Performances des stratégies de distribution des tâches et des données sur Grille

Impact du partage de données

Hamza Adamou[*,**] — Jean-François Méhaut[**]

[*] Département d'Informatique, Faculté de Sciences
Université de Yaoundé 1
BP 812 Yaoundé - Cameroun
hamza.adamou@imag.fr

[**] Laboratoire Informatique de Grenoble (LIG)/ équipe Mescal
Antenne ENSIMAG de Montbonnot, 51 avenue Jean Kuntzmann
38330 Montbonnot - France
jean-francois.mehaut@imag.fr

RÉSUMÉ. On se propose d'étudier les stratégies de répartition de données et d'ordonnancement des applications sur une grille. Nous faisons une analyse des stratégies d'ordonnancement de tâches et de répartition de données sur Grille. Nous définissons deux stratégies de stockage et transfert de données et deux stratégies d'ordonnancement de tâches. Un prototype expérimental pour l'étude comparative de ces stratégies à été développé. Nous comparons sur ce prototype une approche de distribution de données à partir d'un serveur central avec une approche distribuée utilisant BitTorrent. Nous évaluons en particulier l'impact du taux de partage de données sur les performances des stratégies d'ordonnancement de calculs et distribution des données dans une grille

ABSTRACT. We propose to look at strategies for distribution of data and Scheduling applications on a Computational grid. We are doing an analysis strategies for task scheduling and distribution of data on Grid. We define two strategies for storage and transfer of data and two strategies for task scheduling. An experimental prototype for the study comparison of these strategies has been developed. We compare this prototype a distribution data from a central server with a distributed approach using BitTorrent. We evaluate the impact particularly in the rate of data sharing on the performance of scheduling strategies of tasks and distribution data in a Grid

MOTS-CLÉS : Grille de calcul, Stockage distribué, BitTorrent, Ordonnancement

KEYWORDS : Grid computing, Distributed storage, BitTorrent, Scheduling

1. Introduction

Nous assistons à l'émergence des applications de plus en plus gourmandes en ressources de calcul et de stockage et pour lesquelles les plates-formes actuelles s'avèrent insuffisantes. Nous pouvons citer parmi ces applications celles dites "données intensives". Ces applications accèdent et manipulent des données de grandes tailles (allant de plusieurs centaines de méga-octets aux giga-octets) stockées dans des serveurs distants. Une solution à ces exigences consiste à agréger les ressources informatiques disséminées à travers le monde. Cette idée a donné naissance au concept de grille de calcul (*Grid computing*)[1].

L'exécution des ces applications sur ces nouvelles plates-formes nécessite le déplacement des quantités énormes de données du site de stockage vers le site d'exécution, puis le rapatriement des résultats après l'exécution. Certaines données parce que utilisées par plusieurs tâches sont transférées plusieurs fois et souvent de manière simultané d'un serveur de stockage vers les différents sites d'exécution. Il faut donc trouver des mécanismes de distribution de données qui soient efficace pour ces types d'applications.

Parallèlement les sites étant hétérogènes, il faut pouvoir déterminer de manière judicieuse pour une tâche donnée le site de calcul qui lui convienne le mieux. On se pose généralement la question de savoir s'il faut privilégier une approche où on activerait des calculs près du lieu de stockage de données ou plutôt transférer les données vers les sites de calculs ? Il n'y a pas de stratégie générale qui soit efficace pour tous les types d'application. Certaines études ont montré la difficulté de ce problème [3][4]. Une évaluation des différentes techniques est requise pour choisir une solution appropriée pour une application spécifique, d'où la nécessité de disposer d'un outil d'étude et d'évaluation des techniques d'ordonnancement de calcul et de répartition de données dans les environnements de type grille.

Dans ce papier, nous définissons deux stratégies de stockage et transfert de données et deux stratégies d'ordonnancement des tâches. Nous avons implémenté un prototype expérimental pour faciliter l'étude et l'évaluation des stratégies définies. Nous avons par la suite mené sur ce prototype des expérimentations visant à évaluer une approche de stockage central client/serveur avec une approche distribuée de type BitTorrent[5].

