Outil autonome de surveillance de grilles

Laurent Baduel * — Satoshi Matsuoka *,**

** Tokyo Institute of Technology*
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
baduel@smg.is.titech.ac.jp

*** National Institute of Informatics*
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
matsu@is.titech.ac.jp

RÉSUMÉ. *Les grilles de calculs modernes sont devenues particulièrement complexes. Le déploiement et la maintenance des systèmes sont des tâches difficiles qui réclament de nombreux efforts de la part des administrateurs. Nous pensons qu'un réseau pair-à-pair de recouvrement est une base solide pour l'implantation de fonctions de comportements autonomes sur des grilles. Cet article présente la construction d'un outil de surveillance de grilles décentralisé, passant à l'échelle, et efficace. Les composants de cette application négocient au travers d'un réseau pair-à-pair pour fournir un comportement autonome. Notre solution se base sur un protocole de bavardages dirigé dans un graphe hiérarchique, acyclique, et orienté de façon à rapidement disséminer les informations tout en limitant le nombre de messages. L'architecture de cette application est détaillée ; ses performances sont présentées et analysées.*

ABSTRACT. *Grids have become very complex. It makes the deployment and maintenance of systems a difficult task requiring lots of efforts from administrators. We believe that peer-to-peer overlay networks are a valuable basis to support some of the main issues of autonomic computing in the particular case of grids. This article presents the construction of an autonomous, decentralized, scalable, and efficient grid monitoring system. The components of this application negotiate through a peer-to-peer network to provide autonomic behaviors. Our solution is based on a gossip broadcast protocol upon a hierarchical, directed, and acyclic graph to rapidly diffuse information in the system while limiting the number of messages. The software architecture is detailed, and then the first results of its performance are presented and analyzed.*

MOTS-CLÉS : *surveillance, calcul autonome, réseau pair-à-pair, protocole de bavardages.*

KEYWORDS: *monitoring, autonomic computing, peer-to-peer network, gossip protocol.*

DOI:10.3166/ISI.12.3.85-104

RSTI - ISI – 12/2007. Systèmes d'information pair-à-pair, pages 85 à 104

1. Introduction

De nos jours les systèmes issus des technologies de l'information avec les infrastructures de communication et leurs applications de calculs sans cesse grandissantes créent une complexité exponentielle dans leurs opérations de développement et de maintenance. Les systèmes modernes à grande échelle ne permettent plus d'organisations centralisées qui réclament une configuration, un déploiement, et une administration manuels. L'automatisation d'opérations-clés doit être introduite dans de tels systèmes de façon à libérer les développeurs et administrateurs de tâches pénibles et répétitives.

Nous proposons de relever ce défi en utilisant une architecture de calcul autonome dans laquelle les communications sont assurées par un réseau pair-à-pair de recouvrement. Les systèmes pair-à-pair ont déjà prouvé leur efficacité dans de nombreux domaines de l'informatique distribuée : le calcul distribué à grande échelle, SETI@Home (Anderson *et al.*, 2002) ; le stockage de données persistantes à grande échelle, (Rhea *et al.*, 2003) ; et surtout le partage de fichiers en ligne, Napster (Napster, n.d.), eMule (The eMule Project, n.d.), BitTorrent (BitTorrent, n.d.), etc. Les réseaux pair-à-pair offrent également un moyen de communication précieux dans des environnements de grilles de calculs grâce à leur faculté de passage à l'échelle, leur résistance aux fautes, et leur capacité à recouvrir différents domaines administratifs.

Les contributions de cet article sont *(1)* la description de l'utilisation d'un réseau pair-à-pair de recouvrement organisé hiérarchiquement qui fournit une couche de communication adaptée à l'autogestion d'un système autonome sur grille, et *(2)* la conception et l'évaluation d'un outil de surveillance (*monitoring*) décentralisé et passant à l'échelle dédié aux grilles de calculs. Cette application démontre la viabilité de notre approche. Pour disséminer rapidement l'information produite par les senseurs dans l'intégralité du système nous utilisons un protocole de bavardages (*gossip protocol*) qui est un mécanisme décentralisé, tolérant aux pannes, et passant à l´échelle. L'efficacité de ce protocole est accrue par l'organisation des composants de l'application en graphe hiérarchique, orienté et acyclique qui limite le nombre de messages nécessaires pour diffuser l'information. Les performances de cet outil de surveillance sont présentées et commentées.

Le reste de cet article est organisé comme suit. La section 2 présente les systèmes autonomes et les réseaux pair-à-pair de recouvrement comme solution pour fournir une autogestion dans les grilles de calculs. La section 3 décrit la conception d'un système de surveillance pour grilles selon ces principes ; des résultats d'évaluation sont présentés. Dans la section 4 nous présentons les travaux relatifs à la surveillance de grille et leur adaptabilité. Finalement la section 5 conclut l'article et présente nos perspectives.

2. Contexte

Le calcul autonome (*autonomic computing*) et les réseaux pair-à-pair de recouvrement peuvent être combinés pour offrir un cadre efficace pour la construction d'applications à large échelle.

2.1. *Les systèmes autonomes*

Le calcul autonome désigne de grands et complexes systèmes autogérés dans lesquels les composants (eux-mêmes autogérés) interagissent ensemble de manière à s'organiser pour fournir un comportement général satisfaisant à certaines conditions de performance. Les systèmes à grande échelle modernes qui sont distribués sur de multiples domaines administratifs ont atteint une complexité sans précédent. Le calcul autonome offre une solution à ce problème grâce à une automatisation accrue de certaines opérations qui libère les administrateurs de nombreuses activités pénibles.

Comme présenté dans (Kephart *et al.*, 2003) l'autogestion est généralement divisée en quatre catégories : *(1)* l'*autoconfiguration* qui prépare et déploie automatiquement le système ; *(2)* l'*auto-optimisation* qui règle le système pour obtenir les meilleures performances ; *(3)* l'*autosoin* qui détecte, diagnostique, et répare les dysfonctionnements ; et *(4)* l'*autoprotection* qui défend le système contre des attaques malicieuses et des pannes en cascade.

Chaque composant d'un système autonome est responsable de la gestion de son propre état ainsi que de ses interactions avec un environnement constitué de signaux et messages en provenance d'autres composants et du monde extérieur. Dans la première étape de son cycle de vie un composant autonome entre dans le système, s'enregistre auprès d'un service d'annuaire, et cherche à se connecter à des fournisseurs de services. Il devient alors opérationnel et entreprend son activité. En plus de son comportement fonctionnel, le calcul autonome exige une capacité d'auto-observation qui permette de déclencher une procédure d'optimisation de l'activité d'un composant.

Les mécanismes de communication entre composants d'un système autogéré revêtent une grande importance, non seulement pour l'échange de données fonctionnelles de l'application mais aussi au regard des messages échangés pour prodiguer de l'autonomie au système. En effet les composants autonomes interagissent continuellement entre eux, depuis leur création, jusqu'à l'arrêt de leur activité. Un tel comportement entraîne une préoccupation au sujet des mécanismes de communication entre composants.

