Panorama des modèles de flux de données à large échelle

Yann Busnel

Panorama des modèles de flux de données à large échelle

Yann Busnel

IMT Atlantique, IRISA
2, rue de la Châtaigneraie – CS 17607
F-35576 Cesson Sévigné, France
yann.busnel@imt-atlantique.fr

RÉSUMÉ. *L'exploitation en temps réel des données massives du monde numérique nécessite l'appropriation de nouvelles méthodes. L'extraction d'informations pertinentes dans des flux de données massifs, et potentiellement infinis, représente un défi de taille, toujours ouvert à ce jour. L'intérêt d'estimer des métriques ou d'identifier des motifs spécifiques entre différents flux de données est important dans de nombreuses applications traitant des masses de données, incluant l'apprentissage, la fouille de données, les bases de données, la collecte d'informations, ou la surveillance réseau.*
Cet article vise à dresser un panorama aussi complet que possible des différents modèles existants dans la littérature. Nous commençons par présenter le contexte général et la diversité des problématiques liées aux flux de données à grande échelle. Puis, nous proposons un horizon des différents modèles formels de flux ayant été proposés, principalement lors des 2 dernières décennies, ainsi que des exemples d'algorithmes utilisant ces modèles à vocation d'illustration. Le champ des possibles usages de ces méthodes est considérable. Les possibilités de décentralisation de ces algorithmes, l'optimisation par l'apport d'une information sémantique ou l'ouverture à de nouveaux modèles de flux dresseront, en guise de conclusion, des perspectives concrètes de ces modèles.

ABSTRACT. *The real-time exploitation of massive data from the digital world requires the appropriation of new methods. The extraction of relevant information in massive and potentially infinite data streams represents a major challenge, still open today. The value of estimating metrics or identifying specific patterns between different data streams is important in many big data applications, including machine learning, data mining, databases, information retrieval, or network monitoring.*
This article aims to provide an overview as complete as possible of the different models available in the literature. We begin by presenting the general context and the diversity of issues related to large-scale data streams. Then, we propose a survey of the different data streaming models that have been proposed, mainly during the last decade, as well as illustrative examples of algorithms using these. The range of possible uses for these methods is considerable. The ability of decentralization of these algorithms, the optimization by the addition of semantic information or the opening to new data streaming models will draw, as a conclusion, the concrete perspectives of this overview.

MOTS-CLÉS : *Flux de données, Large échelle, Modèles de systèmes complexes, Big data, Systèmes*

distribués, Adversaire byzantin, Algorithme d'approximation.

KEYWORDS: *Data stream, Large scale, Complex system models, Big data, Distributed systems, Byzantine Adversary, Approximation algorithm.*

DOI:10.3166/TSI..1-27

1. Introduction et contexte

La démocratisation et la miniaturisation des réseaux, en partie dues à la croissance exponentielle de l'Internet, ont permis d'envisager des systèmes à très grande échelle, visant à exploiter en commun une multitude de ressources à travers un nombre d'entités toujours croissant. Les systèmes distribués ont donc nécessité une migration de l'approche classique client-serveur vers une informatique répartie à très grande échelle. L'émergence récente du concept de *Cloud Computing* se place dans ce cadre et permet de démocratiser l'accès à ces infrastructures (initialement réservées à des applications de simulation "lourde"), de les rendre accessibles à un spectre de plus en plus large d'applications et donc à un nombre croissant d'acteurs du monde économique. Cette démocratisation est à l'origine du nombre grandissant de données transitant sur nos réseaux, nécessitant des approches novatrices pour le traitement des grandes masses de données.

Par exemple, l'apparition du web a bouleversé les habitudes et la vie de l'humanité et pas seulement des professionnels de l'informatique et de l'information [Mayer-Schönberger, Cukier, 2013]. Le web est lui-même en constante évolution. D'outil de communication et de dissémination, il est devenu éditable et réactif, pour à présent devenir social. Il suffit de voir l'ampleur prise par les réseaux sociaux pour s'en convaincre. Le web est ainsi devenu la première source d'information de tout un chacun.

1.1. Le défi du « Big Data »

Dans le cadre de l'*Internet du futur*, il est ainsi souhaitable de proposer des modèles de systèmes pour la gestion de grande masse de données distribuées à très large échelle par des communautés d'utilisateurs. Le défi est de construire des systèmes et réseaux performants et fiables mais aussi de permettre l'accès à l'information, la personnalisation et l'appropriation des contenus par les utilisateurs. L'émergence du *Big Data* ces dernières années se révèle comme incontournable et particulièrement pertinent pour le développement de ces systèmes [C. Lynch, 2008].

Au cours de la dernière décennie, nous avons donc assisté à l'émergence d'applications numériques nécessitant de faire face à de gigantesques quantités de données, générées de plus en plus rapidement. Ces applications diverses (métrologie des réseaux, biologie et médecine, applications financières, réseaux sociaux, *etc.*) requièrent de plus en plus des techniques capables d'analyser et de traiter ces grandes masses d'information, avec précision et efficacité, en temps contraint. La statistique rejoint ici les sciences du numérique, et plus précisément l'informatique répartie, pour proposer de nouvelles approches, relatives au Big Data. Les techniques et les modèles doivent prendre en compte le volume pléthorique de ces données, mais également leur génération rapide en continu (vélocité) ainsi que l'hétérogénéité de leur format (variété). Il devient inconcevable de stocker l'ensemble des données concernées pour y effectuer ces traitements *a posteriori*. La diversité de ces contenus nécessite donc la proposition de solutions en temps réel innovantes et à faible complexité, à la fois en terme de

mémoire, de temps de traitement, de communication, *etc.* [Muthukrishnan, 2005]. Les modèles de flux qui sont présentés dans cet article s'attachent ainsi principalement à résoudre des problèmes liés à la *vélocité*, mais trouvent également toute leur place dans le traitement rapide de grands *volumes* de données, potentiellement hétérogènes.

Dans toutes ces applications, il est ainsi nécessaire de traiter rapidement et précisément un nombre considérable de données. Dans cet environnement hautement concurrentiel, il est probable qu'un nombre non négligeable de fournisseurs de services tentera de tromper les utilisateurs de ces systèmes, par collusion ou non ; ils peuvent aussi essayer d'attirer plus de données qu'ils ne peuvent raisonnablement traiter. D'un autre côté, par exemple, dans la gestion d'un réseau IP, l'analyse des flux d'entrée permet de détecter rapidement la présence d'anomalies ou de tentatives d'intrusion quand des changements de motifs de communication apparaissent. Concernant la sécurisation de systèmes communicants à grande échelle, nous voudrions être capables de détecter des attaques par surveillance passive des messages transitant sur chaque nœud. Par exemple; s'il était possible de détecter une divergence entre un flux attendu et le flux observé sur un nœud, il serait possible d'alerter le propriétaire de celui-ci. Les faibles capacités de mémoire mises à disposition pour ces opérations de surveillance et métrologie rendent cependant la détection de ces attaques particulièrement complexe.

1.2. L'intérêt du modèle flux

Le problème d'extraire de l'information pertinente dans un (ou des) flux (répartis) de données est similaire au problème d'identifier des motifs qui ne sont pas conformes au comportement attendu. Cette thématique a été un champ de recherche très productif ces dernières décennies. Par exemple, en fonction des spécificités du domaine et du type d'aberrations considérées, différentes méthodes ont été conçues : classification, groupement, plus proche voisin, statistique spectrale et théorie de l'information. Un état de l'art intéressant sur ces techniques, décrivant leurs avantages et inconvénients, est proposé par [Chandola *et al.*, 2009]. Une fonctionnalité commune de ces techniques est leur grande complexité en espace et en coût de calcul, car elles reposent sur des algorithmes « à mémoire non bornée », *i.e.,* nécessitant la connaissance et l'accès complets aux données analysées.

Reposer sur des algorithmes de ce type n'est pas faisable dans un contexte d'extraction d'information en temps réel, avec une très faible capacité en terme de stockage et de traitement. Cependant, deux principales approches existent pour le traitement en temps réel de flux de données massifs, potentiellement répartis.

La première consiste à régulièrement échantillonner le flux, permettant de limiter la quantité de données conservées en mémoire [Alon *et al.*, 1996, Karamcheti *et al.*, 2005, Krishnamurthy *et al.*, 2003, Lakhina *et al.*, 2005]. Cette méthode permet de calculer exactement des fonctions sur ces échantillons. Cependant, la fiabilité de ce calcul, en comparaison avec le résultat exact calculé sur le flux entier, dépend fortement du volume de données qui a été échantillonné, et de la position de ces données sur le flux. Pire encore, un adversaire pourrait facilement tirer avantage de la poli-

tique d'échantillonnage pour masquer ses attaques parmi les paquets qui n'ont pas été échantillonnés, ou de ce fait, éviter que ses paquets « malveillants » puissent être corrélés.

