Publication pour la présente candidature LAFPT

Titre	GrAPP&S Data Grid Une approche de type grille et système pair à pair pour le stockage de données
Auteurs	Thierno Ahmadou DIALLO; Olivier FLAUZAC; Luiz Angelo STEFFENEL; **Samba N'DIAYE**
Résumé	Dans cette article nous présentons GrAPP&S (Grid APPlications & Services), un intergiciel pair à pair (P2P) pour le stockage de tous types de données. En effet, GrAPP&S a pour vocation de gérer tous les types de données, soient-ils des fichiers, bases de données, de documents structurés ou semi-structurés, des flux vidéo/Audio/VoIP, des systèmes de stockage cloud ou encore d'autres réseaux P2P. L'architecture GrAPP&S est très modulaire et fondée sur des protocoles ouverts, ce qui simplifie son déploiement dans n'importe quelle type d'environnement. Ceci permet une gestion totalement décentralisée, une haute disponibilité des données, et une réactivité face aux pannes de ses différents composants.
Mots-Clés	Intergiciel ; système pair à pair ; grille de données
Date de publication	2012
Rang Auteur	4
Correspondant Auteur	thierno80.diallo@ucad.edu.sn
Nom de la revue	CNRIA
Type de revue	Revue à comité de lecture
Portée de la revue	Nationale
Langue de la revue	Français
Référence	Actes du 4e Colloque National sur la Recherche en Informatique et ses Applications (CNRIA 2012)
Pages	
Editeur	CNRIA
DOI	
URL	
Preuve d'indexation	URED ; CNRIA
ISBN - ISSN	0850-2161
Facteur d'impact	

Université, Recherche et Développement (URED) Numéro spécial Actes du CNRIA'2012

URED

UNIVERSITÉ, RECHERCHE ET DÉVELOPPEMENT

Revue Pluridisciplinaire de
l'Université Gaston Berger de Saint-Louis, Sénégal

Numéro Spécial 2012

Actes du CNRIA'2012
Colloque International sur la Recherche
en Informatique et ses applications

4e édition
Du 25 au 27 avril 2012 à Thiès et Bambey, Sénégal

SÉRIE SCIENCES EXACTES

ISSN : 0850-2161

GrAPP&S Data Grid

Une approche de type grille et système pair à pair pour le stockage de données

Thierno Ahmadou Diallo*‡ – Olivier Flauzac* – Luiz Angelo Steffenel* – Samba N'Diaye‡

‡ Département d'Informatique
LMI, Universite Cheikh Anta Diop
5005 Dakar-Fann
SENEGAL
thierno.diallo.sn@gmail.com
samba.ndiaye@ucad.edu.sn

* Département de Mathématique et Informatique
CReSTIC, Université de Reims Champagne-Ardenne
BP 1039, F-51687 Reims Cedex
FRANCE
{olivier.flauzac, Luiz-Angelo.Steffenel}@univ-reims.fr

RÉSUMÉ. Dans cette article nous présentons **GrAPP&S** (**Gr**id **APP**lications & **S**ervices), un intergiciel pair à pair (P2P) pour le stockage de tous types de données. En effet, GrAPP&S a pour vocation de gérer tous les types de données, soient ils des fichiers, bases de données, de documents structurés ou semi-structurés, des flux vidéo/Audio/VoIP, des systèmes de stockage *cloud* ou encore d'autres réseaux P2P. L'architecture GrAPP&S est très modulaire et fondée sur des protocoles ouverts, ce qui simplifie son déploiement dans n'importe quelle type d'environnement. Ceci permet une gestion totalement décentralisée, une haute disponibilité des données, et une réactivité face aux pannes de ses différents composants.

ABSTRACT. In this paper we present **GrAPP&S**(**G**rid **APP**lications & **S**ervices), a peer-to-peer (P2P) middleware designed to deal with any kind of data type. Indeed, data managed by GRAPP&S include files, databases, structured or semi-structured documents, data streams (video/audio/VoIP), cloud storage or even other P2P networks. GrAPP&S architecture is modular and relies on open protocols, which simplifies its deployment on different computational environments. This modularity allows a fully decentralized management, with high data availability and reactivity in the case of components' failures.