Le reste du document est organisé comme suit : La section 2 décrit les différentes approches d'ordonnancement de tâches et répartition de données utilisées. La section 3 présente le prototype et les différents éléments qui le constituent. Les expérimentations réalisées et les résultats obtenus sont présentés à la section 4. La section 5 passe en revue quelques travaux connexes. Et nous terminons par une conclusion.

2. Stratégies d'ordonnancement de tâches et répartition de données sur une grille

Nous considérons une grille comme un ensemble de sites possédant chacun une certaine puissance de calcul et une capacité de stockage. Les sites supportent l'exécution de tâches utilisant des fichiers stockés sur le disque local. Les paramètres d'une tâche se trouvent stockés dans des fichiers d'entrée en lecture seule, les résultats stockés dans des fichiers en écriture.

Une application est constituée d'un ensemble de tâches indépendantes avec des fichiers d'entrée et de sortie. Le problème est d'ordonnancer les tâches vers les sites de la grille, de s'assurer

qu'une tâche possède les fichiers en entrée avant de commencer son exécution et que les résultats d'une tâche sont stockés sur le disque du site. Nous devons donc étudier comment les fichiers sont transférés et stockés sur les disques locaux des sites. Nous devons également étudier comment les sites d'exécution sont choisis pour les tâches. Nous définirons deux stratégies de stockage et transfert de fichiers et deux stratégies d'ordonnancement des tâches.

2.1. Stockage et répartition des fichiers

Nous allons maintenant décrire comment les fichiers sont répartis et stockés sur les différents sites de la grille. Deux approches sont présentées pour effectuer le transfert de fichier d'un site source vers un site destination :

1- transfert point à point : Le fichier est détenu par un site. Un autre site souhaite en obtenir une copie. Une communication point à point entre le site détenteur et le site demandeur va être établie pour transférer le contenu du fichier. Même si le fichier est détenu par plusieurs sites, le site demandeur ne s'adresse qu'à l'un des sites détenteurs (figure 1-a). La limitation peut provenir de la capacité en bande passante du site détenteur. Si le site détenteur est sollicité pour plusieurs transferts, la bande passante sortante est partagée pour les différents transferts.

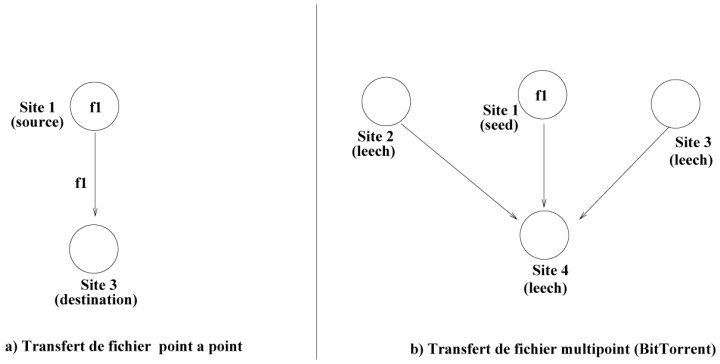

Figure 1. *Exemple de transfert point à point et multipoint*

2- transfert multipoint (BitTorrent) : Le site demandeur d'un fichier récupére le contenu du fichier à partir de tranches que lui enverront plusieurs sites détenteurs. C'est une approche collaborative impliquant plusieurs sites émetteurs (figure 1-b). La limitation ne pouvant maintenant plus se produire au niveau du site demandeur du fichier. Pour mettre en place cette stratégie de transfert multipoints, nous avons utilisé le système de partage de fichiers BitTorrent[5][10].

Les deux approches ainsi présentées sont utilisés pour les transferts de fichiers entre les sites. A l'initialisation du système ces fichiers sont, soit tous stockés sur un seul serveur de fichier, soit distribués (en au moins un exemplaire chacun) sur les différents sites de calcul.

2.2. Répartition de tâches

Nous décrivons ici comment les sites d'exécution sont choisis pour l'exécution des tâches. Deux stratégies sont utilisées pour cela :

1- FIFO (First In First Out) : Un site libre demande une tâche. La première tâche dans la liste de tâches non encore exécutées lui est attribuée. Ainsi l'ordre d'attribution de tâches aux sites est identique à l'ordre de création de ces tâches.