A l'inverse, la seconde approche consiste à traiter à la volée toutes les données du flux, et de ne conserver localement que des résumés, ou *agrégats*, dans lesquels seule l'information essentielle à propos des données est conservée [Chandola *et al.*, 2009]. Cette approche extrait des statistiques sur les flux de données traités, avec des probabilités d'erreurs bornées, sans présupposer aucune contrainte sur l'ordre dans lequel les données sont reçues sur les nœuds (*i.e.*, l'ordre des données pourrait être manipulé par un adversaire omnipotent [Anceaume *et al.*, 2010]). La plupart des recherches existantes utilisant cette approche se sont concentrées sur le calcul de fonctions ou de mesures statistiques avec une erreur ε donnée, utilisant une quantité mémoire poly-logarithmique en la taille du flux et du domaine des données entrantes. Ces fonctions estiment par exemple le nombre de données distinctes présentes dans un flux donné [Bar-Yossef *et al.*, 2002, Flajolet, Martin, 1985, Kane *et al.*, 2010], les moments de fréquence [Alon *et al.*, 1996], les éléments les plus fréquents [Alon *et al.*, 1996, Charikar *et al.*, 2004, Misra, Gries, 1982, Metwally *et al.*, 2005], l'entropie d'un flux [Chakrabarti *et al.*, 2007, Lall *et al.*, 2006], la détection d'attaque par inondation [Salem *et al.*, 2010], la corrélation entre plusieurs flux [Anceaume, Busnel, 2017] ou l'entropie relative entre un flux biaisé et un flux uniforme [Anceaume *et al.*, 2012, Anceaume, Busnel, 2014b].

De surcroit, pour pouvoir étudier théoriquement l'impact des adversaires sur les algorithmes proposés dans la litterature, il est possible d'utiliser différents modèles issus de la communauté des systèmes distribués. Par exemple, nous considérons généralement la présence d'un adversaire *adaptatif*, qui cherche a éviter le bon déroulement de l'algorithme en répartissant ses données judicieusement parmi les différents flux [Anceaume *et al.*, 2015]. Il est ainsi capable d'insérer autant d'information que nécessaire pour augmenter ou réduire la précision d'un algorithme, par rapport à l'état courant du système, ou de réordonner les flux à sa guise. Cependant, généralement, il est considéré que tous les nœuds suivent scrupuleusement les protocoles, lesquels sont des données publiques pour éviter toute sorte de sécurité par obscurité.

1.3. Un modèle, des modèles

Malheureusement, calculer des mesures de théorie de l'information dans le modèle de flux est extrêmement complexe, essentiellement en raison du traitement séquentiel, à la volée, d'une incommensurable quantité de données en utilisant un espace mémoire très restreint (par rapport à la taille du flux) [Muthukrishnan, 2005].

De surcroît, l'analyse doit être temporellement robuste pour détecter les changements soudains sur les flux observés (potentiellement liés à la manifestation d'une attaque par déni de service ou la propagation d'un ver par exemple). L'un des modèles dérivés le plus naturel pour prendre en compte ces aspects est le modèle dit à *fenêtre glissante*, introduit par Datar *et al.* [2002], lequel considère uniquement une portion

récente du flux considéré. Dans de nombreux contextes applicatifs, seuls les éléments reçus le plus récemment sont pertinents au regards de ceux reçus au démarrage du flux. La majorité des problèmes sus-cités trouvent une réponse dans ce modèle comme l'estimation de la variance [Zhang, Guan, 2007], des quantiles [Arasu, Manku, 2004], des éléments surabondants [Cormode, Yi, 2012, Golab *et al*., 2003], la surveillance fonctionnelle [Chan *et al*., 2012], ou la définition d'agrégats [Rivetti, Busnel, Mostéfaoui, 2015].

D'un autre côté, peu de travaux ont pris en compte le modèle de flux répartis, également appelé *problème de surveillance fonctionnelle* (PSF – « functional monitoring ») introduit par Cormode *et al*. [2008]. Celui-ci combine les caractéristiques des modèles de flux et de communication. Comme dans le modèle de flux, les données en entrée sont lues à la volée et traitées en un minimum d'espace et de temps. Cependant, la majorité des travaux proposant une analyse pour un flux unique ne sont pas adaptables au modèle réparti du PSF. Ici, chaque nœud distant reçoit un flux en entrée, effectue quelques traitements en local, et communique avec un coordinateur qui souhaite calculer, ou estimer, de manière continue, une fonction donnée, sur l'union de tous les flux entrants. Le défi de ce modèle est, pour le coordinateur, de calculer cette fonction en minimisant le nombre de bits transmis [Gibbons, Tirthapura, 2001, Anceaume, Busnel, 2014b,a, Anceaume *et al*., 2015]. Cormode *et al*. [2008] innovent en proposant une étude formelle de fonctions réalisées dans ce modèle, se concentrant dans un premier temps sur les trois premiers moments de fréquence, introduits par Alon *et al*. [1996]. Arackaparambil *et al*. [2009] considèrent l'estimation répartie et empirique de l'entropie [Alon *et al*., 1996] et améliorent les travaux de Cormode *et al.* en fournissant une borne inférieure sur la complexité d'estimation des moments de fréquence. Enfin, ils proposent des algorithmes répartis pour compter, à tout temps, le nombre de données qui a été reçu par un noeud depuis le début de leur flux respectif [Haung *et al*., 2012, Liu *et al*., 2012].

Enfin, d'autres modèles ont été proposés ces dernières années, lesquels relâchent certaines hypothèses comme le modèle multi-flux de Gibbons, Tirthapura [2001], ou mixent plusieurs modèles comme le PSF en fenêtre glissante de Cormode, Yi [2012] ou de Gibbons, Tirthapura [2004].

Dans la suite, nous offrons un tour d'horizon parfaitement subjectif des différents formalismes des modèles existants.

L'objectif suivant est la conception et l'implémentation de prototypes d'algorithmes, répartis ou non, efficaces en une seule passe, dans un contexte où l'échange de masse des données est la norme. Ces algorithmes y ont déjà trouvé une utilité pertinente et croissante dans de nombreux contextes, comme la métrologie des réseaux, la sûreté de fonctionnement, les centres de calculs temps réel, *etc.*

Le champ des possibles usages de ces méthodes est considérable. Peu nombreux sont ceux qui peuvent prédire avec certitude ce que seront les besoins de l'Internet dans une décennie. Cependant, les possibilités de décentralisation de ces algorithmes, l'optimisation par l'apport d'une information sémantique ou l'ouverture à d'autres

modèles de flux dessineront, en guise de conclusion, les perspectives concrètes de cet article.

2. Modèles de flux de données

Selon la définition de Csernel *et al*. [2005], un flux de données est « *une séquence infinie d'éléments générés de façon continue à un rythme rapide* ». Dans la communauté de recherche sur les flux de données, de nombreux modèles ont émergé afin de concorder avec les scenarii d'applications de ces approches temps réel. Cette section discute et formalise les modèles qui ont attiré davantage l'attention et qui sont communément utilisés à ce jour.

La taxonomie proposée ci-après reflète une classification personnelle des modèles existants et ne prêtant pas être représentative de l'ensemble des angles de vue de la communauté.

2.1. Modèle de flux unique

Le modèle fondateur considère les flux massifs comme une séquence d'éléments (ou *n-uplets*, ou encore *tuples*) potentiellement infinie $\langle a_1, a_2, \ldots, a_m \rangle$ que nous dénoterons σ par la suite. Les éléments sont tirés d'un univers Ω hypothétiquement très grand, que nous abstrairons, sans perdre en généralité $\{1, 2, \ldots, n\}$, par souci de clarté. Nous avons ainsi défini implicitement par $m = |\sigma|$ la taille (ou longueur) du flux considéré (dans le cas fini), et par $n = |\Omega|$, le cardinal de l'univers considéré. Il est important de noter que ce dernier est habituellement trop grand pour en espérer un stockage en mémoire efficace (*e.g.,* $n = 2^{128}$ pour les adresses IPv6 ou 2^{160} pour la fonction de hachage standard SHA-1).

Les éléments arrivent ainsi rapidement, sans contrainte d'ordre ni d'unicité. Les accès aux éléments suivent donc la séquence initiale (sans possibilité d'accès direct ou aléatoire). Les fonctions d'intérêt, que nous noterons ϕ par la suite, doivent ainsi être calculées en une seule passe (*i.e.,* en ligne et en temps réel).

L'enjeu majeur est d'accomplir le calcul de ϕ en complexité mémoire sous-linéaire en m et n. Formellement, cette dernière doit respecter au minimum $o(\min\{m, n\})$, mais l'objectif principal reste d'assurer $O(\log(m \cdot n))$. Cependant, cet objectif est difficile (voire impossible) à atteindre dans la majorité des problèmes et la garantie conventionnelle est $polylog(m, n)$ [1]. Il convient d'établir ici que la notation « $\log$ » utilisée dans cet article correspond au logarithme en base 2. De plus, la complexité de traitement de chaque élément doit être minimale pour suivre le rythme imposé par le débit du flux.

1. Par abus de langage, sachant que $f(x) = polylog(g(x)) \Leftrightarrow \exists c > 0, f(x) = O(\log^c g(n))$, la notation $polylog(m, n)$ est définie par $\exists c, c' > 0$ tels que $f(x) = O(\log^c(m) + \log^{c'}(n))$.

Dans ce modèle, les solutions calculant de façon continue une fonction donnée sont souvent différenciées de celles effectuant un calcul ponctuel. Muthukrishnan [2005] propose un état de l'art pertinent et relativement exhaustif des recherches dans ce modèle, à l'horizon de 2005, mais toujours d'actualité.

2.2. Modèle à fenêtre glissante

Dans de nombreuses applications, il est plus pertinent de ne prendre en considération que les éléments récemment observés, plutôt que l'ensemble du flux [Chan *et al.*, 2012]. Par exemple, dans une application de métrologie ou de détection d'attaque par déni de service, calculer le nombre d'occurrences d'une adresse destination dans un flux IP sur plusieurs mois n'a pas de sens. Ce constat a motivé la proposition d'un modèle à fenêtre glissante, formalisé par [Datar *et al.*, 2002].