MOTS-CLÉS : Intergiciel, système pair à pair, grille de données.

KEYWORDS: Middleware, peer-to-peer system, data grid.

1. Introduction

Le stockage de grande quantité de données est devenu crucial. De nombreux efforts ont été réalisés envers la proposition de solutions de stockage adaptées. Alors que les clusters, les SAN ou les grilles sont des alternatives pour la résolution de problèmes ponctuels nécessitant une grande capacité de stockage, ces solutions sont limitées par leur caractère fortement centralisé.

Pour résoudre ce problème, certaines approches s'appuient sur des réseaux pair à pair (P2P), qui offrent des propriétés intéressantes pour la tolérance aux pannes et le passage à l'échelle. Les architectures P2P désignent un ensemble d'utilisateurs, appelés nœuds, ainsi que les protocoles utilisés par ces nœuds pour communiquer entre eux. Plus précisément, les nœuds communiquent d'égal à égal, au contraire d'autres protocoles hiérarchiques type client/serveur. Ces architectures P2P permettent non seulement aux utilisateurs de mettre en commun des ressources, mais elles sont intrinsèquement distribuées, passent à l'échelle et conservent des bonnes performances en termes de temps d'accès à l'information. Il existe différent types d'architecture P2P : centralisées, structurées et non structurées [7].

Dans les architectures P2P centralisées, les pairs qui se connectent au réseau envoient une copie de leurs indexes au serveur central. Quand une requête est émise depuis un pair, elle est acheminée jusqu'au serveur qui traite cette requête puis retourne au pair émetteur une liste des pairs qui contiennent l'information recherchée. Le pair émetteur contacte ensuite directement les pairs qui possèdent les informations correspondantes aux critères de recherche afin de les récupérer. L'accès aux informations est donc décentralisé alors que la gestion des requêtes reste centralisée, ce qui rend le système fragile vis-à-vis d'une panne du serveur et pose des problèmes de passage à l'échelle (ce mécanisme limite la taille maximale du réseau). Napster [12], eDonkey [4] et Bittorrent [1] sont des exemples d'architecture P2P centralisée.

Les architectures structurées ont la particularité d'organiser le réseau P2P en une topologie routable, c'est-à-dire que chaque pair dispose d'un identifiant permettant de le localiser en suivant un chemin déterministe parmi les pairs, et nécessitant un minimum de messages. Chaque ressource partagée au sein du réseau possède également un identifiant, qui est le résultat d'une fonction de hachage et qui permet de la localiser rapidement. L'indexation des ressources dans les réseaux P2P structurées utilise le principe des tables de hachage distribuées (DHT), où chaque pair devient responsable des entrées de la table égales ou proches de son identifiant. Les DHT permettent une localisation des pairs et des données très performante. De même, le routage de proche en proche permet de limiter le nombre de messages à O($log\ N$), N étant le nombre de pairs dans le réseau. Les DHT sont implémentées dans bon nombre de système comme Chord [13], Pastry [11] ou Can [10]. L'avantage des réseaux P2P structurées est le

passage à l'échelle et son efficacité dans la recherche d'information. Cependant ils montrent leur limite dans les environnements dynamiques à cause de l'arrivé ou départ de bon nombre de nœuds. Un autre inconvénient vient du fait que les P2P structurées sont limités à des données de même nature (fichier, par exemple) et ne permettent pas des recherches multiparamétriques, notamment à cause des fonctions de hachage utilisées.

Finalement, les architectures P2P non structurées sont basées sur des graphes aléatoires, utilisant une méthode de recherche par inondation. Ces solutions sont résistantes dans les environnements dynamiques du fait des faibles contraintes imposées sur la topologie virtuelle. On peut citer quelques exemples de systèmes non structurés, dont Gnutella 6.0 [12], Fast Track avec son client Kazaa [6] ou FreeNet [1]. Toutes ces solutions P2P sont des architectures planes, i.e., ne sont pas hiérarchiques. Pour améliorer les performances des réseaux P2P, des architectures hiérarchiques ont été proposée dans la littérature [9] [5][3][10]. Cependant la majeure partie d'entre elles se focalisent uniquement sur l'amélioration du temps de latence lors d'une recherche. D'ailleurs, les architectures hiérarchiques existantes ne gèrent que des données de même type, ce qui limite leur domaine d'application.

