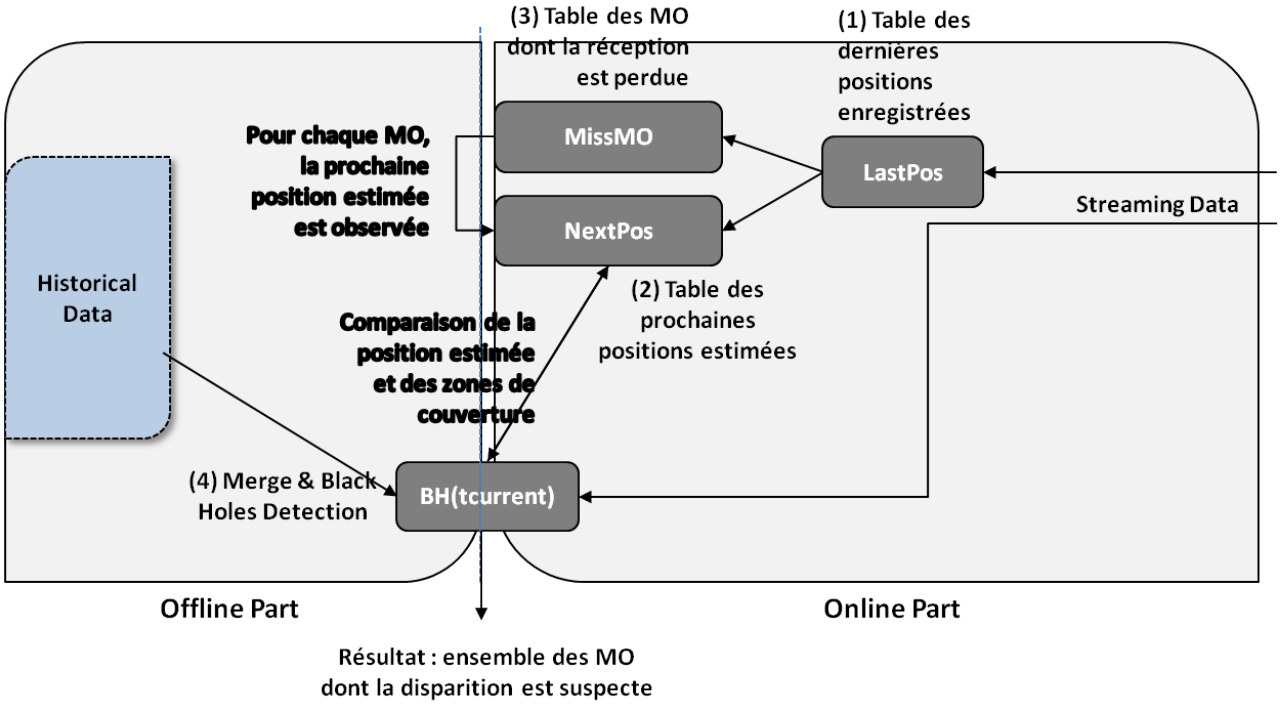

FIGURE 5.4 : Détection de disparition d'objets mobiles

de réapparition) sont d'intérêt pour identifier des éléments ayant éteint leur système et détecter d'éventuelles activités illégales.

A chaque pas de temps τ, la table LastPos est examinée, pour chacun des objets mobiles si l'écart temporel entre l'instant courant et l'instant de dernière position enregistrée est supérieur à une valeur seuil δ_t alors, l'identifiant de l'objet mobile est ajouté à la table des identifiants des objets mobiles dont nous avons perdu le signal MissMO. Dans le même temps, la position estimée de l'objet mobile correspondant contenue dans NextPos est comparée à la table des *Black Holes*, si la position estimée est contenue dans une des zones non couvertes alors l'objet mobile est simplement passé dans une zone non couverte, auquel cas la disparition est "normale". A l'inverse, si l'objet mobile n'a pas émis depuis longtemps, alors même que sa position estimée l'est dans une zone qui devrait être couverte, alors une alarme est soulevée sur l'identifiant de l'objet mobile (celui-ci pouvant soit avoir un système, soit avoir été éteint par son porteur). Si l'objet mobile est censé être dans une zone de non couverture, alors la table NextPos est mise à jour à l'aide de patrons spatio-temporels décrivant les comportements types des objets mobiles passant dans la même configuration à l'endroit donné et est extrait de la partie offline.

5.5 Conclusion

Ce chapitre a traité la résolution d'une nouvelle requête hybride celle-ci permettant de déterminer les zones de couverture et non-couverture de manière périodique alors que les données sont reçues en temps-réel. Cette requête est l'illustration d'un besoin de coupler des informations extraites de données anciennes avec des flux temps-réel pour mieux appréhender et comprendre la situation notamment pour des objets mobiles. Cette requête est générique et peut être appliquée à différents domaines et types d'objets mobiles (animaux, personnes, véhicules ...). Ce cas particulier nécessite le développement d'une approche hybride qui prend avantage à la fois du flot continu de données, mais aussi les incertitudes au niveau des informations reçues. Les principes d'une approche hybride pour la gestion d'objets mobiles ont été décrits d'abord dans [Salmon et al., 2015], ceci nous permettant d'identifier les régions candidats *Black Holes* en fusionnant les informations extraites à la fois de la partie données historiques et la partie temps-réel. C'est en suivant ces principes que nous avons développé le processus de traitement qui (1) permet une analyse de densité et une extraction de connaissances de la partie données historiques, (2) détermine les régions potentiellement *Black Holes* à partir des flux reçus en temps-réel, dans le but de trouver des zones probablement non couvertes (4). Cependant, des mécanismes additionnels sont nécessaires pour la maintenance des synopses concernant les données historiques alors que les données sont reçues en temps-réel et transférées vers la partie offline. Le concept de *Black Hole* défini ici n'avait jusqu'à lors jamais était abordé dans la littérature à notre connaissance. Dans [Hong et al., 2015] les auteurs utilisent le terme de *Black Hole* dans un contexte de réseau contraint orienté pour chercher les noeuds qui ont plus de flux sortants que de flux entrants, mais ce concept est différent de celui exploré dans ce travail de thèse. Un enjeu en relation avec notre détection de *Black Hole* concerne les requêtes de densité qui ont été abordées aussi bien avec une approche temps-réel [Hao et al., 2008] où les auteurs utilisent un quadtree pour enregistrer les nouvelles positions, que d'un point de vue offline [Hadjieleftheriou et al., 2003] où les auteurs cherchent des zones denses associées à des objets mobiles. Avec notre approche, un sous ensemble de *Black Holes* est identifié, mais certains enjeux et leviers d'amélioration restent à explorer concernant la détection de comportements spécifiques que l'on peut deviner à partir de la connaissance de ces *Black Holes* et des techniques d'indexation pourraient être utilisées pour la mise à jour des *Black Holes*. Enfin, la disparition d'objets mobiles a été étudiée et modélisée à l'aide de notre approche hybride, mettant en avant la dynamique et la réactivité du système pour la détection de phénomènes (avec l'utilisation de tables en mémoire et de déclencheurs). Cependant, les mécanismes de mise à jour de l'estimation de prochaine position estimée nécessitent d'être étudiés et étendus dans le cas d'un passage en zone non couverte. Aussi, l'apparition d'un objet mobile dans une zone supposée non couverte pourrait également être étudiée (celui-ci pouvant être par exemple falsifié). Dans la suite nous mettrons en place et appliquerons l'algorithme de détection de

Black Holes proposé sur des données maritimes et en particulier sur des positions de navires. C'est pourquoi, dans la partie suivante, nous présentons plus spécifiquement le domaine maritime ainsi que les besoins et déclinaisons du modèle défendu précédemment au vu de contexte maritime.

Chapitre 6

Modèle pour l'analyse des mobilités maritime

Sommaire

6.1 Introduction

Dans les parties précédentes ont été définies à la fois un modèle générique pour la gestion et l'analyse de données d'objets mobiles, ce modèle s'articule autour de trois axes différents il est d'abord architectural (avec la fusion entre les flux de données temps-réel et données archivées), ensuite en termes de modèle (avec des abstractions supplémentaires qui sont la notion d'événements et de grille) et enfin la définition de requêtes hybrides et les premiers mécanismes de fonctionnement de ce système et les traitements associés illustrés dans le chapitre précédent. Dans cette partie, nous présentons la déclinaison de ce modèle appliqué au domaine maritime sur lequel nous avons travaillé plus particulièrement.

6.2 Contexte Maritime et objets mobiles

Les problématiques de "Big Data" se sont développées également pour le domaine maritime notamment avec l'émergence et la prolifération de capteurs transmettant des données de positions engendrant la production de volumes de données d'ampleur de plus en plus grand et qu'il est nécessaire d'analyser en temps-réel. Le domaine maritime est l'illustration parfaite de ce phénomène d'explosion de données avec le développement rapide de système de positionnement comme l'AIS

(Automatic Identification System), l'AIS satellite, le VMS (Vessel Monitoring System) ou le LRIT (Long Range Identification System) qui dans leur ensemble contribuent tous à la mise à disposition et la disponibilité de ces données en mer. Carrefour des enjeux internationaux, l'espace maritime est soumis à une activité de plus en plus intense (cf Figure 6.1).

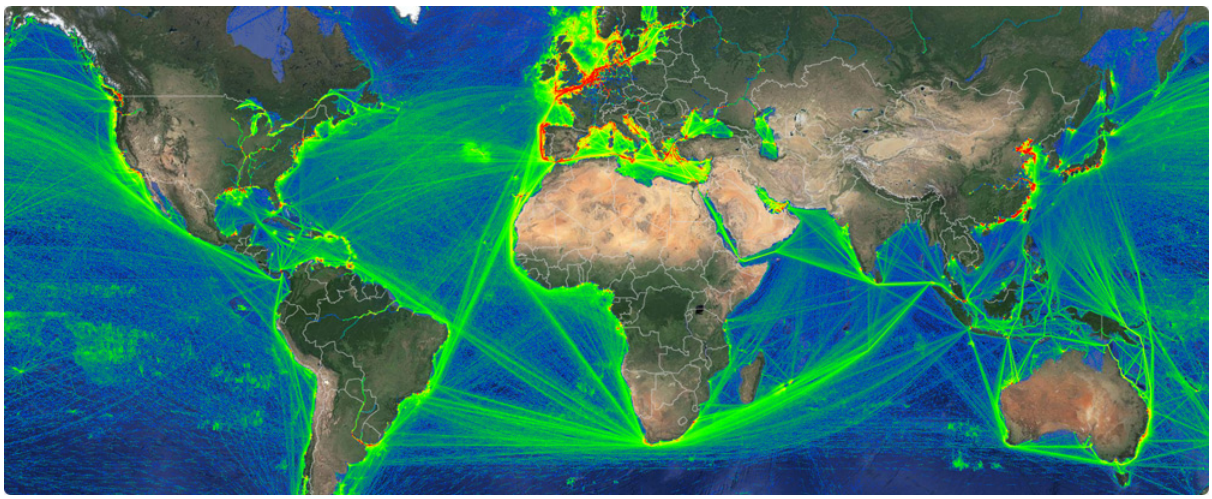

FIGURE 6.1 : Exemple de positions relevées par le système AIS

Les bateaux sont désormais majoritairement observables en temps-réel à l'aide des différents capteurs de positions, l'objectif étant d'identifier et localiser ces bateaux pour détecter des anomalies en mer. La surveillance des zones maritimes notamment côtières pour des raisons de sûreté et de sécurité, de gestion du trafic ou de protection des zones de biodiversité repose en grande partie sur l'identification de positions et de comportements types suivis par les navires qui permettront la détection d'anomalies ou de trajectoires anormales. Ceci pour détecter des activités illégales ou criminelles, les risques encourus (flux de produits illicites, immigration clandestine, surexploitation des ressources halieutiques, pollutions par des matières dangereuses, piraterie, accidents, etc.) et les violations de la réglementation. La multitude des systèmes de report de positionnement des navires mis en œuvre pour contribuer à l'analyse du trafic maritime et la détection d'anomalies en mer et in fine à la sécurisation de l'espace maritime génère désormais quotidiennement des volumes de données sans cesse croissants. En 2012, les reports de position dans les eaux européennes atteignaient déjà près de 5 millions par jour environ 17000 navires. Ceci nécessite de définir un modèle et un ensemble d'opérations et algorithmes pour la surveillance de ce trafic maritime.

Dans le cadre de notre étude nous nous servirons des données AIS (Automatic Identification System) ce dernier ayant été conçu initialement pour éviter les collisions en mer. L'Automatic

Identification System (AIS) est un système qui est embarqué à bord des navires depuis presque dix ans. Conçu comme un système d'anticollision pour les navires, il est obligatoire pour tous les navires soumis à la convention SOLAS ; les bateaux de plus de 300 tonneaux, les bateaux de pêche de plus de 15 mètres et les bateaux transportant des passagers. Le système est essentiellement composé d'un GPS (Global Positioning System), d'un transpondeur et d'une antenne VHF (Very High Frequency) pour communiquer. Les données échangées par un navire équipé du système AIS incluent en particulier des informations statiques (nom du navire, âge, cargaison, etc.) et dynamiques (cap, vitesse, position GPS, etc.). Les informations de positionnement sont transmises à fréquence élevée (2 à 12 secondes pour un navire en mouvement, 3 min pour un navire au mouillage). Le système transmet à une échelle de temps moins régulière les méta-informations liées au navire (identifiant international, nom, taille) et à son déplacement (destination, date et heure prévue d'arrivée) ainsi que des messages de contrôle. Des aides à la navigation, venant de stations à terre, exploitent également le système AIS (phares ou bouées par exemple). Les messages AIS transmis par les navires peuvent être captés par des récepteurs côtiers. La couverture côtière en capteurs est donc un paramètre essentiel permettant d'avoir une vision exhaustive du trafic côtier à une distance de 50-100 km environ. En France, une cinquantaine de capteurs côtiers positionnés dans les sémaphores et les centres opérationnels de surveillance et de sauvetage couvrent les côtes métropolitaines et outremer.

Parmi les 27 types de messages AIS (catégorisés dans le tableau 6.2) les messages relatifs au report de positions sont les plus fréquemment utilisés à cause de leur importance aussi bien pour la navigation que pour l'apport qu'ils peuvent fournir a posteriori en extrayant de l'information. Ces messages de reports de positions sont composés de plusieurs champs (décrits dans le tableau 6.3) tels que la position, l'instant auquel le message a été envoyé, le "speed over ground" (SoG), le "course over ground" (CoG) qui sont des informations d'intérêt dans les applications de navigation.

ID	Type	Description[8]
1-3	Position report	(Assigned) Scheduled position report, or response to interrogation
4	Base station report	Position, UTC, date and current slot number of base station
5	Static and voyage related data	Scheduled static and voyage related vessel data report
6-8	Binary related message	Binary communication
10-11	UTC related message	Request/Response-to UTC/date
12-14	Safety related message	Communication/Acknowledgement/Broadcast safety data
15	Interrogation	Request for special response
21	Aids-to-navigation report	Position and status report for aids-to-navigation
27	Position report for long range applications	Class A and Class B "SO" ship borne mobile equipment outside base station coverage

FIGURE 6.2 : Taxonomie des différents types de messages AIS [Tu et al., 2016]

Pour suivre le trafic maritime et détecter des anomalies, il est nécessaire d'analyser le com-

portement de ces navires via leur déplacement. L'analyse de comportements d'objets mobiles a pour objectif de comprendre les mouvements, les interactions entre les objets, leur environnement et comment il est possible d'interpréter ces comportements à partir des propriétés quantitatives et qualitatives observées des déplacements. L'étude des déplacements a pour objectif de faciliter la compréhension des causes, des mécanismes, des patrons spatio-temporels du mouvement et leur rôle dans l'évolution du système [Nathan et al., 2008]. Les régularités et les irrégularités dans les mouvements peuvent être décrites par des motifs (patrons ou modèles). Un motif peut être considéré comme la synthèse des mouvements, une description des comportements et un modèle de prédiction.

Field	Description[8]
MMSI	The vessel's Maritime Mobile Service Identity (MMSI)
Navigational status	0 = under way using engine, 1 = at anchor, 2 = not under command, 3 = restricted manoeuvre ability, 4 = constrained by her draught, 5 = moored, 6 = aground, 7 = engaged in fishing, 8 = under way sailing, etc.
Rate of turn	Right or left, from 0 to 720 degrees per minute
SOG	Knots(0-102.2) Speed over ground in 1/10 knot steps (0-102.2 knots) 1 023 = not available, 1 022 = 102.2 knots or higher
COG	Degrees(0-359). Course over ground in 1/10 = (0-359). 3600 (E10h) = not available = default. 3601-4095 should't be used
Heading	Degrees (0-359) (511 indicates not available = default)
Longitude	Longitude- to 0.0001 minutes
Latitude	Latitude- to 0.0001 minutes
IMO	IMO ship identification numbera seven digit number that remains unchanged
Draught	Draught of ship 0.1 meter to 25.5 meters
Destination	Destination max. 20 characters
ETA	Estimated time of arrival
Position accuracy	1 = high (<= 10 m) 0 = low (>10 m)
Time stamp	UTC time

FIGURE 6.3 : Composition d'un message de position AIS standard [Tu et al., 2016]

Afin de détecter automatiquement les comportements anormaux, il est possible soit de modéliser ce qui est anormal soit de modéliser ce qui est normal pour identifier les comportements qui s'écartent de cette normalité, ceci en utilisant des données anciennes afin de mettre en contexte les données reçues plus récemment. Modéliser le déplacement d'un navire nécessite de connaître les événements spatio-temporels permettant l'identification des différentes étapes constituant une trajectoire. Dans la littérature, plusieurs modèles existent. Ces derniers varient tant au niveau des thématiques auxquels ils s'appliquent que des besoins auxquels ils répondent. Par exemple, le modèle proposé par du Mouza et Rigaux [Mouza and Rigaux, 2005] permet uniquement l'identification de motifs de trajectoires dont les déplacements seraient déjà connus. Le modèle de Brakatsoulas et son équipe [Brakatsoulas et al., 2004] s'attache quant à lui à définir un modèle sémantique ainsi que les relations potentielles, mais uniquement pour le domaine routier. Du fait de ces différentes limites, le choix du modèle de trajectoire s'est porté sur celui proposé par Spaccapietra [Spaccapietra et al., 2008].

Celui-ci, plus générique, introduit un ensemble de concepts permettant d'attacher des informations à des événements spécifiques (stops, moves) et ainsi d'obtenir une trajectoire sémantiquement enrichie. Plus spécifiquement en ce qui concerne le domaine maritime, le suivi et la détection de comportement anormaux ont été facilités par des systèmes de report de positions comme l'AIS ou le LRIT. La détection d'anomalies par l'étude de trajectoires a notamment été abordée dans les travaux de Etienne [Etienne et al., 2010] et de Ristic [Ristic et al., 2008].

Aujourd'hui, les techniques de collecte, stockage, et d'interrogation des mobilités dans le domaine maritime héritent des travaux sur les bases de données pour objets mobiles (Moving Object Database ; MOD) [Forlizzi et al., 1999]. Ces travaux sont presque exclusivement basés sur un modèle relationnel et intègrent ou exploitent des extensions pour la gestion de ces mobiles (types spatiaux, jointures spatiales, opérateurs, indexation...) [Vodas et al., 2013], [Devogele et al., 2013]. Néanmoins, la plupart des algorithmes de détection nécessitent des trajectoires complètes avant d'effectuer le processus de classification. Cette procédure, dite offline, est une limitation importante pour les domaines nécessitant une réactivité immédiate. De ce fait, de nombreux travaux ont été réalisés sur la définition d'algorithmes online (ou séquentiel) permettant la détection d'anomalies même sur des trajectoires incomplètes [Patroumpas et al., 2015]. L'analyse se fait alors en même temps que l'objet mobile évolue [Laxhammar, 2008]. L'avantage de ces solutions de détection online a notamment été abordé dans Piciarelli et Foresti [Piciarelli and Foresti, 2006] ainsi que dans [Rhodes et al., 2007]. Nous avons dans cette thèse pris le parti de combiner ces deux approches, via l'architecture présentée précédemment à la fois pour avoir une meilleure compréhension du déplacement des navires, pour ingérer les forts volumes de données qui s'accroissent au fur et à mesure que les navires sont équipés et enfin pour que le système détecte d'éventuelles anomalies en temps-réel.

Les enjeux dans le cadre de l'étude de ces données maritimes concernent plus les problématiques de vélocité et de véracité. En effet, les volumes de données traitées ne sont pas du même ordre que ce qui peut être manipulé dans le cadre d'autres travaux de recherche [Ma et al., 2009], [Sun et al., 2013]. Cependant les traitements à effectuer sur les données d'objets mobiles sont bien plus coûteux que ceux intervenant dans le cadre des travaux cités précédemment. La surveillance du trafic maritime s'accompagne de problématiques opérationnelles et le besoin de réponses en temps-réel est plus que nécessaire pour aider les agents surveillant le transit des navires d'autant plus au vu de la forte densité de navires à certains moments (cf Figure 6.4). Pour cela, il est nécessaire d'avoir un système orienté temps-réel comme nous l'avons décrit précédemment (cf Section 4.3), mais qui puisse croiser les flux temps-réel et les examiner à la lumière des événements passés. De plus, les données AIS utilisées sont "imparfaites", notamment avec certains champs inexploitables (le champ destination mal rempli par les personnels à bord ou non rempli par exemple) avec d'autres qui ne respectent

pas la norme (des travaux montrent l'existence de différents bateaux avec le même identifiant ou MMSI) ou encore des champs dont les valeurs sont erronées à la réception du message (vitesse, cog ...). La notion de véracité est donc une composante importante à prendre en considération dans le cadre du traitement de données AIS. Enfin, même si cette problématique n'est pas développée dans cette thèse, l'utilisation de données complémentaires pour la compréhension des dynamiques maritimes telles que les données de courant ou météorologiques peuvent permettre d'extraire plus d'informations de traitement et constituent sans conteste une voie d'amélioration pour le suivi et l'étude du trafic maritime. Une prise en compte de l'aspect variété et la fusion avec des données extérieures autres que celles de positions récupérées via l'AIS peuvent compléter les informations extraites via nos algorithmes et contrebalancer le manque fiabilité parfois du système AIS.

