Université – Nancy II

Étude comparative des algorithmes de découverte de la topologie de la grille

MÉMOIRE

soutenu le 29 Juin 2006

pour l'obtention du

Master Recherche Informatique de Lorraine - École Doctorale IAEM Lorraine

par

Ahmed HARBAOUI

Composition du jury

	Dominique Mery	Professeur UHP-Nancy1 et ESIAL
	Claude Godart	Professeur UHP-Nancy1
	Didier Galmiche	Professeur UHP-Nancy1
	Noëlle Carbonell	Professeur UHP-Nancy1
	Guy Perrier	Professeur à l'Université Nancy 2
Encadrants :	Martin Quinson	Maître de Conférences à l'UHP et ESIAL

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Remerciements

Au terme de ce travail, je tiens à assurer de ma profonde reconnaissance tous ceux dont j'ai eu à solliciter la compétence et l'expérience.

Ma sincère gratitude va tout d'abord à Martin Quinson mon encadrant au cours de ce stage, pour ses conseils et son soutien dans le choix de mes orientations.

J'adresse ensuite mes remerciements les plus vifs à Jens Gustedt, responsable du projet ALGORILLE, qui m'a offert l'opportunité de mener à bien mon stage de master reherhe au sein du laboratoire Loria.

Je ne saurai oublier Arnaud Legrand pour sa disponibilité et ses réponses pertinentes.

Enfin, mes remerciments s'adressent aussi à Tchimou N'Takpé, Flavien Vernier et à tous les membres de l'équipe ALGORILLE pour l'amabilité avec laquelle ils m'ont intégré dans leur équipe.

Je dédie ce travail
à ma mère, à mon père,
à ma soeur, à mon frère,
à la mémoire de ma grande mère
et à tous mes amis

Résumé

L'optimisation des applications distribuées et l'ordonnancement des tâches pour le calcul parallèle dans le cadre de grille de calcul nécessite la connaissance de la topologie et des performances du réseau. Plusieurs outils d'administration et de cartographie du réseau permettent aux administrateurs de détecter des éventuelles pannes. Néanmoins, ces outils ne leurs permettent pas d'obtenir une estimation des performances que les utilisateurs peuvent escompter du réseau. De telles estimations pourraient être exploité par les composants de la grille pour s'adapter automatiquement aux changements d'état de ces plates-formes fortement hétérogènes et dynamiques. Pour cela, il faut avoir une vue applicative plutôt que physique (interconnexion) de la topologie de la grille. Seuls quelques travaux théoriques ont été proposés pour résoudre ce problème dans le cadre des systèmes pair-à-pair et des grilles de calcul. Ces travaux fournissent une description de la topologie sous forme de graphes annotés par les performances des liens et/ou des machines. La comparaison quantitative et qualitative des ces travaux et la détermination de la meilleure topologie découverte font l'objet de mon stage de master.

Mots-clés: Grille de calcul, cartographie automatique, topologie applicative.

Abstract

The optimization of distributed applications and tasks scheduling in the context of grid computing require the knowledge of the network topology and performances. Several administration and cartography tools for networks enable the administrators to detect possible breakdowns. Nevertheless, these tools can not provide performances estimation about the network. These estimations could be exploited by the grid components to automatically adapt to the unpredictable changes in heterogeneous and dynamic platforms. For that, it is necessary to have an applicative point of view rather than a physical one about the grid topology.

To date, only some theoretical works were proposed to solve this problem in the context of peer-to-peer systems and grid computing. They provide a topology description based on links/hosts' performances annotated graphs. Both quantitative and qualitative comparison of these related works and the determination of the best topology discovered are the main focus of the present dissertation.

Keywords: Grid computing, topology mapping, applicative topology.

Table des matières

Table des figures

Introduction

Le développement du Web a révolutionné la façon d'accéder à l'information. Il est aujourd'hui possible d'accéder à n'importe quelle information électronique dans le monde via le web. La grille de calcul propose une révolution similaire pour les ressources de calcul et de stockage. Un des ouvrages fondateurs de la grille de calcul [FK99] décrit la grille comme un ensemble de super-calculateurs dispersés partout dans le monde, reliés par Internet et disposant d'énormes capacités de stockage. Cela autorise une classe d'applications entièrement nouvelle. Ces applications étaient inimaginables auparavant puisqu'elles demandaient des ressources qui semblaient démesurées. En rassemblant la puissance de calcul de milliers, voire des centaines de milliers, de machines et des espaces de stockage sur plusieurs sites, la grille de calcul propose donc une solution réaliste et déjà mise en pratique. Cette approche s'applique par exemple aux simulations scientifiques à grande échelle, à des programmes de réalité virtuelle ou encore à l'analyse de données issues de microscopes électroniques puis la reconstruction tridimensionnelle d'images pour une visualisation en temps réel des résultats.