Dans cet article, nous proposons une architecture hiérarchique nommée GrAPP&S, structurée sur deux niveaux : un niveau d'interconnexion et un niveau de communautés. Le niveau d'interconnexion est utilisé pour relier les différentes communautés, chacune composée par des nœuds qui peuvent communiquer mutuellement et qui partagent les mêmes propriétés. Cette approche permet la diversification des types de données stockées par GrAPP&S, qui permet ainsi la gestion de différentes sources de données, selon les propriétés de chaque communauté, tout en gardant l'interconnexion des sources grâce à la couche d'interconnexion.

Cet article est organisé comme suit : la section 2 est consacré à l'état de l'art et la section qui suit présente l'architecture GrAPP&S. La section 4 est consacrée à la gestion des nœuds en fin nous allons conclure dans la section 5

2. Etat de l'art

Différentes architectures hiérarchique P2P ont été proposée dans [9][5][3][10]. Dans [5], l'auteur propose une architecture hiérarchique à deux niveaux. Au niveau inférieur il regroupe les pairs d'une même région, organisés sur un anneau de Chord et coordonnés par un super nœud. Au niveau supérieur se trouvent les supers nœuds de chaque région. [5] propose un algorithme de recherche régionale basé sur les super-nœuds, qui gardent une table de routage bidirectionnelle dans le but de réduire efficacement la redondance de la table de routage originale de Chord. Toutefois, même si l'architecture passe à l'échelle, elle ne résiste pas dans un environnement dynamique. Dans [3] les auteurs ont

proposé un modèle hiérarchique de DHT (HDHT), où les pairs sont organisés en groupes. L'objectif des HDHTs est d'améliorer l'architecture plane DHT conventionnelle, par une exploitation ressources hétérogènes des pairs, la mise en cache des infrastructures, la transparence et l'autonomie de différentes parties du système, afin de rendre la recherche de clés plus efficace tout en générant moins de trafic.

Les auteurs de [10] proposent l'architecture hiérarchique CBT pour améliorer le protocole de téléchargement de fichier de Bittorrent dans un réseau de grande échelle. Il utilise trois types de pairs : les sources, les téléchargeurs et les super pair. Tous les supers pairs sont connectés au *torrent tracker* (gérant de l'index des ressources) pour former un réseau dédié. La disponibilité des informations est fortement liée au *torrent tracker*, car si ce dernier tombe en panne tout le service disparait.

Les travaux effectués dans [5][3][10] reposent sur une architecture hiérarchique à deux niveau. Leur objectif est d'améliorer efficacement les performances tels que la latence lors d'une recherche et générer moins de trafic réseau. Cependant, ces architectures montrent leur limite dans les environnements dynamiques à cause de l'instabilité des nœuds. Une alternative est proposée dans [9], où les auteurs proposent HP2P, une architecture hiérarchique hybride à deux niveaux, combinant DHT et systèmes P2P non structuré. HP2P utilise dans son premier niveau Chord et au deuxième niveau Kazaa, et procède par inondation lors d'une recherche. Ceci fait de HP2P un système robuste car elle combine les avantages des DHT et des systèmes non structurés, mais reste aussi limité par leurs inconvénients, comme la dépendance aux ressources de même type ou le nombre de messages lors d'une recherche. Face à ces travaux, nous sommes motivés à proposer la spécification d'une architecture hiérarchique plus générique, qui permet le stockage de tous les formats de données tout en conservant les avantages vis-à-vis de la performance réseau.

3. Présentation de l'architecture GrAPP&S

3.1. Description

Afin de présenter notre architecture, nous introduisons dans un premier temps quelques notations. Une **communauté (*C*)** est une entité autonome, qui regroupe des nœuds qui peuvent se communiquer et qui partagent une propriété définie : même localisation, même autorité d'administration (par exemple, des serveurs distants appartenant à la même entreprise) ou même domaine d'application (base de données métier, par exemple).