2.2. *Une architecture pair-à-pair*

Le concept du calcul autonome est apparu pour aider les administrateurs à gérer de grands systèmes sophistiqués. Les grilles de calculs tombent précisément dans cette catégorie. Les grilles dressent des contraintes et des besoins particuliers notamment

au sujet de la communication. Une technique devenant de plus en plus pertinente est d'utiliser des réseaux pair-à-pair.

Les grilles et le pair-à-pair ont tous deux une approche identique dans l'accomplissement de leurs objectifs : l'utilisation de structures de recouvrement. Cependant nous faisons une distinction entre les approches des grilles et du pair-à-pair :

– les *grilles* fournissent une large quantité de services à une communauté de taille moyenne avec généralement une haute qualité de service. A cause de leurs organisations statiques et localement centralisées les grilles sont vulnérables aux fautes ;

– par opposition les *systèmes pair-à-pair* fournissent des services limités et spécialisés à un très grand nombre d'utilisateurs. Vus dans leur ensemble les systèmes pair-à-pair sont fortement résistant aux pannes mais ne fournissent qu'une faible qualité de service. Cette limitation résulte du fait que les services ne sont pas de nature à être fournis en masse, et se heurtent donc à divers problèmes tels que la volatilité des nœuds, la performance *best-effort* du réseau internet, etc.

Tel que rapportée par (Foster *et al.*, 2003a) la nature complémentaire des forces et faiblesses de ces deux approches suggère un intérêt réciproque de l'une vers l'autre qui tend à les rapprocher. Les buts majeurs des grilles actuelles sont d'augmenter leur capacité à passer à l'échelle et fournir une meilleure tolérance aux fautes. Symétriquement les systèmes pair-à-pair visent à élargir le spectre des services qu'ils proposent.

Les mécanismes de communication requis par un système autonome doivent être adaptés à l'environnement dans lequel il évolue. Les infrastructures de communication des composants internes des grilles actuelles sont fragilisées principalement par une organisation statique et l'absence de chemins alternatifs de communication. Ces infrastructures de communication ont besoin d'être soutenues par des recouvrements appropriés, passant à l'échelle, et résistant aux fautes. Les réseaux pair-à-pair sont un élément de réponse aux préoccupations des communications dans des environnements de grilles. Avec l'ajout de comportements spécifiques tels que la gestion autonome des connectivités, les réseaux pair-à-pair deviennent une solution efficace à ce problème. La figure 1 présente la manière dont nous organisons les applications sur une grille : une couche de communication pair-à-pair assure la communication entre les composants autonomes et leur permet d'interagir efficacement.

3. Un outil de surveillance pour les grilles

Surveiller un système consiste à observer des évènements et à les communiquer à qui est intéressé par eux. Deux systèmes sont communément présents sur une grille. Selon (Zanikolas *et al.*, 2005), un *système de surveillance* gère des données très dynamiques telles que la charge d'un processeur ou le débit d'un lien de communication. La grande dynamicité des données qu'il gère rend le système de surveillance différent du *système d'information* qui lui gère des données plus statiques comme par exemple la configuration matérielle d'un nœud. Toutefois les infrastructures de grilles bénéficient

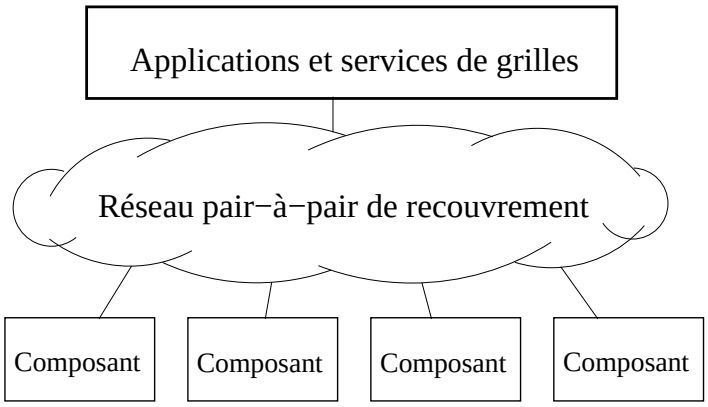

Figure 1. *Architecture d'application*

d'un système qui unie ces deux rôles. La connaissance globale résultant de l'union des deux systèmes aide à la planification des tâches et au partage des ressources : un outil de réservation peut être attaché à ce système pour distribuer les ressources et pour garantir une certaine qualité de service.

La surveillance des grilles de calculs est sujette à un intérêt grandissant. Les grilles à grandes échelles récentes ne supportent plus efficacement les outils de surveillance existants. La plupart des solutions actuelles ne sont que des adaptations d'outils de surveillance pour les grappes de calculs et posent des problèmes de passage à l'échelle et de tolérance aux pannes. Comme détaillé dans la section 4 le *Network Weather Service* est construit autour d'un contrôleur centralisé, et le *Globus Monitoring and Discovery System* souffre de problèmes de performance et de passage à l'échelle à cause de son architecture LDAP. Au contraire, l'outil de surveillance que nous proposons est parfaitement adapté aux environnements de grilles grâce à sa faculté à passer à l'échelle, à résister aux pannes, et bien sûr à s'autogérer.

3.1. *L'architecture de surveillance de grilles*

Le *Global Grid Forum (GGF)* a introduit la *Grid Monitoring Architecture (GMA)* (Tierney *et al.*, 2002) qui offre la flexibilité et la capacité de passer à l'échelle requises par un outil destiné aux grilles. Cette architecture décrit simplement trois types de composants comme présentés dans la figure 2 :

– les *producteurs* récupèrent diverses informations sur un appareil et les rendent disponibles aux autres composants GMA ; un exemple peut être un senseur qui rapporte la charge du processeur d'un nœud de calcul ;

– les *consommateurs* demandent des données surveillées pour lesquelles ils ont un intérêt ; par exemple un courtier qui cherche des ressources de calculs libres ;

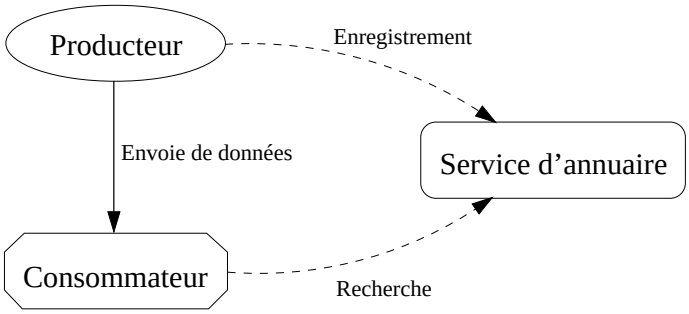

Figure 2. *Les composants de la GMA*

– le *service d'annuaire* assure la publication d'informations, la découverte de composants ainsi que de données surveillées. Un service d'annuaire est l'endroit où les producteurs annoncent leurs données et où les consommateurs annoncent leurs besoins.