2- NDP (Node Data Present) : La tâche qui est attribuée à un site libre est celle pour laquelle il y a au moins un des fichiers d'entrée sur le site demandeur. Les tâches sont donc attribuées selon la localité des fichiers qu'elles utilisent. La stratégie FIFO est utilisée dans le cas où le site demandeur de tâches ne contient aucun des fichiers des tâches non encore exécutées.

Avec les stratégies de répartition (de données et tâches) ainsi définies, nous voulons évaluer l'impact de chacune d'elles sur les performances du calcul sur grille. Quel serait l'avantage d'avoir un stockage initial distribué par rapport à un stockage centralisé ? Quel est l'impact du taux de partage de fichiers sur chacune de stratégies définies ? Pour répondre à ces questions, nous avons besoin d'une plate-forme qui nous permettra de mettre en œuvre puis d'évaluer les différentes stratégies proposées. Cette plate-forme devra prendre en entrée une application constituée d'un ensemble de tâches manipulant de données et un ensemble de stratégies à évaluer. Puis exécuter ces tâches sur les sites de calcul disponibles suivant les stratégies définies et produit en sortie les résultats et le temps d'exécution total. L'implémentation du premier prototype de cette plate-forme est détaillée dans la section suivante.

3. Prototype expérimental

Au vu de ce qui précède, nous avons soulevé la nécessité de disposer d'un outil expérimental qui va permettre l'exécution et l'évaluation des différentes stratégies d'ordonnancement de tâches et de distribution de données. Nous présentons donc dans cette section notre prototype expérimental, les différents éléments qui le constituent et leur interaction.

Le but du prototype est de faciliter la mise en œuvre et l'évaluation des stratégies de répartition de données et d'ordonnancement de calculs dans les environnements distribués de type grille. Il est conçu de manière modulaire pour faciliter sa modification, son extension et son adaptation. Il met en relation principalement trois composants (figure 2), ce découpage a été choisi pour faciliter son évolution et l'intégration de nouvelles stratégies. La modification d'un seul composant suffit pour changer une stratégie ou bien en intégrer une nouvelle.

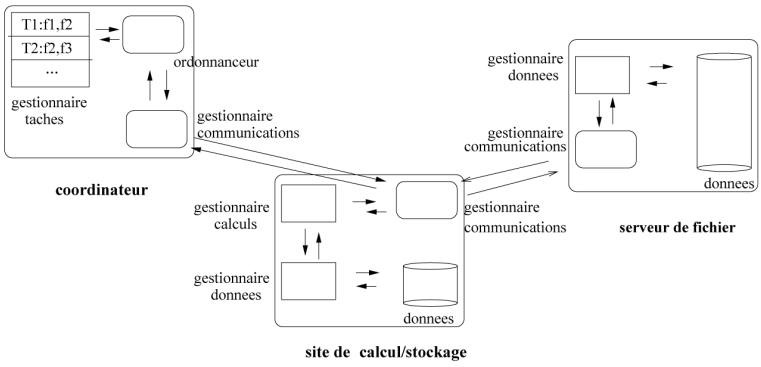

Figure 2. *Architecture du prototype*

1- Coordinateur : Le coordinateur est le composant qui assure la mise en relation des demandes de ressources (traitements et fichiers) avec les ressources disponibles (sites de calcul). Il déploie les calculs sur des ressources géographiquement distribuées et assure la coordination globale du système. Il est constitué d'un gestionnaire de tâches pour l'ajout et la suppression des tâches, d'un ordonnanceur et d'un gestionnaire de communication pour l'interaction avec les autres composants. En fonction des données déjà présentes sur un site et la liste des tâches non encore exécutées, l'ordonnanceur choisit une tâche pour le site suivant la stratégie donnée.

2- Serveur de fichiers : Le serveur de fichiers est celui qui détient initialement l'ensemble de fichiers disponibles pour l'exécution des tâches. Il est constitué d'un gestionnaire de données qui s'occupe de la politique d'accès aux données et d'un gestionnaire de communication. Le gestionnaire de communication assure la communication entre le serveur et les sites de calcul. Il reçoit des connexions des sites demandant le transfert d'une donnée, contacte le gestionnaire des données qui lui renvoie la donnée en question qu'il transmet au site demandeur.

