Déploiement de fichiers sur Grid'5000

Raphaël Jamet, Guillaume Huard
Équipe MOAIS, LID - INRIA Rhône-Alpes
jamet.raphael@gmail.com - guillaume.huard@imag.fr

***Résumé*—L'objectif de ce stage est d'améliorer les procédés de diffusion de fichiers utilisés dans Grid'5000. Plutôt qu'un transfert en pipeline[1], nous avons essayé d'utiliser les arbres de diffusion[2] afin de créer un outil plus performant et robuste, adapté à la structure réseau.**

***Index Terms*—Architecture distribuée, Grid'5000, arbre de diffusion, transfert de données**

I. Introduction

TRIVIAL comme problème dans le cas d'un ordinateur classique, le transfert de données devient problématique dès que l'on entre dans le cas de calculs distribués entre de nombreuses machines (dans une grille). Nous avons ici étudié l'envoi de données d'un noeud à un ensemble d'autres, scénario rencontré dans le travail quotidien d'un administrateur par exemple. Il est important ici de réduire les temps de transferts, principalement en répartissant au maximum les tâches d'envoi de données.

II. Procédés existants

Dans toutes les solutions existantes, on retrouve un découpage du fichier en parties, afin de mieux distinguer l'état courant et de simplifier les algorithmes.

Deux paramètres sont notables : le degré de centralisation, et le degré de déterminisme. La solution la plus communément utilisée actuellement utilise un pipeline fixe de transferts[1][3], et est donc déterministe. Tout transfert suivra une topologie linéaire déterminée au lancement : une fois la première partie reçue par un noeud, celui-ci la fera suivre au suivant tout en recevant la suivante. A l'opposé, les procédés pair à pair comme Bittorrent[4][5] sont relativement décentralisés (un serveur central assure la mise en relation des pairs) et non déterministes. Ce protocole entraîne cependant des surcoûts importants et n'utilisera pas intelligemment la structure réseau dans sa forme canonique.

Nous utiliserons comme notation n le nombre de nœuds total, racine comprise, p le nombre de parties dans lesquelles le fichier à transférer est découpé, d le débit et f la taille du fichier.

III. Solution adoptée

A. Philosophie

Dans un premier temps, il a été décidé de ne pas décentraliser complètement le processus. La structure de Grid'5000 est fiable, et le déploiement de fichiers n'a d'intérêt que lorsqu'il est intégral : le crash d'un nœud est improbable à notre échelle (quelques minutes au plus), et compromet de toute manière l'intégralité du processus. De la même manière, la fiabilité, l'immuabilité et l'homogénéité (locale aux clusters) de l'environnement encouragent l'utilisation d'un procédé avec des connexions fixes, réduisant ainsi les coûts d'établissement de ces connexions.

Manuscrit écrit dans le cadre d'un stage de Magistère 1ère année dans le Laboratoire Informatique et Distribution de Montbonnot, sous la tutelle de l'INRIA. Nous tenons à remercier Jean-Noël Quintin, Nicolas Gast et Gaël Gorgo pour leur aide précieuse.

B. Algorithme

L'algorithme choisi pour répondre à ces décisions est dérivé de celui décrit par Karp, Sahay, Santos et Schauser[2]. Il se base sur l'entrelacement de deux arbres binaires afin d'obtenir deux graphes de diffusion de paquets d'arité minimale, qui s'alterneront au cours du processus (le temps est discrétisé, l'unité est un temps de transfert, et nous utilisons différemment les temps pairs et impairs).

On place tous les n nœuds dans un vecteur (numéroté de 0 à $n-1$), avec la racine comme nœud 0. Dans un premier temps, la racine envoie en temps pair au nœud 1 et en temps impair au nœud nb. Ensuite, chaque nœud l non racine envoie en temps pair au nœud $2l$ et en temps impair au noeud $2l+1$ s'ils existent. Enfin, chacun des nœuds envoient à $2l-n$ et $2l-n-1$ selon leur existence et leurs disponibilités réciproques. A noter que les envois suivent une certaine logique, puisqu'on distingue au cours du processus une logique dans les parties envoyées par un nœud. Nous ne nous sommes cependant pas servis de cette propriété.

IV. Temps de diffusion théoriques

Pour qu'un transfert soit complet d'un nœud à un autre, il nous faudra trivialement f/d unités de temps. Prenons désormais le cas de n nœuds placés en chaine : pour amener un fichier entier d'un bout à l'autre de la chaine, il faudra $(n-1)f/d$ unités de temps. Ensuite, décomposons ce fichier, afin de ne pas attendre le transfert entier pour faire suivre des données. Sachant que p parties doivent traverser, on obtient :

$$t = \frac{f}{d} + (n-1)\frac{f/p}{d}$$

Considérons maintenant la structure décrite dans la partie précédente. Une partie envoyée dans un des arbres n'empruntera que les chemins de cet arbre : la longueur maximale d'une chaine de transferts est donc égale à la profondeur maximale d'un arbre, soit $(\lceil \log_2(n) \rceil)$. Mais, étant donné que les envois sont alternés entre les fils d'un même nœud, il faut compter le double de cette valeur si on veut connaître le temps de traversée de cet arbre pour une partie. On obtient donc pour le temps total (le temps de transfert étant constant) :

$$t = \frac{f}{d} + (2\lceil \log_2(n) \rceil)\frac{f/p}{d}$$

Le gain est significatif, puisque l'ordre de grandeur du temps de propagation est changé de n à $2\lceil \log_2(n) \rceil$. Ajoutons maintenant les surcoûts liés au réseau et au protocole, considérés comme uniformes (avec o le surcoût pour un transfert) :