En plus de ces trois composants de base, la GMA laisse la possibilité de construire des *composants intermédiaires*. Ces composants sont à la fois producteur et consommateur. Ils permettent une agrégation, un filtrage, une ré-émission, ou une diffusion des données fournies par un autre producteur. Souvent les outils de surveillance utilisent des *agrégateurs* qui collectent localement des données et les retransmettent sous forme agrégée. Ces composants aident au passage à l'échelle du système en évitant la formation de goulots d'étranglements au niveau des communications entre les producteurs et le service d'annuaire.

La capacité à passer à l'échelle de la GMA est assurée par la séparation des tâches de publication, de recherche, et de découverte. La GMA ne spécifie ni la manière dont les composants communiquent entre eux (contenu des messages ou protocole réseau), ni le format utilisé par le service d'annuaire pour stocker de l'information. Nous avons choisi de baser notre outil de surveillance de grilles autonome sur la GMA pour sa capacité à passer à l'échelle et pour sa flexibilité comme initialement présenté dans (Baduel *et al.*, 2007).

3.2. *Description du modèle*

Les principaux problèmes des outils de surveillance actuels sont leur centralisation et leur organisation statique (voir section 4). La centralisation introduit une faiblesse dans un système par le risque de voir le point de centralisation tomber en panne. De plus cela amène aussi fréquemment à des difficultés de passage à l'échelle en créant un goulot d'étranglement. Une organisation statique quant à elle rend l'administration des grands systèmes particulièrement difficile aux administrateurs.

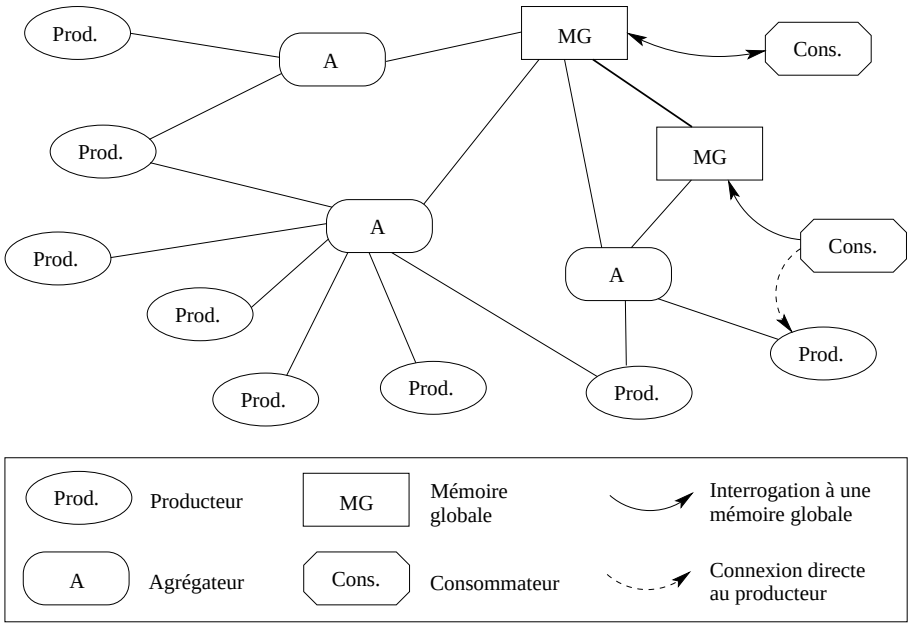

Figure 3. *Structure d'un outil de surveillance de grilles basé sur une architecture pair-à-pair*

Comme la GMA ne spécifie pas de mécanisme de communication pour l'interaction entre composants ou le transfert de données nous choisissons d'utiliser des communications pair-à-pair. La figure 3 présente la structure de notre outil de surveillance basée sur un réseau pair-à-pair. Les producteurs sont en contact avec des agrégateurs. Pour une diffusion rapide et sûre des données chaque producteur est en contact avec plusieurs agrégateurs (plus de détails plus loin). Les agrégateurs communiquent avec le service d'annuaire de la même façon. Le service d'annuaire est constitué de composants qui enregistrent des informations sur les producteurs et sur les données qui sont émises. Nous appelons ces composants *mémoires globales*. Le service d'annuaire est donc un service distribué. Pour les mêmes raisons qui font qu'un producteur référence plusieurs agrégateurs, un agrégateur est en contact avec plusieurs mémoires globales. Enfin les consommateurs s'adressent directement au service d'annuaire (c-à-d aux mémoires globales) pour rechercher les informations qui les intéressent. Un consommateur peut se renseigner sur la localisation d'un producteur pour un certain type d'évènement, et contacter directement ce producteur pour recevoir directement les données produites. Un consommateur peut aussi demander directement au service d'annuaire des données enregistrées par celui-ci.

En introduisant de la distribution et des liens de communication indirects au travers d'un réseau pair-à-pair, nous devons nous préoccuper des performances. Le défi majeur de notre outil de surveillance pour grilles est de diffuser le plus rapidement possible les données fournies par un producteur vers les mémoires globales. Des données trop anciennes sont inutiles car elles ne reflètent plus l'état d'un nœud qui peut avoir radicalement changé. Un système de surveillance se doit également de ne consommer que peu de ressources. Les activités de surveillance ne doivent avoir aucun effet sur l'exécution des applications sur la grille. Pour cela la consommation de temps processeur, de mémoire, et de bande passante doit rester significativement basse.

3.3. *Autogestion du système*

En accord avec les principales recommandations de (White *et al.*, 2004), nous traitons les quatre propriétés d'un système autogéré présentées dans la section 2.1 comme suit.

3.3.1. *Autoconfiguration*

Quand un nouveau composant entre dans le système il doit suivre quelques étapes. D'abord le composant entre dans le réseau pair-à-pair. Ensuite il peut éventuellement télécharger le code qu'il souhaite exécuter au sein même du réseau pair-à-pair ; d'autres pairs peuvent fournir ce service. Enfin le composant recherche les composants appropriés avec qui il doit interagir en tenant compte de leur efficacité.

Illustrons cette procédure avec le cas d'un nouveau producteur qui entre dans l'outil de surveillance. En premier lieu ce nouveau composant cherche à rejoindre le réseau de pairs. Le système offre deux façons de découvrir ce réseau : *(1)* en diffusant un appel dans le réseau local (ce service est souvent proposé par les bibliothèques de pair-à-pair), ou *(2)* en assumant l'existence d'un groupe de démarrage constitué de nœuds supposés être continuellement joignables. Les adresses de ces nœuds sont enregistrées sous une seule entrée dans un DNS. Ce second mécanisme est nécessaire à un pair qui appartient à un réseau privé sur lequel aucun nœud du réseau pair-à-pair n'est présent. Le problème du démarrage a lieu dans tous les réseaux pair-à-pair décentralisés.

Tous les appareils pouvant être surveillés n'ont pas forcément à implanter leurs propres senseurs mais peuvent en récupérer sur le réseau. Par exemple les senseurs des nœuds de calculs sur une grille sont relativement similaires quel que soit leur domaine administratif : ils fournissent des données sur la charge du processeur, le débit de communication, le système d'opération, etc. Grâce à la faculté de certains réseaux pair-à-pair à rechercher et télécharger du code publié à distance, un producteur peut automatiquement récupérer le code d'un senseur depuis un autre producteur.