Dans ce modèle, les éléments arrivent continuellement mais expirent après $\dddot{m}$ étapes. Les données pertinentes représentent donc les éléments reçus durant les $\dddot{m}$ dernières étapes. Une étape est définie soit :

– par l'arrivée d'un nouvel élément. La fenêtre glissante contient alors exactement $\dddot{m}$ éléments et la fenêtre est dite à taille fixe;

– par un tic d'horloge (logique ou physique). La fenêtre est alors dite temporelle.

Formellement, en supposant qu'au plus un élément est reçu par étape, le flux $\sigma(t)$ considéré à l'étape t (ou au temps t) est la séquence $\langle a_1, \ldots, a_i, a_j, \ldots, a_{t'} \rangle$ où $i \leq t - \dddot{m}$, $j > t - \dddot{m}$ et $t' \leq t$ (lequel correspond au temps d'arrivée du dernier élément reçu). La fonction considérée devra ainsi être évaluée uniquement sur le sous-flux $\sigma_{\dddot{m}}(t) = \langle a_i \rangle_{i>t-\dddot{m}}$. La fenêtre glissante se réfère donc à la séquence des éléments valides.

Deux autres modèles simplifiés de la littérature méritent également d'être cités, dits à fenêtre à jalon et à fenêtre sautante. Dans le premier, une étape particulière est choisie comme point de repère et la fonction est calculée sur le sous-flux reçu à partir de celui-ci. A l'inverse, dans le modèle à fenêtre sautante, la fenêtre courante est divisée en plusieurs sous-fenêtres (habituellement de taille fixe), et progresse uniquement quand la sous-fenêtre courante est pleine, représentant un « saut en avant ».

L'ensemble de ces modèles peut être utilisé dans le cadre d'un calcul ponctuel ou continu. Ils héritent par ailleurs des contraintes de mémoire sous-linéaire et d'accès séquentiel du modèle précédent (*cf.* Section 2.1).

2.3. Modèle à flux multiples

Le modèle à flux multiples a été formalisé par Gibbons, Tirthapura [2001]. Nous considérons ici un ensemble de nœuds recevant chacun un flux de données distinct et coopérant par l'intermédiaire d'un coordinateur central.

Formellement, nous considérons un ensemble de nœuds $\mathcal{S} = \{1, \ldots, s\}$, recevant chacun un flux massif $\sigma^{(i)}$ ($i \in \mathcal{S}$) composé d'éléments tirés dans un univers commun Ω. Chaque sous-flux étant potentiellement non-bornés, les nœuds ne peuvent stocker localement qu'une quantité minime d'informations, au regard de m et n.

Dans ce modèle, les nœuds ne peuvent communiquer directement entre eux, mais doivent passer par l'intermédiaire d'un coordinateur. Ce dernier peut être vu comme une station de base dans un réseau de capteurs, ou un leader choisi dans un contexte décentralisé. En fonction de l'application, le mode de communication peut ainsi être modélisé comme uni- ou bi-directionnel. La figure 2.3 illustre le fonctionnement de ce modèle.

A nouveau, nous partageons les contraintes du modèle pionnier de la Section 2.1 avec comme objectif de minimiser la complexité de communication (en nombre de bits échangés), pour calculer ponctuellement une fonction d'intérêt ϕ sur l'union de ces flux:

$$\phi(\sigma) = \phi(\sigma^{(1)} \cup \ldots \cup \sigma^{(s)}).$$

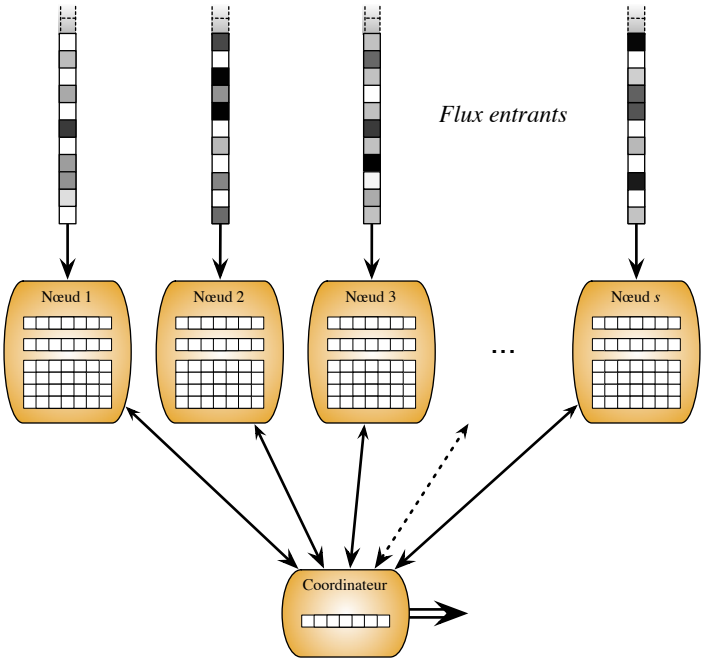

Figure 1. Illustration des modèles à flux répartis

2.4. Modèle de flux répartis continus

Cormode *et al*. [2008] ont récemment proposé le modèle de flux répartis continus, également connu sous le nom de problème de surveillance fonctionnelle, dont ils dressent un état de l'art des possibilités dans [Cormode *et al*., 2011, Cormode, 2011].

Ce modèle est similaire au précédent à flux multiples, mais plutôt qu'une évaluation ponctuelle et globale, celui-ci considère une évaluation en continu de la fonction d'intérêt ϕ sur l'union des flux.

Formellement, l'objectif est de calculer à tout instant t,

$$\phi(\sigma(t)) = \phi(\sigma^{(1)}(t) \cup \ldots \cup \sigma^{(s)}(t)).$$

Cormode *et al*. [2011] bornent cependant ϕ à être monotone et distinguent deux classes de problèmes :

– la supervision à seuil, dont l'objectif est de lever une alerte si $\phi(\sigma(t))$ dépasse une certaine valeur limite donnée ;

– la supervision de valeur, qui calcule une approximation de la valeur exacte de $\phi(\sigma(t))$.

Les auteurs affirment néanmoins que la seconde classe de problèmes peut être réduite à la première, avec un facteur $O(\frac{1}{\varepsilon} \log R)$ bits de communication, où R représente la plus grande valeur atteinte de ϕ. Toutefois, la question reste ouverte pour le calcul de fonctions non monotones, telles que l'entropie par exemple.

2.5. Combinaison de modèles

Comme introduit dans la Section 1.3, un intérêt significatif existe pour un modèle permettant de combiner les aspects multi-flux et fenêtre glissante. Gibbons, Tirthapura [2004] proposent une formalisation de celui-ci, tout en soulevant le problème de la définition d'une fenêtre commune dans un environnement synchrone. Les auteurs proposent ainsi trois définitions d'une fenêtre de $\ddot{m}$ éléments sur s flux répartis:

1. Une fenêtre est composée des $\ddot{m}$ derniers éléments de chacun des s flux. La fenêtre représente donc une taille globale de $s\ddot{m}$ éléments.

2. Les flux locaux sont extraits d'un flux global unique arrivant à l'entrée du système et répartis arbitrairement sur les différents nœuds participant. Chaque élément reçu localement est donc enrichi d'un numéro de séquence global. Ainsi, une fenêtre représente $\ddot{m}$ éléments consécutifs dans le flux entrant unique.

3. Les $\ddot{m}$ derniers éléments de chacun des s flux locaux sont fusionnés pour produire un unique flux sortant global, également de taille $\ddot{m}$. Les éléments à l'étape t de ce flux combiné résultent du *ou* logique appliqué sur les s éléments des flux locaux à l'étape t (*i.e.,* $a_t = a_t^{(1)} \vee \ldots \vee a_t^{(s)}$).

Si les deux premières définitions diffèrent formellement, elles peuvent être résolues par adaptation d'algorithmes existants, à la fois dans les flux uniques ou multiples.

Cependant, la dernière ouvre un champ de recherche stimulant mais ambitieux qui reste à explorer, au même titre que leur adaptation au modèle asynchrone.

2.6. Modèles d'adversaires

De manière transversale, dans n'importe quel modèle cité ci-dessus, nous pouvons supposer la présence de nœuds malveillants (ou Byzantins) qui cherchent collectivement à subvertir le système, en manipulant les algorithmes considérés.

Ce comportement malveillant est modélisé par un adversaire qui contrôle et manipule intégralement ces nœuds. Nous supposons que l'adversaire est adaptatif, en ce sens qu'il peut manipuler, autant qu'il le souhaite, le flux entrant de n'importe quel nœud dans $\mathcal{S}$, en observant, insérant, ou réorganisant les éléments de ce flux.

Afin d'éviter toute forme de sécurité par obscurité, l'algorithme utilisé est considéré comme une connaissance publique. Cependant, l'adversaire ne peut avoir connaissance des valeurs aléatoires générées localement par l'algorithme (*e.g.,* le choix aléatoire des fonctions de hachage à partir d'une famille de fonctions 2-universelles [2] [Carter, Wegman, 1979]).

Généralement, la résilience face à un adversaire est révélée par l'effort requis par ce dernier pour renverser l'algorithme considéré (*i.e.,* pour enfreindre ses garanties de précision). D'autre part, dans le cas des flux répartis, l'adversaire peut également mettre en œuvre une stratégie pour faire du coordinateur une cible d'attaque par déni de service distribué (DDoS), en maximisant les interactions entre lui et les s nœuds du système. Il peut y parvenir en réorganisant les éléments des flux et/ou en injectant stratégiquement des éléments à des positions pertinentes [Anceaume *et al.*, 2015].