La connaissance de la topologie du réseau mise à disposition de la grille est indispensable pour exploiter efficacement ces plates-formes. Leur nature à la fois dynamique et hétérogène impose aux applications de s'adapter aux variations des conditions extérieures. La communauté réseau dispose de plusieurs outils de découverte. Néanmoins, ces outils sont non exploitables dans le cadre d'une plate-forme à très large échelle. La surveillance des caractéristiques quantitatives de la plate-forme (bande passante, disponibilité, processeur et mémoire, etc) a été largement étudiée ces dernières années. Des outils spécialisés comme le NWS [WSH99] ou GANGLIA [MCC04] existent maintenant. Cependant, pour que cette découverte soit plus adaptée aux applications, il paraît judicieux d'avoir une vue applicative plutôt que physique (interconnexion) de cette topologie : une vue de la topologie telle que les applications la voient ou souhaitent la voir. Il s'agit donc de la topologie applicative. Peu de travaux (principalement théoriques) ont été proposés pour résoudre ce problème dans le cadre des systèmes pair-à-pair et des grilles.

L'objectif de ce mémoire est donc de mener une étude comparative des techniques disponibles pour la découverte de la topologie applicative de la grille. À partir de cette étude, découle un autre objectif à atteindre. Il s'agit d'évaluer expérimentalement le meilleur algorithme par rapport à la topologie découverte et par rapport à plusieurs critères indispensables pour l'optimisation des applications de la grille de calcul.

Dans le premier chapitre, nous commençons par présenter le contexte de travail et la problématique. Nous abordons dans le deuxième chapitre une étude de l'état de l'art. Nous

commençons donc par présenter les diffèrentes méthodes et outils de cartographie automatique des réseaux et plus precisement dans le cadre des grilles de calcul. Nous choisissons ensuite les algorithmes de découverte de la topologie qui répondent à nos besoins et qui offrent une vue applicative de la plate-forme. Ces algorithmes seront détaillés et comparés quantitativement dans un premier temps.

Dans le troisième chapitre, nous présentons notre méthodologie de comparaison des algorithmes de découverte de la topologie applicative. Cette approche consiste en la comparaison qualitative des algorithmes en utilisant des *Testcases*. Le choix de ces *Testcases* doit se faire de façon à garantir une bonne évaluation des algorithmes suivant plusieurs critères qui nous expliciterons à ce niveau.

Enfin, pour déterminer le meilleur algorithme de découverte de la topologie applicative, l'évaluation et l'analyse des résultats de l'expérimentation seront discutées dans le dernier chapitre. Nous terminons par conclure et dégager les perspectives du travail effectué.

Chapitre 1

Contexte et Problématique

Nous présentons brièvement dans ce chapitre le concept des grilles et ses caractéristiques. Ensuite nous poserons la problématique de recherche de la meilleure solution de découverte de la topologie de la grille.

1.1 Qu'est ce qu'une grille de calcul ?

La grille de calcul est une fédération hétérogène de machines géographiquement éloignées, qui permet d'optimiser et de gérer globalement l'accès et le partage de ressources, de données, et d'applications à grande échelle. Le principe en est de permettre le partage de moyens de calcul, de stockage et d'applications à travers un réseau de communication. Il peut être imaginé à l'échelle mondiale d'Internet ou plus modestement à l'échelle d'une entreprise qui, en interne, cherche à optimiser l'utilisation de ses moyens informatiques.

La grille doit donc faciliter l'accès aux ressources disponibles en incluant non seulement les ressources classiques, processeurs, mémoire et stockage, mais aussi les applications. Elle doit également assurer le partage de ces ressources et la collaboration au sein d'organisations virtuelles dans un environnement distribué, entre groupes de travail et entreprises indépendantes. Elle doit enfin rendre transparent l'hétérogénéité des composants et la complexité de l'infrastructure : systèmes d'exploitation et localisations variées, réseaux complexes, politiques de sécurité variables, etc. Les besoins en calcul intensif et de manipulation de très grandes quantités d'informations rendent nécessaire l'utilisation de moyens de traitement de l'information de plus en plus performants. Le calcul sur la grille apporte une solution dont le rapport performance / coût est extrêmement avantageux, en comparaison avec l'achat massif de super-calculateurs par exemple.

1.1.1 Caractéristiques de la grille

Dans [Sob03], les auteurs résument les différences entre la grille précédemment décrite et les super-calculateurs classiques de la manière suivante :

1. **L'hétérogénéité :** contrairement au super-calculateurs qui sont assez homogènes, une grille de calcul comprend généralement des nœuds hétérogènes par rapport au types de processeur, de capacité mémoire, etc.