Le **nœud communicateur** (***c***) joue un rôle essentiellement lié au transport d'informations et à l'interconnexion entre différentes communautés, comme par exemple

lors du passage de messages à travers des pare-feux. C'est le point d'entrée de la communauté, et il assure sa sécurité vis-à-vis de l'extérieur. Un nœud *c* peut être par exemple un serveur php, un serveur J2EE, un serveur .Net. Il dispose d'un identifiant unique (ID) à partir duquel on construit les identités des autres nœuds de la communauté. Ce nœud ne stocke de données et ne fait pas d'indexation.

Les nœuds **ressource manager (*RM*)** assurent l'indexation et l'organisation des données dans la communauté. Ils reçoivent les requêtes des utilisateurs et assurent leur prétraitement. Les nœuds ***RM*** participent à la recherche de données dans la communauté.

Les nœuds **data manager (*DM*)** stockent des données qui peuvent être dans différents supports tels que les bases de données (objet ou relationnelle), les documents (texte/XML/multimédia), des flux (vidéo, audio, VoIP), des données issus de capteurs ou encore une service *cloud*. Un nœud *DM* est un service qui dispose des composants suivants : *(i)* une source de données (un disque dur, un serveur WebDAV, FTP, SFTP, une base de données, un stockage sur *cloud* type Dropbox, etc.) ; *(ii)* un gestionnaire de requêtes qui permet d'exprimer des requêtes locales ou globales et *(iii)* un gestionnaire de communication qui permet au nœud *DM* de communiquer avec le nœud *RM* auquel il est connecté. Cette communication est assurée par des services RMI, WSDL et des protocoles réseaux TCP, HTTP. Le nœud *DM* dispose d'un protocole de connexion spécifique au type de donnée, par exemple JDBC, ODBC, FTP, etc.

Les nœuds de GRAPP&S sont organisés de façon hiérarchique dans une communauté. Dans chaque communauté, il y'a un seul nœud ***c*** sur lequel est connecté un ou plusieurs nœuds ***RM***. Sur chaque nœud ***RM*** est connecté un ou plusieurs nœuds ***DM***. Les nœuds communiquent par message. Les communautés ***C*** sont reliées par un réseau d'interconnexion.

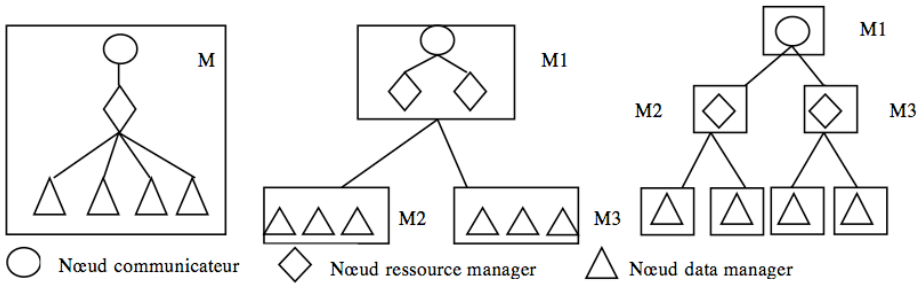

Figure 1 - Organisation des nœuds (a) dans une machine, (b) dans un cluster et (c) dans un réseau

3.2. Formation des communautés

GrAPP&S peut etre déployé dans plusieurs types d'architecture selon le placement des nœuds. ***(i)*** les nœuds peuvent etre regroupés dans une seule machine physique (voir Figure 1a). C'est l'exemple typique d'une machine d'un particulier, qui souhaite héberger une communauté de l'architecture. Le placement des nœuds sous cette forme peut être justifié par sa simplicité à mettre en œuvre lors de sa phase d'implémentation, en utilisant les concepts d'héritage et de polymorphisme. Les nœuds sont interconnectés par des sockets, des solutions RPC pour qu'ils puissent communiquer par message dans les deux sens entre deux nœuds. Dans ***(ii)*** les nœuds sont organisés dans une ferme de serveurs telle qu'un cluster, ce qui est caractéristique des réseaux HPC (Figure 1b). Finalement, ***(iii)*** les nœuds peuvent être regroupés s'ils partagent une même propriété de localisation ou d'administration. Ils sont proches géographiquement les uns des autres (voir Figure 1c). Ceci est l'exemple d'un réseau formé par les nœuds d'une entreprise ou d'un département.