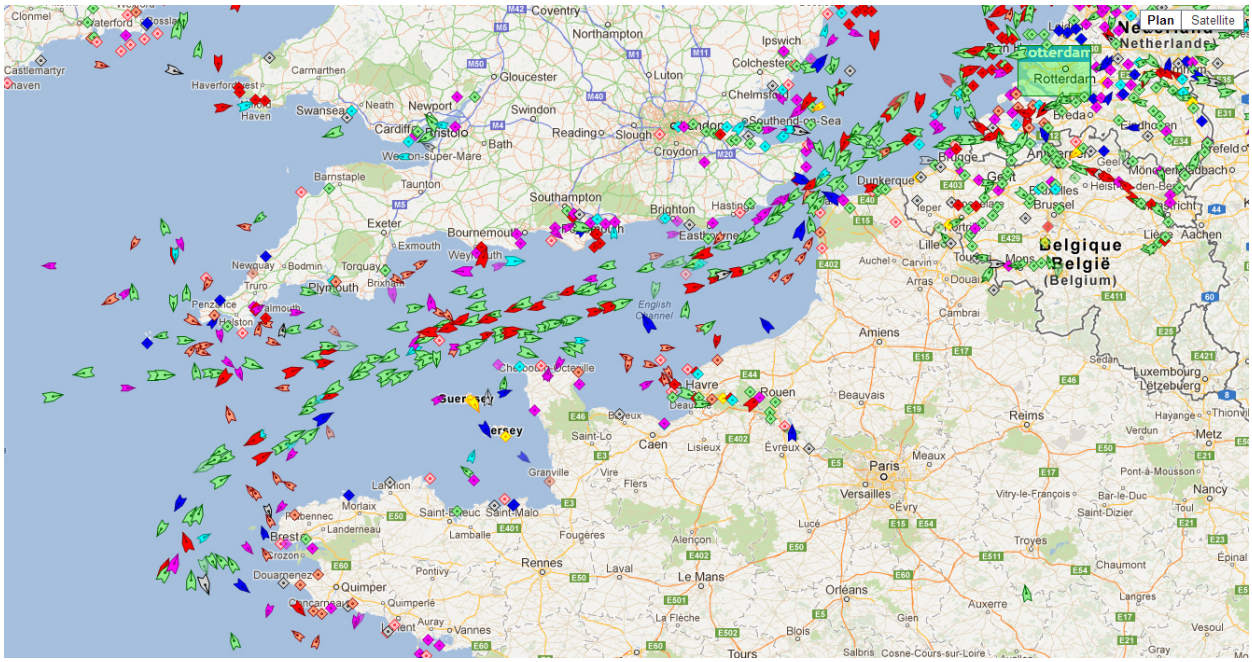

FIGURE 6.4 : Image extraite de Marine Traffic montrant la forte densité de navires

Malgré la multiplication des moyens de suivi des bateaux, des travaux récents montrent la possibilité de malversations au niveau de l'AIS [Ray et al., 2015]. Ceci nécessite la mise en place d'un système de suivi des bateaux et de détections d'anomalies ou de comportements anormaux dus à des attaques du système. Ceci nécessite dans la suite de définir des algorithmes spécifiques pour la surveillance du trafic maritime et la détection d'anomalies en mer. Parmi les quelques fraudes en mer répertoriées, Figure 6.5 nous pouvons identifier différents cas, comme un changement de la position, une usurpation d'identité ou des bateaux qui n'émettent plus leurs positions durant un temps donné.

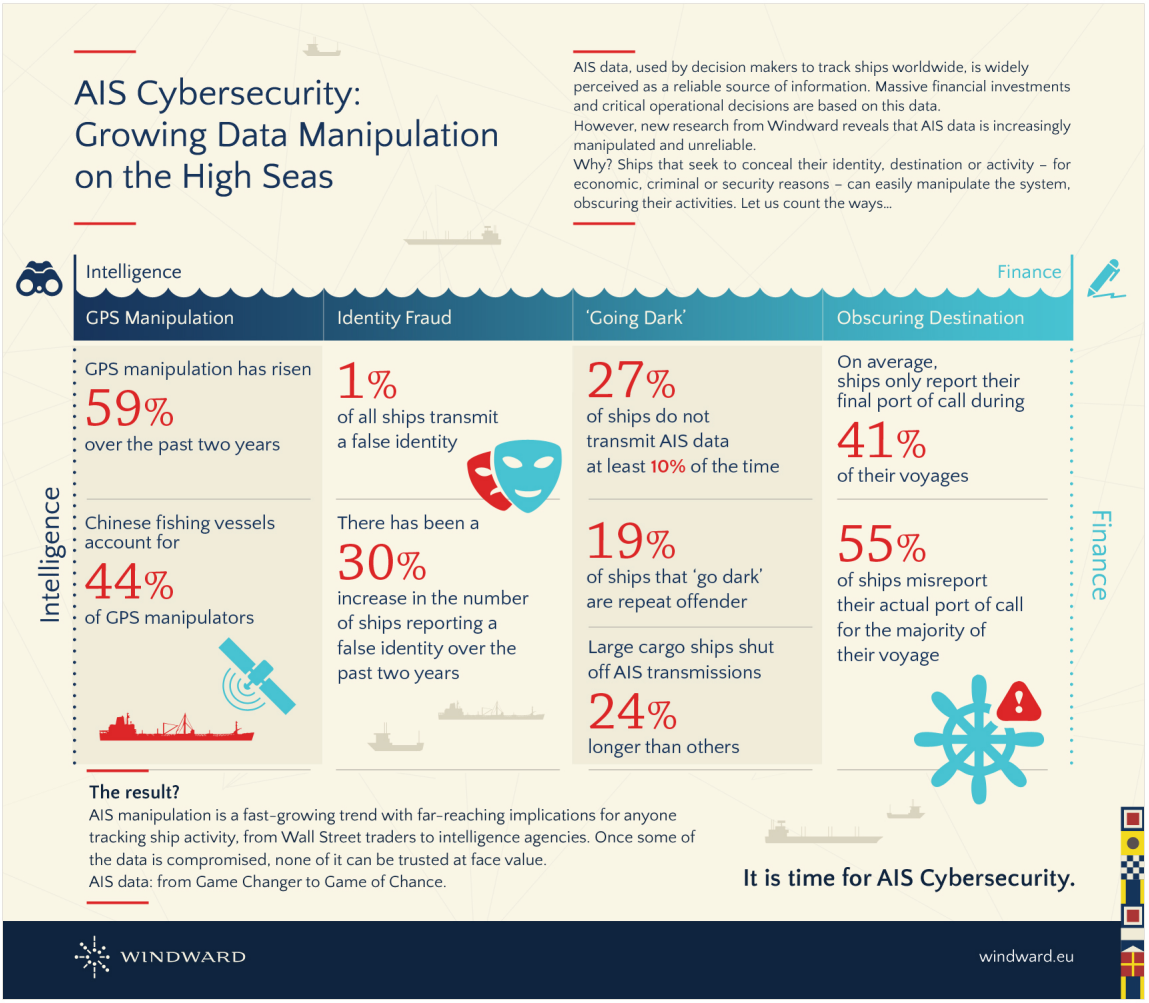

FIGURE 6.5 : Quelques statistiques sur les fraudes de l'AIS [Windward, 2014]

D'autres travaux récents à l'image de ceux entrepris dans le projet H2020 Datacron [1] ont adopté une approche proche de la nôtre. Dans [Claramunt et al., 2017], les auteurs soulèvent les problématiques associées à l'étude de données maritimes et proposent des alternatives pour traiter ces flux de données de plus en plus importants tout en les croisant avec des données supplémentaires dans le but de les visualiser et détecter des phénomènes d'intérêt (pêche illégale, migration illégale etc ...). Dans le cadre de ce projet, certains éléments de principe sont sensiblement les mêmes que les nôtres, notamment l'usage d'un processus de traitement hybride avec des flux reçus en temps-réel couplés à des résumés de données et des données archivées (Figure 6.6) [Santipantakis et al., 2018]. De plus, les auteurs utilisent également Flink pour le traitement des données reçues en temps-réel, celles-ci sont alors synthétisées sous formes de synopses et permettent l'identification de certains événements [Patroumpas et al., 2015]. Cependant, la détection d'événements est gérée à plusieurs

1. http ://datacron-project.eu/

niveaux d'abstraction, les flux étant assimilés en tant que position, puis transformées en synopses et éventuellement rejouées pour être interprétées en tant qu'événements, ceci permettant d'identifier des motifs spatio-temporels spécifiques de déplacements maritimes [Pitsikalis et al., 2018]. A l'inverse de notre proposition, la notion d'ontologie est abordée avec une base de données rdf permettant entre autres de croiser les données et de traiter l'aspect de variété des données et des sources [Santipantakis et al., 2017], tout en traitant l'aspect de volume massif de données à l'aide de Spark [Nikitopoulos et al., 2018]. Contrairement à ces travaux, nous définissons des requêtes pouvant être continues ou exécutées seulement une fois, les unes étant liées aux autres, et l'arrivée de données nouvelles mettant à jour certains des éléments puis finalement les résultats de l'ensemble des requêtes en cours. Ensuite, nous tirons partie de la notion de grille pour construire des résumés de données permettant d'appréhender le contexte maritime différemment tout en disposant d'un index commun aux parties online et offline

6.2.1 Modèle de données pour l'étude des mobilités maritimes

Contrairement à l'étude d'autres objets mobiles, les navires se déplacent en milieu dit "semi-ouvert" ce qui implique un modèle de données ou une représentation des données différent. En effet, en domaine urbain, la ville peut facilement être considérée comme un graphe dont les noeuds sont les intersections et les arrêtes les routes, dans ce cadre si une voiture se déplace sur une route avec une certaine vitesse v, il est facile de prédire sa position dans cinq minutes en l'absence d'intersection. À l'inverse pour le maritime, les objets mobiles peuvent changer de direction avec plus de liberté ce qui engendre une incertitude plus marquée sur la position de l'objet à l'instant t.

Pour étudier le trafic maritime, une grille uniforme est construite de manière analogue à celle définie dans la partie précédente (Section 4.4), séparant l'espace en cellules disjointes régulières $C_{i,j}$ de taille fixée $size_{cell}$ pour déterminer par exemple le nombre de positions par cellule et finalement identifier celles qui ne sont pas couvertes. La taille des cellules est évaluée suivant la norme [(international telecommunication union), 2014] décrivant la durée entre deux enregistrements provenant d'un même bateau considérant sa position actuelle, son cap et sa vitesse. La taille nécessaire des cellules est choisie de manière à empêcher les bateaux d'émettre deux positions successives dans deux cellules non adjacentes (i.e. à l'extérieur de zones non couvertes). Pour ce faire, le pire cas a été considéré, soit celui où un bateau est située aux frontières d'une cellule et avance selon un cap droit et avec une vitesse max notée V_{max} et $T_{t->t+1}$ l'intervalle de temps entre deux positions enregistrées. Dans ce cas, il est nécessaire d'envisager une taille fixe plus grande pour éviter l'enregistrement de deux positions successives dans deux cases non adjacentes. Pour minimiser le problème d'"answer loss" décrit dans [Jensen et al., 2006] qui peut arriver dans le cas d'une grille régulière, il est nécessaire de

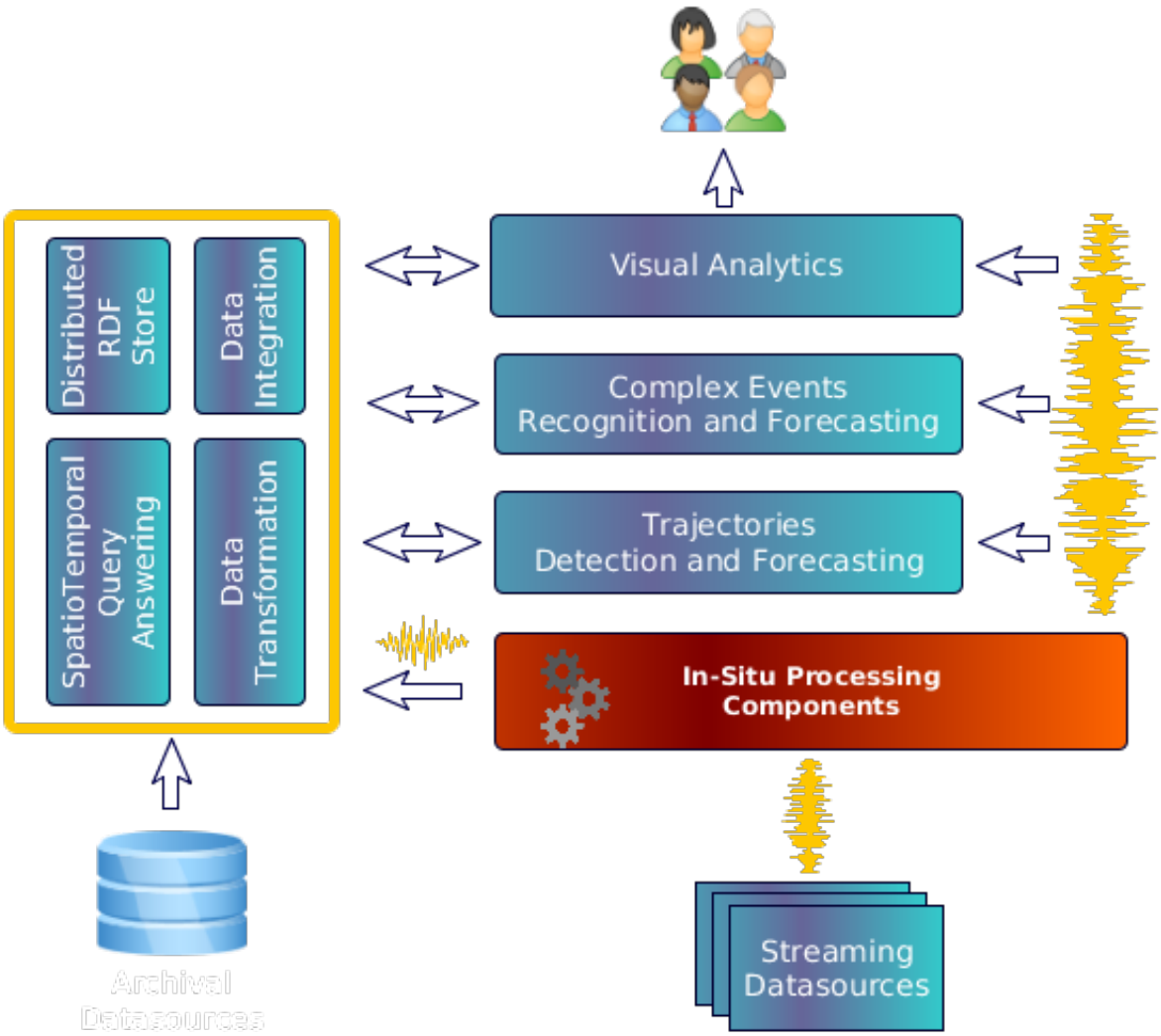

FIGURE 6.6 : Proposition pour le traitement et le stockage massif de données spatio-temporelles maritimes [Claramunt et al., 2017]

choisir les cellules les plus petites possible. En effet, en prendre des cellules plus fines augmente les temps de calcul tandis que prendre des cellules plus larges réduit les coûts, mais l'identification de certains événements peut être entachée. Par conséquent, nous avons défini la taille de cellule $size_{cell}$ telle que $size_{cell} = V_{max} * T_{t->t+1}$. Différentes notations additionnelles relatives à cette grille sont développées dans la suite.

Pour chaque cellule $C_{i,j}$ relative à la i^e abscisse et la j^e ordonnée pour i $\in 1..n$, j $\in 1..n$, les indices de mesure suivants sont introduits :

— $Npos_{i,j}$ τ correspond au nombre de positions qui ont été enregistrées dans la cellule $C_{i,j}$ pendant l'intervalle de temps $\tau \subset T$, plus formellement nous avons $Npos_{i,j}$ (τ) =

$\sum_{Id=1}^{nbpositions} t_{Id}$ during τ /

$$t_{Id} = \begin{cases} 1, & \text{if} p_k \in C_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

où nbpositions est le nombre de positions de navires qui ont été enregistrées pendant l'intervalle de temps τ.

— $Nvessel_{i,j}$ τ donne le nombre de bateaux distincts qui ont enregistré leur position dans la cellule $C_{i,j}$ pendant la période de temps $\tau \subset T$, plus formellement $Nvessel_{i,j}$ $(\tau) = \sum_{Id=1}^{nbvessels} t_{Id}$ pendant τ /

$$t_{Id} = \begin{cases} 1, & \text{if } \exists (p_k, t_k) \in Traj(Id) / p_k \in C_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

où nbvessels est le nombre de bateaux distincts dont la position a été enregistrée sur la période de temps τ.

— $Ncross_{i,j}(\tau)$ donne le nombre de bateaux distincts qui ont traversé la cellule $C_{i,j}$ pendant l'intervalle de temps $\tau \subset T$, ce qui signifie ici le nombre de bateaux dont la trajectoire (modélisée en tant que polyligne) a intersecté le cellule correspondante $C_{i,j}$, plus formellement $Ncross_{i,j}(\tau) = \sum_{Id=1}^{nbvessels} t_{Id}$ durant l'intervalle de temps τ /

$$t_{Id} = \begin{cases} 1, & \text{if } Poly(Id) \cap C_{i,j} \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

où nbvessels est le nombre de bateaux distincts dont la position a été enregistrée sur la période de temps τ.

Pour le maritime et notamment avec l'étude des positions AIS nous différencions, en raison de déficiences au niveau de la réception des messages, les cellules qui ont été traversées par un navire de celles où l'enregistrement d'une position s'est effectuée. Cela nous permet de gérer un cas éventuel où un bateau aurait transité par une cellule sans y enregistrer de position.

6.3 Modèle de traitement des mobilités maritimes

Requêtes typiques pour l'analyse du trafic maritime

Dans notre système, les requêtes peuvent être persistantes ou exécutées une seule fois en fonction de si elles sont exécutées en continu ou simplement une fois. Les requêtes persistantes sont la pierre angulaire de notre système, car elles permettent de produire des vues et des informations mises à jour qui synthétisent des résultats intéressants pour la compréhension et l'analyse du trafic.

Ces synopses peuvent être utilisés directement pour répondre à d'autres requêtes. Dans la suite, nous présentons une illustration de ces différentes requêtes qui ne saurait être exhaustive.

6.3.1 Requêtes persistantes typiques pour l'analyse du trafic maritime

Requête persistante relative à la reconstruction de trajectoires.
Pour chaque trajectoire, quand une nouvelle position est reçue par le système, la nouvelle position est ajoutée à la trajectoire. Quand l'enregistrement est fait sur un nœud différent de celui de l'enregistrement précédent alors les vues et les requêtes associées sont transférées sur le nouveau nœud d'appartenance du navire. La vue correspondant à la reconstruction de trajectoires est l'ensemble des différentes positions enregistrées jusqu'à durant les L dernières minutes, avec L la taille de la fenêtre glissante qui nécessite d'être définie. L'utilisation de vues nécessite certains mécanismes pour qu'elles soient mises à jour et qu'à chaque instant elles soient représentatives de la situation actuelle. Pour la reconstruction des trajectoires, les mises à jour sont faites en ajoutant les nouveaux enregistrements à la liste des positions tandis que les données les plus anciennes sont supprimées de la fenêtre via les *Evictors* décrits précédemment.

Requête persistante concernant la prochaine position d'un navire.
Avec le système AIS, chaque enregistrement n'est pas fait de manière linéaire, mais varie en fonction du cap et de la vitesse du navire. Ainsi, les bateaux lorsqu'ils se déplacent rapidement doivent émettre leur position de manière plus régulière que ceux dont le mouvement est linéaire et modéré [(international telecommunication union), 2014]. De cette manière, la prochaine position d'un bateau peut être estimée ainsi que son temps d'arrivée dans le système, et ce dernier peut lever et émettre des alertes si aucune position n'est reçue de la part d'un bateau depuis un moment ou si la position reçue est aberrante au vu de celle estimée grâce au cap, la vitesse et la dernière position enregistrée. Pour estimer la prochaine position d'un navire, le type de bateau, le cap précédent, la vitesse sont nécessaires tout en prenant en compte la notion d'incertitude. Ainsi, nous pouvons de manière analogue à certains travaux [Potamias et al., 2006] supprimer des données qui ne sont pas utiles pour le système (i.e. lorsque la position effectivement reçue peut être calculée à partir de la position précédente).