Enfin le nouveau producteur cherche les composants avec qui il peut interagir : les agrégateurs (au travers desquels il communique avec les mémoires globales). Le producteur demande au réseau pair-à-pair de trouver un nombre d'agrégateurs prédéfini par l'administrateur. Ensuite le producteur mesure le temps d'aller-retour de ses

messages (*Round Trip Time (RTT)*) vers les agrégateurs. Nous utilisons le RTT comme métrique pour exprimer la proximité des pairs : notre supposition est qu'un RTT faible signifie que les deux pairs sont proches et font partie d'une même grappe, inversement un RTT élevé signifie que les pairs sont éloignés et qu'ils appartiennent à deux domaines distincts de la grille. Le producteur garde contact avec un sous-ensemble des agrégateurs formé *(1)* des agrégateurs « proches » (les plus faibles RTT) de façon à communiquer efficacement, mais aussi *(2)* de quelques agrégateurs « éloignés » (plus forts RTT) pour communiquer ses données vers des domaines distants. Ainsi un composant producteur est automatiquement configuré pour diffuser ces données efficacement au travers de la grille.

3.3.2. *Auto-optimisation*

Une politique évidente d'auto-optimisation est le dimensionnement dynamique de la taille du service en fonction de la variation de sa charge. Cela consiste à ajouter ou supprimer des composants du service en correspondance avec l'augmentation ou la diminution de la charge du service. Lorsqu'un composant devient surchargé il doit rediriger une partie de son travail vers un composant apte à traiter cette surcharge. Si aucun autre composant n'est susceptible d'accomplir ce travail supplémentaire, le composant surchargé demande la création d'un nouveau composant sur lequel il va déléguer une partie de ses activités.

Les agrégateurs et les mémoires globales sont un exemple d'un tel mécanisme. Un agrégateur (ou une mémoire globale) devient surchargé lorsqu'il reçoit plus de messages qu'il ne peut en traiter. Le composant détecte la surcharge quand la file d'attente de ses messages atteint une certaine limite et continue à croître. Il est possible d'exprimer une surcharge selon d'autres critères tels que l'utilisation des ressources de l'hôte par le composant par exemple. A cet instant le composant cherche sur le réseau pair-à-pair un composant similaire qui puisse recevoir plus de travail sans devenir surchargé à son tour. Si cette recherche est un succès le composant en surcharge informe une partie des producteurs qui sont en contact avec lui de référencer le composant libre. Si aucun composant capable de prendre en charge le travail en excédant n'est trouvé la décision est prise de créer un nouveau composant. Ce nouveau composant est instancié sur l'hôte le plus propice qui satisfait des conditions de puissance processeur, capacité mémoire, bande passante, etc. Cet hôte est trouvé par une recherche sur le réseau pair-à-pair. La moitié des producteurs connectés au composant en surcharge sont redirigés vers le nouveau composant. Les critères de choix sont similaires à ceux de l'autoconfiguration : les producteurs avec tous types de RTT, faibles et élevés, sont redirigés vers le nouveau composant.

Un autre aspect de l'auto-optimisation consiste en l'observation régulière des performances du système. Si ces performances n'obéissent pas aux contraintes fixées par l'administrateur le système doit prendre des mesures. Par exemple si le temps de communication entre un agrégateur et une mémoire globale dépasse une certaine valeur, l'agrégateur peut migrer vers un nouvel emplacement depuis lequel il pourra respecter la contrainte. Un autre exemple est la réduction de la fréquence de communication de

façon à éliminer une congestion des liens de communication du réseau. Si les senseurs du réseau avertissent de la présence d'un engorgement sur une partie du réseau, les composants implantés dans cette zone peuvent augmenter la période entre deux communications jusqu'à ce que la congestion disparaisse.

3.3.3. *Autosoin*

Une politique d'autosoin évidente consiste en la détection automatique de la panne d'un composant et son remplacement dynamique en garantissant que le système demeure cohérent. Nous distinguons deux cas : la panne d'un producteur et celle d'un agrégateur ou d'une mémoire globale. Quand un producteur ne parvient pas à s'exécuter ou qu'il termine de façon inattendue le système tente de le relancer. Si le démarrage ou l'activité du composant échoue encore après plusieurs essais, le nœud hôte est considéré comme nœud en panne et est écarté du système jusqu'à une intervention extérieure.

Quand un agrégateur ou une mémoire globale tombe en panne, la défaillance est détectée par les producteurs. Lorsqu'un producteur échoue à communiquer avec un composant, la panne de celui-ci est suspectée. Le producteur diffuse alors dans le réseau pair-à-pair un message indiquant la suspicion d'une défaillance en spécifiant l'identité du composant suspecté. Si un ou plusieurs autres producteurs confirment la panne du composant un mécanisme d'élection choisit un nouveau nœud où redémarrer le composant. Ce mécanisme d'élection cherche dans le réseau pair-à-pair l'hôte disposant des meilleures caractéristiques de puissance processeur, capacité mémoire, bande passante, etc. Si la panne d'un composant n'est pas détectée, le mécanisme d'auto-optimisation peut observer une surcharge de certains composants et décider de la création d'un nouveau composant.

3.3.4. *Autoprotection*

Pour rendre le système résistant aux pannes en cascade nous nous basons sur la robustesse de certains mécanismes de communication pair-à-pair capables de re-router les messages si un lien de communication tombe où d'adresser des messages à un pair qui a été déconnecté du réseau pendant quelques instants. Nous nous basons aussi fortement sur la réplication des services d'agrégation et de mémoire globale.

3.4. *Protocole de diffusion d'information*

Le modèle de communication de notre outil de surveillance doit passer à l'échelle et consommer le moins de ressources possibles. Il n'est pas envisageable qu'un producteur émette plusieurs fois les mêmes données à destination de chaque agrégateur. Cependant les données émises doivent être communiquées le plus vite possible au travers de tout le système. Notre solution repose sur un protocole de bavardages (Jenkins *et al.*, 2001) pour diffuser rapidement les données. Ce protocole est à la fois performant, décentralisé, et passe à l'échelle.