2. **Réseau et transmission :** une grille de calcul a un réseau de transmission fortement hétérogène et non équilibré, comportant un mélange de différents réseaux et d'une variété de raccordements d'Internet dont les caractéristiques de bande passante et de temps de latence peuvent varier avec le temps et l'espace.
3. **Problème de disponibilité :** la grille a un problème de disponibilité de ses ressources puisque des nœuds peuvent quitter, tomber en panne ou s'ajouter à la grille.
4. **Passage à l'échelle** des applications dédiées à la grille : le grand nombre de machines regroupées par une grille impose à ses applications la contrainte de passage à l'échelle. Ceci rend caduque plusieurs algorithmes distribués développés dans le cadre des plates-formes plus modestes.
5. **Administration :** chaque nœud de la grille partageant ses ressources avec les autres, ne peut donner des privilèges spécifiques aux administrateurs de la grille, ce qui rend la tâche de gestion et d'administration très difficile.

1.2 Problèmatique

L'optimisation des applications distribuées et l'ordonnancement des tâches pour le calcul parallèle dans le cadre de grille de calcul nécessite la connaissance des performances de la grille ainsi que la topologie du réseau. Plusieurs outils d'administration et de cartographie du réseau permettent aux administrateurs de détecter des éventuels mauvais fonctionnements. Néanmoins, ces outils ne leurs permettent pas d'obtenir une estimation des performances que les utilisateurs peuvent escompter du réseau. Des estimations que tous les composants de la grille peuvent utiliser pour s'adapter automatiquement aux changements d'état de ces plates-formes fortement hétérogènes et dynamiques. Ceci rend impose que cette découverte soit de plus en plus adaptée aux applications et qu'elle ait une vue applicative plutôt que physique (interconnexion) de la topologie de la grille : on parle alors d'une topologie applicative. Seuls quelques travaux (principalement théoriques) ont été proposés pour résoudre ce problème dans le cadre des systèmes pair-à-pair et des grilles de calcul.

Nous nous intéressons dans ce rapport aux travaux de découverte de la topologie applicative du réseau, et leurs effets sur les communications de niveau application. Ces travaux fournissent une description de la topologie sous forme de graphes a nnotés par les performances des liens et/ou des machines. La comparaison de ces résultats et le choix de la meilleure solution est importante pour la détection des éventuelles limites et avantages de chaque solution et la validation des ses algorithmes par rapport au besoin de la grille. Toutefois la comparaison reste une tâche difficile vu la diversité de ces graphes et des différences des significations de leurs composantes. Nous présentons alors dans ce travail une méthodologie permettant la comparaison qualitative de ces solutions et la détermination de la meilleure topologie applicative découverte par rapport à des critères qui seront précisés ultérieurement.

Conclusion

Dans ce chapitre, nous avons présenté la notion de grille de calcul et nous avons précisé les différentes caractéristiques de ses plateformes à savoir l'hétérogénéité et les contraintes de passage à l'échelle imposées à ses applications. Nous avons donné ensuite les enjeux qui ont

motivé le travail effectué dans ce rapport, à savoir la découverte de la topologie applicative et la détermination du meilleur algorithme de découverte.

Chapitre 2

État de l'art

Nous présentons dans ce chapitre un tour d'horizon des travaux et des outils de découverte de la topologie dans le cadre des réseaux et des grilles de calcul. Nous introduisons dans un premier temps les méthodologies et les outils de cartographie des réseaux. Nous présentons ensuite les algorithmes de découverte dédiés aux plates-formes très large échelle. Nous comparons enfin la phase d'acquisition des mesures et la phase de construction de graphes de ces algorithmes.

2.1 Cartographie automatique

2.1.1 Méthodologie de mesure

On distingue des mesures actives et d'autres qui sont passives :

1. **Les mesures actives :** Le principe des mesures actives consiste à générer du trafic dans le réseau à étudier et à observer les effets des composants et des protocoles réseaux et transport sur le trafic : taux de perte, délai, Round Trip Time (RTT), topologie, etc.
2. **Les mesures passives :** leur principe est de regarder le trafic existant et d'étudier ses propriétés en un ou plusieurs points du réseau. L'avantage des mesures passives est qu'elles ne sont absolument pas intrusives et ne changent rien à l'état du réseau. En revanche, il est très difficile de déterminer les informations manquantes pour avoir une vue globale du réseau.

2.1.2 Outils de cartographie pour les réseaux

Outils utilisant l'approche active

Parmi les outils de mesure active, les plus connus et certainement les plus utilisés sont *ping*[1] et *traceroute*[2]. Ces programmes permettent respectivement de déterminer la latence et l'itinéraire qui séparent deux nœuds de réseau.