Requêtes relatives aux motifs de trajectoire récurrents.
Ici, nous souhaitons que le système puisse extraire en tant que synopses les trajets représentatifs utilisés par des navires lorsqu'ils traversent une zone de manière similaire à [Etienne et al., 2012]. Par exemple, il peut être intéressant de déterminer la route type utilisée par les cargos dans l'Océan

Atlantique, il peut être pertinent d'agréger les trajectoires de tous les cargos ayant traversé la zone pour en extraire les différents motifs et comportements représentatifs. D'autres algorithmes de fouille de données spécifiques aux objets mobiles et trajectoires peuvent être utilisés à l'image de DBSCAN ou des techniques de *clustering* pour déterminer des motifs spatio-temporels [Devogele et al., 2013]. Nous proposons d'utiliser l'algorithme FpGrowth [Han et al., 2000] dont l'objectif est de trouver les motifs fréquents pour un ensemble de transactions ou des séries temporelles. Cette technique de fouille de données semble appropriée et pertinente pour gérer la notion d'événements introduite précédemment (Section 3.4). En effet, en séparant l'espace et le considérant comme une grille uniforme, l'ensemble des données est réparti suivant cette grille sur les différents nœuds de l'architecture. Il suffit ensuite d'utiliser un algorithme de détection de motifs fréquents comme FpGrowth pour trouver des motifs spatio-temporels[Morzy, 2007]. Nous pouvons limiter la notion d'événement basique au fait qu'un navire a enregistré sa position dans un élément de la grille à un instant précis, tandis que l'événement complexe est la suite d'événements basiques associés à un navire dans le cas présent la séquence des différents "carrés" de la grille traversés par ce navire. Ainsi, un arbre des motifs fréquents est construit (trajectoires et sous-trajectoires fréquentes) depuis la base de transactions (l'ensemble des trajectoires).

D'autres requêtes sont formulées par les agents maritimes lorsqu'ils observent des comportements suspects. Notre système bénéficiant des vues dérivées des requêtes persistantes peut répondre plus efficacement et rapidement à ces requêtes ad-hoc. Dans la suite, nous définissons un ensemble de requêtes représentatives qui là encore ne saurait être exhaustif.

6.3.2 Requêtes ad-hoc typiques

Requête Ad-Hoc : "Est ce que ce bateau a une trajectoire "normale" ?"
Le rôle des agents maritimes est de surveiller le trafic maritime dans le but d'empêcher des accidents en mer ou réagir rapidement pour réguler le trafic maritime lorsqu'un problème se produit. Dans ce contexte où les agents doivent chercher parmi des centaines de bateaux voire plus pour des zones très denses, une requête permettant de détecter automatiquement des comportements étranges ou atypiques peut être utile. Pour déterminer la "normalité" d'une trajectoire, une des méthodes peut être de comparer les trajectoires actuelles avec les motifs spatio-temporels stockés dans notre système en temps que vues ou synopses.

Nous pouvons de cette manière utiliser les vues relatives à l'extraction de motifs et calculer la distance avec chacune des trajectoires "récentes" en utilisant par exemple une distance de Jaccard. Si la distance de Jaccard entre la trajectoire décrite par le navire et le couloir spatio-temporel extrait

est plus élevé qu'une certaine valeur seuil, alors nous pouvons "zoomer" et regarder les déplacements de ce bateau sur les derniers jours, pour voir s'il y a des motifs qui reviennent de manière périodique ou si d'autres anomalies ont été détectées sur une plage temporelle plus grande (i.e. quelques jours). Voici l'intérêt de ces requêtes continues et des vues dérivées, qui permettent de faire un filtrage sur l'ensemble des éléments et de se concentrer uniquement sur les comportements ou navires suspects.

Requête Ad-Hoc : "Quelles sont les zones de pêche proches de Brest ?"
Identifier les zones de pêche est pertinent parce que cela permet de distinguer les bateaux en situation de pêche légale de ceux qui sont en train de pêcher illégalement. Ceci est une problématique grandissante d'intérêt pour la préservation des espèces et de la biodiversité. Pour ce faire, nous restreignons l'espace d'étude aux bateaux de pêche dont le comportement est spécifique et peut être facilement identifiable lorsqu'ils sont en train de pêcher. Lorsqu'ils sont en activité, ces navires se déplacent à vitesse réduite et décrivent souvent un mouvement erratique, pour chaque navire nous pouvons extraire l'emprise spatiale correspondant à la zone de pêche et ainsi par comparaison et intersection avec celles des autres bateaux obtenir une carte de densité des pêches effectuées dans la zone [Le Guyader et al., 2016].

Ces aires de pêche extraites depuis le traitement de données archivées et mises à jour en fonction des données reçues en temps-réel sont comparées avec les bateaux actuellement en transit pour identifier les bateaux en situation de pêche illégale. Un des problèmes est le fait que certains pêcheurs coupent leur signal de transmission pour pouvoir pêcher dans des zones illégales sans émettre leur position ou être repérés. Pour étendre cette requête, une des pistes est la détection de période de non-émission de signal par les bateaux pour permettre d'identifier les fraudeurs.

6.4 Conclusion

Ce chapitre étend notre modèle défini précédemment pour prendre en compte les spécificités intrinsèques au domaine maritime, à savoir l'imprécision des données, un déplacement en milieu semi-ouvert et des cas pratiques ou événements spécifiques qui ont pu être modélisés et détaillés. Les données reçues en temps-réel et un contexte opérationnel de surveillance du trafic et de sauvetage en mer et l'incertitude quant aux données reçues nous ont fait mettre l'accent sur les aspects de vélocité et de véracité des données, là ou d'autres travaux concernent la variété et le volume des données appliquées à des contextes maritimes. Nous avons également montré la pertinence et l'aspect générique de notre approche en l'étendant donc au contexte maritime et en définissant des requêtes typiques concernant la surveillance du trafic maritime, donnant des principes de fonctionnement

notamment online. Celle-ci tire partie par exemple d'un échantillonnage spatio-temporel spécifique (par rapport à des données à caractère non spatial), d'un traitement incrémental des données ou bien de l'usage de la notion d'événements pour l'identification de comportements types et par suite la détection d'anomalies dans un contexte de trafic maritime. Cependant, même si elles permettent d'illustrer les processus et mécanismes intervenant dans notre système sur un cas concret, cette notion de requête persistante nécessite d'être enrichie par la notion d'événement, afin d'identifier des comportements normaux et à terme détecter des anomalies dans le trafic maritime. Il apparait au final que cet ensemble de requêtes non exhaustif mérite d'être enrichi et pourrait servir à définir d'autres requêtes ou à la détection d'autres phénomènes en mer comme les rendez vous, migrations illégales ou dégazage en mer.

Chapitre 7

Mise en place de l'architecture et analyse des résultats

Sommaire

7.1 Introduction

Dans les parties précédentes différentes notions ont été abordées d'abord le modèle de gestion d'objets mobiles proposé, ensuite les mécanismes d'un traitement hybride au travers d'une requête spécifique puis enfin le contexte maritime et la déclinaison de notre modèle pour le suivi du trafic maritime. Dans cette partie nous présentons et explicitons la façon dont notre approche a été mise en place, l'application de l'algorithme détection de Black Holes à des données réelles de positions de navires ainsi que l'analyse des résultats de cette requête.

7.2 Architecture générale et choix techniques

Dans cette partie, nous introduisons les éléments opérationnels de notre système. Celui-ci dérive d'Apache Flink abordé dans la section 4.2.1, car il semble plus approprié pour répondre à

une problématique opérationnelle temps-réel tout en permettant un traitement hybride des données (cf Figure 4.2). La solution proposée prend également en compte éléments et principes formulés précédemment (cf. Figure 4.3 and Figure 4.4). Flink fournit deux abstractions nommées "DataSet" et "DataStream" (respectivement pour les parties offline et online) semblables aux RDD décrits précédemment via une API intégrée en scala un langage de programmation fonctionnel pour la machine virtuelle java. Nous avons choisi scala car il combine à la fois une écriture concise (ce qui est pratique pour un usage interactif) et efficace (car typé de manière statique).

Pour la gestion des objets mobiles, nous avons décidé de choisir Flink décrit précédemment et ceci pour plusieurs raisons. D'abord, car contrairement à Spark envisagé un temps, Flink est spécifiquement dédié au traitement temps-réel là où Spark à une approche "orientée batch" et bien qu'avec son extension temps-réel Spark Streaming il soit possible de traiter des données temps-réel via le concept de "mini-batch" nous avons préféré Flink. Ce système plus orienté temps-réel possède des mécanismes intéressants notamment au niveau du fenêtrage et des triggers spécifiques. Enfin, Flink possède une extension pour la gestion d'événements, permettant de traiter les données reçues en temps-réel en tant qu'événements, notion importante ou intéressante pour permettre le couplage entre données temps-réel et anciennes.

Différents éléments présents dans Flink et la façon dont les données sont gérées dans le système Apache Flink sont décrits ci-après. Un *Window Assigner* est responsable de l'affectation des éléments à une ou plusieurs fenêtres de manière analogue à notre *Window Manager* défini dans la figure 4.4. Chaque fenêtre possède un déclencheur qui lui est associé, "forçant" l'évaluation ou l'exécution d'une requête ou un traitement sur les données présentes dans la fenêtre. Le déclencheur est "actionné" soit lorsqu'une donnée est entrante dans le système, soit lorsqu'un événement particulier arrive (par exemple lorsqu'aucune activité n'a été enregistrée pendant une certaine période de temps.). Une fenêtre peut être évaluée plusieurs fois, en fonction de la nature du déclencheur et des données entrant dans le système. L'*Evictor* est un complément optionnel responsable de la suppression des données les plus anciennes de la fenêtre et peut intervenir indépendamment ou après l'action du déclencheur.

Flink permet de gérer des processus itératifs de manière spécifique. Pour un algorithme itératif classique comme les algorithmes de *machine learning*, l'entièreté des données en entrée est consommée à travers une chaîne d'opérateurs pour produire un ensemble de résultats intermédiaires, eux-mêmes traités par cette chaîne jusqu'à ce que les conditions d'arrêt soient atteintes (critère de convergence, nombre maximum d'itérations ...). Flink introduit la notion de "delta itération" qui peut être très appropriée à notre contexte, c'est-à-dire qu'au lieu de réévaluer l'entièreté des données à chaque étape, les données nouvellement reçues par le système sont évaluées et fusionnées avec les résultats dérivant de la précédente itération.

La Figure 7.1 présente l'architecture de Flink, le programme géré au niveau du client étant traduit en termes de workflow d'exécution.. Le *JobManager* est responsable de l'affectation des tâches, du partitionnement et de la coordination des tâches, tandis que zookeeper est utilisé pour la gestion de la communication entre les noeuds, la conservation des métadonnées et la tolérance aux pannes. Chacun des *Task Manager* est en charge localement de l'exécution de traitement et de la gestion de la mémoire, qui sont ensuite traduites "physiquement" sur des noeuds d'une architecture distribuée.

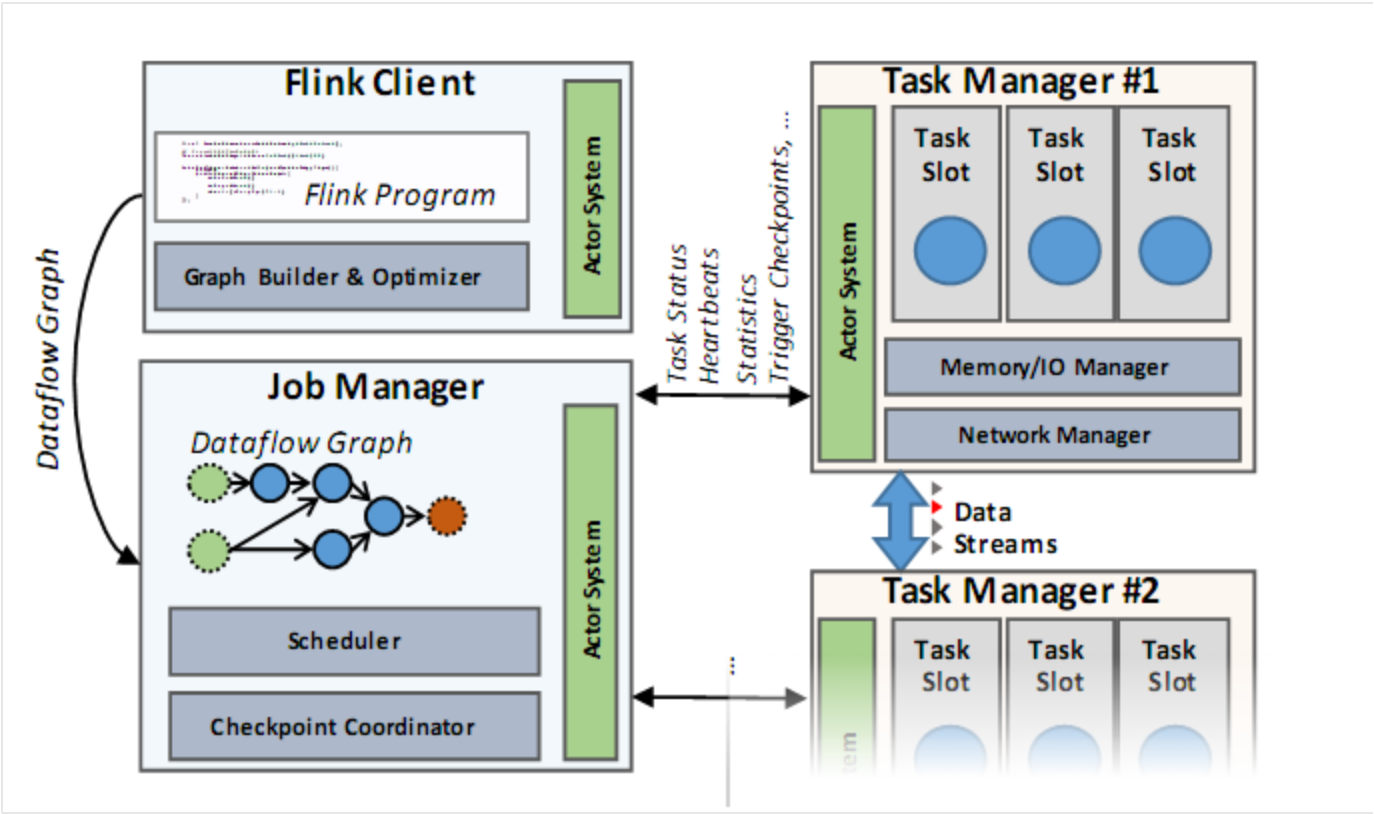

FIGURE 7.1 : Architecture "physique" de Flink [Carbone et al., 2016]

La Figure 7.2 présente elle l'ensemble des éléments de Flink, avec les différentes librairies présentes à un niveau d'abstraction supérieur notamment pour faire du traitement de graphes, du machine learning, exécuter des requêtes SQL, de manipuler des tables en mémoire ou bien encore d'utiliser la notion d'événements. À un niveau plus bas d'abstraction, les "DataSet" et "DataStream" constituent des conteneurs sur lesquels des opérations notamment ensemblistes sont admises. Les opérations s'effectuent nativement de manière parallèle sur ces "DataStream" ou "DataSet" permettant également à l'utilisateur de "découper" ces ensembles de façon à pouvoir distribuer les traitements à sa guise. C'est à ce niveau que se place notre contribution, en encapsulant l'ensemble de nos objets spatio-temporels dans ces DataStream" ou "DataSet" et implémentant des

méthodes spécifiques bénéficiant du traitement parallélisé de Flink. Enfin la partie runner est le workflow en lui-même (cf Figure 7.1), l'ensemble des opérations décrites en tant que "DataStream" ou "DataSet" étant traduites sous forme de graphe de manière optimisée. L'exécution de ce graphe peut être déployé physiquement, en local ou bien via un cluster sur lequel les opérations seront effectuées.

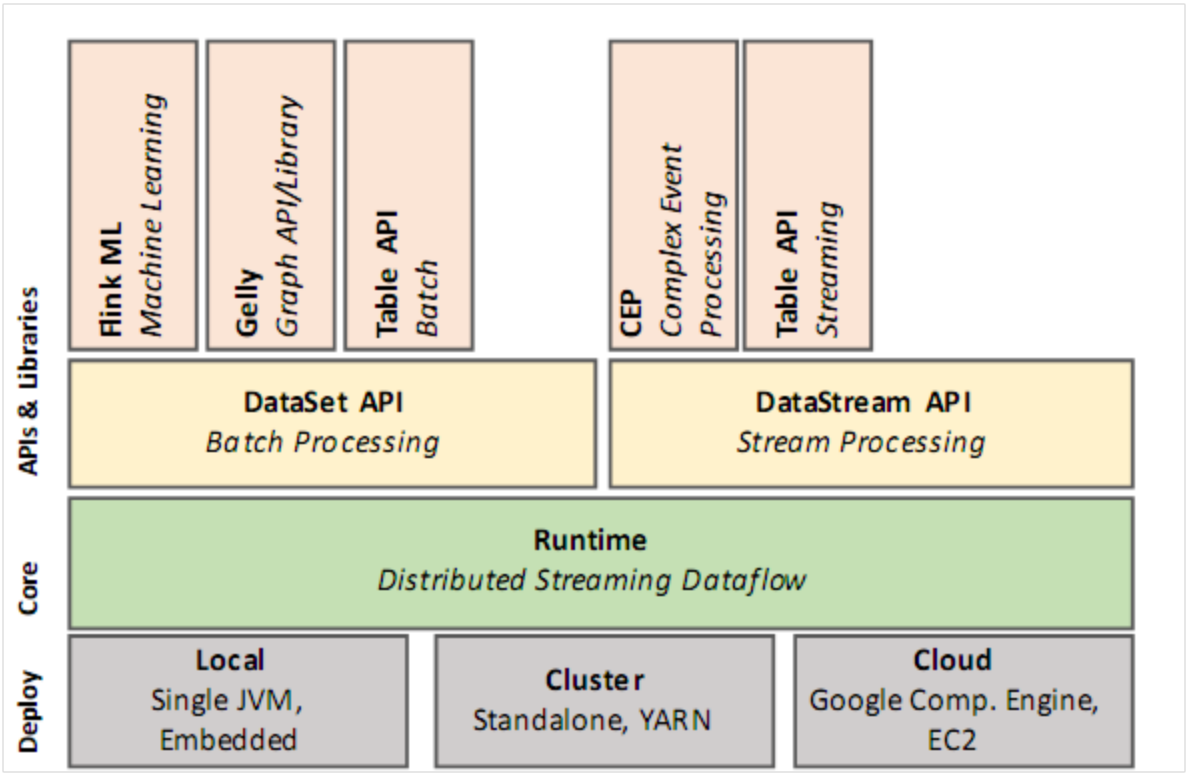

FIGURE 7.2 : Couches logicielles associées à Flink [Carbone et al., 2016]

Pour assimiler et simuler un flux, nous prenons des données stockées dans un fichier que nous "rejouons" via Kafka (Figure 7.3 présentant l'implémentation développée dans le but d'identifier des Black Holes en mer). L'intérêt d'utiliser Kafka pour assimiler le flux de données est qu'il pourra permettre de diviser le fichier d'entrée en deux, trois ... n sources pour vérifier l'efficacité des méthodes implémentées et aussi simuler un environnement dans lequel les données proviennent de différents récepteurs. Les données sont ensuite assimilées et traitées "à la volée" comme suit pour générer des synopses et tables en temps-réel pouvant être modifiés et utilisés. Pour la partie online l'API "DataStream" est utilisée pour assimiler et traiter les flux, tandis que pour la partie offline l'API "DataSet" est utilisée et lit directement l'ensemble du fichier pour générer la grille, les trajectoires et les indices de densité relatives à la grille. Nous définissons les structures de données définies précédemment à savoir la grille ainsi que les trajectoires à différents niveaux d'abstraction notamment comme une succession de points et une succession de segments. La notion d'événements abordée

n'a pas été mise en place, et pourra faire l'objet de travaux postérieurs. Nous avons également implémenté les différents opérateurs et requêtes spatio-temporelles introduites précédemment (Section 3.2) notamment contient, distance, intersection ainsi que range query et requête k plus proche voisin à l'aide de librairie JTS pour les aspects spatiaux et la librairie Joda pour les aspects temporels. L'intérêt de la librairie JTS est son interopérabilité avec notre implémentation en scala et le fait qu'elle implémente le modèle géométrique et les standards OGC, avec les objets spatiaux basiques (point, lignes, géométries etc ...) et des index spatiaux (R-Tree, Quad-tree etc ...). De manière analogue, Joda permet la manipulation de notions temporelles basiques (instant, intervalle etc...) ce qui est nécessaire pour manipuler nos données d'objets mobiles.