Un nœud du système qui n'est pas malveillant est dit *correct*. Cependant, aucun nœud correct ne peut distinguer *a priori* les autres nœuds corrects des malveillants. Traditionnellement, un adversaire n'est pas en mesure de supprimer ou de modifier le contenu d'un message échangé entre deux nœuds corrects sans être détecté [Bortnikov *et al.*, 2009]. L'existence d'un mécanisme d'authentification est ainsi supposée pour assurer l'authenticité et l'intégrité des messages, correspondant au modèle de défaillance Byzantine authentifiée [N. Lynch, 1996]. De même, le contrôle de multiples nœuds est très coûteux pour l'adversaire, qui doit ainsi interagir avec une autorité cen-

2. Une famille $\mathcal{H}$ de fonctions de hachage $h : [\![M]\!] \to [\![N]\!]$ est dite *ℓ-indépendante* si pour tout ℓ éléments distincts $x_1, \ldots, x_\ell \in [\![M]\!]$ et pour tout ℓ hachés (non nécessairement distincts) $y_1, \ldots, y_\ell \in [\![N]\!]$,

$$\mathbb{P}_{h\in\mathcal{H}}\{h(x_1) = y_1 \wedge \ldots \wedge h(x_\ell) = y_\ell\} \leq \frac{1}{N^\ell}.$$

Une famille $\mathcal{H}$ de fonctions de hachage $h : [\![M]\!] \to [\![N]\!]$ est dite *universelle* si pour tout couple d'éléments distincts $x, y \in [\![M]\!]$,

$$\mathbb{P}_{h\in\mathcal{H}}\{h(x) = h(y)\} \leq \frac{1}{N}.$$

trale pour recevoir les certificats adéquats, *i.e.,* le modèle d'attaque Sybil [Douceur, 2002].

3. Méthodes d'approximation et algorithmes

L'ensemble des contraintes de ces modèles a été précédemment introduit, tel que la nécessité de fonctionner à mémoire restreinte, en ligne, avec un accès séquentiel, une faible complexité de calcul et de communication, *etc.* Dans ces conditions, la plupart du temps, il est impossible d'obtenir les valeurs exactes de la fonction d'intérêt ϕ considérée. Par conséquent, il est d'usage de recourir à des algorithmes d'approximation, permettant de garantir un résultat estimé avec une marge d'erreur relative bornée. Cette section présente ainsi brièvement les techniques courantes d'estimation.

Comme introduit dans la section 1, deux techniques d'approximation, que nous présentons succinctement ci-après, sont principalement utilisées dans les solutions existantes.

3.1. Échantillonage

L'approche par échantillonnage consiste à prélever un sous-ensemble restreint d'éléments reçus sur le flux, et d'appliquer la fonction d'intérêt ϕ à ce sous-ensemble. Conventionnellement, l'enjeu persiste à restreindre l'espace de stockage à une taille poly-logarithmique en la longueur du flux. Couramment, des méta-données sont aussi associées aux éléments échantillonnés [Alon *et al.*, 1996, Karamcheti *et al.*, 2005, Krishnamurthy *et al.*, 2003, Lakhina *et al.*, 2005, Misra, Gries, 1982, Anceaume *et al.*, 2013].

Il existe de nombreuses méthodes pour choisir quel élément doit être échantillonné, lesquelles peuvent être déterministes ou aléatoires [Muthukrishnan, 2005]. Cependant, la présence d'un adversaire contraint fortement la politique d'échantillonnage, celui-là pouvant tirer profit de cette dernière pour augmenter ses chances de réussir une attaque. Afin de limiter l'impact de ces attaques, une méthode classique pour augmenter l'entropie d'un plan de sondage est de réordonner aléatoirement le flux avant d'échantillonner [Chauvet, 2012]. Malheureusement, ce processus est incompatible avec les modèles présentés ci-dessus.

3.2. Projections aléatoires et agrégats

A l'inverse, les techniques de projection et d'agrégat examinent chaque élément du flux d'entrée à la volée et conservent localement un résumé de l'information pertinente pour la fonction ϕ à estimer.

Les méthodes usuelles reposent sur des *projections (pseudo-)aléatoires* permettant de réduire significativement la taille des données. L'empilement de projections aléatoires permet souvent d'affiner le résultat final. Les vecteurs de projection reposent

principalement sur les fonctions de hachage utilisées pour y extraire les statistiques désirées et sont générés en utilisant des calculs à faible mémoire, sur des variables aléatoires indépendantes [Charikar *et al.*, 2004, Cormode, Muthukrishnan, 2005].

Un *agrégat* est défini comme une structure de données (telle que les projections aléatoires) qu'il est possible de fusionner sans perte d'information supplémentaire, en comparaison à l'élaboration de celle-ci sur les données globales. Formellement, une structure de données SD est un agrégat s'il existe un algorithme de fusion COMB tel que, pour tout couple de flux (σ_1, σ_2),

$$\texttt{COMB}(\texttt{SD}(\sigma_1), \texttt{SD}(\sigma_2)) = \texttt{SD}(\sigma_1 \cup \sigma_2),$$

où $\sigma_1 \cup \sigma_2$ représente l'union des éléments de σ_1 et σ_2 selon n'importe quelle intrication. De plus, une structure de données est un *agrégat linéaire* si elle prend ses valeurs dans un espace vectoriel de dimension finie, souvent dépendant de n, et que l'algorithme d'agrégation est une fonction linéaire de ϕ.

Dans le cadre des modèles à fenêtre glissante, les définitions d'agrégat sont bien entendu limitées à l'intrication des éléments des flux σ_1 et σ_2 sur les $\dot{m}$ derniers éléments seulement.

3.3. Algorithmes d'approximation

Ainsi, étant donné les contraintes fortes des modèles présentées, dans la plupart des cas, il est impossible d'avoir recours à un algorithme déterministe. Les solutions couramment proposées reposent donc sur des algorithmes d'approximation, dont l'erreur et l'intervalle de confiance sont garantis de manière déterministe.

Un algorithme $\mathcal{A}$ est une *ε-approximation* d'une fonction d'intérêt ϕ sur un flux σ si, pour toute séquence d'éléments de σ, l'algorithme $\mathcal{A}$ retourne une estimation $\hat{\phi}$ telle que

$$\mid \hat{\phi}(\sigma) - \phi(\sigma) \mid \leq \varepsilon\phi(\sigma),$$

où $\varepsilon > 0$ est un paramètre de précision donné de $\mathcal{A}$.

De même, un algorithme aléatoire $\mathcal{A}$ est une *(ε, δ)-approximation (relative)* d'une fonction d'intérêt ϕ sur un flux σ si, pour toute séquence d'éléments de σ, l'algorithme $\mathcal{A}$ retourne une estimation $\hat{\phi}(\sigma)$ telle que

$$\mathbb{P}\{\mid \hat{\phi}(\sigma) - \phi(\sigma) \mid > \varepsilon\phi(\sigma)\} < \delta,$$

où $\varepsilon, \delta > 0$ représentent respectivement les paramètres de précision et de sensibilité de $\mathcal{A}$.

Enfin, dans certains cas, cette condition est trop forte, par exemple lorsque les valeurs de ϕ sont proches de ou égales à 0. Dans les hypothèses ci-dessus, un algorithme aléatoire $\mathcal{A}$ est une *(ε, δ)-approximation absolue* d'une fonction d'intérêt ϕ sur un flux σ si

$$\mathbb{P}\{\mid \hat{\phi}(\sigma) - \phi(\sigma) \mid > \varepsilon\} < \delta.$$

3.4. Exemples d'algorithmes d'approximation

Parmi les nombreuses références cités dans les sections précédentes, nous pouvons illustrer cette approche en citant quelques exemples de problème, et des propositions associées.

3.4.1. Moment de fréquence

Un flux $\sigma = \langle a_1, a_2, \ldots, a_m \rangle$ définit implicitement un vecteur de fréquences $\mathbf{f} = (f_1, \ldots, f_n)$, où f_u représente le nombre d'occurrences d'un élément u dans σ.

Ce vecteur de fréquences peut aussi être assimilé à la distribution de probabilité empirique des éléments de Ω sur le flux

$$\mathbf{p} = (p_u)_{u \in \Omega} \quad \text{où} \quad p_u = \frac{f_u}{\sum_{v \in \Omega} f_v} = \frac{f_u}{m}.$$

Les moments de fréquence d'un flux sont des mesures statistiques importantes, introduites par Alon *et al*. [1996]. La valeur des moments de fréquences F_k fournit une quantification du taux de biais de la distribution du flux. Pour tout $k \geq 0$, le $k^{\text{ième}}$ moment de fréquence F_k du flux σ est défini par

$$F_k(\sigma) = \sum_{u \in \Omega} f_u^k.$$

Parmi les moments marquants, il est intéressant de relever que

– $F_0(\sigma)$ représente le nombre d'éléments distincts de σ (où par convention $0^0 = 0$ [Jacquelin, 2007]);

– $F_1(\sigma)$ représente la longueur m de σ;

– $F_2(\sigma)$ représente « l'asymétrie » de la distribution de σ (*i.e.,* sa divergence par rapport à la loi uniforme);

– $F_\infty(\sigma)$ représente l'élément le plus fréquent dans σ [Flajolet, Martin, 1985].