3.3. Adressage des nœuds de GrAPP&S

Chaque nœud de GrAPP&S a un identifiant (ID) unique. L'adresse IP ou MAC ne peuvent pas régler l'identification des nœuds car ils peuvent etre déployés dans seule machine. La solution la plus intéressante est celle proposée par JXTA [8] pour identifier ses pairs et qui utilise une chaine de 128 bits. Chaque nœud dispose ainsi d'une chaîne unique ID_local, sous la forme *« urn:nom_communaute:uuid:chaine-de-bit »*, de taille 128 bits. L'expression de l'adressage hiérarchique se fait par la concaténation des IDs sous forme de préfixe, i.e., l'*ID* du nœud $\boldsymbol{c_i}$ est équivalent à son *ID_local*, l'ID du nœud $\mathbf{RM_i}$ est formé par la *« ID_c_i/ID_RM_i »*, et l'ID du nœud $\mathbf{DM_i}$ présente la forme *« ID_c_i/ID_RM_i/ID_DM_i »*.

4. Gestion des nœuds

La topologie du réseau change fréquemment à cause de la mobilité des nœuds. Nous travaillons dans l'hypothèse où les nœuds ***c*** et ***RM*** sont fixes dès la construction de la communauté ; ainsi, tout autre nœud qui arrive dans le réseau est un nœud *DM*.

Arrivé d'un nœud

Quand un nœud *DM* arrive dans le réseau il dispose de deux moyens pour trouver un nœud RM sur lequel il peut se connecter. ***(i)*** Si le nœud DMi connait un ou plusieurs nœuds *RM*, il envoie un message de diffusion *REQ()* et collecte toutes les identités des nœuds *RM*, qu'il garde dans un tableau ordonné par l'identifiant. Il peut ainsi se connecter au nœud *RM* qui a l'identifiant le plus grand. Si ce dernier se déconnecte,

alors le *DMi* le supprime du tableau et se connecte au nœud RM suivant ; ***(ii)*** Si par contre le nœud *DM* ne connait aucun nœud *RM*, il doit effectuer une découverte sur le réseau local (par exemple, grâce à un multicast) ou contacter un service d'annuaire qui peut indiquer l'identifiant d'un nœud RM_i. Comme la manière de trouver le nœud RM_i dépend de l'implémentation, elle n'est pas précisée dans notre architecture.

Panne d'un nœud

Les nœuds peuvent subir deux types de pannes : des pannes volontaires ou des pannes involontaires. Les pannes sont détectées soit par des messages périodiques de type *Pull* (ou *heartbeat*), soit à la demande, par des messages *Push* (*ping-pong*).

Dans le cas d'une déconnexion involontaire, les nœuds RM_i $\{RM_1,\ldots,RM_n\}$ envoient des messages $req_i(ping)$ respectivement $\{req_1,\ldots,req_n\}$ au nœud **c** et initialisent des temporisateurs d'attente $\{ta_1,\ldots,ta_n\}$. ***(i)*** Si le nœud RM_i ne reçoit pas de message $req_i(pong)$ du nœud **c** jusqu'à l'expiration du temps ta_i, alors le nœud RM_i interroge son voisin directe $\boldsymbol{RM_j}$ avec un message jeton initialisé à faux. ***(ii)*** Si $\boldsymbol{RM_j}$ reçoit le message $req_j(pong)$ du nœud ***c*** avant l'expiration de son temps d'attente $\boldsymbol{ta_j}$, alors $\boldsymbol{RM_j}$ modifie la valeur du *jeton* à **vrai** et retourne le message jeton à son émetteur RM_i. Ceci signifie (indirectement) que le nœud ***c*** n'est pas déconnecté. Alors le nœud RM_i peut envoyer à nouveau un message au nœud ***c***. ***(iii)*** Sinon $\boldsymbol{RM_j}$ fait suivre le message jeton à son voisin directe avec la valeur **faux.** ***(iv)*** Si le jeton revient avec la valeur **faux** à $\boldsymbol{RM_i}$ alors tous les nœuds **RM** $\{RM_1,\ldots,RM_n\}$ constatent le départ du nœud **c** et exécutent la procédure d'élection d'un nouveau nœud **c**.