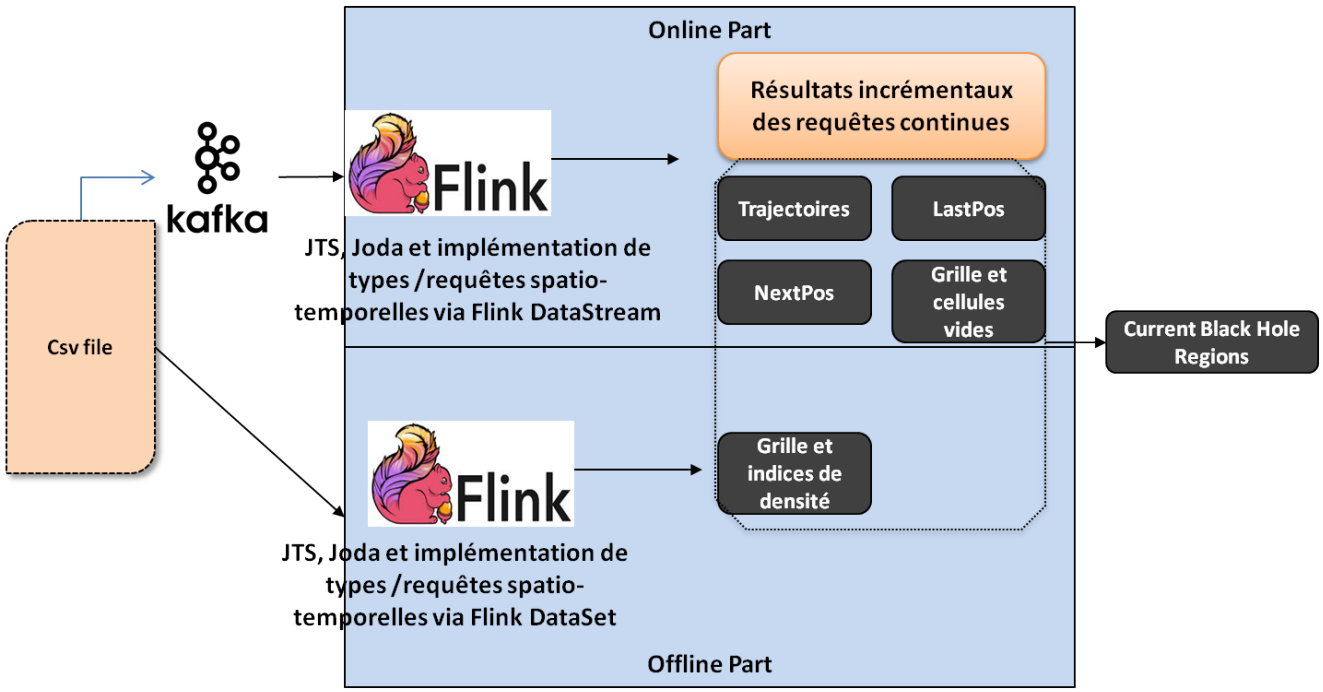

FIGURE 7.3 : Implémentation de types spatio-temporels et des éléments pour l'identification de Black Holes en mer

Concernant la distribution des données, celles-ci sont réparties sur les différents nœuds de l'architecture en fonction de leur emprise spatiale. L'espace est découpé en utilisant une grille uniforme, malgré les phénomènes de disparité de densité qui nécessiterait l'utilisation de techniques spécifiques pour distribuer les données. L'avantage de répartir les données et requêtes en fonction de leur emprise spatiale est que si un objet mobile considéré ne quitte pas la zone de laquelle il dépend le nombre de données à examiner est moindre et ne requiert pas le concours de plusieurs nœuds de l'architecture desquels il faudrait extraire de l'information et ensuite croiser les résultats obtenus. Le problème intervient lorsqu'un objet mobile quitte cette aire spatiale, nécessitant de transférer données et requêtes relatives à cet objet mobile sur un autre nœud de l'architecture

responsable de la maintenance des données sur la nouvelle aire d'appartenance de l'objet mobile d'une manière analogue au modèle QTP introduit dans [Xiong et al., 2007]. L'intérêt ici par rapport à un partitionnement dynamique que les données n'ont pas à être réallouées dynamiquement pour que la répartition des données sur les noeuds soit équilibrée.

7.3 Identification de Black Holes en mer

Comme nous l'avons vu précédemment, l'espace maritime est soumis à une activité de plus en plus intense. La surveillance des zones maritimes notamment côtières pour des raisons de sûreté et de sécurité, de gestion du trafic ou de protection des zones de biodiversité repose en grande partie sur l'identification de positions et de comportements types suivis par les navires qui permettront la détection d'anomalies ou de trajectoires anormales. Cependant, la couverture géographique de ces systèmes de positionnement n'est pas parfaite et trouve ses limites. Étant donné cette vulnérabilité du système de report de positions AIS ceci laisse l'opportunité à des personnes malveillantes de perturber le système. Parmi les quelques fraudes en mer répertoriées, Figure 6.5 nous pouvons identifier différents cas, comme un changement de la position, une usurpation d'identité ou des bateaux qui n'émettent plus leurs positions durant un temps donné. Dans la suite, nous nous focaliserons sur le dernier enjeu à savoir la disparition volontaire d'un navire, ceci implique d'abord de connaître la couverture géographique de l'AIS qui est imparfaite et trouve ses limites.

En effet, il y a toujours des zones desquelles les positions émises par les bateaux ne peuvent être reçues par les stations à terre ou ne sont pas couvertes par une antenne, ce qui est typiquement le cas pour l'AIS sur lequel nous allons travailler. Ceci rend difficile le processus d'identification de navires éteignant leur signal de manière volontaire, ne sachant pas si les personnes à bord du bateau ont coupé sciemment leur signal ou bien si le navire pénètre simplement dans une zone non couverte par les stations à terre [Salmon et al., 2016].

Pour détecter ces *Black Holes* une grille est construite pour permettre d'identifier plus formellement ces zones. La figure 7.4 illustre un autre phénomène de *Black Hole*, la carte ayant été obtenue cette fois à partir de positions AIS enregistrées dans les eaux grecques dans la zone des Cyclades. Une grille régulière est appliquée à la carte et la distinction peut être faite entre la région B où aucun signal n'a été reçu alors que dans les cellules adjacentes (A et C) des positions sont enregistrées et décrivent des trajectoires allant de A à C en passant par B, mais sans signal émis en B.

Le concept de *Black Holes* et les notions de couverture et non-couverture associées sont liés et d'une importance cruciale pour la surveillance du trafic maritime. En effet, l'analyse des mobilités et des trajectoires en mer est particulièrement importante pour des raisons de sécurité et de sûreté

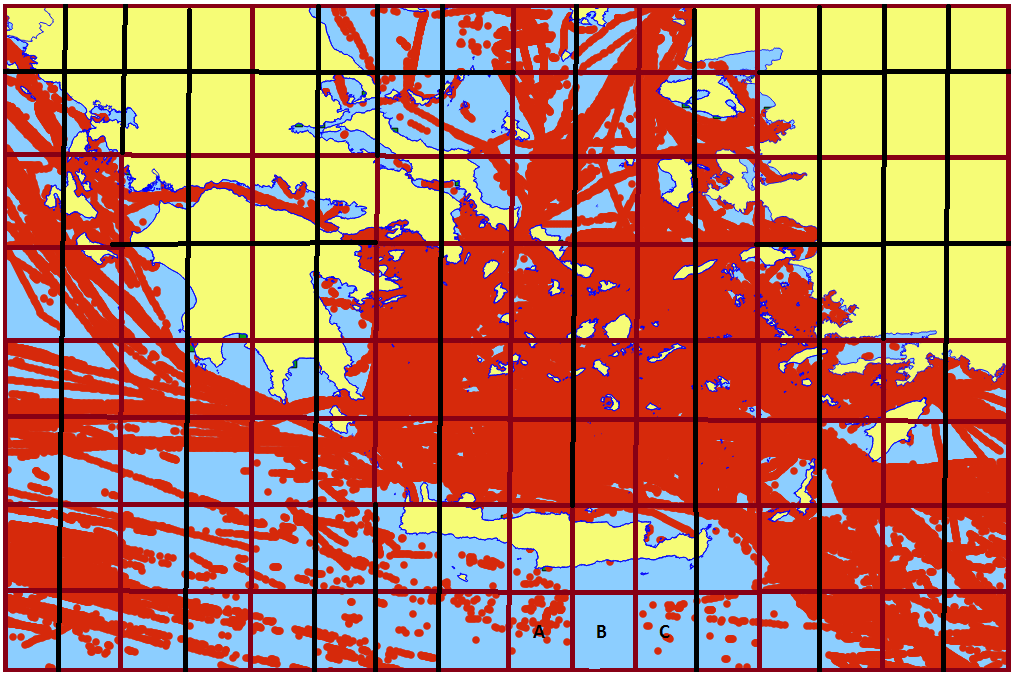

FIGURE 7.4 : Exemple de *Black Hole* dans la mer Égée (Grèce)

du trafic maritime. L'analyse de mobilité et de comportement peut dans ce contexte permettre de détecter des activités criminelles ou illégales en mer (flux de produits illicites, immigration illégale, surpêche, pollution, piraterie ...) et plus généralement toute violation aux règles de régulation du trafic maritime [Pallotta et al., 2013].

Cependant, ces problématiques sont loin d'être évidentes ou résolues. D'abord, parce que ces *Black Holes* ont des frontières qui sont "floues" et qui fluctuent au cours du temps alors que la météo et les conditions atmosphériques changent, ce qui peut impacter considérablement la transmission et l'envoie de messages AIS. En effet, les systèmes AIS et leurs capteurs sont alimentés par des ondes Electro-Magnétiques (EM) traversant l'atmosphère et sont donc sujets à un certain nombre de facteurs environnementaux susceptibles de modifier leur comportement nominal. Ainsi, les conditions météorologiques, le rayonnement solaire et l'état de mer affectent les communications radio, le positionnement du navire et de son système antennaire et des services tels que le GPS. Ces facteurs impactent directement la portée et la visibilité mutuelle des mobiles, mais également les capacités de détection des systèmes côtiers ainsi que la qualité du positionnement. De plus, le comportement de l'AIS avec des reports de positions qui sont irréguliers et des limitations en termes de couverture renforce les difficultés rencontrées pour identifier formellement ces zones. En fait, le concept de *Black Holes* doit être étudié à différents moments pour permettre de déterminer ces zones couvertes et non couvertes. À un temps donné, idéalement, il devrait être possible de savoir quelles sont ces zones non couvertes pour détecter des anomalies ou des comportements suspects, c'est-à-dire ici faire la

différence entre les bateaux qui ont leur transpondeur AIS éteints, ceux qui traversent une zone spécifique où aucun signal émis n'est reçu par les stations de réception à terre ou encore d'éventuels faux messages [Ray et al., 2015].

Pour identifier ces *Black Holes*, les données historiques aussi bien que les flux temps-réels sont nécessaires pour fournir de manière appropriée un indice sur les fluctuations de ces zones de non-couverture. En effet, la manière qui amène à déterminer la couverture ou non couverture d'une zone ne dépend pas seulement des conditions météorologiques, mais aussi de l'analyse des données anciennes qui permet de fournir plus d'informations sur la couverture des données dans l'espace et le temps. Par exemple, certaines régions dont nous savons qu'elles sont généralement couvertes peuvent devenir momentanément des *Black Holes* si les conditions météorologiques ne sont pas favorables. D'un autre côté, si sont considérées uniquement les données reçues en temps-réel, des zones vides (i.e. où aucun bateau ne passe effectivement) pourraient être considérées de manière abusive comme des *Black Hole* alors qu'en l'occurrence elles n'en sont pas. Pour résumer, le manque de données de positions nécessite d'être interprété comme la non-couverture de cette zone ou bien simplement l'absence de trafic maritime dans la zone considérée. L'objectif du cas d'application actuel est donc une approche particulière dont le but principal est d'obtenir à chaque période de temps considérée une image globale de la couverture en signal AIS de l'ensemble des régions maritimes et ainsi de détecter des anomalies potentielles dans le trafic maritime.

Nous avons nommé ces zones spécifiques desquelles les positions émises par les bateaux ne sont pas reçues *Black Holes*. La figure 7.5 illustre ce phénomène observé au niveau du cap de la chèvre, cette carte ayant été obtenue par extraction et visualisation des positions AIS dans les eaux aux abords de la presqu'île de Crozon.

7.4 Résultats expérimentaux

Cette section décrit les résultats extraits de la partie historique du jeu de données correspondant à la région maritime de Brest et qui contient plus de 18 millions de positions sur une période de six mois.

7.4.1 Description du jeu de données

Le jeu de données utilisé pour l'évaluation de notre traitement hybride et la détection de *Black Holes* dérive d'une étude préliminaire du trafic maritime à l'ouest de la France. Le jeu de données couvre une période de temps de six mois, soit entre Octobre 2014 et Mars 2015. Ces données ont été

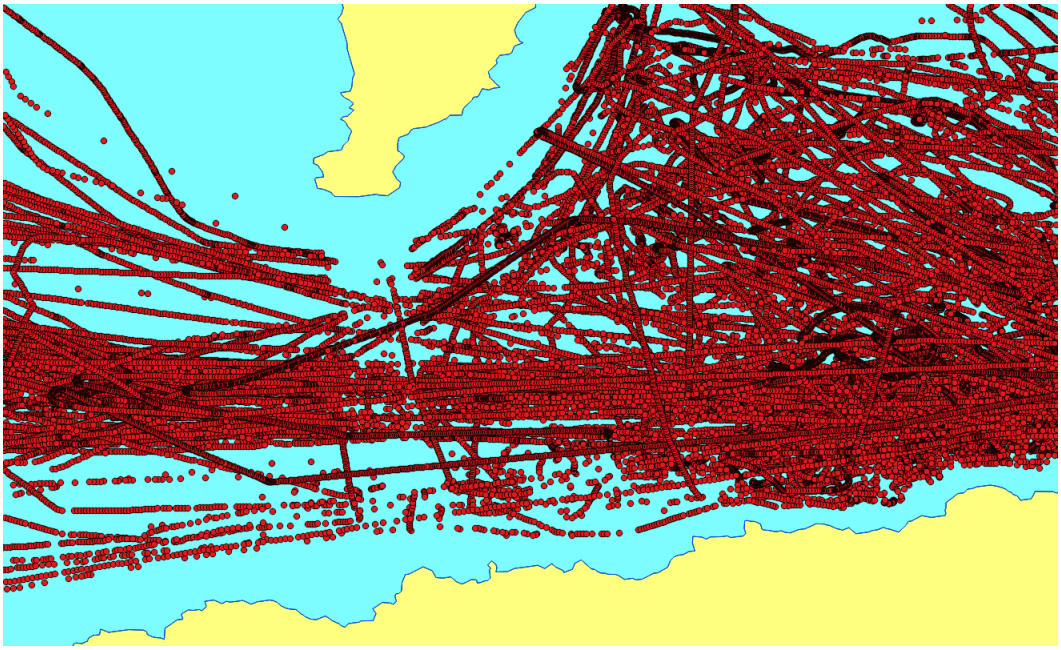

FIGURE 7.5 : Exemple de Black Hole au niveau du cap de la chèvre (Brest, France)

reçues et "parsées" en temps-réel, puis stockées dans un fichier servant d'entrée pour notre algorithme de détection. Sur cette période de six mois 24,467,196 messages AIS ont été reçus (i.e. en moyenne une valeur et demie par seconde), et 1,316,689 d'entre eux (soit environ 6%) ne respectent pas la norme AIS [(international telecommunication union), 2014]. L'ensemble de ces messages erronés ont été retirés de notre jeu de données. Le système AIS compte 27 différents types de messages. Le jeu de données est composé uniquement des reports de positions, ce type de messages représentant en moyenne 81.3% des messages reçus (avec en moyenne 7.3% d'erreur sur les reports de position). Notre jeu de données contient finalement 18,115,534 positions. L'emprise spatiale de ces reports de positions est donnée par la Figure 7.6. Tandis que la plupart des positions sont dans un rayon de portée radio (15,991,493 localisées à l'intérieur d'un cercle de 50 km), la couverture de l'AIS évolue de manière continue au gré des conditions météorologiques, le récepteur peut obtenir des positions bien au-delà de la ligne de vue (2,124,041). La distance maximale au récepteur considérée pour cette étude est restreinte à 1000 km (les autres points étant considérés comme des outliers).

7.4.2 Description de la méthodologie

L'ensemble des expérimentations ont été faites en utilisant Flink [Alexandrov et al., 2014], une plateforme "Big Data" fournissant une architecture permettant de traiter aussi bien les données d'un point de vue offline qu'online. Cependant, ce système ne dispose d'aucun support ou opérateur spatial ou spatio-temporel. Nos propres opérateurs ont été implémentés en accord avec l'architecture

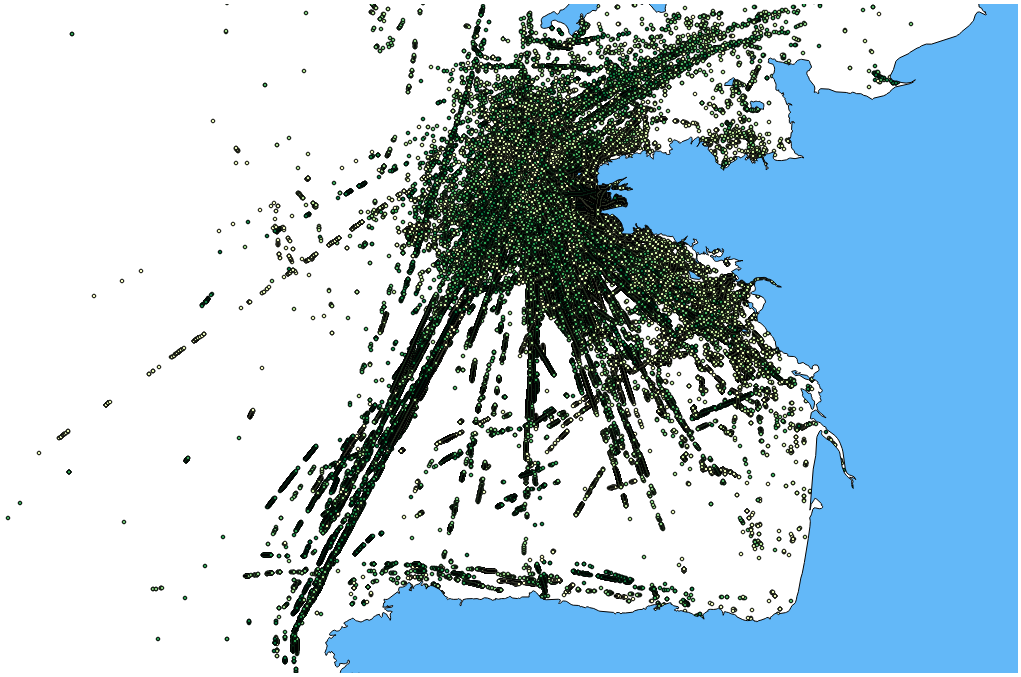

FIGURE 7.6 : Illustration de l'ensemble des messages reçus durant l'étude

proposée dans [Salmon et al., 2015]. La recherche de *Black Hole* permet d'illustrer le type de problèmes particuliers que nous souhaitons pouvoir traiter avec le principe d'une approche hybride. La partie offline doit permettre de synthétiser des informations ou une connaissance et de réduire l'espace des solutions tandis que la partie online doit permettre de donner une réponse en prenant en compte les données reçues en temps-réel en les croisant avec les informations de la partie offline. La taille des cellules de la grille a été fixée à 0.2*0.2 degré et le quantile θ choisi pour la partie offline est de 5%. Pour définir l'ensemble des données archivées, nous considérons les cinq premiers mois comme appartenant à la partie offline et le dernier mois est utilisé pour la partie temps-réel. Pour la partie temps-réel, les données sont "lues" comme si elles arrivaient dans le système avec une fréquence plus élevée et avec des estampilles temporelles ordonnées (considérer des estampilles non ordonnées par rapport au temps pourra faire l'objet de travaux futurs).

7.4.3 Expérimentations sur la partie données historiques

Comme mentionné précédemment, pour chaque cellule de la grille, un nombre de bateaux distincts ayant enregistré leurs positions est généré. Dans la figure 7.7 chaque point (cellule de la grille) représente un nombre le nombre de positions $Npos_{i,j}$ enregistrées dans la cellule $C_{i,j}$, la couleur rouge pour les cellules enregistrant le plus de positions et en blanc celles sans enregistrement. Il apparaît que la zone proche de Brest est relativement dense et les routes fréquemment utilisées

par les bateaux qui traversent la zone sont facilement identifiables à l'image du rail d'Ouessant. Les cellules candidates *Black Holes* sont celles en blanc et celles en orange avec les plus petites valeurs d'enregistrement.

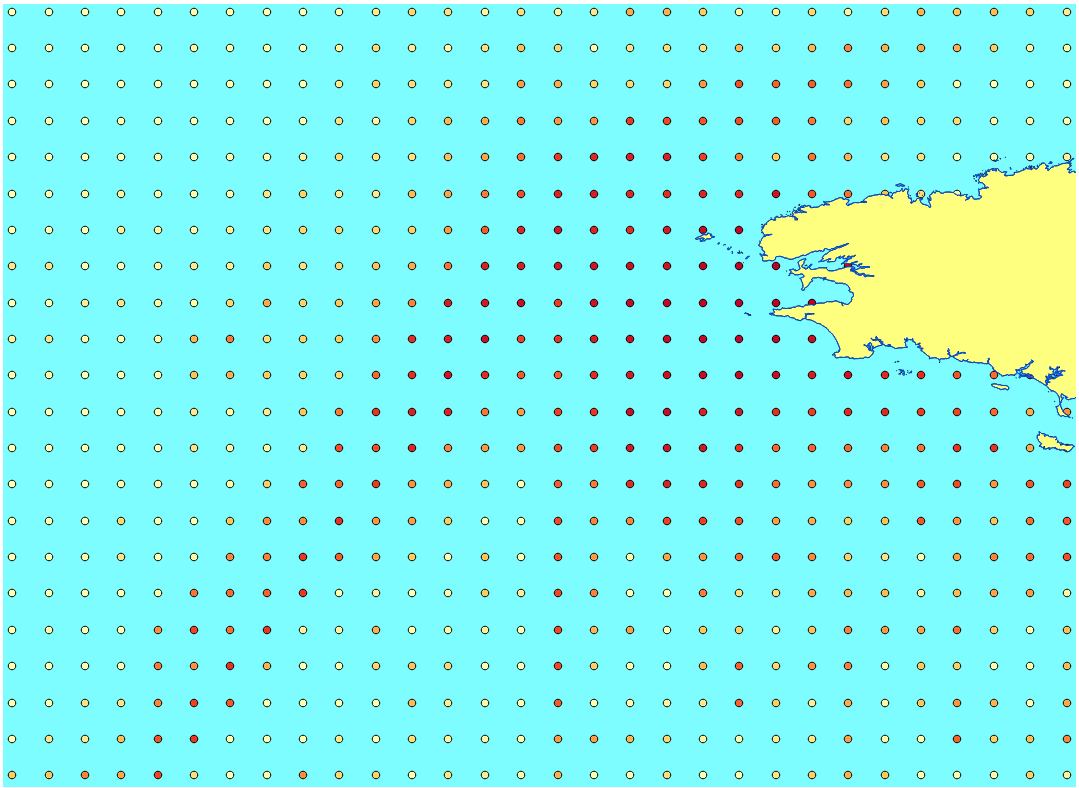

FIGURE 7.7 : Vue macroscopique de l'ensemble des positions AIS reçues

7.4.4 Fortes fluctuations de couverture et granularité

Un agrégat de l'ensemble des données a été considéré pour la partie offline, mais des agrégats intermédiaires devraient aussi être pris en compte pour détecter des tendances à l'échelle d'un jour, d'une semaine ou d'un mois pour estimer par exemple des valeurs de densité. Dans un système orienté temps-réel, les données récentes devraient avoir plus d'importance que les données plus anciennes, partant du postulat que les données plus récentes sont plus représentatives et permettent l'identification d'événements spécifiques en temps-réel. Considérons dans notre cas d'étude les candidats *Black Holes* pour deux jours successifs, chaque point correspondant là encore à une cellule $C_{i,j}$ appartenant à l'ensemble des candidats *Black Holes* (Figure 7.8).