L'algorithme proposé par Alon *et al*. [1996] n'est certes pas l'algorithme le plus précis pour résoudre ce problème, mais son principe a été largement réutilisé et étendus depuis [Bar-Yossef *et al*., 2002, Cormode *et al*., 2011, Anceaume, Busnel, 2014b]. Le fondement est de construire un estimateur non biaisé X dont la valeur attendue est égale au $k^{\text{ième}}$ moment de fréquence, et de réduire sa variance en exécutant en parallèle plusieurs instances de cet estimateur de base. Cet algorithme est une (ε, δ)-approximation du $k^{\text{ième}}$ moment de fréquence, mais il n'est pas optimal en complexité mémoire de l'ordre de $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta} k n^{1-1/k} (\log n + \log m))$ bits. Plus récemment, Cormode *et al*. [2011] ont proposé un algorithme permettant d'estimer les moments de fréquence dans le modèle des flux répartis continus (*cf.* Section 2.4).

3.4.2. Entropie

Intuitivement, l'entropie est une mesure du caractère aléatoire d'un flux σ [Basseville, Cardoso, 1995]. L'entropie $H(\sigma)$ est minimale (*i.e.,* égale à 0) quand tous les

éléments du flux sont identiques, et elle atteint son maximum quand tous les éléments du flux sont distincts. Formellement, l'entropie du flux σ est définie par

$$H(\sigma) = -\sum_{u\in\Omega} p_u \log p_u.$$

Par convention, il est établi ici que $0 \log 0 = 0$.

Par analogie, la norme de l'entropie d'un flux σ est définie par

$$F_H(\sigma) = \sum_{u\in\Omega} f_u \log f_u.$$

Ces métriques peuvent être utilisées pour détecter l'occurrence d'événements indésirables, tels que des intrusions dans la gestion du réseau IP [Lall *et al*., 2006]. Chakrabarti *et al*. [2007] ont proposé une solution pour estimer l'entropie lorsqu'elle est strictement inférieure à 1. De tels flux ont peu d'éléments avec une fréquence d'occurrence élevée alors que tous les autres éléments apparaissent avec une fréquence faible.

Entropie relative

L'entropie relative, aussi dénommée dans la littérature en tant que divergence de Kullback-Leibler (KL-divergence) [Kullback, Leibler, 1951], est une métrique robuste qui estime la différence statistique entre deux flux de données. La KL-divergence fait partie de la classe des distances de Ali-Silvey [Ali, Silvey, 1966] et possède la particularité intéressante de mettre en exergue un faible nombre de faux positifs quand deux flux de données sont significativement différents.

Etant donnés deux flux σ_1 et σ_2 ayant pour distribution de probabilité respectivement $\mathbf{p}$ et $\mathbf{q}$ sur Ω, la KL-divergence de σ_1 vers σ_2 est définie comme valeur attendue du rapport de vraisemblance de $\mathbf{p}$ par rapport à $\mathbf{q}$:

$$\mathcal{D}(\sigma_1||\sigma_2) = \sum_{u\in\Omega} p_u \log \frac{p_u}{q_u} = H(\sigma_1, \sigma_2) - H(\sigma_1),$$

où $H(\sigma_1, \sigma_2) = -\sum p_u \log q_u$ est l'entropie croisée de p vers q. La KL-divergence est donc nulle si et seulement si $\mathbf{p} = \mathbf{q}$.

Dans [Anceaume, Busnel, 2014b], nous avons proposé un algorithme qui est une (ε, δ)-approximation relative de la KL-divergence, utilisant

$$\mathcal{O}\left(\log n + \frac{1}{\varepsilon^2} + \left(\frac{1}{\varepsilon} + \frac{\mu}{\varepsilon^2}\log\frac{1}{\delta}\right)(\log n + \log m)\right)$$

bits de mémoire où $\mu = (\log m + \log e - 1)$ si $F_H^e \geq \frac{m^2}{\Delta m_e}$ (et $\mu = (n - \frac{1}{\varepsilon} - 1)$ sinon), Δ est une constante strictement positive, m_e représente le nombre d'occurrences des éléments non-surabondants dans le flux (*i.e.,* $m_e = m - \sum_{i\in\hat{B}} f_i$) et F_H^e est la norme

de l'entropie pour ces éléments uniquement. Nous avons proposé également une extension de cet algorithme au modèle de flux répartis continus, ayant une complexité en communication de $O(s \log m \log n)$ bits (avec m représentant la longueur maximale des s flux entrants).

3.4.3. Éléments fréquents

Nous avons précédemment touché du doigt la problématique des éléments surabondants. La détection de ces éléments très fréquents est d'autant plus fondamentale dans des domaines d'application tels que la sûreté de fonctionnement (notamment pour les attaques DDoS [Manjhi *et al*., 2005, Yi, Zhang, 2013]), ou les bases de données réparties (pour l'estimation de charge impliquée par une jointure entre clés surabondantes [Alon *et al*., 1999, Poosala, Ioannidis, 1996, Swami, Schiefer, 1994, Vengerov *et al*., 2015]).

Misra, Gries [1982] fournissent un algorithme d'échantillonnage élégant pour ce problème avec un paramètre de précision ε de la sortie. L'algorithme conserve un ensemble clé/valeur MG où les clés sont les éléments et les valeurs sont des compteurs d'occurrences. Pour chaque échantillon u, le compteur associé dans MG est augmenté s'il existe, sinon il est créé s'il y a de la place. Quand $\mid MG \mid \geq \lceil \frac{1}{\varepsilon} \rceil - 1$, pour chaque nouvel échantillon (*i.e.*, pas encore dans MG), tous les compteurs sont diminués. Si un compteur atteint 0, la paire est retirée de MG. Cet algorithme propose ainsi une ε-approximation avec une complexité mémoire de $O(\frac{1}{\varepsilon}(\log(m) + \log(n))$ bits. L'estimation de fréquence $\hat{f}_u$ est la valeur du compteur associé à u dans MG et 0 sinon. Cette estimation satisfait l'encadrement suivant : $f_j - \varepsilon m \leq \hat{f}_j \leq f_j$.

Par la suite, Metwally *et al*. [2005] proposent une amélioration déterministe de cette solution, fondée également sur des compteurs pour estimer la fréquence des éléments surabondants. Cet algorithme, appelé SPACESAVING, prend deux paramètres : Θ et ε, tels que $0 < \varepsilon < \Theta \leq 1$. Il maintient $\lceil 1/\varepsilon \rceil$ couples $\langle tuple, compteur \rangle$, et retourne tous les éléments dont la fréquence d'occurrences est supérieure ou égale à Θm. Metwally *et al*. [2005] ont prouvé, qu'après avoir reçu m éléments, l'approximation faite par l'algorithme sur la fréquence estimée $\hat{f}_u$ de l'élément fréquent u vérifie $\hat{f}_i - f_i \leq \varepsilon m$ pour tout $0 < \varepsilon < \Theta$.

Plus spécifiquement, le problème de détection (et d'identification) d'icebergs a été formalisé par Estand, Varghese [2003], puis adapté dans différents contextes [Manjhi *et al*., 2005, Zhao *et al*., 2010]. Ce problème consiste, pour un coordinateur, à identifier rapidement tout élément u excédant globalement une proportion Θ donnée du flux. Nous avons proposé dans [Anceaume *et al*., 2015] une extension de ce problème au cadre du modèle de flux répartis continus, et proposé une (ε, δ)-approximation permettant de résoudre le problème de la détection d'icebergs, pour tout $\varepsilon \in [0, 1]$ et $\delta \leq 1/2$, avec une faible complexité mémoire. De plus, nous avons fourni une borne supérieure précise sur le nombre de bits échangés par notre solution.

Pour un panorama plus exhaustif de ce cas d'usage spécifique, le lecteur intéressé pourra consulter l'excellent état de l'art proposé par Cormode, Hadjieleftheriou [2010].

3.5. Systèmes de gestion de flux de données

Les systèmes distribués de traitement de flux (SDTF) sont aujourd'hui considérés comme une technologie courante pour construire des architectures d'analyse en temps réel de données massives, tels que le pionnier Aurora [Abadi *et al*., 2003] ou les célèbres solutions récentes comme STREAM [B. Babcock *et al*., 2016], Apache Storm [2014], Apache Spark Streaming [2013] ou Apache Flink [2014]. Leur capacité à produire des résultats d'analyse avec des latences inférieures à la seconde, associée à leur évolutivité, en a fait le choix favori de nombreuses grandes sociétés de données.

Une application s'exécutant dans un SDTF est généralement modélisée sous la forme d'un graphe acyclique dirigé (appelé topologie) où les opérateurs de données, représentés par des nœuds, sont interconnectés par des flux de n-uplets contenant les données à analyser, représentés par des arcs. Le succès d'un tel système peut être attribuée à leur capacité d'exécuter des applications complexes passant à l'échelle sur des grappes de matériels standards. L'évolutivité est généralement atteinte à la phase de déploiement où chaque opérateur de données peut être parallélisé en plusieurs instances, chacune d'elles traitant un sous-ensemble des n-uplets transportés par le flux entrant de l'opérateur. Il est ainsi primordial d'équilibrer la charge entre les instances parallèle d'un opérateur, car cela permet une meilleure utilisation des ressources et donc l'obtention de débits plus importants et une réduction des temps de traitement des n-uplets. Ainsi, le provisionnement correct des ressources de calcul pour un SDTF est loin d'être une tâche insignifiante. Les concepteurs de systèmes doivent tenir compte de plusieurs facteurs : la complexité de calcul des opérateurs, les surcoûts induits par le système et les caractéristiques des flux d'entrée. Ce dernier aspect est souvent le plus critique, car les flux de données en entrée peuvent changer de façon imprévisible dans le temps, tant au niveau du débit que du contenu [Rivetti, Querzoni *et al*., 2015, Rivetti *et al*., 2016, J. Xu *et al*., 2014].