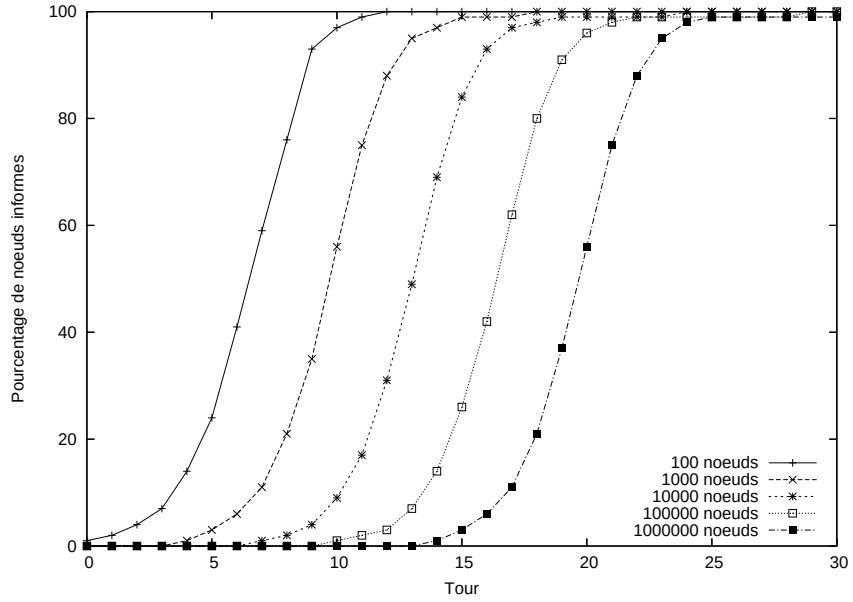

Figure 4. *Diffusion par protocole de bavardages*

Un protocole de bavardages pur est composé de tours successifs dans chacun desquels un processus choisit aléatoirement un autre processus et lui communique ses informations. Il existe plusieurs implantations de ce mécanisme. Dans un bavardage poussant (*push*) le processus initiant la diffusion *infecte* la cible choisie avec ses informations. (Pittel, 1987) a démontré que dans un groupe de *n* participants une information nouvelle apparaissant dans un participant nécessite *O(log n)* tours pour que tous les participants soient infectés par cette information.

La figure 4 présente la dissémination d'une information depuis un nœud vers une population entière (chaque courbe de population est la moyenne de 20 expérimentations). Nous observons que même dans des systèmes extrêmement larges (1 000 000 de nœuds) l'intégralité de la population est informée en moins de 30 tours. De plus 90 % de la population est déjà informée au 22^{e} tour. Les protocoles de bavardages sont un outil très efficace qui fournit une diffusion rapide, fiable, décentralisée, et passant à l'échelle.

Dans notre système le protocole de bavardages est modifié pour palier son principal défaut : la génération de trop nombreux messages. Dans notre système tous les nœuds n'ont pas à recevoir une donnée nouvellement produite, seulement les mémoires globales. Ainsi les bavardages sont limités entre les producteurs et les agrégateurs et entre les agrégateurs et les mémoires globales. Cette structure hiérarchique, qui est un graphe orienté acyclique (*Directed Acyclic Graph (DAG)*) offre un double bénéfice : en réduisant le nombre de nœuds à informer le passage à l'échelle est amé-

lioré alors que le temps et le nombre de messages nécessaires pour informer tous les destinataires (les mémoires globales) sont réduits. Comme expliqué précédemment un des défis de notre système est de pouvoir répandre au plus vite les données issues d'un producteur vers l'ensemble des mémoires globales.

3.5. *Performance du modèle*

Nous avons déployé notre système sur 100 nœuds d'une grappe de calculs constituée de bi-AMD Opteron 64 bits cadencés à 2 GHz avec 2 GB de mémoire et interconnectés par un réseau Ethernet à 1 Gb/s. Les nœuds opérent sous un système Linux Red Hat 9 avec un noyau 2.4.21. Nous déployons équitablement les composants sur tous les nœuds. Pour mesurer les performances des très larges systèmes (100 000 et 1 000 000 producteurs) nous avons légèrement modifié le code des producteurs de façon à ce qu'un seul processus joue le rôle de tous les producteurs qui auraient dû être normalement déployés sur un nœud. En effet, la présence de plus de 1 000 processus producteurs sur un seul nœud nuit aux performances de ce nœud. La structure et le nombre des messages transmis restent inchangés.

Dans notre première expérience nous cherchons à observer l'adaptation du nombre d'agrégateurs en fonction de l'augmentation du nombre de producteurs. A partir d'un système contenant 100 producteurs et un seul agrégateur nous ajoutons 10 nouveaux producteurs à chaque itération jusqu'à ce que le système contienne 1 000 producteurs. Une nouvelle itération est déclenchée toutes les 30 secondes. Nous marquons un palier de 20 itérations à 500 producteurs pour nous assurer que le nombre d'agrégateurs reste stable si le nombre de producteurs reste stable. Nous considérons qu'un agrégateur est surchargé s'il reçoit plus de 100 messages à traiter dans une même itération. En cas de surcharge un nouvel agrégateur est créé et hérite de la moitié des producteurs communiquant avec l'agrégateur en surcharge.

La figure 5 présente le nombre d'agrégateurs et de producteurs (par centaines) présents dans le système tout au long de l'expérience. Les deux courbes restent proches : cela atteste que la quantité d'agrégateurs augmente dans les mêmes proportions que les producteurs. Le rapport entre les agrégateurs et les producteurs reste approximativement le même pendant toute la durée de l'expérience : en moyenne 1 agrégateur pour 85,02 producteurs.

La deuxième expérience vise à observer le temps requis par les mémoires globales pour être informées de la présence des producteurs. Cela signifie le temps requis par chaque mémoire globale pour recevoir au moins une donnée de chaque producteur dans le système. Le temps est exprimé en itérations. Durant une itération, chaque producteur bavarde avec un agrégateur, chaque agrégateur bavarde avec une mémoire globale, et chaque mémoire globale bavarde avec une autre mémoire globale. Le graphe des composants est déjà formé (et équilibré) comme suit. Les ratios sont fixés à 1 agrégateur pour 100 producteurs, et 1 mémoire globale pour 5 agrégateurs. Chaque producteur est en contact avec 10 agrégateurs en utilisant une distribution uniforme, de la

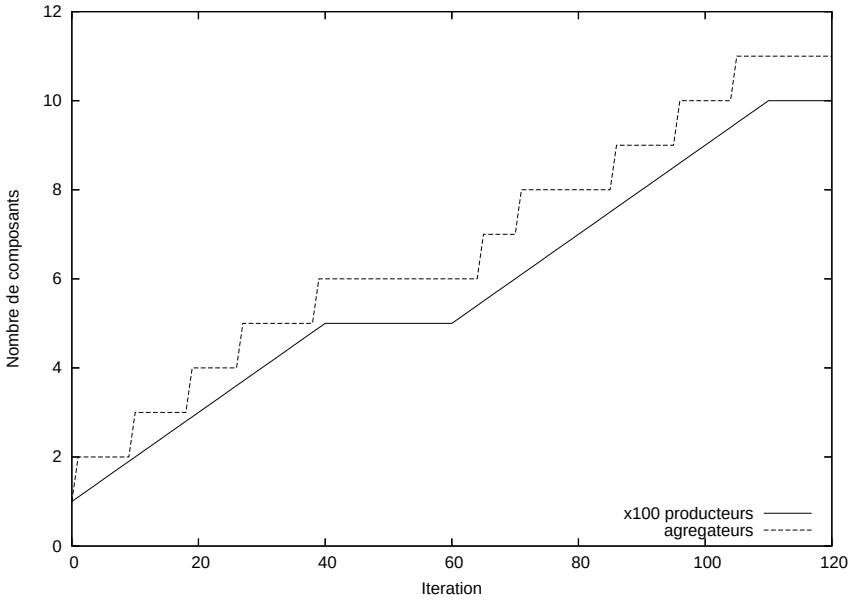

Figure 5. *Adaptation de la taille du système*

même façon chaque agrégateur référence 10 mémoires globales, et chaque mémoire globale 10 autres mémoires globales. La figure 6 présente les résultats. Les courbes tracent le pourcentage moyen de la connaissance du système par l'ensemble des mémoires globales : 100 % indique que toutes les mémoires globales ont connaissance (c-à-d reçu au moins une donnée) de tous les producteurs.