3- Site de calcul : Un site de calcul est l'entité qui s'occupe du calcul et/ou du stockage. Il dispose d'un espace pour stocker les données à utiliser pour le calcul et d'une unité de traitement pour exécuter les tâches. Il est constitué d'un gestionnaire de stockage qui gère la liste de fichiers présents sur le site, d'un gestionnaire de calcul pour l'exécution des tâches et d'un gestionnaire de communication pour la connexion avec les autres composants.

A l'issue de l'implémentation du prototype, nous avons réalisé une campagne d'expériences combinant les stratégies de répartition tâches et de données présentées dans la section précédente. Les expériences réalisées et les résultats obtenus sont présentés dans la section suivante.

4. Expérimentations et Résultats

Dans cette section, il est question de l'évaluation des stratégies définies sur le prototype. Nous présentons d'abord le détail des expériences réalisées, puis les différents résultats obtenus.

4.1. Plan d'expérimentations

Les expérimentations ont été menées sur les applications synthétiques, les tâches et les données sont générées de manière aléatoire. Chaque tâche utilise deux fichiers d'entrée pour s'exécuter. Un fichier est dit partagé lorsqu'il est utilisé par plusieurs tâches différentes. Le taux de partage de fichiers est le rapport entre le nombre de fichiers partagés et le nombre total de fichiers. Le temps d'exécution d'une tâche est la somme du temps de transfert de données et du temps de calcul de la tâche. Le temps global d'exécution est quant à lui la différence entre la fin de la dernière tâche exécutée et le début de la première.

Une série d'expérimentations a été menée en faisant varier le nombre de site de calcul et le taux partage des fichiers. Pour chacun des cas, deux types de stockage initial ont été utilisé :

1- Stockage initial centralisé : Les données sont stockées au niveau d'un seul serveur centralisé. Chaque site y accède afin de récupérer les données nécessaires à l'exécution d'une tâche. Le serveur se charge seul du transfert des données vers tous les sites demandeurs. Ces échanges se font soit en point à point, soit en multipoint (BitTorrent).

2- Stockage initial distribué : Ici, les données sont directement stockées au niveau des différents sites. Chaque site devient donc le serveur des données qu'il héberge. Un site demandeur

récupère le fichier chez le serveur le détenant. Comme précédemment les données sont transférées en point à point ou en multipoint.

4.2. Résultats

Après avoir présenté en quoi consistait nos expérimentations, nous présentons dans cette partie l'environnement d'exécution utilisé et les différents résultats obtenus.

Les expériences ont été mené sur la grille de calcul du projet Grid5000 [11]. Cette grille fédère un ensemble de ressources de calcul reparties sur dix sites en France. Les ressources utilisées pour nos expériences étaient reparties sur les quatre grappes du site de Rennes.

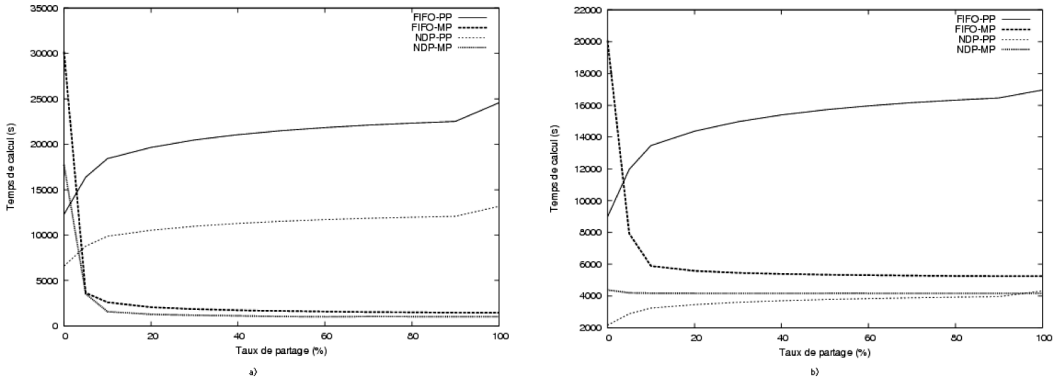

Figure 3. *Résultats sur 10 sites*

La figure 3 présente les résultats sur dix sites. FIFO-PP, FIFO-MP, NDP-PP et NDP-MP représentent respectivement la combinaison des approches de répartition de tâches FIFO et NDP avec les approches de distribution de données PP et MP. Pour chacun des cas, les expérimentations ont été réalisées avec un stockage initial centralisé et un stockage initial distribué.