Avec l'apparition de MapReduce [Dean, Ghemawat, 2008], une manière différente de distribuer les opérateurs s'est développée. MapReduce a l'avantage de fournir un paradigme de programmation hautement parallèle facile d'utilisation. Les opérateurs sont simples à définir et la gestion du parallélisme est cachée aux utilisateurs. Dans ce contexte, certains SDTF basés sur le cadre MapReduce ont été proposés et fournissent différentes visions du traitement des flux. Une étude intéressante proposée par Kotto-Kombi *et al*. [2015] vise à explorer les différents problèmes rencontrés par les systèmes de gestion de flux de données existants, à apporter une vision plus précise du traitement continu de requêtes et à faciliter la comparaison des différentes solutions connues. Les auteurs proposent ainsi une classification des SDTF selon 3 critères principaux: (1) fondés sur un workflow, sur MapReduce ou hybride, (2) fenêtrage par lots ou or incrémental et (3) répartis ou centralisés sur des multi-cœurs.

Par exemple, Aurora et STREAM sont des systèmes centralisés multi-coeur, reposant sur des workflow mais se différentient en étant respectivement à fenêtre par lots ou incrémentale. Storm, T-storm, Spark Streaming et Flink sont toutes des solutions réparties, les deux premières étant à fenêtre par lots, et reposent sur un workflow, alors que les deux suivantes sont en fenêtre incrémentale et reposent respectivement sur une approche MapReduce et hybride. Le choix du SDTF utilisé dépendra donc fortement de l'application et de la nature des flux de données à traiter.

4. Perspectives

Nous proposons, en guise de conclusion, de revisiter les diverses facettes du modèle de flux, à la lumière des travaux présentés précédemment, pour présenter les perspectives de recherche ouvertes.

4.1. Vers la décentralisation des modèles répartis

Compte tenu du nombre potentiellement croissant de sources de données et d'unités de traitement dans les centres de données, il apparaît indispensable de développer des solutions entièrement décentralisées, répartissant la charge entre les participants et permettant de passer à l'échelle en terme de nombre d'entités déployées. Par conséquent, dans le contexte des flux répartis à grande échelle, l'approche collaborative dans la conception de systèmes dédiés apparaît nécessaire. En effet, ces derniers sont la plupart du temps développés dans un cadre applicatif précis. La généralisation de ces déploiements est pertinente comme par exemple le projet [SocioPlug, 2013], dont l'objectif est fournir des modèles pour programmer, interroger et sécuriser des fédérations d'entités à faibles ressources en préservant la symétrie d'utilisation, l'équité et le respect de la vie privée. Ces fédérations doivent ainsi être capables de délivrer un service sans nécessiter d'intervention extérieure (humaine ou matérielle), la collaboration devant être conçue de manière *auto-organisante*. Cette dernière notion a pour optique l'émergence d'un comportement général et global du système, à partir d'acteurs indépendants et possédant uniquement des informations restreintes et locales du système. Ainsi, la charge est répartie sur l'ensemble du système, et l'augmentation du nombre de participants n'entraîne pas la formation de goulots d'étranglement, ni de points de défaillance critique (par l'unicité d'un serveur), couramment observés sur les systèmes centralisés.

L'attrait de la décentralisation et de l'auto-organisation a été fortement en vogue depuis le début des années 2000 [Dolev *et al.*, 2007, Ratnasamy *et al.*, 2001, Gordon, 2010]. L'ouverture récente du monde des réseaux à des solutions auto-organisantes illustre pleinement la convergence des communautés systèmes, réseaux, télécommunications, bases de données, *etc.* Soutenus par 3GPP (*3rd Generation Partnership Project*) et le NGMN (*Next Generation Mobile Networks*), deux exemples récents intègrent des solutions de réseaux auto-organisants : LTE (*Long Term Evolution*), ainsi qu'une amélioration de l'UMTS (*Universal Mobile Telecommunications System*) [Brunner, Flore., 2009, Feng, Seidel, 2008]).

Dans le contexte présenté précédemment, il devient nécessaire de s'abstraire de l'organe coordinateur des modèles à flux répartis (*cf.* sections 2.3 et 2.4), en proposant des algorithmes totalement décentralisés. L'objectif est d'améliorer la sûreté de fonctionnement de nos algorithmes en s'affranchissant du point central de défaillance de ces algorithmes. De nombreux algorithmes répartis permettent de mettre en œuvre de manière décentralisée des mémoires partagées et du consensus, avec de très bonnes garanties, tels que Paxos [Lamport, 1998] ou les travaux de [Chandra, Toueg, 1996, Castro, Liskov, 2002]. Ces protocoles ne sont cependant que modérément extensibles dans un contexte à large échelle. La proposition de nouvelles extensions décentralisées dans le cadre des modèles à flux répartis est une piste d'investigation stimulante. De surcroît, cette dernière doit porter une attention spécifique à la recherche d'optimalité en terme de surcoût de communication.

4.2. Intégrer la sémantique des données

Une seconde perspective, orthogonale et complémentaire à la section précédente, repose sur l'intégration d'informations sémantiques dans les algorithmes de traitement de flux de données. En effet, la grande majorité des solutions proposées dans ce contexte ne présuppose aucune connaissance du contenu des flux *a priori*. Tous les modèles reposent sur une abstraction des éléments qui les composent.

Néanmoins, l'ajout d'informations sémantiques au moment même de la conception des algorithmes permettrait assurément d'optimiser les futures solutions. L'objectif serait ainsi de concevoir des méthodologies et algorithmes innovants, s'appuyant sur des approches développées dans la « science des données », laquelle partage les richesses de la statistique et de l'informatique.

La première piste concrète envisagée repose sur les données massives en santé (DMS), lesquelles ne se limitent pas à un grand volume de données ou de sources. Les données biomédicales se réfèrent à des notions de complexités, d'enjeux, et d'opportunités portées par l'analyse combinée de ces données [National Institutes of Health, 2015]. En recherche biomédicale, ces sources de données sont diverses, hétérogènes, désorganisées, massives et multimodales, étant produites par diverses entités (chercheurs, praticiens, établissements hospitaliers, appareils nomades, *etc.*). Les DMS ont radicalement changé la manière dont l'information est traitée par les chercheurs en médecine [Toubiana, Cuggia, 2014], les données collectées et produites étant à présent potentiellement partagées et réutilisables. Cette quantité d'information promet l'émergence de nouvelles fonctionnalités médicales, incluant l'aide à la décision clinique, la pharmacovigilance, l'épidémiologie, la gestion sanitaire [Raghupathi, Raghupathi, 2014]. De nouvelles métriques adéquates pourraient être proposées pour traiter en parallèle de grands volumes de données, en temps réel ou non. Par exemple, il serait envisageable de concevoir des agrégats reposant sur des espaces de Hilbert [Berlinet, Thomas-Agnan, 2004].

Une seconde voie de recherche intéressante incluant un enrichissement sémantique concerne les données transitant sur les réseaux mobiles. Le développement d'applica-

tions dans ce contexte a majoritairement suivi un patron de conception sous-optimal, se limitant le plus souvent à répliquer des services préexistants sur les réseaux filaires tels que les jeux en ligne ou les interactions ubiquitaires avec les réseaux sociaux. Quelques exceptions notables ont vu le jour comme des assistants de navigation, lesquels utilisent des informations générées par les capteurs embarqués dans les appareils nomades (en particulier, les récepteurs GPS), afin de faciliter la mobilité des utilisateurs [Raptis *et al.*, 2005]. Bien que cette approche améliore la productivité et la satisfaction des utilisateurs, les capacités grandissantes de ces appareils sont souvent sous-utilisées (en terme de puissance de calcul, de communication ou de capteurs).

Nous avons dernièrement conjecturé que le déploiement de services répartis pour appareils mobiles nécessite un changement de paradigme [Busnel *et al.*, 2013]. Nous y avons notamment proposé un programme de recherche, ayant un impact potentiel à la fois scientifique et sociétal. L'objectif repose sur trois concepts: *(i)* la possibilité de partager les ressources calculatoires avec les autres appareils à proximité, *(ii)* une communication « permissive » dont les motifs protocolaires seront appris en ligne, et finalement, *(iii)* la mise en œuvre d'un stockage coopératif de la connaissance commune acquise précédemment. Ces pistes de recherches se situent au croisement des réseaux ad-hoc, du calcul ubiquitaire et du traitement de données à la volée. Au niveau social, la mise à disposition de bibliothèques logicielles et de nouveaux standards de communication ouvrirait la voir à un déploiement viral d'une nouvelle classe d'applications, qui pourrait contribuer à améliorer la qualité d'expérience des utilisateurs.

Dans des travaux futurs, un comportement adaptatif à la qualité d'expérience pourrait être développé par les fournisseurs de services en fonction du débit prédit [Samba *et al.*, 2016]. Un des problèmes ouverts de cette approche repose sur la prise en compte des brusques changements de couverture durant une connexion, lors d'un changement de relais par exemple. La connaissance du contexte environnemental de l'utilisateur considéré représente une information sémantique primordiale à prendre en considération. Des méthodes additionnelles tolérantes aux évolutions des conditions de couverture et permettant la prédiction de la mobilité (*e.g.,* par des études de corrélations ou d'apprentissage statistique) représentent ainsi de bons candidats pour répondre à ce défi.

4.3. Ouverture vers d'autres modèles de flux

En dernier lieu, une perspective de recherche particulièrement attractive se trouve au croisement du traitement des flux de données et de l'algorithmique des graphes. En effet, les systèmes à large échelle et les grands réseaux de communications sont souvent représentés par des graphes de communication ou d'interaction sociales par exemple. La dimension de ces objets explose littéralement dans les systèmes à grande échelle, tels que les réseaux de téléphonie mobile, les réseaux sociaux ou le routage sur internet, rendant les solutions classiques de la théorie des graphes inadaptées.