Dans un système de 100 000 producteurs, 10 itérations sont nécessaires pour que tous les producteurs soient connus de toutes les mémoires globales. Dans un système contenant 1 000 000 de producteurs, après 10 itérations, 95,02 % des producteurs sont connus. La diffusion de l'information est plus rapide que dans un protocole de bavardages pur présenté dans la figure 4. Une première raison est qu'une itération de notre système ne correspond pas exactement à un tour classique d'un protocole à bavardages pur (présenté dans la section 3.4). En effet lors d'une itération une nouvelle donnée est relayée non pas une, mais trois fois : du producteur à un agrégateur, de l'agrégateur à une mémoire globale, et de la mémoire globale à une autre. Une seconde raison de l'efficacité vient du fait que notre protocole oriente les messages vers un sous-ensemble des nœuds participants : les mémoires globales. L'évaluation de l'achèvement de la diffusion d'information ne prend en compte que ce sous-ensemble de composants.

La troisième expérience consiste à évaluer l'âge des données dans les mémoires globales. Puisqu'un consommateur peut questionner directement les mémoires globales pour obtenir des données produites, il est important que ces informations ne soient pas périmées, en d'autres termes il est souhaitable que ces données soient les

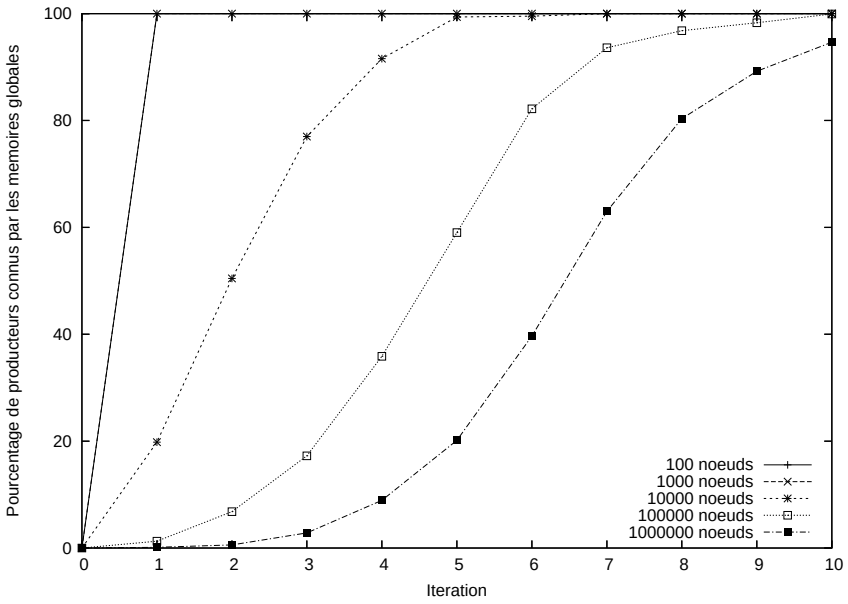

Figure 6. *Connaissance du système par les mémoires globales*

plus récentes possibles. Les conditions de cette expérience sont les mêmes que pour la précédente, à la différence près que le système a fonctionné pendant 80 itérations avant d'être arrêté et observé. La figure 7 présente l'âge des données contenues dans les mémoires globales d'un système de 100 000 producteurs. L'instant de l'observation est noté itération 0.

Les données dans les mémoires globales sont récentes. La courbe montre un pic autour de l'itération -2 (valeur maximum avec 7,52 %). 52,89 % des données sont âgées de moins de 7 itérations, 81,07% de moins de 16 itérations, et 94,97 % de moins de 30 itérations. Lors de nos expériences la durée d'une itération était de 30 secondes : l'âge moyen des données dans l'intégralité du système est d'approximativement 5 minutes (10,16 itérations), ce qui est très satisfaisant pour un système de cette envergure.

3.6. *Implantation*

Notre implantation du système de surveillance peut être basée sur la plupart des bibliothèques pair-à-pair. Nous avons cependant choisi JXTA comme protocole par défaut. Nous présentons ici cette bibliothèque et motivons notre choix.

JXTA (Gong, 2002 ; Traversat *et al.*, 2003) est un ensemble de protocoles pair-à-pair libres qui permet à n'importe quels appareils connectés sur le réseau de communiquer et collaborer en tant que pairs. Les protocoles de JXTA sont indépendants des

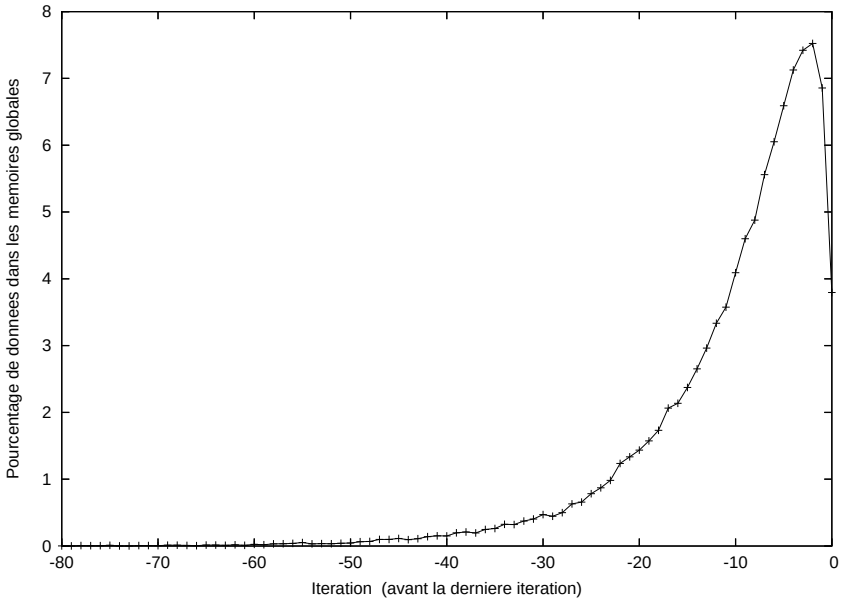

Figure 7. *Age des données dans les mémoires globales*

langages de programmation, et plusieurs implantations (appelées *bindings*) existent pour différents environnements. JXTA est donc parfaitement adapté aux environnements hétérogènes qui composent une grille de calculs.

JXTA possède son propre mécanisme d'adressage et de nommage : un pair peut se déplacer dans le réseau, changer son protocole de communication et son adresse réseau, même être temporairement déconnecté, et pourtant être toujours joignable par les autres pairs. Cette capacité offre une résistance accrue face à la volatilité des nœuds dans une grille. De plus JXTA fournit des communications et des accès sécurisés aux ressources. JXTA permet également de franchir des pare-feux sous la condition que certains pairs supportent des communications HTTP. Cette fonctionnalité est précieuse dans les environnements de grilles composés de plusieurs domaines administratifs différents.