Dans les combinaisons incluant la stratégie PP (i.e. FIFO-PP et NDP-PP), le temps global d'exécution croit avec le taux de partage, alors que dans les combinaisons incluant la stratégie MP le temps décroît. Ceci s'explique par le fait que dans la stratégie point à point, la communication ne s'établit qu'entre le détenteur du fichier et celui qui en demande. Plus on a des fichiers partagés, plus la bande passante du détenteur sera partagée. Cela pourra créer des goulots d'étranglement qui ralentissent très grandement les transferts. C'est pour cela que dans ce cas le temps d'exécution croit avec le taux de partage. Contrairement à la stratégie point à point, les temps de transfert décroît considérablement en multipoint lorsque les fichiers sont partagés. Plus les fichiers sont partagés, plus ils seront sollicités, ainsi les temps de transferts seront réduits.

Nous observons ensuite un temps de calcul très élevé dans les combinaisons incluant le multipoint (FIFO-MP,NDP-MP) lorsque le taux de partage tend vers 0. Cela s'explique par le fait qu'avec un taux de partage tendant vers 0, le nombre de fichiers partagés est très réduit. Les transferts multipoint deviennent presque des transferts point à point ; à cela s'ajoute un surcoût lié à la phase d'initialisation de chaque transfert. Le surcoût lié à la phase d'initialisation des transferts en multipoint avec BitTorrent (récupération du torrent, récupération de la liste des

sites chez le *tracker* et connexion aux différents sites) est énorme et n'est pas compensé. Ce surcoût est compensé pour les taux de partage dépassant 5%.

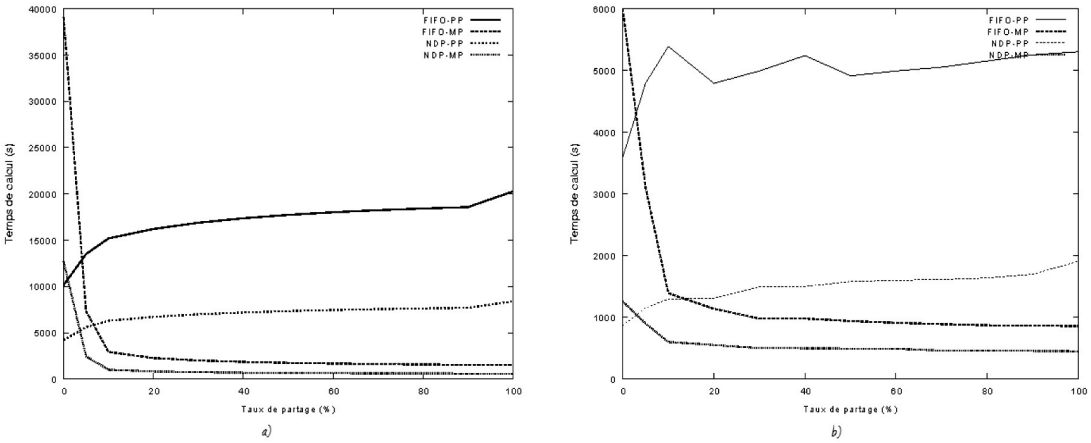

Figure 4. *Résultats sur 30 sites*

Les mêmes expériences ont été reprises avec 30 sites de calcul, les résultats sont présentés sur la figures 4

La prise en compte de la localité des données à travers les combinaisons incluant la stratégie NDP (NDP-PP et NDP-MP) améliore de manière significative les temps d'exécution. Avec les tâches cherchant les sites où se situent leurs données, le nombre de transferts est réduit. Le taux de partage a très peu d'influence dans ce cas. Les tâches utilisant les mêmes données sont envoyées sur les mêmes sites. Ainsi les approches NDP sont meilleures pour n'importe quel taux de transfert. La combinaison NDP-MP est la plus intéressante.

5. Travaux connexes

Le problème d'ordonnancement de tâches et de répartition de données sur grille a été intensivement étudié dans la littérature. Ces études ont été faites dans plusieurs contextes. On peut citer entre autres : l'ordonnancement des tâches sur les grilles de calcul (*Computational grids*)[6], l'ordonnancement découplé des tâches et des données[7] et l'ordonnancement simultané des tâches et des données [8].