Dans le cas d'une déconnexion volontaire, ***(i)*** un nœud **DM** qui décide de quitter le réseau il informe son nœud **RM**, qui va mettre à jour son indexe. Durant la déconnexion le nœud **DM** en question ignore les messages entrants. ***(ii)*** Si par contre il s'agit d'un nœud **RM** (ou **c**) qui décide de quitter le réseau, il doit informer ses voisins **DM** (respectivement **c**). Ces derniers exécutent un algorithme d'élection entre eux pour élire un nouveau **RM** (respectivement **c**) qui prendra en charge les éventuels DM (ou RM) orphelins.

5. Conclusion

Nous avons présenté les premiers éléments de la spécification de GrAPP&S, notre architecture de gestion de données. Cette architecture est flexible et modulaire, et adapte une approche orientée services permettant de gérer l'hétérogénéité des données. La validation de cette spécification nous permettra de définir les lignes qui guideront la suite de nos travaux. Nous souhaitons définir des stratégies pour le stockage pour le stockage des données, la réplication des données pour assurer leur disponibilité et aussi le routage des requêtes de recherche.

6. Bibliographie

[1] Clarke, O. Sandberg, B. Wiley, and T. W. Hong. "Freenet: A distributed anonymous information storage and retrieval system". *In Workshop on Design Issues in Anonymity and Unobservability*, ICSI, Berkeley, CA, USA, July 2000, 311–320, *Lecture Notes in Computer Science* No 2009, 2001.

[2] B. Cohen, "Incentives build robustness in Bittorrent". *In Workshop on Economics of Peer to Peer Systems*, Berkeley, USA, May, 2003.

[3] L. Garcés-Erice, P. Felber, E. W. Biersack, G. Urvoy-Keller and K. W. Ross: "*Data Indexing in Peer-to-Peer DHT Networks*". ICDCS 2004, p. 200-208, 2004.

[4] O. Heckmann, A. Bock, A. Mauthe, R. Steinmetz. " The eDonkey File-Sharing Network" *in Proceedings of Workshop on Algorithms and Protocols for Efficient Peer-to-Peer Applications*, Informatik 2004, September 2004.

[5] M. Ji. "Hierarchical Bidirectional Chord". *In 2010 International Conference on Educational and Information Technology (ICEIT 2010)*, p. 486-489. IEEE Xplore, 2010.

[6] J. Liang, R. Kumar, and K.W. Ross, "The Kazaa Overlay: A Measurement Study", *Elsevier Computer Networks Journal (special issue on Overlays)*, 2005

[7] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim. "A survey and comparison of peer-to-peer overlay network schemes". *IEEE Communication Survey and Tutorial*, march 2004.

[8] S. Oaks, B. Traversat and L. Gong. "*JXTA in a Nutshell*", O'Reilly, 2002.

[9] Z. Peng, Z. Duan, J.-J. Qi, Y. Cao, and E. Lv. "HP2P: A hybrid hierarchical P2P network". *ICDS '07*, pages 18–, Washington, DC, USA, 2007. IEEE Computer Society.

[10] S Ratnasamy, P Francis, M Handley, R Karp, S Shenker, "A Scalable Content-Addressable Network", *ACM SIGCOMM Computer Communication Review* 31 (4), 161-172

[11] Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, pages 329-350, November, 2001.

[12] S. Saroiu, P. K. Gummadi, S. D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts". *In Multimedia Systems Journal*, Volume 9, Number 2, pp. 170-184 August 2003, Springer-Verlag.

[13] Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. Frans Kaashoek, F. Dabek and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for Internet applications". *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, pp. 17-32, February 2003.

[14] Yu and M. Li: "CBT: A proximity-aware peer clustering system in large-scale BitTorrent-like peer-to-peer networks". *Computer Communications* 31:(3). p. 591-602, 2008.