Au vu de l'écart entre les deux jours successifs pris en exemple, les fluctuations de couverture semblent trop importantes pour considérer seulement les données les plus récentes et peuvent ne pas

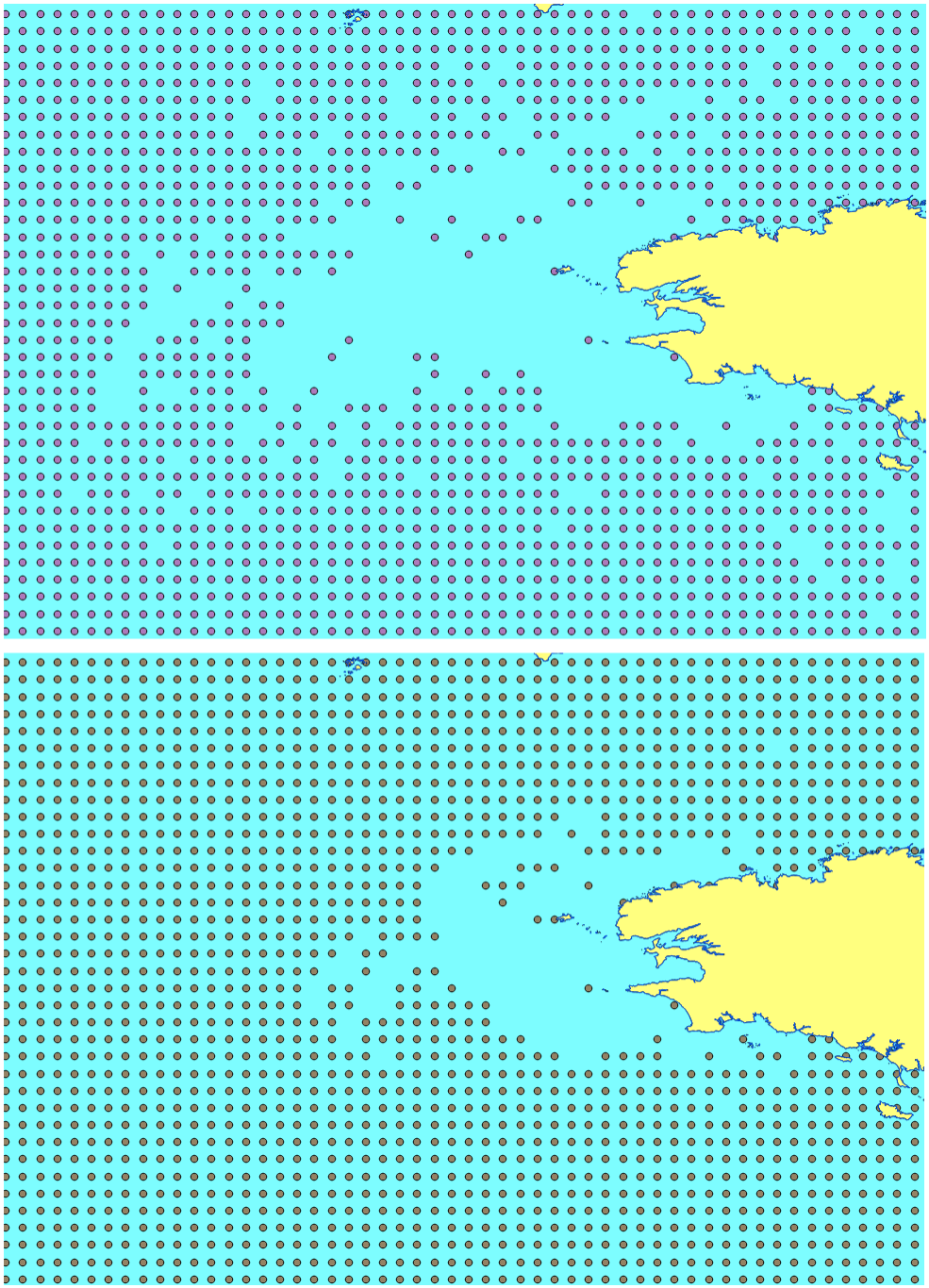

FIGURE 7.8 : Résultats expérimentaux (28 et 29 Mars 2015)

être représentatives de la couverture actuelle. Ceci est une des raisons justifiant le recours à d'autres informations additionnelles et notamment un agrégat sur l'ensemble des données plus anciennes. Cet exemple illustre parfaitement la dynamique de changement de couverture pouvant être élevée même sur des courtes périodes de temps.

7.4.5 Analyse de l'influence de la taille de la fenêtre temporelle

Les différentes observations ont montré que les changements météorologiques peuvent s'effectuer très rapidement et affecter la couverture des zones associées. La granularité temporelle considérée est d'un jour comme la limite supérieure pour la taille de la fenêtre temporelle. Trois tailles de fenêtres temporelles sont étudiées et discutées : un jour, douze heures et six heures. Pour un jour, 82% des cellules sont identifiées comme vides, 3% sont identifiées formellement comme *Black Holes* et 15% sont formellement identifiées comme couvertes. Pour une fenêtre de douze heures, de la partie temps-réel 88% des cellules sont considérées comme vides, 1% formellement identifiées comme *Black Holes* et 11% formellement identifiées comme couvertes. Enfin, pour une taille de six heures nous observons respectivement 95%, 1% et 4% de cellules vides, de cellules *Black Holes* et de cellules couvertes identifiées de manière formelle.
La figure 7.9 illustre et met en exergue la différence entre la proportion de cellules vides pour six heures (représenté par l'ensemble des triangles et des points) et une journée (seulement les triangles).

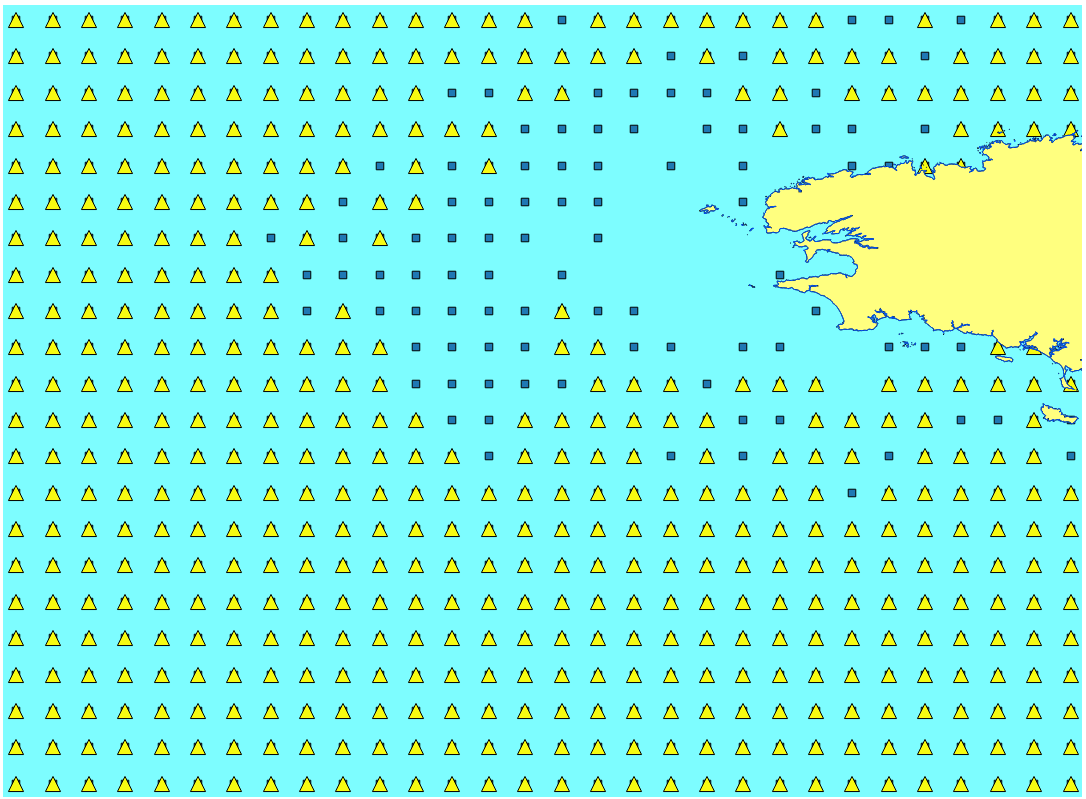

FIGURE 7.9 : Cellules vides observées pour six heures et un jour

7.4.6 Analyse de l'influence de la taille de cellules

Parmi les tests que nous avons menés, trois différentes tailles de cellules ont été considérées avec un pourcentage de candidat *Black Holes* à 0.05%. Pour un quadrillage composé de cellules de résolution 0.1*0.1 degré l'analyse de la partie temps-réel fournit approximativement 90% de cellules vides, 4% de cellules identifiées formellement comme *Black Holes* et 6% de cellules formellement couvertes. Pour un quadrillage composé de cellules de résolution 0.2*0.2 degré l'analyse de la partie temps-réel fournit approximativement 82% de cellules vides, 3% de cellules identifiées formellement comme *Black Holes* et 15% de cellules formellement couvertes. Enfin, pour un quadrillage composé de cellules de résolution 0.25*0.25 degré l'analyse de la partie temps-réel fournit approximativement 79% de cellules vides, 3% de cellules identifiées formellement comme *Black Holes* et 18% de cellules formellement couvertes. Le cas 0.1*0.1 (Figure 7.10) montre une taille de cellules qui est trop petite et génère un nombre trop important de cellules vides qu'il est difficile de discriminer par la suite en utilisant la partie offline, ce qui constitue l'un des enjeux principaux. Le cas 0.25*0.25 (Figure 7.11) est représentatif de cellules trop larges pouvant contenir à la fois des parties couvertes non couvertes en leur sein.

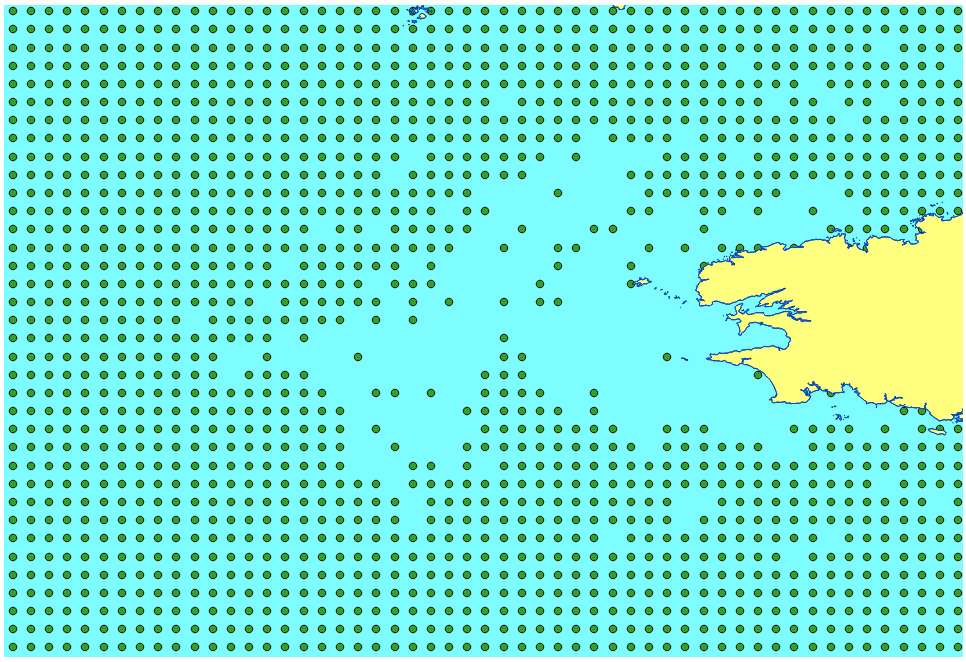

FIGURE 7.10 : Cellules vides observées pour des mailles de taille 0.1*0.1

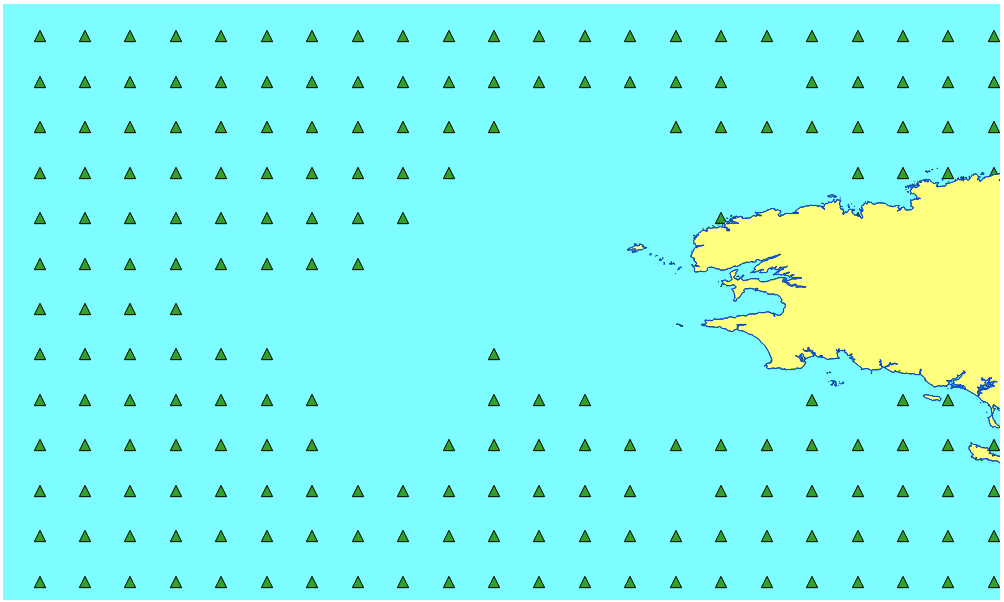

FIGURE 7.11 : Cellules vides observées pour des mailles de taille 0.25*0.25

7.4.7 Analyse des résultats et discussion

La figure 7.12 illustre quant à elle notre classification en utilisant une fenêtre temporelle de six heures pour l'identification de *Black Holes*. Les cellules vides qui sont indéfinies (i.e. vides et qu'on ne peut classifier en tant que *Black Holes* supposées ou couvertes supposées au vu des données plus anciennes) sont représentées par des triangles, tandis que les cellules supposées couvertes (identifiées à l'aide des données historiques d'un mois) sont représentées par des carrés. Les cellules formellement identifiées *Black Holes* (identifiées par des pentagones sur la carte) sont détectées, car des bateaux traversent la zone sans qu'aucune position correspondante n'ait été enregistrée pour la zone et enfin les cellules formellement couvertes correspondent aux régions proches des côtes de Brest qui ne sont représentées par aucune icône.

Ces deux figures montrent que plus la période de temps considérée est restreinte pour la fenêtre temporelle, plus le nombre de cellules vides augmente. Quelque soit le niveau de granularité choisi, notre algorithme de détection nécessite l'apport d'informations additionnelles extraites des données plus anciennes pour pouvoir distinguer des cellules qui sont vides simplement par absence de passage de bateaux de celles qui sont traversées par des navires, mais qui ne sont pas couvertes.

Nous pouvons observer que notre algorithme est très sensible notamment à la taille des cellules et la fenêtre temporelle choisies, et dans certains cas (les moins favorables à savoir pour des cellules très petites et une période temporelle faible) bon nombre de cellules sont constatées vides sur le laps de temps d'observation sans que nous puissions déterminer si celles-ci semblent être couvertes ou non.

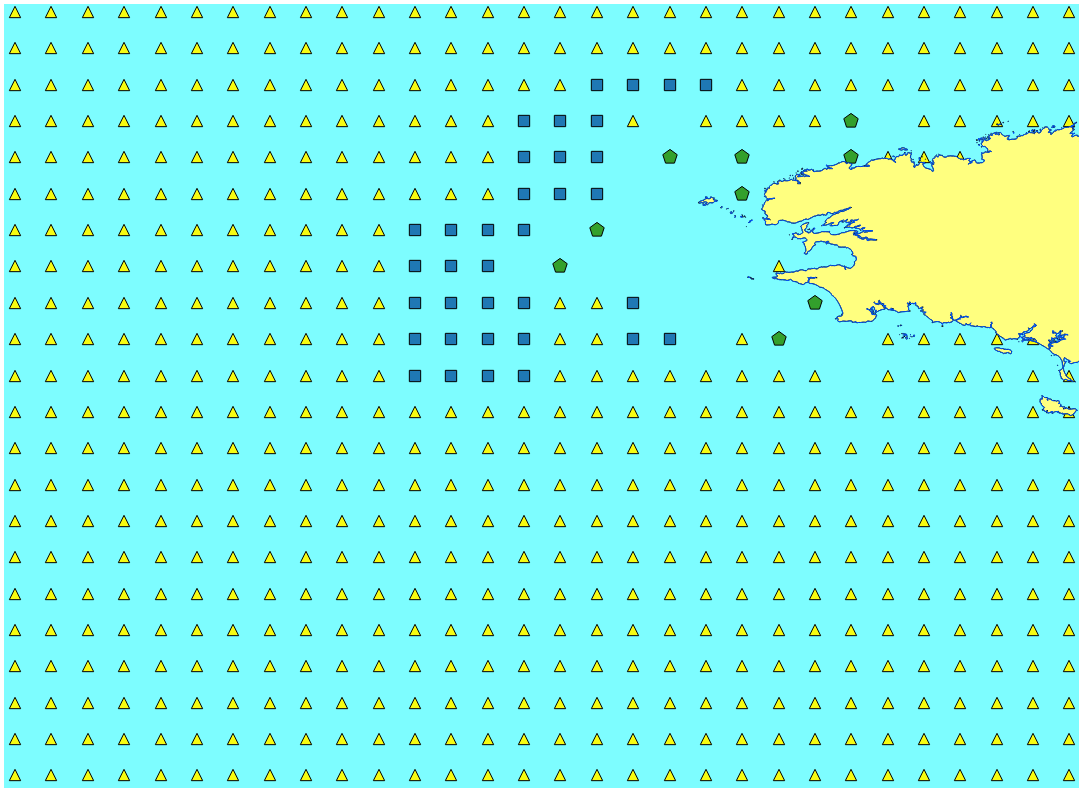

FIGURE 7.12 : Classification des cellules considérant une fenêtre temporelle de six heures et un mois de données historiques

Ceci s'explique notamment car en prenant des tailles de cellules très petites et avec le jeu de données utilisé, le trafic maritime n'est pas assez conséquent pour que chacune des cellules soient fréquentées assez régulièrement pour déterminer s'il y a bien couverture ou non. Pour s'assurer l'efficacité de notre solution, il faudrait tester notre algorithme de détections de zones de non-couverture sur un jeu de données plus dense en termes de trafic et voir si le même biais persiste ou non.

Enfin, toujours pour s'assurer de la fiabilité de notre algorithme mais aussi se servir de la connaissance de ces zones non couvertes, la mise en place d'algorithmes et mécanismes de détection de disparition de navires en mer pourrait être étudié (cf Section 5.4). En effet, quand l'identification des actuels *Black Holes* est réalisée, ces derniers peuvent être utilisés pour détecter des comportements anormaux. Si un des capteurs d'objets mobiles n'émet plus de positions, ceci peut être la cause de différents événements, soit l'objet mobile est passé dans une zone de non-couverture, soit le capteur est défaillant ou hors service et n'émet plus ou alors il y a eu perte du message. Dans le cas des navires, les messages AIS sont émis depuis les bateaux selon les règles internationales et la norme définie [(international telecommunication union), 2014]. Malheureusement, les transpondeurs responsables de l'émission de ces messages peuvent être éteints par l'équipage à bord. En l'absence

de détection ou d'identification de ces *Black Holes* il n'est pas possible de faire la différence entre des objets mobiles pénétrant dans une zone non couverte de ceux dont l'émetteur est hors service ou a été éteint.