Par exemple, le modèle des flots de liens permet de représenter l'activité d'un système complexe. Au cours du temps, un lien apparaît lorsque deux entités du système

entrent en interaction. Ces flots de liens peuvent notamment représenter la dynamique des interactions humaines, lesquelles impliquent de nombreuses applications, de la diffusion épidémique aux modèles de mobilités [Starnini *et al*., 2013]. L'étude de ces flots de liens, par leur dynamique, la détection de motifs récurrents, ou le groupement de sommets (*clustering*), représente un défi passionnant et pertinent. [Viard *et al*., 2015] proposent par exemple d'étudier ces grands graphes dynamiques (*i.e.,* dont la topologie évolue au cours du temps) à travers le prisme des sciences des réseaux et des propriétés structurelles de ces objets.

Seulement, l'analyse de ces grands graphes en temps réel nécessite de nombreuses ressources de calcul. L'une des approches en vogue récemment repose sur le groupement de sommets (respectivement de liens) pour diviser le traitement sur des opérateurs parallèles [Chen *et al*., 2015, Vaquero *et al*., 2013, Xie *et al*., 2014]. Cette technique de pré-traitement des graphes permet d'améliorer l'équilibrage de charge entre opérateurs. Cependant, le coût du partitionnement du graphe complet est absolument prohibitif. De nouvelles contributions pour découper le graphe à la volée ont ainsi vu le jour récemment, avec un paradigme issu du modèle flux [N. Xu *et al*., 2014, Petroni *et al*., 2015]. Ces algorithmes permettent ainsi de répartir les sommets (respectivement les arcs) du graphe rapidement, de manière adaptative et naturellement parallélisable. Néanmoins, des stratégies trop naïves conduisent rapidement à répliquer une grande proportion des éléments en entrée sur différentes instances, impliquant de fait un besoin de synchronisation important pour garantir la cohérence des données, entamant sérieusement les performances [Petroni, 2015].

L'élaboration et l'analyse de solutions innovantes dans ces nouveaux modèles représentent un défi motivant, enthousiasmant et séduisant.

Pour conclure, la quantité de questions ouvertes dans le champ de l'analyse et du traitement des flux de données à la volée reste pléthorique. Les cas d'utilisation semblent illimités dans notre société où le numérique prend une place de plus en plus importante tous les jours [Gartner Inc., 2015].

Remerciements

Ces travaux ont été partiellement financés par le projet ANR SocioPlug (ANR-13-INFR-0003). Ce panorama a pu être réalisé à la suite de travaux qui ont été réalisé avec de nombreux co-auteurs, que je souhaite remercier ici. Je souhaite également témoigner d'une reconnaissance particulière à Emmanuelle Anceaume et Nicoló Rivetti di Val Cervo, avec qui nous avons principalement développé cette activité de recherche.

Bibliographie

Abadi D. J., Carney D., Çetintemel U., Cherniack M., Convey C., Lee S. *et al.* (2003). Aurora: a new model and architecture for data stream management. *The International Journal on Very Large Data Bases (VLDB)*, vol. 12, n° 2.

Ali S. M., Silvey S. D. (1966). General Class of Coefficients of Divergence of One Distribution from Another. *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 28,

n° 1, p. 131–142.

Alon N., Gibbons P. B., Matias Y., Szegedy M. (1999). Tracking join and self-join sizes in limited storage. In *Proceedings of the 18th ACM Symposium on Principles of Database Systems (PODS)*, p. 10–20.

Alon N., Matias Y., Szegedy M. (1996). The space complexity of approximating the frequency moments. In *Proceedings of the 28th ACM Symposium on Theory of computing (STOC)*, p. 20–29.

Anceaume E., Busnel Y. (2014a). Deviation estimation between distributed data streams. In *Proceedings of the 10th European Dependable Computing Conference, Newcastle (EDCC)*, p. 35–45.

Anceaume E., Busnel Y. (2014b). A distributed information divergence estimation over data streams. *IEEE Transaction on Parallel and Distributed System*, vol. 25, n° 2, p. 478–487.

Anceaume E., Busnel Y. (2017). Lightweight metric computation for distributed massive data streams. *Transaction on Large-Scale Data- and Knowledge-Centered Systems*, vol. 33, p. 1–39.

Anceaume E., Busnel Y., Gambs S. (2010). Uniform and Ergodic Sampling in Unstructured Peer-to-Peer Systems with Malicious Nodes. In *Proceedings of the 14th International Conference on Principles of Distributed Systems (OPODIS)*, vol. 6490, p. 64–78.

Anceaume E., Busnel Y., Gambs S. (2012). Ankle: Detecting attacks in large scale systems via information divergence. In *Proceedings of the 9th European Dependable Computing Conference (EDCC).*

Anceaume E., Busnel Y., Rivetti N., Sericola B. (2015). Identifying global icebergs in distributed streams. In *Proceedings of the 34th IEEE Symposium on Reliable Distributed Systems (SRDS)*, p. 266–275.

Anceaume E., Busnel Y., Sericola B. (2013). Uniform node sampling service robust against collusions of malicious nodes. In *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).*

Apache Flink. (2014). *http://flink.apache.org*. The Apache Software Foundation. (En ligne; accédé en Novembre 2018)

Apache Spark Streaming. (2013). *https://spark.apache.org/streaming/*. The Apache Software Foundation. (En ligne; accédé en Novembre 2018)

Apache Storm. (2014). *http://storm.apache.org*. The Apache Software Foundation. (En ligne; accédé en Novembre 2018)

Arackaparambil C., Brody J., Chakrabarti A. (2009). Functional monitoring without monotonicity. In *Proceedings of the 36th ACM International Colloquium on Automata, Languages and Programming: Part 1 (ICALP).*

Arasu A., Manku G. S. (2004). Approximate counts and quantiles over sliding windows. In *Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS).*

Bar-Yossef Z., Jayram T. S., Kumar R., Sivakumar D., Trevisan L. (2002). Counting distinct elements in a data stream. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques (RANDOM)*, p. 1–10. Springer-Verlag.

Basseville M., Cardoso J.-F. (1995). On entropies, divergences, and mean values. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT).*

B. Babcock A. A. anbd, Babu S., Cieslewicz J., Datar M., Ito K., Motwani R. *et al.* (2016). Stream: The stanford data stream management system. In *Data stream management*, p. 317–336.

Berlinet A., Thomas-Agnan C. (2004). *Reproducing kernel hilbert spaces in probability and statistics*. Springer-Verlag.

Bortnikov E., Gurevich M., Keidar I., Kliot G., Shraer A. (2009). Brahms: Byzantine Resilient Random Membership Sampling. *Computer Networks*, vol. 53, p. 2340–2359.

Brunner C., Flore. D. (2009). Generation of pathloss and interference maps as son enabler in deployed umts networks. In *Proceedings of the 69th IEEE Vehicular Technology Conference (VTC Spring).*

Busnel Y., Cruz N., Gillet D., Holzer A., Miranda H. (2013). Reinventing mobile community computing and communication. In *Proceedings of the 12th IEEE International Conference on Ubiquitous Computing and Communications (IUCC)*, p. 1450–1457.

Carter J. L., Wegman M. N. (1979). Universal classes of hash functions. *Journal of Computer and System Sciences*, vol. 18, p. 143–154.

Castro M., Liskov B. (2002). Practical byzantine fault tolerance and proactive recovery. *ACM Transaction on Computer System*, vol. 20, n° 4, p. 398–461.

Chakrabarti A., Cormode G., McGregor A. (2007). A near-optimal algorithm for computing the entropy of a stream. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 328–335.

Chan H.-L., Lam T.-W., Lee L.-K., Ting H.-F. (2012). Continuous monitoring of distributed data streams over a time-based sliding window. *Algorithmica*, vol. 62, n° 3-4, p. 1088-1111.

Chandola V., Banerjee A., Kumar V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, vol. 41, n° 3, p. 1–58.

Chandra T. D., Toueg S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of ACM*, vol. 43, n° 2, p. 225–267.

Charikar M., Chen K., Farach-Colton M. (2004). Finding frequent items in data streams. *Elsevier Theoretical Computer Science*, vol. 312, n° 1, p. 3-15.

Chauvet G. (2012). On a characterization of ordered pivotal sampling. *Bernoulli*, vol. 18, n° 4, p. 1320–1340.

Chen R., Shi J., Chen Y., Chen H. (2015). Powerlyra: Differentiated graph computation and partitioning on skewed graphs. In *Proceedings of the 10th European Conference on Computer Systems (EuroSys)*, p. 1:1–1:15.

Cormode G. (2011). Continuous distributed monitoring: A short survey. In *Proceedings of the 1st International Workshop on Algorithms and Models for Distributed Event Processing (AlMoDEP).*

Cormode G., Hadjieleftheriou M. (2010, février). Methods for finding frequent items in data streams. *The VLDB Journal*, vol. 19, n° 1, p. 3–20.

Cormode G., Muthukrishnan S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, vol. 55, n° 1, p. 58–75.

Cormode G., Muthukrishnan S., Yi K. (2008). Algorithms for distributed functional monitoring. In *Proceedings of the 19th ACM-SIAM Symposium On Discrete Algorithms (SODA).*

Cormode G., Muthukrishnan S., Yi K. (2011). Algorithms for distributed functional monitoring. *ACM Transaction on Algorithms*, vol. 7, n° 2, p. 21:1–21:20.