Les *modules* sont une abstraction pour représenter des morceaux de code. Chaque pair, groupe, service, et module sont décrits par une annonce (*advertisement*), un document XML représentant les métadonnées. Le système de découverte de JXTA se réfère aux annonces pour trouver un composant particulier dans le réseau. Dans notre système de surveillance chaque composant fournissant un service enregistre une annonce pour être par la suite recherché par des clients. De plus les senseurs sont empaquetés en tant que module pour être automatiquement recherchés, téléchargés, et exécutés.

L'utilisation de JXTA comme couche de communication pour des calculs à haute performance n'est pas satisfaisante ; notamment pour des applications fortement communicantes. Les principales raisons sont *(1)* que l'hétérogénéité des nœuds constituant le réseau pair-à-pair en termes de puissance de calcul et de capacité de communication crée des déséquilibres dans le système, et que *(2)* la latence des communications JXTA est trop élevée comme souligné dans (Antoniu *et al.*, 2005). Cependant pour une application qui consomme peu de ressources, bien en deçà de la capacité des nœuds, l'hétérogénéité n'est pas un problème. De la même façon pour une application qui ne communique pas intensément, la latence plus élevée n'est pas non plus un facteur gênant. C'est pourquoi au vu des services que propose JXTA (enregistrement et découverte, module, nommage et adressage), et au peu d'impact de ses défauts sur l'application que nous construisons nous l'avons choisi comme bibliothèque par défaut.

L'implantation de notre outil de surveillance a été écrite en Java et utilise la version 2.3.7 de la bibliothèque JXTA. Dans sa version actuelle les données observées se limitent aux informations concernant le processeur, la mémoire, et le système d'exploitation des nœuds hôtes.

4. Précédents travaux

Cette section présente une étude sommaire des outils de surveillance majoritairement déployés et de leurs capacités.

4.1. *Network weather service (NWS)*

Le « service de météorologie du réseau » (Wolski *et al.*, 1999) est un système distribué qui surveille une collection de machines et leur réseau. NWS émet périodiquement des prévisions sur les performances du réseau et sur les ressources de calcul qui peuvent être offertes sur certains intervalles de temps. Les composants de NWS sont :

– un *serveur de nommage* qui est un contrôleur centralisé qui garde les enregistrements de tous les composants du système et des activités surveillées ;

– des *senseurs* qui produisent les données sur les ressources et activités surveillées ;

– des *mémoires* qui enregistrent les données ;

– des *météorologues* qui analysent les données et produisent des prévisions.

Les limites du système NWS viennent du serveur de nommage. La centralisation de service introduit un goulot d'étranglement, et un point de panne critique. De plus les connexions entre les senseurs et le serveur de nommage sont entièrement à la charge de l'administrateur qui doit configurer manuellement l'intégralité du système avant de le démarrer. Le serveur de nommage ne contient aucun mécanisme de sécurité. Enfin

l'utilisation de NWS est difficile dans des environnements à grande échelle à cause de l'absence d'un vrai système de base de données.

(Shirose *et al.*, 2004) présentent un outil de configuration autonome pour NWS. Cet outil construit automatiquement les *clique groups* de NWS en mesurant la proximité des nœuds par leur RTT. Ces groupes représentent des ensembles de nœuds parmi lesquels un nœud est choisi comme représentant de l'intégralité du groupe pour la surveillance de la bande passante. Sans la constitution de ces groupes la complexité pour mesurer les bandes passantes croîtrait en fonction du carré de la taille de la grille.

4.2. *Globus monitoring and discovery system (MDS)*

Le système de surveillance et de découverte est un composant des services d'information de Globus. Il fournit des informations au sujet des ressources disponibles sur une grille et de leur état. MDS se base sur le protocole d'interrogation et de modification d'annuaire LDAP (*Lightweight Directory Access Protocol*). La nature distribuée du LDAP semble intéressante : les informations sont organisées hiérarchiquement et l'arbre qui en résulte peut être distribué sur plusieurs serveurs. Dans le cas des grilles, cette hiérarchie reflète souvent l'organisation même de la grille : du réseau international vers les réseaux nationaux vers les réseaux locaux. Les feuilles étant des ressources telles que des grappes, des nœuds de calculs, ou des unités de stockage.

L'architecture LDAP est appropriée pour enregistrer des informations statiques comme le nombre de processeurs d'une grappe ou la taille de la partition d'un disque. Lorsque les données sont plus dynamiques l'architecture LDAP est un mauvais choix car elle est inefficace pour effectuer de nombreuses opérations d'écriture. Sous cette condition l'architecture LDAP souffre de problèmes sérieux de performance et de passage à l'échelle.

4.3. *Relational grid monitoring architecture (R-GMA)*

Le langage structuré de requête SQL (*Structured Query Language*) permet de manipuler une base de données relationnelle. Plusieurs implantations de ce type de base ont été conçues pour des applications très exigeantes. Elles offrent un bon compromis entre leur capacité à être distribuées et le coût de traitement des requêtes. De plus une base de données peut-être dupliquée de manière à rendre le service plus disponible et à résister aux pannes. R-GMA (Cooke *et al.*, 2004) a été développée dans ce but. L'introduction de composants qui combinent les données de la base et les gardent en cache accroît la capacité du système à passer à l'échelle. R-GMA est une implantation du modèle GMA du GGF (voir section 3.1). Elle supporte les requêtes qui combinent des informations sur des objets de différentes classes (une opération *JOINT*). R-GMA est maintenant développée au sein du projet EGEE (*Enabling Grids for E-science in Europe*).

4.4. *Ganglia*

Ganglia (Massie *et al.*, 2004) est un système de surveillance distribué pour les systèmes de calculs à haute performance tels que les grappes et également les grilles. Ganglia est basé sur une organisation hiérarchique constituée de fédérations de grappes. Au sein de chaque grappe un protocole multicast est utilisé pour publier l'état des ressources, alors qu'un arbre de connexions point-à-point relie des nœuds choisis comme représentant sur chaque grappe.

L'implantantion de Ganglia consiste en la collaboration de deux démons. `gmond` s'exécutant sur chaque nœud fournit la surveillance d'une grappe en diffusant et enregistrant les données par multicast. `gmetad` quant à lui fournit la fédération de plusieurs grappes en prenant en charge la communication avec des démons présents sur d'autres grappes et en agrégeant les données.

Ganglia se révèle performant et simple d'utilisation pour des grappes et des grilles de tailles moyennes. Cependant le maintient sur chaque nœud d'une copie de l'état entier du système par `gmond` ainsi que l'utilisation d'un protocole multicast peut entrainer des problemes de passage à l'échelle sur des grappes de grandes dimmensions (plus de 2 000 nœuds). Une partie alors trop importante des ressources de chaque nœud et du réseau serait alors consacrée au système de surveillance. De plus la mise en place et le maintient d'un système double `gmond`/`gmetad` nécessitent une certaine expertise de la part des administrateurs.