C'est pourquoi des travaux futurs pourront concerner dans un cadre maritime l'identification d'extinction volontaire de transpondeurs AIS et la détection de faux messages. En effet, quand aucune position n'est reçue de zones dont nous savons qu'elles sont couvertes, il est raisonnable de penser que le transpondeur a été éteint de manière volontaire. Ce genre de comportements est souvent associé à des malversations comme le trafic de produits illicites, l'immigration illégale, la pêche illégale ou des activités illégales polluantes. La détection de faux messages est le cas inverse, où l'apparition de messages dans des zones dont nous savons qu'elles sont non couvertes peut poser question et mettre en doute la véracité du message reçu.

7.5 Conclusion

Ce chapitre décrit les détails de l'implémentation permettant la mise en place (partielle) de l'architecture présentée précédemment(cf Chapitre 4). Cette implémentation est appliquée à ce contexte maritime (cf Chapitre 6) pour la détection de *Black Holes* en mer en utilisant la méthode d'identification de zones de couverture (cf Chapitre 5). Ce problème pratique montre l'importance d'un processus de traitement hybride et de technologies dites "Big Data" face à la vélocité des données reçues et l'incertitude quant aux messages AIS reçus. L'approche utilisée comporte certaines limites notamment par le fait que le processus de traitement soit sensible à la taille des cellules de la grille, mais également par le fait que la fenêtre temporelle impacte fortement les résultats. Cette sensibilité à la granularité et à la localité choisies reste l'un des enjeux essentiels et communs à tout problème traitant d'objets mobiles. Les limites de notre approche sont principalement dues au fait que la densité des données et des positions reçues n'est pas suffisante si bien que nombre de cellules sont considérées comme vides sans savoir s'il s'agit d'une absence de navires dans la zone ou bien d'une absence de couverture. Des travaux futurs pourraient concerner l'utilisation de cet algorithme sur un jeu de données plus dense pour en vérifier l'efficacité. Des données météorologiques (dont la propagation du signal dépend) pourraient également être croisées avec la cartographie obtenue des zones non couvertes. Enfin, l'utilisation de différentes sources de données de positions et des processus de parallélisation spécifiques à cet algorithme restent encore à étudier.

Conclusion Générale

Cette thèse a étudié les enjeux tout comme elle a exploré des solutions pour la gestion et le traitement d'objets mobiles requérant à la fois des connaissances extraites de données historiques et des flux de données reçus en temps-réel.

Comme discuté dans le chapitre 2 des approches à la fois offline et online ont été progressivement développées pour répondre aux problématiques de volumétrie et de vélocité. Ces travaux ne permettent cependant pas de prendre en compte les dimensions spatiales et spatio-temporelles.

C'est pourquoi dans le chapitre 3 nous nous sommes intéressés aux travaux dédiés au stockage et au traitement de ces données spécifiques. Pour la partie offline, il existe des travaux dits "Big Data" qui intègrent principalement la composante spatiale, mais ne prennent pas en considération la partie temporelle, ce qui rend difficile leur capacité à stocker et traiter des données d'objets mobiles. Pour la partie dite online, la plupart des travaux actuels n'intègrent pas la dimension "Big Data", mais permettent plutôt la gestion d'objets mobiles et donc les aspects spatiaux et temporels.

Afin d'avoir une analyse fine et une compréhension à un autre niveau des déplacements d'objets mobiles, il est nécessaire de coupler des informations issues de données historiques avec des flux de données reçus en temps-réel. Les informations extraites de données archivées permettent en effet de mettre en contexte les données reçues en temps-réel. C'est la problématique que nous explorons dans le chapitre 4 en rappelant d'abord les travaux qui ont proposé un "couplage" offline/online concernant des données où la composante spatiale n'est cependant pas prise en compte. Partant de ce constat nous avons défini un modèle de gestion et de traitement des objets mobiles dans un contexte dit "Big Data". Ce modèle est défini à plusieurs niveaux : d'abord d'un point de vue architectural pour que les traitements soient distribués, mais aussi pour qu'il y ait différentes interactions entre les parties offline et online. Au-delà de cet aspect "technique" il est nécessaire de définir un modèle de données qui permette ce couplage de connaissances extraites de données historiques avec des flux de données temps-réel, notamment pour pouvoir comparer les connaissances passées avec les faits "actuels" et/ou mettre en contexte ces informations. Une définition et une taxonomie de requêtes

d'objets mobiles ont été proposées pour cette gestion d'objets mobiles à partir de ce modèle de données.

Dans le chapitre 5, une requête spécifique concernant les zones de couvertures de signal a été étudiée. Celle-ci elle nécessite le couplage de données historiques et temps-réel pour fournir une réponse "satisfaisante" et met en exergue le besoin de ces deux composantes offline et online pour répondre à une requête de type objets mobiles.

Dans le chapitre 6, nous avons vu le cas particulier du domaine maritime et l'application du modèle défini dans le chapitre 4, avec toutes les requêtes et problématiques spécifiques associées. Cette application met en avant l'aspect générique de la proposition faite précédemment tout en en montrant l'efficacité.

Dans le chapitre 7, nous présentons la façon dont le modèle proposé a été mis en place, puis a été testé sur des données de navires pour identifier les zones de couverture maritime celles-ci dépendant des conditions météorologiques. Cet exemple particulier montre la pertinence de notre approche hybride et le besoin de ce processus de traitement dans le cadre de gestion et de suivi d'objets mobiles.

Contributions

Cette thèse aborde et développe la gestion et le traitement continu de données massives d'objets mobiles, et ce, à différents niveaux. Les différents apports de notre recherche sont développés et discutés dans cette section.

Conception d'une architecture hybride dédiée au traitement d'objets mobiles.
Nous avons construit les éléments de principe pour la conception d'une architecture hybride dédiée au traitement d'objets mobiles. Cette approche permet de lier des résultats extraits de données historiques avec des flux de données reçus en temps-réel. Elle s'appuie essentiellement sur une partie temps-réel pour avoir un système réactif et la partie gestion et traitement de données massives spatio-temporelles qui permet de "mettre en contexte" et de compléter les informations traitées "à la volée". L'intérêt de cette approche est double : d'une part elle permet de limiter le temps de réponse aux différentes requêtes s'exécutant en continu, et d'autre part elle permet de répondre à des requêtes spécifiques que des systèmes temps-réels ou orientés stockage ne peuvent pas traiter à eux seuls, fournissant ainsi une meilleure compréhension des déplacements d'objets mobiles.

Modèle de données à base d'événements pour la gestion et le traitement d'objets mobiles.
Un modèle de données à base d'événements pour la gestion et le traitement d'objets mobiles a été

proposé. Celui-ci a l'avantage de faciliter le couplage entre données historiques et flux temps-réel en utilisant une grille spatiale régulière pour localiser l'ensemble des événements. Il apparaît en effet plus performant de comparer des motifs spatio-temporels repérés par des événements plutôt que de comparer une structure de résumé de données avec des flux temps-réel. La notion d'événement spatio-temporels permet de caractériser plus précisément le déplacement des objets mobiles et leurs interactions afin notamment de favoriser la fouille de données ou de la détection d'anomalies. Nous avons appliqué ce modèle et l'avons étudié dans le cadre de l'étude du déplacement de navires.

Requêtes hybrides pour la gestion de données spatio-temporelles.
Nous avons défini la notion de requête hybride pour la gestion de données spatio-temporelles dérivant de l'architecture hybride proposée précédemment avec une étude de plusieurs cas pratiques. Le premier cas étudié concerne la "disparition" d'un objet mobile dont le signal a été "perdu" en construisant une "carte des couvertures" du signal à chaque période de temps. Le second cas concerne la détection d'extinction ou de défaillance du capteur découlant de ces "cartes de couverture" et l'estimation de la position d'un objet mobile en cas de perte du signal.

Implémentation et mise en place d'un système hybride de gestion des mobilités.
Nous avons partiellement implémenté et étudié l'efficacité de notre modèle en appliquant notre processus de traitement ainsi que nos algorithmes à des données maritimes. Différentes structures spatio-temporelles et requêtes ont été implémentées et testées dans le cadre de cette approche hybride. Cette expérimentation a permis de mettre en exergue le besoin d'une approche hybride de traitement ainsi que d'observer les limites de notre algorithme de résolution concernant l'identification des zones de non-couverture.

Perspectives

Les leviers d'amélioration et de perspectives concernant les travaux présentés dans ce manuscrit sont multiples. Nous proposons une liste de ces enjeux encore à étudier, ils sont classés par thématique :

Modèle de données et notion d'événement

Une perspective de recherche attenante aux travaux de recherche présentés dans ce manuscrit concerne l'implémentation et l'utilisation de la notion d'événement pour la gestion en temps-réel d'objets mobiles. Dans le cadre de notre recherche, nous avons défini un modèle théorique d'événement pour permettre un couplage plus "facile" entre les parties online et offline, cependant celui-ci n'a pas été utilisé dans le chapitre 7 relatif à l'implémentation et l'application des algorithmes pour la

résolution des requêtes. Contrairement à d'autres travaux [Patroumpas et al., 2015] et grâce à Flink utilisé dans ces travaux de thèse, il est possible d'avoir un système gérant les flux de données entrant dans le système comme des événements plutôt que de le gérer à différents niveaux d'abstraction. L'avantage de ce paradigme d'événement est double ; il devrait permettre de fusionner connaissance de la partie offline et flux entrants, mais aussi de définir les requêtes et donner un niveau de compréhension de plus "haut niveau" d'abstraction pour mieux caractériser le déplacement d'objets mobiles.

Contexte maritime et détection de Black Holes en mer

Concernant la requête d'identification de *Black Holes* en mer il est possible de croiser des données de positions avec d'autres données (notamment météorologiques) pour en tirer une meilleure information face à l'influence de la granularité si bien temporelle que spatiale, ces dernières ayant une forte influence sur la qualité résultante des résultats. Alors que notre approche s'est principalement focalisée sur les aspects de vélocité et de véracité (avec des données qu'il faut traiter en temps-réel et en présence de données erronées ou manquantes), l'aspect variété, et donc de fusion de données, n'a pas pu être abordé alors même qu'il pourrait enrichir la qualité de notre processus de traitement. Effectuer des analyses sur d'autres jeux de données pourrait également être pertinent pour étudier la sensibilité aux granularités et localités choisies. Intégrer des données satellitaires AIS pourrait aussi (et donc non sensibles à ces problèmes de couvertures) être d'intérêt pour vérifier l'efficacité de notre algorithme de détection de non-couverture de zones. En effet, nous pourrions ainsi obtenir des données satellitaires reçues dans les zones non couvertes par les stations à terre. Enfin, l'identification de ces zones non couvertes pourrait permettre d'étudier des phénomènes attenants comme la disparition ou l'apparition d'un signal. En l'absence d'une cartographie des zones couvertes et non couvertes, il est en effet difficile de distinguer des objets mobiles ayant disparu car ces derniers ont pénétré une zone non couverte de ceux qui auraient éteint leur signal ou qui auraient un émetteur défaillant. Dans ce contexte maritime, de telles approches pourraient permettre d'identifier des navires ayant des activités illégales et qui éteignent le transpondeur à bord du navire pour ne pas être repérés.

Architecture pour le stockage et le couplage de données temps-réel et historiques.

La composante gestion et stockage de données nécessite d'être développée, notamment en ce qui concerne les mécanismes d'indexation, de couplage avec une base de données orientée colonne comme Cassandra, mais aussi la maintenance et la mise en place de concepts de "vues" pour le couplage entre les données de flux entrants et les données historiques. Il s'agit aussi bien d'étudier les mécanismes de résumé de données, que leur couplage avec des données temps-réels, et que la mise à jour de ces résumés de données pour la gestion d'objets mobiles.

Gestion de requêtes multiples et d'ordonnancement.

La problématique relative à la gestion de requêtes multiples mérite d'être développée plus en avant, la proposition d'architecture proposée devrait permettre de gérer ces mécanismes, mais n'a pas été complètement implémentée dans le cadre de cette thèse où nous nous sommes essentiellement préoccupés par la définition d'un modèle de données et d'une architecture hybride tout comme en montrer la pertinence. Tout l'enjeu ici consiste à définir des similarités entre requêtes spatio-temporelles et traiter celles-ci sur de mêmes noeuds logiques afin d'éviter d'effectuer plusieurs fois le même traitement.

Gestion de flux multiples et traitement distribué.

La problématique relative à la gestion de flux multiples nécessite d'être étudiée notamment pour évaluer si le système développé est à la fois robuste et si la qualité de réponse est la même dans un tel contexte. En effet, dans ce travail de thèse, le flux considéré est unique et les algorithmes implémentés ne prennent pas pleinement avantage des possibilités de distribution des données et de traitements. La notion de fusion de données a été principalement abordée dans cette thèse, à défaut de la parallélisation et de l'optimisation des requêtes en termes de temps de réponse.

Langage et fouille de données.

Les mécanismes relatifs au langage doivent notamment permettre à l'utilisateur de réaliser sa requête et que celle-ci soit traduite directement en deux processus : l'un portant sur les données temps-réel et l'autre sur les données historiques. La définition d'un tel langage de haut niveau peut s'accompagner de processus de traitements issus de la fouille de données ou de l'apprentissage pour permettre d'avoir une meilleure compréhension des phénomènes observés. Une idée serait d'avoir un système permettant également de faire de la fouille de données sur des données spatio-temporelles, le but n'étant plus seulement de pouvoir contrôler et observer le trafic, mais de pouvoir en faire une analyse plus profonde et complète. Une extension possible peut être l'utilisation d'une grille spatiale de référence ou de détection de motifs (i.e. utilisant des expressions régulières) pour retrouver les objets mobiles répondant à un comportement type.

List of Publications

Articles revues

Salmon, L., Ray, C., "Design principles of a stream-based framework for mobility analysis" , *Geoinformatica Volume 21, Number 2,Special issue on Geostreaming.* 25 pages, Avril 2017.

Conférences internationales

Claramunt, C. Ray, C, Camossi, E., Jousselme, A. Hadzagic, M. Andrienko, G. Andrienko, N. Theodoridis, Y. Vouros, G. Salmon, L., "Maritime data integration and analysis : recent progress and research challenges" *Proceedings of the EDBT 20th International Conference on Extending Database Technology,* Venise, Italie Mars 2017.

Ateliers dans des conférences internationales

Salmon, L., Ray, C., Claramunt, C. , "Continuous detection of black holes for moving objects at sea" , textit7th ACM Sigspatial International Workshop on Geostreaming (IWGS), 24th November ACM SIGSPATIAL International Conference on advances in Geographic Information Systems 10 pages, Novembre 2016 San Francisco États-Unis.

Salmon, L., Ray, C., Claramunt, C. , "A hybrid approach combining real-time and archived data for mobility analysis" , textit6th ACM Sigspatial International Workshop on Geostreaming (IWGS), 23rd ACM SIGSPATIAL International Conference on advances in Geographic Information Systems 6 pages, Novembre 2015, Seattle, Etats-Unis

Salmon, L., Ray, C., Claramunt, C. ,"A holistic approach combining real-time and historical data for maritime traffic monitoring" , textitACM SIGSATIAL 2015 PhD Symposium, 23rd ACM SIGSPATIAL International Conference on advances in Geographic Information Systems 4 pages, Novembre 2015, Seattle, Etats-Unis

Conférences nationales

Salmon, L. , "Une approche holistique combinant flux temps-réel et données archivées pour la France, gestion et le traitement d'objets mobiles" *Session jeunes chercheurs, Base de données 14-17 avancées 30e édition* 2 pages, Octobre 2014, Grenoble, France.

Communications

Salmon, L., Ray, C., Claramunt, C. , "Gestion et traitement simultané de flux temps-réel et archivés de données spatio-temporelles : Application à l'analyse du trafic maritime" , *Atelier Big Spatial Data, Colloque international Géomatique, une vision prospective des territoires* 6 pages, Juillet 2014, Orléans France

Bibliographie

[Abadi et al., 2003] Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Erwin, C., Galvez, E. F., Hatoun, M., Maskey, A., Rasin, A., Singer, A., Stonebraker, M., Tatbul, N., Xing, Y., Yan, R., and Zdonik, S. B. (2003). Aurora : A data stream management system. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, page 666.

[Abadi et al., 2008] Abadi, D. J., Madden, S. R., and Hachem, N. (2008). Column-stores vs. row-stores : How different are they really ? In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 967–980, New York, NY, USA. ACM.

[Aji et al., 2013] Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., and Saltz, J. (2013). Hadoop gis : A high performance spatial data warehousing system over mapreduce. pages 1009–1020.

[Akidau et al., 2013] Akidau, T., Balikov, A., Bekiroglu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., and Whittle, S. (2013). Millwheel : Fault-tolerant stream processing at internet scale. In *Very Large Data Bases*, pages 734–746.

[Akidau et al., 2015] Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R. J., Lax, R., McVeety, S., Mills, D., Perry, F., Schmidt, E., and Whittle, S. (2015). The dataflow model : A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proc. VLDB Endow.*, 8(12) :1792–1803.

[Alexandrov et al., 2014] Alexandrov, A., Bergmann, R., Ewen, S., Freytag, J., Hueske, F., Heise, A., Kao, O., Leich, M., Leser, U., Markl, V., Naumann, F., Peters, M., Rheinländer, A., Sax, M. J., Schelter, S., Höger, M., Tzoumas, K., and Warneke, D. (2014). The stratosphere platform for big data analytics. *VLDB J.*, 23(6) :939–964.

[Ali et al., 2010] Ali, M. H., Chandramouli, B., Raman, B. S., and Katibah, E. (2010). Spatio-temporal stream processing in microsoft streaminsight. *IEEE Data Eng. Bull.*, 33(2) :69–74.

[Ali et al., 2009] Ali, M. H., Gerea, C., Raman, B. S., Sezgin, B., Tarnavski, T., Verona, T., Wang, P., Zabback, P., Kirilov, A., Ananthanarayan, A., Lu, M., Raizman, A., Krishnan, R., Schindlauer,

R., Grabs, T., Bjeletich, S., Chandramouli, B., Goldstein, J., Bhat, S., Li, Y., Nicola, V. D., Wang, X., Maier, D., Santos, I., Nano, O., and Grell, S. (2009). Microsoft CEP server and online behavioral targeting. *PVLDB*, 2(2) :1558–1561.

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11) :832–843.

[Anselin, 1989] Anselin, L. (1989). What is special about spatial data ? alternative perspectives on spatial data analysis. pages 63–77.

[Arasu et al., 2004] Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J. (2004). Stream : The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab.

[Arasu et al., 2016] Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J. (2016). *STREAM : The Stanford Data Stream Management System*, pages 317–336. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Armbrust et al., 2015] Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., and Zaharia, M. (2015). Spark sql : Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1383–1394, New York, NY, USA. ACM.

[Avnur and Hellerstein, 2000] Avnur, R. and Hellerstein, J. M. (2000). Eddies : Continuously adaptive query processing. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 261–272.

[Baas, 2012] Baas, B. (2012). Nosql spatial : Neo4j versus postgis.

[Babcock et al., 2004] Babcock, B., Datar, M., and Motwani, R. (2004). Load shedding for aggregation queries over data streams. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, pages 350–, Washington, DC, USA. IEEE Computer Society.

[Balazinska et al., 2007] Balazinska, M., Kwon, Y., Kuchta, N., and Lee, D. (2007). Moirae : History-enhanced monitoring. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pages 375–386.

[Barber et al., 1996] Barber, C. B., Dobkin, D. P., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4) :469–483.

[Beckmann et al., 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The r*-tree : An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, pages 322–331, New York, NY, USA. ACM.

[Biem et al., 2010] Biem, A., Bouillet, E., Feng, H., Ranganathan, A., Riabov, A., Verscheure, O., Koutsopoulos, H. N., and Moran, C. (2010). IBM infosphere streams for scalable, real-time,

intelligent transportation services. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 1093–1104.

[Boykin et al., 2014] Boykin, P. O., Ritchie, S., O'Connell, I., and Lin, J. (2014). Summingbird : A framework for integrating batch and online mapreduce computations. *PVLDB*, 7(13) :1441–1451.

[Brakatsoulas et al., 2004] Brakatsoulas, S., Pfoser, D., and Tryfona, N. (2004). Modeling, storing, and mining moving object databases. In *8th International Database Engineering and Applications Symposium (IDEAS 2004), 7-9 July 2004, Coimbra, Portugal*, pages 68–77.

[Cangialosi et al., 2005] Cangialosi, F. J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A. S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., and Zdonik, S. (2005). The Design of the Borealis Stream Processing Engine. In *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, Asilomar, CA.

[Carbone et al., 2016] Carbone, Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., and Tzoumas, K. (2016). Apache flink tm : Stream and batch processing in a single engine paris.

[Cattell, 2011] Cattell, R. (2011). Scalable sql and nosql data stores. *SIGMOD Rec.*, 39(4) :12–27.

[Chakka et al., 2003] Chakka, V. P., Everspaugh, A., and Patel, J. M. (2003). Indexing large trajectory data sets with SETI. In *CIDR 2003, First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*.

[Chandramouli et al., 2016] Chandramouli, B., Fernandez, R. C., Goldstein, J., Eldawy, A., and Quamar, A. (2016). Quill : Efficient, transferable, and rich analytics at scale. *Proc. VLDB Endow.*, 9(14) :1623–1634.

[Chandramouli et al., 2014] Chandramouli, B., Goldstein, J., Barnett, M., DeLine, R., Fisher, D., Platt, J. C., Terwilliger, J. F., and Wernsing, J. (2014). Trill : A high-performance incremental query processor for diverse analytics. *Proc. VLDB Endow.*, 8(4) :401–412.

[Chandrasekaran et al., 2003] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Reiss, F., and Shah, M. A. (2003). Telegraphcq : Continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, page 668.

[Chandrasekaran and Franklin, 2004] Chandrasekaran, S. and Franklin, M. (2004). Remembrance of streams past : Overload-sensitive management of archived streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, VLDB '04, pages 348–359.