Nous avons aussi un certain nombre de travaux qui ont été réalisés pour évaluer le protocole de partage collaborative BitTorrent sur les *grilles de PC*(*Desktop grid*) [9]. Les résultats obtenus montrent une bonne performance de BitTorrent dans les transferts de fichiers de grandes tailles par un grand nombre de machines, mais des performances faibles lors de transfert de petits fichiers. A partir de ces observations, quelques heuristiques d'ordonnancement de tâches indépendantes utilisant des grandes quantités de données, puis un modèle de performance qui

aide à décider le meilleur entre le protocole BitTorrent et FTP selon la taille de données et le nombre de nœuds ont été proposés.

6. Conclusion

Dans ce papier, après avoir présenté des techniques de répartition de tâches et de données sur une grille, nous avons proposé un prototype pour l'évaluation de ces techniques. Sur ce prototype a été réalisée une étude comparative de deux stratégies de distribution de données (centralisée et distribuée) couplée avec deux stratégies de répartition de tâches.

Les résultats des expérimentations montrent que, la stratégie de répartition de tâches FIFO se comporte bien pour les applications avec un faible taux de partage (moins de 10%). L'attribution des tâches aux sites sans prise en compte de la localité des données est mauvaise pour les applications où les tâches partagent énormément des données. Cela reste mauvais même lorsque le transfert des fichiers se fait en multipoint. La stratégie de répartition de tâches NDP se comporte mieux par rapport à FIFO. Les meilleurs résultats sont obtenus lorsque la stratégie de répartition de tâches NDP est combinée au transfert en multipoint avec un taux de partage de fichiers de plus de 10%. Ainsi, le choix d'une stratégie de répartition de tâches et de données à un impact significatif dans les performances des calculs sur grille.

La métrique de comparaison prise en compte jusqu'ici a été le temps d'exécution global, nous envisageons prendre en compte dans les travaux futurs d'autres métriques comme : le nombre de données transférées, l'impact des performances réseaux, la capacité de calcul de chaque site, la consommation de la mémoire, ... Nous envisageons aussi évaluer d'autres stratégies de répartition des tâches et de données et de proposer des stratégies mixtes et/ou adaptatives permettant de combiner plusieurs stratégies existantes ou bien de choisir la meilleure stratégie en fonction d'un paramètre donné.

7. Bibliographie

I. Foster et C. Kesselman. " The Grid : Blueprint for a New Computing Infrastructure ". Morgan Kaufmann, 1999.

Samer Al-Kiswany, Matei Ripeanu, Adriana Iamnitchi et Sudharshan Vazhkudai. " Are P2P Data-Dissemination Techniques Viable in Today's Data-Intensive Scientific Collaborations ? " In Euro-Par, pages 404-414, 2007

U. Cibej, B. Slivnik et B. Robic. " The Complexity of Static Data Replication in Data Grids ". Parallel Computing 31 (2005), 900-912.

A. Legrand. " Algorithmique Parallèle Hétérogène Et Techniques D'ordonnancement : Approches Statiques Et Dynamiques ". PhD Thesis, école Normale Supérieure de Lyon, dec 2003.

B. Cohen. " Incentives build robustness in BitTorrent. In Proc ". 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, June 2003.

H. Casanova, A. Legrand, D. Zagorodnov, et F. Berman. " Heuristics for scheduling parameter sweep applications in grid environments ". In Proceedings of the 9th Heterogeneous Computing Workshop,Cancun, Mexico, 2000.

K. Ranganathan et I. Foster. " Decoupling computation and data scheduling in distributed data-intensive applications ". In Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, Edinburgh, Scotland, 2002.

F. Desprez et A. Vernois. " Simultaneous Scheduling of Replication and Computation for Data-Intensive Applications on the Grid ". Journal Of Grid Computing 4, numéro 1 (mars 2006), 19 ?31.

Wei, B., Fedak, G., Cappello, F. : " Towards efficient data distribution on computational desktop grids with BitTorrent ". Future Gener. Comput. Syst.,23(8) :983-989, 2007

Bittorrent Protocol Specification. http ://wiki.theory.org/BitTorrentSpecification.

Grid'5000. https ://www.grid5000.fr.