5. Conclusion et perspectives

Nous proposons une solution pour offrir une gestion autonome dans un environnement de grille. Notre approche est basée sur l'utilisation d'un réseau pair-à-pair de recouvrement qui fournit aux composants d'une application un moyen de communication résistant aux pannes et passant à l'échelle. Nous avons détaillé la construction d'un outil de surveillance basé sur le modèle GMA du GGF. Les composants autonomes de cette application interagissent grâce au réseau pair-à-pair. La mise en œuvre des mécanismes d'autogestion a été décrite. Des mesures de performances attestent de l'efficacité du système pour diffuser rapidement les données dans des systèmes de grandes tailles.

Nous envisageons de poursuivre le développement de cette application en affinant ses capacités d'autogestion. Des améliorations de notre protocole de bavardages telles que la sélection des données à émettre en fonction des données précédemment émises permettront de diminuer encore la quantité de messages émis sur le réseau. De plus si le choix aléatoire du contact avec qui communiquer était pondéré par le temps depuis lequel ce contact n'a pas reçu de message la diffusion des données serait encore accélérée et son uniformité serait accrue. Nous souhaitons également utiliser une base de données relationnelle pour enregistrer les données dans les mémoires globales. Cela aurait le double avantage de permettre un meilleur passage à l'échelle, et d'autoriser des requêtes plus complexes impliquant des jointures sur différents champs de la base.

Dans un cadre plus général nous visons à intégrer les services de notre outil de surveillance selon le modèle OGSA (*Open Grid Software Architecture*) (Foster *et al.*, 2003b) qui est en passe de devenir le moyen standard d'accès aux services de grille. Enfin nous souhaitons nous préoccuper des mécanismes de sécurité. A l'heure actuelle nous nous reposons uniquement sur les mécanismes de sécurité offerts par la couche de communication, c'est-à-dire la bibliothèque pair-à-pair. Un système sécurisé se doit de fournir des mécanismes d'authentification, de confiance, et de certification de l'intégrité des données. La sécurité du système est le thème majeur de l'autoprotection.

6. Bibliographie

Anderson D. P., Cobb J., Korpela E., Lebofsky M., Werthimer D., « SETI@home : An Experiment in Public-Resource Computing », *Communications of the ACM*, vol. 45, n° 11, p. 56-61, November, 2002.

Antoniu G., Jan M., Noblet D. A., « Enabling the P2P JXTA Platform for High-Performance Networking Grid Infrastructures », *Proceedings of High Performance Computing and Communications (HPCC)*, vol. 3276 of *LNCS*, Sorrento, Italy, p. 429-439, September, 2005.

Baduel L., Matsuoka S., « A Peer-to-Peer Infrastructure for Autonomous Grid Monitoring », *Proceedings of the third International Workshop on Hot Topics in Peer-to-Peer Systems, at IPDPS*, Long Beach, California, USA, March, 2007.

BitTorrent, http ://www.bittorrent.com, n.d.

Cooke A. W., Gray A. J. G., Nutt W., Magowan J., Oevers M., Taylor P., Cordenonsi R., Byrom R., Cornwall L., Djaoui A., Field L., Fisher S., Hicks S., Leake J., Middleton R., Wilson A. J., Zhu X., Podhorszki N., Coghlan B. A., Kenny S., O'Callaghan D., Ryan J., « The Relational Grid Monitoring Architecture : Mediating Information about the Grid », *Journal of Grid Computing*, vol. 2, n° 4, p. 323-339, December, 2004.

Czajkowskiy K., Fitzgeraldz S., Foster I., Kesselmany C., « Grid Information Services for Distributed Resource Sharing », *Proceedings of the 10th international symposium on High Performance Distributed Computing*, San Francisco, California, USA, p. 181-194, August, 2001.

Foster I., Iamnitchi A., « On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing », *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, California, USA, February, 2003a.

Foster I., Kesselman C., Nick J. M., Tuecke S., *Grid Computing, Making the Global Infrastructure a Reality*, John Wiley & Sons, chapter The physiology of the Grid, p. 217-249, 2003b.

Gaidioz B., Wolski R., Tourancheau B., « Synchronizing Network Probes to avoid Measurement Intrusiveness with the Network Weather Service », *Proccedings of the 9th international symposium on High Performance Distributed Computing*, Pittsburgh, Pennsylvania, USA, p. 147-154, August, 2000.

Gong L., Project JXTA : A Technology Overview, Technical report, Sun Microsystem, Inc., October, 2002 .

Jenkins K., Hopkinson K., Birman K., « A Gossip Protocol for Subgroup Multicast », *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS)*, Phoenix, Arizona, USA, April, 2001.

Kephart J. O., Chess D. M., « The Vision of Autonomic Computing », *IEEE Computer*, vol. 36, n° 1, p. 41-52, January, 2003.

Massie M. L., Chun B. N., Culler D. E., « The Ganglia Distributed Monitoring System : Design, Implementation, and Experience », *Journal of Parallel Computing*, vol. 30, n° 7, p. 817-840, July, 2004.

Napster, http ://www.napster.com, n.d.

Pittel B., « On Spreading a Rumor », *SIAM Journal of Applied Mathematics*, vol. 47, n° 1, p. 213-223, March, 1987.

Rhea S., Eaton P., Geels D., Weatherspoon H., Zhao B., Kubiatowicz J., « Pond : the OceanStore Prototype », *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, San Francisco, California, USA, March, 2003.

Shirose K., Matsuoka S., Nakada H., Ogawa H., « Autonomous Configuration of Grid Monitoring Systems », *Proceedings of the international Symposium on Applications and the Internet (SAINT Workshop)*, Tokyo, Japan, p. 651-657, January, 2004.

The eMule Project, http ://www.emule-project.net, n.d.

Tierney B., Aydt R., Gunter D., Smith W., Swany M., Taylor V., Wolski R., A Grid Monitoring Architecture, Technical report, Global Grid Forum, January, 2002.

Tierney B., Crowley B., Gunter D., Holding M., Lee J., Thompson M., « A Monitoring Sensor Management System for Grid Environments », *Proceedings of the 9th international symposium on High Performance Distributed Computing*, Pittsburgh, Pennsylvania, USA, p. 97-104, August, 2000.

Traversat B., Arora A., Abdelaziz M., Duigou M., Haywood C., Hugly J., Pouyoul E., Yeager B., Project JXTA 2.0 Super-Peer Virtual Network, Technical report, Sun Microsystem, Inc., May, 2003.

White S. R., Hanson J. E., Whalley I., Chess D. M., Kephart J. O., « An Architectural Approach to Autonomic Computing », *Proceedings of the 1st International Conference on Autonomic Computing*, New York, New York, USA, p. 2-9, May, 2004.

Wolski R., Spring N., Hayes J., « The Network Weather Service : A Distributed Resource Performance Forecasting Service for Metacomputing », *Journal of Future Generation Computing Systems*, vol. 15, n° 5–6, p. 757-758, October, 1999.

Zanikolas S., Sakellariou R., « A Taxonomy of Grid Monitoring Systems », *Future Generation Computer Systems*, vol. 21, n° 1, p. 163-188, January, 2005.