[Chandrasekaran and Franklin, 2003] Chandrasekaran, S. and Franklin, M. J. (2003). Psoup : a system for streaming queries over streaming data. *VLDB J.*, 12(2) :140–156.

[Chang et al., 2006] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2006). Bigtable : A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 15–15, Berkeley, CA, USA. USENIX Association.

[Cheatham et al., 1995] Cheatham, T., Fahmy, A. F., Stefanescu, D. C., and Valiant, L. G. (1995). Bulk synchronous parallel computing-a paradigm for transportable software. In *28th Annual Hawaii International Conference on System Sciences (HICSS-28), January 3-6, 1995, Kihei, Maui, Hawaii, USA*, pages 268–275.

[Claramunt, 1998] Claramunt, C. (1998). *A Spatial View Model for a Flexible Representation of Geographical Data*. Theses, Universite de Bourgogne.

[Claramunt et al., 2017] Claramunt, C., Ray, C., Camossi, E., Jousselme, A., Hadzagic, M., Andrienko, G. L., Andrienko, N. V., Theodoridis, Y., Vouros, G. A., and Salmon, L. (2017). Maritime data integration and analysis : recent progress and research challenges. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017.*, pages 192–197.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM*, 13(6) :377–387.

[Cohn et al., 1997] Cohn, A. G., Bennett, B., Gooday, J., and Gotts, N. M. (1997). Qualitative spatial representation and reasoning with the region connection calculus. In *PROCEEDINGS OF THE DIMACS INTERNATIONAL WORKSHOP ON GRAPH DRAWING, 1994. LECTURE NOTES IN COMPUTER SCIENCE*, pages 89–4.

[Condie et al., 2010] Condie, T., Conway, N., Alvaro, P., Hellerstein, J. M., Elmeleegy, K., and Sears, R. (2010). Mapreduce online. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 21–21, Berkeley, CA, USA. USENIX Association.

[de Almeida et al., 2006] de Almeida, V. T., Guting, R. H., and Behr, T. (2006). Querying moving objects in secondo. In *Proceedings of the 7th International Conference on Mobile Data Management*, MDM '06, pages 47–52. IEEE Computer Society.

[De Francisci Morales and Bifet, 2015] De Francisci Morales, G. and Bifet, A. (2015). Samoa : Scalable advanced massive online analysis. *J. Mach. Learn. Res.*, 16(1) :149–153.

[de Souza Baptista et al., 2014] de Souza Baptista, C., Pires, C. E. S., Leite, D. F. B., and de Oliveiraa, M. G. (2014). Nosql geographic databases : An overview. *Geographical Information Systems : Trends and Technologies*, page 73.

[Dean and Ghemawat, 2004] Dean, J. and Ghemawat, S. (2004). Mapreduce : Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting*

Systems Design & Implementation - Volume 6, OSDI'04, pages 10–10, Berkeley, CA, USA. USENIX Association.

[DeCandia et al., 2007] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo : Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6) :205–220.

[Deng et al., 2011] Deng, K., Xie, K., Zheng, K., and Zhou, X. (2011). Trajectory indexing and retrieval. In *Computing with Spatial Trajectories*, pages 35–60.

[Deshpande et al., 2007] Deshpande, A., Ives, Z., and Raman, V. (2007). Adaptive query processing. *Found. Trends databases*, 1(1) :1–140.

[Devogele et al., 2013] Devogele, T., Etienne, L., and Ray, C. (2013). Maritime monitoring. In *Mobility Data : Modeling, Management, and Understanding*, pages 221–239.

[Diestel, 2012] Diestel, R. (2012). *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.

[Dindar et al., 2009] Dindar, N., Güç, B., Lau, P., Özal, A., Soner, M., and Tatbul, N. (2009). Dejavu : declarative pattern matching over live and archived streams of events. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 1023–1026.

[Doulkeridis and Nørvåg, 2014] Doulkeridis, C. and Nørvåg, K. (2014). A survey of large-scale analytical query processing in mapreduce. *VLDB J.*, 23(3) :355–380.

[Duggan et al., 2015] Duggan, J., Elmore, A. J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., Madden, S., Maier, D., Mattson, T., and Zdonik, S. (2015). The bigdawg polystore system. *SIGMOD Rec.*, 44(2) :11–16.

[Egenhofer, 1991] Egenhofer, M. J. (1991). Reasoning about binary topological relations. In *Proceedings of the Second International Symposium on Advances in Spatial Databases*, SSD '91, pages 143–160, London, UK, UK. Springer-Verlag.

[Eldawy et al., 2015] Eldawy, A., Alarabi, L., and Mokbel, M. F. (2015). Spatial partitioning techniques in spatial hadoop. *PVLDB*, 8(12) :1602–1605.

[Eldawy et al., 2013] Eldawy, A., Li, Y., Mokbel, M. F., and Janardan, R. (2013). Cg_hadoop : computational geometry in mapreduce. In *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*, pages 284–293.

[Eldawy and Mokbel, 2013] Eldawy, A. and Mokbel, M. F. (2013). A demonstration of spatialhadoop : An efficient mapreduce framework for spatial data. *PVLDB*, pages 1230–1233.

[Eldawy and Mokbel, 2014] Eldawy, A. and Mokbel, M. F. (2014). Pigeon : A spatial mapreduce language. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1242–1245.

[Eldawy and Mokbel, 2015] Eldawy, A. and Mokbel, M. F. (2015). The era of big spatial data. In *31st IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2015, Seoul, South Korea, April 13-17, 2015*, pages 42–49.

[Elmongui et al., 2005] Elmongui, H. G., Mokbel, M. F., and Aref, W. G. (2005). Spatio-temporal histograms. In *Advances in Spatial and Temporal Databases, 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005, Proceedings*, pages 19–36.

[Etienne et al., 2010] Etienne, L., Devogele, T., and Bouju, A. (2010). Spatio-temporal trajectory analysis of mobile objects following the same itinerary. In *Joint International Conference on Theory, Data Handling and Modelling in GeoSpatial Information Science*, pages pp 86–91, Hong Kong, Hong Kong SAR China.

[Etienne et al., 2012] Etienne, L., Devogele, T., and Bouju, A. (2012). Spatio-temporal trajectory analysis of mobile objects following the same itinerary. *Advances in Geo-Spatial Information Science*, 10 :47–57.

[Ewen et al., 2013] Ewen, S., Schelter, S., Tzoumas, K., Warneke, D., and Markl, V. (2013). Iterative parallel data processing with stratosphere : an inside look. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 1053–1056.

[Ewen et al., 2012] Ewen, S., Tzoumas, K., Kaufmann, M., and Markl, V. (2012). Spinning fast iterative data flows. *CoRR*, abs/1208.0088.

[Fernandez et al., 2015] Fernandez, R. C., Pietzuch, P. R., Kreps, J., Narkhede, N., Rao, J., Koshy, J., Lin, D., Riccomini, C., and Wang, G. (2015). Liquid : Unifying nearline and offline big data integration. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*.

[Forlizzi et al., 1999] Forlizzi, L., Güting, R. H., Nardelli, E., and Schneider, M. (1999). A data model and data structures for moving objects databases. pages 319–330.

[Foster and Karonis, 1998] Foster, I. and Karonis, N. T. (1998). A grid-enabled mpi : Message passing in heterogeneous distributed computing systems. In *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*, SC '98, pages 1–11, Washington, DC, USA. IEEE Computer Society.

[Fox et al., 2013] Fox, A. D., Eichelberger, C. N., Hughes, J. N., and Lyon, S. (2013). Spatio-temporal indexing in non-relational distributed databases. In *Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA*, pages 291–299.

[Galic et al., 2014] Galic, Z., Baranovic, M., Krizanovic, K., and Meskovic, E. (2014). Geospatial data streams : Formal framework and implementation. *Data Knowl. Eng.*, 91 :1–16.

[Garzó et al., 2013] Garzó, A., Benczúr, A. A., Sidló, C. I., Tahara, D., and Wyatt, E. F. (2013). Real-time streaming mobility analytics. In Hu, X., Lin, T. Y., Raghavan, V., Wah, B. W., Baeza-Yates, R. A., Fox, G., Shahabi, C., Smith, M., 0001, Q. Y., Ghani, R., Fan, W., Lempel, R., and Nambiar, R., editors, *BigData Conference*, pages 697–702. IEEE.

[Ghanem et al., 2006] Ghanem, T. M., Aref, W. G., and Elmagarmid, A. K. (2006). Exploiting predicate-window semantics over data streams. *SIGMOD Record*, 35(1) :3–8.

[Ghanem et al., 2010] Ghanem, T. M., Elmagarmid, A. K., Larson, P., and Aref, W. G. (2010). Supporting views in data stream management systems. *ACM Trans. Database Syst.*, 35(1).

[Ghanem et al., 2007] Ghanem, T. M., Hammad, M. A., Mokbel, M. F., Aref, W. G., and Elmagarmid, A. K. (2007). Incremental evaluation of sliding-window queries over data streams. *IEEE Trans. Knowl. Data Eng.*, 19(1) :57–72.

[Ghemawat et al., 2003] Ghemawat, S., Gobioff, H., and Leung, S.-T. (2003). The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 29–43, New York, NY, USA. ACM.

[Giannotti et al., 2011] Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F., Renso, C., Rinzivillo, S., and Trasarti, R. (2011). Unveiling the complexity of human mobility by querying and mining massive trajectory data. *The VLDB Journal*, 20(5) :695–719.

[Giannotti et al., 2007] Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. (2007). Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 330–339. ACM.

[Gilbert and Lynch, 2002] Gilbert, S. and Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2) :51–59.

[Golab, 2006] Golab, L. (2006). Sliding window query processing over data streams. Technical report.

[Golab and Johnson, 2013] Golab, L. and Johnson, T. (2013). Data stream warehousing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 949–952.

[Golab and Özsu, 2003] Golab, L. and Özsu, M. T. (2003). Issues in data stream management. *SIGMOD Rec.*, pages 5–14.

[Güting et al., 2000] Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1) :1–42.

[Guttman, 1984] Guttman, A. (1984). R-trees : A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, pages 47–57, New York, NY, USA. ACM.

[Hadjieleftheriou et al., 2003] Hadjieleftheriou, M., Kollios, G., Gunopulos, D., and Tsotras, V. J. (2003). On-line discovery of dense areas in spatio-temporal databases. In *Advances in Spatial and Temporal Databases, 8th International Symposium, SSTD 2003, Santorini Island, Greece, July 24-27, 2003, Proceedings*, pages 306–324.

[Hagedorn et al., 2017] Hagedorn, S., Götze, P., and Sattler, K. (2017). The STARK framework for spatio-temporal data analytics on spark. In *Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Proceedings*, pages 123–142.

[Hagedorn and Sattler, 2016] Hagedorn, S. and Sattler, K.-U. (2016). Piglet : Interactive and platform transparent analytics for rdf &#38 ; dynamic data. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 187–190, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Hammad et al., 2008] Hammad, M. A., Aref, W. G., and Elmagarmid, A. K. (2008). Query processing of multi-way stream window joins. *VLDB J.*, 17(3) :469–488.

[Hammad et al., 2003] Hammad, M. A., Franklin, M. J., Aref, W. G., and Elmagarmid, A. K. (2003). Scheduling for shared window joins over data streams. In *VLDB*, pages 297–308.

[Hammad et al., 2004] Hammad, M. A., Mokbel, M. F., Ali, M. H., Aref, W. G., Catlin, A. C., Elmagarmid, A. K., Eltabakh, M. Y., Elfeky, M. G., Ghanem, T. M., Gwadera, R., Ilyas, I. F., Marzouk, M. S., and Xiong, X. (2004). Nile : A query processing engine for data streams. In *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA*, page 851.

[Han et al., 2000] Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 1–12.

[Hao et al., 2008] Hao, X., Meng, X., and Xu, J. (2008). Continuous density queries for moving objects. In *Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access, Mobide 2008, June 13, 2008, Vancouver, British Columbia, Canada, Proceedings*, pages 1–7.

[Hewitt et al., 1973] Hewitt, C., Bishop, P., and Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, pages 235–245, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Hindman et al., 2011] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R., Shenker, S., and Stoica, I. (2011). Mesos : A platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 295–308, Berkeley, CA, USA. USENIX Association.

[Hong et al., 2015] Hong, L., Zheng, Y., Yung, D., Shang, J., and Zou, L. (2015). Detecting urban black holes based on human mobility data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 35. ACM.

[Huang and Zhang, 2008] Huang, Y. and Zhang, C. (2008). New data types and operations to support geo-streams. In *Geographic Information Science, 5th International Conference, GIScience 2008, Park City, UT, USA, September 23-26, 2008. Proceedings*, pages 106–118.

[Huang and Zhang, 2009] Huang, Y. and Zhang, C. (2009). Interval-based nearest neighbor queries over sliding windows from trajectory data. In *MDM 2009, Tenth International Conference on Mobile Data Management, Taipei, Taiwan, 18-20 May 2009*, pages 212–221.

[Hueske et al., 2013] Hueske, F., Krettek, A., and Tzoumas, K. (2013). Enabling operator reordering in data flow programs through static code analysis. *CoRR*, abs/1301.4200.

[Hueske et al., 2012] Hueske, F., Peters, M., Sax, M., Rheinländer, A., Bergmann, R., Krettek, A., and Tzoumas, K. (2012). Opening the black boxes in data flow optimization. *CoRR*, abs/1208.0087.

[Hunt et al., 2010] Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. (2010). Zookeeper : Wait-free coordination for internet-scale systems. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 11–11, Berkeley, CA, USA. USENIX Association.

[(international telecommunication union), 2014] (international telecommunication union), I. (2014). Technical characteristics for an automatic identification system using time division multiple access in the vhf maritime mobile frequency band.

[Isard et al., 2007] Isard, M., Budiu, M., Yu, Y., Birrell, A., and Fetterly, D. (2007). Dryad : Distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 59–72, New York, NY, USA. ACM.

[Jacox and Samet, 2007] Jacox, E. H. and Samet, H. (2007). Spatial join techniques. *ACM Trans. Database Syst.*, 32(1).

[Jensen et al., 2006] Jensen, C. S., Lin, D., Ooi, B. C., and Zhang, R. (2006). Effective density queries on continuouslymoving objects. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 71.

[Johnson and Shkapenyuk, 2015] Johnson, T. and Shkapenyuk, V. (2015). Data stream warehousing in tidalrace. In *CIDR*.

[Kalashnikov et al., 2002] Kalashnikov, D. V., Prabhakar, S., Hambrusch, S. E., and Aref, W. G. (2002). Efficient evaluation of continuous range queries on moving objects. In *Database and Expert Systems Applications, 13th International Conference, DEXA 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings*, pages 731–740.

[Kallman et al., 2008] Kallman, R., Kimura, H., Natkins, J., Pavlo, A., Rasin, A., Zdonik, S. B., Jones, E. P. C., Madden, S., Stonebraker, M., Zhang, Y., Hugg, J., and Abadi, D. J. (2008). H-store : a high-performance, distributed main memory transaction processing system. *PVLDB*, 1 :1496–1499.

[Kazemitabar et al., 2010] Kazemitabar, S. J., Demiryurek, U., Ali, M. H., Akdogan, A., and Shahabi, C. (2010). Geospatial stream query processing using microsoft SQL server streaminsight. *PVLDB*, 3(2) :1537–1540.

[Khandekar et al., 2009] Khandekar, R., Hildrum, K., Parekh, S., Rajan, D., Wolf, J. L., Wu, K., Andrade, H., and Gedik, B. (2009). COLA : optimizing stream processing applications via graph partitioning. In *Middleware 2009, ACM/IFIP/USENIX, 10th International Middleware Conference, Urbana, IL, USA, November 30 - December 4, 2009. Proceedings*, pages 308–327.

[Kreps, 2014] Kreps, J. (2014). “questioning the lambda architecture”.

[Kreps et al., 2011] Kreps, J., Narkhede, N., and Rao, J. (2011). Kafka : a distributed messaging system for log processing.

[Kulkarni et al., 2015] Kulkarni, S., Bhagat, N., Fu, M., Kedigehalli, V., Kellogg, C., Mittal, S., Patel, J. M., Ramasamy, K., and Taneja, S. (2015). Twitter heron : Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’15, pages 239–250, New York, NY, USA. ACM.

[Lakshman and Malik, 2010] Lakshman, A. and Malik, P. (2010). Cassandra : A decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 44(2) :35–40.

[Lam et al., 2012] Lam, W., Liu, L., Prasad, S., Rajaraman, A., Vacheri, Z., and Doan, A. (2012). Muppet : Mapreduce-style processing of fast data. *Proc. VLDB Endow.*, 5(12) :1814–1825.

[Landset et al., 2015] Landset, S., Khoshgoftaar, T. M., Richter, A. N., and Hasanin, T. (2015). A survey of open source tools for machine learning with big data in the hadoop ecosystem. *Journal of Big Data*, 2(1) :24.

[Laxhammar, 2008] Laxhammar, R. (2008). Anomaly detection for sea surveillance. In *2008 11th International Conference on Information Fusion*, pages 1–8.

[Le Guyader et al., 2016] Le Guyader, D., Ray, C., and Brosset, D. (2016). Defining fishing grounds variability with Automatic Identification System (AIS). In *2nd International Workshop on Maritime Flows and Networks (WIMAKS’16)*, page 96, Paris, France.

[Li et al., 2012] Li, B., Mazur, E., Diao, Y., McGregor, A., and Shenoy, P. (2012). Scalla : A platform for scalable one-pass analytics using mapreduce. *ACM Trans. Database Syst.*, 37(4) :27 :1–27 :43.

[Li, 2014] Li, Z. (2014). Spatiotemporal pattern mining : Algorithms and applications. In *Frequent Pattern Mining*, pages 283–306.

[Loglisci and Malerba, 2014] Loglisci, C. and Malerba, D. (2014). Mining dense regions from vehicular mobility in streaming setting. In *Foundations of Intelligent Systems - 21st International Symposium, ISMIS 2014, Roskilde, Denmark, June 25-27, 2014. Proceedings*, pages 40–49.

[Lu and Güting, 2013] Lu, J. and Güting, R. H. (2013). Parallel SECONDO : practical and efficient mobility data processing in the cloud. In *Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA*, pages 17–25.

[Lu and Güting, 2013] Lu, J. and Güting, R. H. (2013). Parallel secondo : Practical and efficient mobility data processing in the cloud. In *BigData Conference*, pages 17–25. IEEE.

[Ma et al., 2009] Ma, Q., Yang, B., Qian, W., and Zhou, A. (2009). Query processing of massive trajectory data based on mapreduce. In *Proceedings of the First International Workshop on Cloud Data Management*, CloudDB '09, pages 9–16, New York, NY, USA. ACM.

[Malewicz et al., 2010] Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G. (2010). Pregel : A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA. ACM.

[Marz, 2013] Marz, N. (2013). *Big data : principles and best practices of scalable realtime data systems*. O'Reilly Media, [S.l.].

[Meehan et al., 2015] Meehan, J., Tatbul, N., Zdonik, S., Aslantas, C., Cetintemel, U., Du, J., Kraska, T., Madden, S., Maier, D., Pavlo, A., Stonebraker, M., Tufte, K., and Wang, H. (2015). S-store : Streaming meets transaction processing. *Proc. VLDB Endow.*, 8(13) :2134–2145.

[Meskovic et al., 2014] Meskovic, E., Osmanovic, D., Galic, Z., and Baranovic, M. (2014). Generating spatio-temporal streaming trajectories. In *37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014, Opatija, Croatia, May 26-30, 2014*, pages 1130–1135.

[Miller et al., 2011] Miller, J., Raymond, M., Archer, J., Adem, S., Hansel, L., Konda, S., Luti, M., Zhao, Y., Teredesai, A., and Ali, M. H. (2011). An extensibility approach for spatio-temporal stream processing using microsoft streaminsight. In *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings*, pages 496–501.

[Mokbel and Aref, 2005] Mokbel, M. F. and Aref, W. G. (2005). Gpac : Generic and progressive processing of mobile queries over mobile data. In *Proceedings of the 6th International Conference on Mobile Data Management*, MDM '05, pages 155–163, New York, NY, USA. ACM.

[Mokbel and Aref, 2008] Mokbel, M. F. and Aref, W. G. (2008). SOLE : scalable on-line execution of continuous queries on spatio-temporal data streams. *VLDB J.*, 17(5) :971–995.

[Mokbel et al., 2004] Mokbel, M. F., Xiong, X., and Aref, W. G. (2004). Sina : Scalable incremental processing of continuous queries in spatio-temporal databases. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 623–634, New York, NY, USA. ACM.

[Mokbel et al., 2005] Mokbel, M. F., Xiong, X., Hammad, M. A., and Aref, W. G. (2005). Continuous query processing of spatio-temporal data streams in place. *Geoinformatica*, pages 343–365.

[Morzy, 2007] Morzy, M. (2007). Mining frequent trajectories of moving objects for location prediction. In *Machine Learning and Data Mining in Pattern Recognition, 5th International Conference, MLDM 2007, Leipzig, Germany, July 18-20, 2007, Proceedings*, pages 667–680.

[Mouza and Rigaux, 2005] Mouza, C. and Rigaux, P. (2005). Mobility patterns. *Geoinformatica*, 9(4) :297–319.

[Nathan et al., 2008] Nathan, R., Getz, W. M., Revilla, E., Holyoak, M., Kadmon, R., Saltz, D., and Smouse, P. E. (2008). A movement ecology paradigm for unifying organismal movement research. *Proceedings of the National Academy of Sciences*, 105(49) :19052–19059.