$$t = \frac{f}{d} + po + 2\lceil \log_2(n) \rceil (\frac{f/p}{d} + o)$$

Ces résultats sont assez proches de ceux obtenus par Mundinger, Weber et Weiss dans leur étude statistique[5]. On voit désormais qu'il existe un p optimal en fonction de d, f, n et o, qu'on peut trouver via la formule suivante :

$$p_{ideal} = \sqrt{\frac{2\lceil \log_2(n) \rceil . f}{od}}$$

V. Implémentation

A. Protocole

Pour commencer, un protocole de transfert à utiliser a été choisi. Nous avons écarté FTP par souci de légèreté, TFTP parce qu'il utilise UDP, et SCP parce qu'il est inutilement crypté : au final, un protocole ad hoc a été choisi et conçu. Il est plutôt primitif, avec un souci de lisibilité plutôt que de performance, et s'appuie sur TCP.

B. Processus de construction des arbres

Il a fallu trouver un moyen fiable de générer la chaîne de connexions nécessaire au déroulement des transferts, sans interblocages. Nous procédons arbre par arbre, en commençant par mettre tous les nœuds en écoute de leurs parents sur l'arbre choisi. La racine se connecte à son fils, qui à son tour se connecte à ses fils, jusqu'au bout, et idem pour le second arbre, évitant tout interblocage. Il faut au pire $2.\lceil \log_2(n) \rceil$ connexions séquentielles pour établir l'interconnexion voulue.

C. Résultats expérimentaux

Grid'5000 [1] est la grille de l'INRIA [2]. C'est un outil conçu pour l'étude des systèmes parallèles à grande échelle. Tous les essais pratiqués dans le cadre de cet article ont été réalisés sur le site de Grenoble (détails sur le site). Actuellement, le logiciel est fonctionnel, mais la performance n'atteint ni la stabilité ni les niveaux attendus comme le montre la figure 1. Nous cherchons les causes de ce ralentissement.

D. Evolution vers une dynamicité accrue

L'implémentation actuelle a été effectuée en ayant à l'esprit une future extension vers d'avantage de dynamicité. Premièrement, nous ne nous servons pas du déterminisme dans les envois de parties : chaque noeud émetteur choisit une partie à envoyer selon ce que son interlocuteur a. Au niveau de la fin d'un transfert, un noeud dira à ses émetteurs qu'il n'a plus besoin de leurs services, ce qui terminera le dialogue.

1. http://www.grid5000.fr
2. http://www.inria.fr

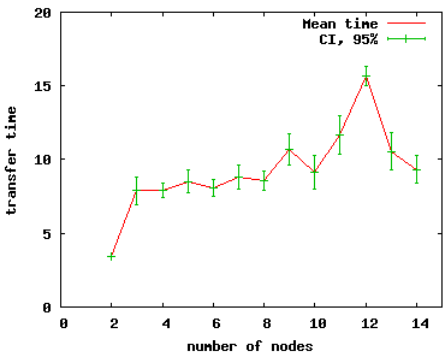

FIGURE 1. Temps de tranfert d'un fichier de 350 Mo en fonction du nombre de noeuds impliqués

E. Extensions envisageables

La plupart des extensions possibles nécéssitent un choix : préfère-t-on la performance, la robustesse, ou la capacité d'adaptation ?

La robustesse du procédé est un point grandement améliorable : par exemple, si un nœud client plante, le programme aurait du mal à le contourner. Dans le même registre, le processus n'est pas capable de reconstruire les connexions coupées temporairement, et un ralentissement local freinera exagérément le reste des nœuds. Une architecture pair à pair serait grandement avantageuse de ce point de vue, mais on perdrait beaucoup en performance dans les cas de réseaux fiables et homogènes.

L'optique initiale de ce logiciel était de permettre un déploiement entre divers sites ou clusters. Le problème des points d'engorgement dans le réseau et de l'hétérogénéité de la plateforme se pose, et la réponse que nous avons considéré utilise un anneau de transferts entre les frontend, suivi d'un transfert en arbre dans les divers clusters. Il faudra cependant permettre aux nœuds de désynchroniser leurs transferts entre leurs deux "phases" pour que cette solution soit efficace.

VI. Conclusion

Les résultats théoriques sont pour l'instant très prometteurs, et l'utilité d'un tel logiciel n'est pas à démontrer. Il reste également beaucoup de pistes à développer, laissant espérer un outil pouvant être aussi bien spécialisé pour une grille particulière que d'utilisation générale.

Références

[1] B. Videau, C. Touati, and O. Richard, "Toward an experiment engine for lightweight grids," in *Proceedings of the MetroGrid workshop : Metrology for Grid Networks*, 2007, pp. 142–153.

[2] R. M. Karp, A. Sahay, E. E. Santos, and K. E. Schauser, "Optimal broadcast and summation in the logp model," in *In Proc. 5th ACM Symp. on Parallel Algorithms and Architectures*, 1993, pp. 142–153.

[3] T. Hoshino, K. Taura, and T. Chikayama, "An adaptive file distribution algorithm for wide area network," *AGridM 2003 : Workshop on Adaptive Grid Middleware*, 2003.

[4] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Understanding bittorrent : An experimental perspective," *INRIA Sophia Antipolis/INRIA Rhone-Alpes-PLANETE*, 2005.

[5] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," *J. of Scheduling*, vol. 11, no. 2, pp. 105–120, 2008.