Cormode G., Yi K. (2012). Tracking distributed aggregates over time-based sliding windows. In *Proceedings of the 24th International Conference on Scientific and Statistical Database Management (SSDBM).*

Csernel B., Clerot F., Hebrail G. (2005). Traitement des flux de donnees. In *Proceedings of the 37emes Journees de Statistique (SFDS).*

Datar M., Gionis A., Indyk P., Motwani R. (2002). Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, vol. 31, n° 6, p. 1794–1813.

Dean J., Ghemawat S. (2008). Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, vol. 51, n° 1, p. 107–113.

Dolev D., Hoch E. N., Van Renesse R. (2007). Self-stabilizing and Byzantine-tolerant Overlay Network. In *Proceedings of the 11th International conference on Principles of Distributed Systems (OPODIS)*, p. 343–357. Springer-Verlag.

Douceur J. (2002). The Sybil Attack. In *Proceedings of the 1rst International Workshop on Peer-to-Peer Systems (IPTPS)*, p. 251–260.

Estand C., Varghese G. (2003). New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, vol. 21, n° 3, p. 270–313.

Feng S., Seidel E. (2008). *Self-organizing networks (son) in 3gpp long term evolution*. Rapport technique. Nomor Research GmbH.

Flajolet P., Martin G. N. (1985). Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, vol. 31, n° 2, p. 182–209.

Gartner Inc. (2015). *Hype Cycles.* http://www.gartner.com/technology/research/hype-cycles/. (En ligne; accédé en Juin 2016)

Gibbons P. B., Tirthapura S. (2001). Estimating simple functions on the union of data streams. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, p. 281-291.

Gibbons P. B., Tirthapura S. (2004). Distributed streams algorithms for sliding windows. *Theory of Computing Systems*, vol. 37, n° 3, p. 457-478.

Golab L., DeHaan D., Demaine E. D., Lopez-Ortiz A., Munro J. I. (2003). Identifying frequent items in sliding windows over on-line packet streams. In *Proceedings of the 3rd ACM SIGCOMM Internet Measurement Conference (IMC)*.

Gordon D. M. (2010). *Ant encounters: Interaction networks and colony behavior*. Princeton University Press.

Haung Z., Yi K., Zhang Q. (2012). Randomized algorithms for tracking distributed count, frequencies and ranks. In *Proceedings of the 31st ACM Symposium on Principles of Database Systems (PODS)*.

Jacquelin J. (2007). Zéro puissance zéro. *Quadrature*, vol. 66, p. 34–36.

Kane D. M., Nelson J., Woodruff D. P. (2010). An optimal algorithm for the distinct element problem. In *Proceedings of the 29th ACM Symposium on Principles of Database Systems (PODS)*.

Karamcheti V., Geiger D., Kedem Z., Muthuskrishnan S. (2005). Detecting malicious network traffic using inverse distribution of packet contents. In *Proceedings of the 1st Workshop on Mining Network Data (MineNet)*.

Kotto-Kombi R., Lumineau N., Lamarre P., Caniou Y. (2015). *Parallel and Distributed Stream Processing: Systems Classification and Specific Issues*. Rapport technique. Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS).

Krishnamurthy B., Sen S., Zhang Y., Chen Y. (2003). Sketch-based change detection: Methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM Internet Measurement Conference (IMC)*, p. 234–247.

Kullback S., Leibler R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, vol. 22, n° 1, p. 79–86.

Lakhina A., Crovella M., C.Diot. (2005). Mining anomalies using traffic feature distributions. In *Proceedings of the 28th ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*.

Lall A., Sekar V., Ogihara M., Xu J., Zhang H. (2006). Data streaming algorithms for estimating entropy of network traffic. In *Proceedings of the 27th joint International conference on Measurement and modeling of computer systems (SIGMETRICS)*. ACM.

Lamport L. (1998). The part-time parliament. *ACM Transaction on Computer System*, vol. 16, n° 2, p. 133–169.

Liu Z., Radunovic B., Vojnovic M. (2012). Continuous distributed counting for non-monotonic streams. In *Proceedings of the 31st ACM Symposium on Principles of Database Systems (PODS)*.

Lynch C. (2008). Big data: How do your data grow? *Nature*, vol. 455, n° 7209, p. 28–29.

Lynch N. (1996). *Distributed algorithms*. Morgan Kaufmann Publishers.

Manjhi A., Shkapenyuk V., Dhamdhere K., Olston C. (2005). Finding (recently) frequent items in distributed data streams. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*.

Mayer-Schönberger V., Cukier K. (2013). *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt.

Metwally A., Agrawal D., El Abbadi A. (2005). Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th International Conference on Database Theory (ICDT)*.

Misra J., Gries D. (1982). Finding repeated elements. *Science of Computer Programming*, vol. 2, n° 2, p. 143–152.

Muthukrishnan S. (2005). *Data streams: Algorithms and applications*. Now Publishers Inc.

National Institutes of Health. (2015). *Data Science at NIH: What is Big Data?* https://datascience.nih.gov/bd2k/about/what. (En ligne; accédé en Juin 2016)

Petroni F. (2015). *Mining at scale with latent factor models for matrix completion*. Thèse de doctorat non publiée, Sapienza University of Rome.

Petroni F., Querzoni L., Daudjee K., Kamali S., Iacoboni G. (2015). HDRF: Stream-Based Partitioning for Power-Law Graphs. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, p. 243–252.

Poosala V., Ioannidis Y. E. (1996). Estimation of query-result distribution and its application in parallel-join load balancing. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB)*, p. 448–459.

Raghupathi W., Raghupathi V. (2014). Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, vol. 2, n° 1, p. 1–10.

Raptis D., Tselios N., Avouris N. (2005). Context-based design of mobile applications for museums: a survey of existing practices. In *Proceedings of the 7th ACM International Conference on Human Computer Interaction with Mobile Devices & Services (MobileHCI)*, p. 153–160.

Ratnasamy S., Francis P., Handley M., Karp R., Shenker S. (2001). A scalable content-addressable network. In *Proceedings of the 24th ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*.

Rivetti N., Anceaume E., Busnel Y., Querzoni L., Sericola B. (2016). Proactive Online Scheduling for Shuffle Grouping in Distributed Stream Processing Systems. In *Proceedings of the 13th ACM/IFIP/USENIX International Conference on Middleware (Middleware)*.

Rivetti N., Busnel Y., Mostéfaoui A. (2015). Efficiently summarizing data streams over sliding windows. In *Proceedings of the 14th IEEE International Symposium on Network Computing and Applications (NCA)*, p. 151–158.

Rivetti N., Querzoni L., Anceaume E., Busnel Y., Sericola B. (2015). Efficient key grouping for near-optimal load balancing in stream processing systems. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems (DEBS)*, p. 80–91.

Salem O., Vaton S., Gravey A. (2010). A scalable, efficient and informative approach for anomaly-based intrusion detection systems: theory and practice. *International Journal of Network Management*, vol. 20, n° 5, p. 271–293.

Samba A., Busnel Y., Blanc A., Dooze P., Simon G. (2016). Throughput prediction in cellular networks: Experiments and preliminary results. In *Actes de la 1ères Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication (CoRes).*

SocioPlug. (2013). *Cloud social sur des réseaux de plugs, pour un accès à l'information symétrique et respectueux de la vie privée.* http://socioplug.univ-nantes.fr/ – Agence Nationale pour la Recherche – ANR-13-INFR-003. (En ligne; accédé en Juin 2016)

Starnini M., Baronchelli A., Pastor-Satorras R. (2013). Modeling human dynamics of face-to-face interaction networks. *Physical Review Letters*, vol. 110, p. 168701–168705.

Swami A., Schiefer K. B. (1994). On the estimation of join result sizes. In *Proceedings of the 4th International Conference on Extending Database Technology: Advances in Database Technology (EDBT)*, p. 287–300.

Toubiana L., Cuggia M. (2014). Big data and smart health strategies: findings from the health information systems perspective. *Yearbook of Medical Informatics*, vol. 9, n° 1, p. 125–127.

Vaquero L., Cuadrado F., Logothetis D., Martella C. (2013). Adaptive partitioning for large-scale dynamic graphs. In *Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC)*, p. 35:1–35:2.

Vengerov D., Menck A. C., Zait M., Chakkappen S. P. (2015). Join size estimation subject to filter conditions. *VLDB Endowment – Proceedings of the 41st International Conference on Very Large Data Bases (VLDB)*, vol. 8, n° 12, p. 1530–1541.

Viard J., Latapy M., Magnien C. (2015). Revealing contact patterns among high-school students using maximal cliques in link streams. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, p. 1517–1522.

Xie C., Yan L., Li W.-J., Zhang Z. (2014). Distributed power-law graph computing: Theoretical and empirical analysis. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, p. 1673–1681.

Xu J., Chen Z., Tang J., Su S. (2014). T-storm: Traffic-aware online scheduling in storm. In *Proceedings of the 34th IEEE International Conference on Distributed Computing Systems (ICDCS)*, p. 535–544.

Xu N., Chen L., Cui B. (2014). LogGP: A Log-based Dynamic Graph Partitioning Method. *VLDB Endowment – Proceedings of the 40th International Conference on Very Large Data Bases (VLDB)*, vol. 7, n° 14, p. 1917–1928.

Yi K., Zhang Q. (2013). Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica*, vol. 65, p. 206–223.

Zhang L., Guan Y. (2007). Variance estimation over sliding windows. In *Proceedings of the 26h ACM Symposium on Principles of Database Systems (PODS).*

Zhao Q., Lall A., Ogihara M., Xu J. (2010). Global iceberg detection over distributed streams. In *Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE).*