[Nehme and Rundensteiner, 2006] Nehme, R. V. and Rundensteiner, E. A. (2006). SCUBA : scalable cluster-based algorithm for evaluating continuous spatio-temporal queries on moving objects. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, pages 1001–1019.

[Nehme and Rundensteiner, 2007] Nehme, R. V. and Rundensteiner, E. A. (2007). *ClusterSheddy* : Load shedding using moving clusters over spatio-temporal data streams. In *Advances in Databases : Concepts, Systems and Applications, 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007, Proceedings*, pages 637–651.

[Neumeyer et al., 2010] Neumeyer, L., Robbins, B., Nair, A., and Kesari, A. (2010). S4 : Distributed stream computing platform. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 170–177, Washington, DC, USA. IEEE Computer Society.

[Nguyen-Dinh et al., 2010] Nguyen-Dinh, L.-V., Aref, W. G., and Mokbel, M. F. (2010). Spatio-temporal access methods : Part 2 (2003 - 2010). *IEEE Data Eng. Bull.*, 33(2) :46–55.

[Ni and Ravishankar, 2007] Ni, J. and Ravishankar, C. V. (2007). Pointwise-dense region queries in spatio-temporal databases. In *Proceedings of the 23rd International Conference on Data*

Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007, pages 1066–1075.

[Nidzwetzki and Güting, 2015] Nidzwetzki, J. K. and Güting, R. H. (2015). Distributed SECONDO : A highly available and scalable system for spatial data processing. In *Advances in Spatial and Temporal Databases - 14th International Symposium, SSTD 2015, Hong Kong, China, August 26-28, 2015. Proceedings*, pages 491–496.

[Nikitopoulos et al., 2018] Nikitopoulos, P., Vlachou, A., Doulkeridis, C., and Vouros, G. A. (2018). Distrdf : Distributed spatio-temporal RDF queries on spark. In *Proceedings of the Workshops of the EDBT/ICDT 2018 Joint Conference (EDBT/ICDT 2018), Vienna, Austria, March 26, 2018.*, pages 125–132.

[Nishimura et al., 2011] Nishimura, S., Das, S., Agrawal, D., and Abbadi, A. E. (2011). Md-hbase : A scalable multi-dimensional data infrastructure for location aware services. In *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management - Volume 01*, MDM '11, pages 7–16, Washington, DC, USA. IEEE Computer Society.

[Noghabi et al., 2017] Noghabi, S. A., Paramasivam, K., Pan, Y., Ramesh, N., Bringhurst, J., Gupta, I., and Campbell, R. H. (2017). Samza : Stateful scalable stream processing at linkedin. *Proc. VLDB Endow.*, 10(12) :1634–1645.

[Olston et al., 2011] Olston, C., Chiou, G., Chitnis, L., Liu, F., Han, Y., Larsson, M., Neumann, A., Rao, V. B., Sankarasubramanian, V., Seth, S., Tian, C., ZiCornell, T., and Wang, X. (2011). Nova : Continuous pig/hadoop workflows. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1081–1090, New York, NY, USA. ACM.

[Olston et al., 2008] Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. (2008). Pig latin : A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1099–1110, New York, NY, USA. ACM.

[Owen et al., 2011] Owen, S., Anil, R., Dunning, T., and Friedman, E. (2011). *Mahout in Action*. Manning Publications Co., Greenwich, CT, USA.

[Ozsu, 2007] Ozsu, M. T. (2007). *Principles of Distributed Database Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

[Pallotta et al., 2013] Pallotta, G., Vespe, M., and Bryan, K. (2013). Vessel pattern knowledge discovery from AIS data : A framework for anomaly detection and route prediction. *Entropy*, 15(6) :2218–2245.

[Patroumpas, 2013] Patroumpas, K. (2013). Multi-scale window specification over streaming trajectories. *J. Spatial Information Science*, pages 45–75.

[Patroumpas et al., 2015] Patroumpas, K., Artikis, A., Katzouris, N., Vodas, M., Theodoridis, Y., and Pelekis, N. (2015). Event recognition for maritime surveillance. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 629–640.

[Patroumpas and Sellis, 2004] Patroumpas, K. and Sellis, T. K. (2004). Managing trajectories of moving objects as data streams. In *Spatio-Temporal Database Management, 2nd International Workshop STDBM'04, Toronto, Canada, August 30, 2004*, pages 41–48.

[Patroumpas and Sellis, 2010] Patroumpas, K. and Sellis, T. K. (2010). Multi-granular time-based sliding windows over data streams. In *TIME 2010 - 17th International Symposium on Temporal Representation and Reasoning, Paris, France, 6-8 September 2010*, pages 146–153.

[Patroumpas and Sellis, 2011] Patroumpas, K. and Sellis, T. K. (2011). Subsuming multiple sliding windows for shared stream computation. In *Advances in Databases and Information Systems - 15th International Conference, ADBIS 2011, Vienna, Austria, September 20-23, 2011. Proceedings*, pages 56–69.

[Pelekis et al., 2008] Pelekis, N., Frentzos, E., Giatrakos, N., and Theodoridis, Y. (2008). Hermes : Aggregative lbs via a trajectory db engine. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1255–1258, New York, NY, USA. ACM.

[Pelekis et al., 2006] Pelekis, N., Theodoridis, Y., Vosinakis, S., and Panayiotopoulos, T. (2006). Hermes - a framework for location-based data management. In *In Proceedings of EDBT*, pages 1130–1134.

[Peng et al., 2011] Peng, S., Li, Z., Li, Q., Chen, Q., Pan, W., Liu, H., and Nie, Y. (2011). Event detection over live and archived streams. In Wang, H., Li, S., Oyama, S., Hu, X., and Qian, T., editors, *Web-Age Information Management*, pages 566–577, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Pfoser et al., 2000] Pfoser, D., Jensen, C. S., and Theodoridis, Y. (2000). Novel approaches in query processing for moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 395–406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Piciarelli and Foresti, 2006] Piciarelli, C. and Foresti, G. L. (2006). On-line trajectory clustering for anomalous events detection. *Pattern Recogn. Lett.*, 27(15) :1835–1842.

[Pitsikalis et al., 2018] Pitsikalis, M., Kontopoulos, I., Artikis, A., Alevizos, E., Delaunay, P., Pouessel, J., Dreo, R., Ray, C., Camossi, E., Jousselme, A., and Hadzagic, M. (2018). Composite event patterns for maritime monitoring. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence, SETN 2018, Patras, Greece, July 09-12, 2018*, pages 29 :1–29 :4.

[Potamias et al., 2006] Potamias, M., Patroumpas, K., and Sellis, T. K. (2006). Sampling trajectory streams with spatiotemporal criteria. In *18th International Conference on Scientific and Statistical Database Management, SSDBM 2006, 3-5 July 2006, Vienna, Austria, Proceedings*, pages 275–284.

[Potamias et al., 2007] Potamias, M., Patroumpas, K., and Sellis, T. K. (2007). Online amnesic summarization of streaming locations. In *Advances in Spatial and Temporal Databases, 10th International Symposium, SSTD 2007, Boston, MA, USA, July 16-18, 2007, Proceedings*, pages 148–166.

[Pritchett, 2008] Pritchett, D. (2008). Base : An acid alternative. *Queue*, 6(3) :48–55.

[Ray et al., 2015] Ray, C., Gallen, R., Iphar, C., Napoli, A., and Bouju, A. (2015). Deais project : Detection of ais spoofing and resulting risks. In *OCEANS 2015 - Genova*, pages 1–6.

[Reiss et al., 2007] Reiss, F., Stockinger, K., Wu, K., Shoshani, A., and Hellerstein, J. M. (2007). Enabling real-time querying of live and historical stream data. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, SSDBM '07, pages 28–, Washington, DC, USA. IEEE Computer Society.

[Rhodes et al., 2007] Rhodes, B. J., Bomberger, N. A., and Zandipour, M. (2007). Probabilistic associative learning of vessel motion patterns at multiple spatial scales for maritime situation awareness. In *2007 10th International Conference on Information Fusion*, pages 1–8.

[Ristic et al., 2008] Ristic, B., Scala, B. F. L., Morelande, M. R., and Gordon, N. J. (2008). Statistical analysis of motion patterns in ais data : Anomaly detection and motion prediction. In *FUSION*, pages 1–7. IEEE.

[Rundensteiner et al., 2004] Rundensteiner, E. A., Ding, L., Sutherland, T. M., Zhu, Y., Pielech, B., and Mehta, N. K. (2004). CAPE : continuous query engine with heterogeneous-grained adaptivity. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 1353–1356.

[Sakr and Güting, 2014] Sakr, M. A. and Güting, R. H. (2014). Group spatiotemporal pattern queries. *GeoInformatica*, 18(4) :699–746.

[Salmon et al., 2014] Salmon, Ray, and Claraumnt (2014). Gestion et traitement simultané de flux temps-réel et archivés de données spatio-temporelles : Application à l'analyse du trafic maritime. In *Colloque international Géomatique, une vision prospective des territoires*, Atelier Spatial Big Data.

[Salmon and Ray, 2017] Salmon, L. and Ray, C. (2017). Design principles of a stream-based framework for mobility analysis. *GeoInformatica*, 21(2) :237–261.

[Salmon et al., 2015] Salmon, L., Ray, C., and Claramunt, C. (2015). A hybrid approach combining real-time and archived data for mobility analysis. In *Proceedings of the 6th ACM SIGSPATIAL*

International Workshop on GeoStreaming, IWGS 2015, Bellevue, WA, USA, November 3-6, 2015, pages 43–48.

[Salmon et al., 2016] Salmon, L., Ray, C., and Claramunt, C. (2016). Continuous detection of black holes for moving objects at sea. In *Proceedings of the 7th ACM SIGSPATIAL International Workshop on GeoStreaming*, IWGS '16, pages 2 :1–2 :10, New York, NY, USA. ACM.

[Samet, 1984] Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2) :187–260.

[Santipantakis et al., 2018] Santipantakis, G. M., Kotis, K. I., Vouros, G. A., and Doulkeridis, C. (2018). Rdf-gen : Generating RDF from streaming and archival data. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, WIMS 2018, Novi Sad, Serbia, June 25-27, 2018*, pages 28 :1–28 :10.

[Santipantakis et al., 2017] Santipantakis, G. M., Vouros, G. A., Glenis, A., Doulkeridis, C., and Vlachou, A. (2017). The datacron ontology for semantic trajectories. In *The Semantic Web : ESWC 2017 Satellite Events - ESWC 2017 Satellite Events, Portorož, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, pages 26–30.

[Sellis et al., 1987] Sellis, T. K., Roussopoulos, N., and Faloutsos, C. (1987). The r+-tree : A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, VLDB '87, pages 507–518, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Shah et al., 2004] Shah, M. A., Hellerstein, J. M., and Brewer, E. A. (2004). Highly-available, fault-tolerant, parallel dataflows. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 827–838.

[Shah et al., 2003] Shah, M. A., Hellerstein, J. M., Chandrasekaran, S., and Franklin, M. J. (2003). Flux : An adaptive partitioning operator for continuous query systems. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pages 25–36.

[Sharifzadeh and Shahabi, 2006] Sharifzadeh, M. and Shahabi, C. (2006). The spatial skyline queries. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 751–762. VLDB Endowment.

[Shekhar et al., 2012] Shekhar, S., Gunturi, V., Evans, M. R., and Yang, K. (2012). Spatial big-data challenges intersecting mobility and cloud computing. In *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDE '12, pages 1–6, New York, NY, USA. ACM.

[Shoji, 2014] Shoji, N. (2014). Key-value store "md-hbase" enables multi-dimensional range queries.

[Shvachko et al., 2010] Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA. IEEE Computer Society.

[Sistla et al., 1997] Sistla, A. P., Wolfson, O., Chamberlain, S., and Dao, S. (1997). Modeling and querying moving objects. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K.*, pages 422–432.

[Spaccapietra et al., 2008] Spaccapietra, S., Parent, C., Damiani, M. L., de Macêdo, J. A. F., Porto, F., and Vangenot, C. (2008). A conceptual view on trajectories. *Data Knowl. Eng.*, 65(1) :126–146.

[Sun et al., 2013] Sun, X., Yaagoub, A., Trajcevski, G., Scheuermann, P., Chen, H., and Kachhwaha, A. (2013). P2est : Parallelization philosophies for evaluating spatio-temporal queries. In *Proceedings of the 2Nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '13, pages 47–54, New York, NY, USA. ACM.

[Sutherland et al., 2005a] Sutherland, T. M., Liu, B., Jbantova, M., and Rundensteiner, E. A. (2005a). D-CAPE : distributed and self-tuned continuous query processing. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 217–218.

[Sutherland et al., 2005b] Sutherland, T. M., Zhu, Y., Ding, L., and Rundensteiner, E. A. (2005b). An adaptive multi-objective scheduling selection framework for continuous query processing. In *Ninth International Database Engineering and Applications Symposium (IDEAS 2005), 25-27 July 2005, Montreal, Canada*, pages 445–454.

[Tatbul et al., 2003] Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M., and Stonebraker, M. (2003). Load shedding in a data stream manager. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 309–320. VLDB Endowment.

[Thusoo et al., 2009] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., and Murthy, R. (2009). Hive : A warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, 2(2) :1626–1629.

[Toshniwal et al., 2014] Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S., and Ryaboy, D. V. (2014). Storm@twitter. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 147–156.

[Tu et al., 2016] Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., and Huang, G. (2016). Exploiting AIS data for intelligent maritime navigation : A comprehensive survey. *CoRR*, abs/1606.00981.

[Urhan and Franklin, 2001] Urhan, T. and Franklin, M. J. (2001). Dynamic pipeline scheduling for improving interactive query performance. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 501–510.

[Vatsavai et al., 2012] Vatsavai, R. R., Ganguly, A., Chandola, V., Stefanidis, A., Klasky, S., and Shekhar, S. (2012). Spatiotemporal data mining in the era of big spatial data : Algorithms and applications. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '12, pages 1–10, New York, NY, USA. ACM.

[Vavilapalli et al., 2013] Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B., and Baldeschwieler, E. (2013). Apache hadoop yarn : Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5 :1–5 :16, New York, NY, USA. ACM.

[Vazirgiannis et al., 1998] Vazirgiannis, M., Theodoridis, Y., and Sellis, T. K. (1998). Spatio-temporal composition and indexing for large multimedia applications. *Multimedia Syst.*, 6(4) :284–298.

[Vo et al., 2014] Vo, H., Aji, A., and Wang, F. (2014). Sato : A spatial data partitioning framework for scalable query processing. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 545–548, New York, NY, USA. ACM.

[Vodas et al., 2013] Vodas, M., Pelekis, N., Theodoridis, Y., Ray, C., Karkaletsis, V., Petridis, S., and Miliou, A. (2013). Efficient ais data processing for environmentally safe shipping. *SPOUDAI Journal of Economics and Business*, 63(3-4) :181–190.

[Whitby et al., 2017] Whitby, M. A., Fecher, R., and Bennight, C. (2017). Geowave : Utilizing distributed key-value stores for multidimensional data. In *Advances in Spatial and Temporal Databases - 15th International Symposium, SSTD 2017, Arlington, VA, USA, August 21-23, 2017, Proceedings*, pages 105–122.

[Windward, 2014] Windward (2014). Ais data on the high seas : an analysis of the magnitude and implications of growing data manipulation at sea.

[Wolf et al., 2008] Wolf, J. L., Bansal, N., Hildrum, K., Parekh, S., Rajan, D., Wagle, R., Wu, K., and Fleischer, L. (2008). SODA : an optimizing scheduler for large-scale stream-based distributed computer systems. In *Middleware 2008, ACM/IFIP/USENIX 9th International Middleware Conference, Leuven, Belgium, December 1-5, 2008, Proceedings*, pages 306–325.

[Wolfson et al., 1999] Wolfson, O., Sistla, A. P., Xu, B., Zhou, J., and Chamberlain, S. (1999). DOMINO : databases for moving objects tracking. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 547–549.

[Worboys and Duckham, 2004] Worboys, M. F. and Duckham, M. (2004). *GIS - a computing perspective (2. ed.)*. Taylor & Francis.

[Xie et al., 2016] Xie, D., Li, F., Yao, B., Li, G., Chen, Z., Zhou, L., and Guo, M. (2016). Simba : spatial in-memory big data analysis. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016, Burlingame, California, USA, October 31 - November 3, 2016*, pages 86 :1–86 :4.

[Xiong et al., 2007] Xiong, X., Elmongui, H. G., Chai, X., and Aref, W. G. (2007). Place : A distributed spatio-temporal data stream management system for moving objects. In *8th International Conference on Mobile Data Management (MDM 2007), Mannheim, Germany, May 7-11, 2007*, pages 44–51.

[Xiong et al., 2005] Xiong, X., Mokbel, M. F., and Aref, W. G. (2005). SEA-CNN : scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, pages 643–654.

[Yang et al., 2017] Yang, F., Merlino, G., Ray, N., Léauté, X., Gupta, H., and Tschetter, E. (2017). The radstack : Open source lambda architecture for interactive analytics. In *50th Hawaii International Conference on System Sciences, HICSS 2017, Hilton Waikoloa Village, Hawaii, USA, January 4-7, 2017*.

[Yang et al., 2014] Yang, F., Tschetter, E., Léauté, X., Ray, N., Merlino, G., and Ganguli, D. (2014). Druid : A real-time analytical data store. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 157–168, New York, NY, USA. ACM.

[Yu et al., 2015a] Yu, J., Wu, J., and Sarwat, M. (2015a). Geospark : a cluster computing framework for processing large-scale spatial data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Bellevue, WA, USA, November 3-6, 2015*, pages 70 :1–70 :4.

[Yu et al., 2016] Yu, J., Wu, J., and Sarwat, M. (2016). A demonstration of geospark : A cluster computing framework for processing big spatial data. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 1410–1413.

[Yu et al., 2005] Yu, X., Pu, K. Q., and Koudas, N. (2005). Monitoring k-nearest neighbor queries over moving objects. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, pages 631–642.

[Yu et al., 2015b] Yu, Z., Liu, Y., Yu, X., and Pu, K. Q. (2015b). Scalable distributed processing of K nearest neighbor queries over moving objects. *IEEE Trans. Knowl. Data Eng.*, 27(5) :1383–1396.

[Zaharia et al., 2012] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets : A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on*

Networked Systems Design and Implementation, NSDI'12, pages 2–2, Berkeley, CA, USA. USENIX Association.

[Zaharia et al., 2010] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark : Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*.

[Zaharia et al., 2013] Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., and Stoica, I. (2013). Discretized streams : Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 423–438, New York, NY, USA. ACM.

[Zhang et al., 2009] Zhang, C., Huang, Y., and Griffin, T. (2009). Querying geospatial data streams in SECONDO. In *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*, pages 544–545.

[Zhu et al., 2004] Zhu, Y., Rundensteiner, E. A., and Heineman, G. T. (2004). Dynamic plan migration for continuous queries over data streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 431–442.

Titre : Une approche holistique combinant flux temps-réel et données archivées pour la gestion et le traitement d'objets mobiles : application au trafic maritime.

Mots clés : Objets mobiles, traitement temps-réel, bases de données spatio-temporelles, système distribué.

Résumé

La numérisation de nos espaces de vie et de mobilités s'est largement accentuée durant la dernière décennie. La multiplication des capteurs de toute nature permettant de percevoir et de mesurer notre espace physique en est le levier principal. L'ensemble de ces systèmes produit aujourd'hui de grands volumes de données hétérogènes sans cesse croissants, ce qui soulève de nombreux enjeux scientifiques et d'ingénierie en termes de stockage et de traitement pour la gestion et l'analyse de mobilités. Les travaux dans le domaine d'analyse des données spatio-temporelles ont largement été orientés soit vers la fouille de données historiques archivées, soit vers le traitement continu. Afin d'éviter les écueils de plus en plus prégnants dus à l'augmentation de ces volumes de données et de leur vélocité (temps de traitement trop long, modèles conceptuellement plus adaptés, analyse approximative des données), nous proposons la conception d'une approche hybride distribuée permettant le traitement combiné de flux temps-réel et de données archivées. L'objectif de cette thèse est donc de développer un nouveau système de gestion et de traitement distribué pour l'analyse des mobilités en particulier maritimes. La solution proposée répond principalement à des contraintes de temps-réel, les données archivées et les informations qui en sont extraites permettant d'améliorer la qualité de réponse. Une proposition de paradigme d'événements est également développée pour permettre ce traitement hybride mais aussi pour caractériser et identifier plus facilement des comportements types d'objets mobiles. Enfin, une requête appliquée sur des zones de couverture de signal pour des objets mobiles a été étudiée et testée sur des données maritimes mettant en exergue le besoin d'une approche hybride pour le traitement de trajectoires.

Title : A hybrid approach combining real-time and archived data for mobility analysis : application to maritime trafic

Keywords : Moving objects, geostreaming, spatio-temporal databases, Distributed systems.

Abstract

Over the past few years, the rapid proliferation of sensors and devices recording positioning information regularly produces very large volumes of heterogeneous data. This leads to many research challenges as the storage, distribution, management, processing and analysis of the large mobility data generated still needs to be solved. Current works related to the manipulation of mobility data have been directed towards either mining archived historical data or continuous processing of incoming data streams. The aim of this research is to design a holistic system whose objective is to provide a combined processing of real time data streams and archived data positions. The proposed solution is real-time oriented, historical data and informations extracted from them allowing to enhance quality of the answers to queries. A event paradigm is discussed to facilitate the hybrid approach and to identify typical moving objects behaviors. Finally, a query concerning signal coverage of moving objects has been studied and applied to maritime data showing the relevance of a hybrid approach to deal with moving object data processing.