– **ping :** il s'appuie sur le protocole ICMP pour mesurer le temps séparant l'émission d'une requête de la réception de sa réponse. La principale limite de cet outil est que les paquets

[1] http ://www.die.net/doc/linux/man/man8/ping.8.html.
[2] http ://www.die.net/doc/linux/man/man8/traceroute.8.html.

ICMP peuvent être utilisés pour mener des dénis de Service (DoS) et que beaucoup d'administrateurs préfèrent donc bloquer ces paquets.

– **traceroute :** C'est la durée de vie des paquets ICMP (Time-To-Live ou TTL) qui est principalement mise à contribution dans le cas de la commande traceroute. En effet, en incrémentant progressivement la valeur de celle-ci, il est possible d'obtenir des informations sur toutes les machines traversées en chemin. De par son ancienneté, cet outil a été utilisé dans de nombreuses études visant à établir la topologie du réseau. En effet, bien que la méthode utilisée par traceroute ne permette pas de connaître les routes asymétriques, elle reste la plus simple à mettre en oeuvre. De plus, la présence de plusieurs dizaines de serveurs répartis dans le monde permet d'avoir une vision plus complète de la topologie du réseau.

D'autres outils sont apparus en apportant des améliorations à ceux que nous venons d'évoquer dont *TReno*, *Pathchar* et *Sting* sont les plus connus. En revanche, ces outils ainsi que le traceroute présentent plusieurs problèmes. Nous en citons principalement deux : (1) Ils se basent surtout sur le protocole ICMP que certains administrateurs l'interdisent. (2) Ces outils ne traitent pas le trafic concurrent qui présente un critère très important pour déterminer une vue applicative du réseau.

Outils utilisant l'approche passive

Parmi les outils de mesure passive, nous citons Tcpdump [JLM89] et CoralReef [KMK+01].

– **Tcpdump :** Il s'appuie sur la bibliothèque Pcap qui propose une interface de capture de paquets adaptée à la plupart des systèmes modernes. Pour chaque paquet capturé, tcpdump affiche une ligne le décrivant. Il est aussi possible d'obtenir une trace complète de tous les paquets observés pour une analyse ultérieure.

– **CoralReef :** Cette solution complète (à la fois logicielle et matérielle) a été conçue pour surveiller des réseaux de grande capacité. Cet outil permet d'obtenir un nombre de métriques intéressantes (latence, bande passante, etc) mais interdit l'analyse des protocoles applicatifs.

Le principal avantage de ces outils est qu'ils n'injectent aucun trafic supplémentaire dans le réseau, limitant ainsi au minimum leur intrusivité. En revanche, les mesures fournies étant incomplètes, il est très difficile de construire une vue globale du réseau par cette approche.

2.2 Algorithmes de découverte des grilles de calcul

La dynamicité, l'hétérogénéité et la taille de la topologie de la grille rendent les solutions de cartographie du réseau présentées précédemment inadaptées au contexte de grid computing. Néanmoins, ces dernières années beaucoup de travaux de recherche ont étudié la cartographie et la surveillance des réseaux dans le contexte des grilles. Ces algorithmes peuvent être classés en deux grandes classes :

1. Les algorithmes qui visent à déterminer le schéma d'interconnexion du réseau : ces approches sont utilisées dans le cadre d'administration et de surveillance de ces réseaux.
2. Les algorithmes qui visent à déterminer une topologie applicative du réseau : ces approches sont utilisées dans les cadre d'optimisation des communications collectives et des applications distribuées s'exécutant sur ces réseaux.

2.2.1 Choix des algorithmes

Les algorithmes visant la détermination d'un shéma d'interconnexion du réseau ont été largement étudiés. En effet, des outils spécialisés comme le NWS ou GANGLIA existent maintenant. Les informations collectées par ces outils sont plutôt quantitatives que qualitatives.

Par ailleurs, seuls quelques travaux visant la topologie applicative ont été proposés et ils sont principalement théoriques. Parmi ces algorithmes, nous pouvons citer l'algorithme de LATENCES, ENV, INCA. Ces algorithmes cherchent à découvrir une topologie adaptée aux applications : une topologie telle que les applications l'aperçoivent.

Dans la suite de cette étude, nous nous concentrerons sur les algorithmes de la deuxième classe. Nous commençons par présenter les solutions de LATENCES, ENV et INCA.

2.3 Algorithme ENV

ENV [SBW99, ENV04] (Effective Network View) a été développé par Gary Shao sous la direction de Francine Berman et Richard Wolski à l'Université de Californie à San Diego (UCSD) dans le cadre de projet de AppLeS. Son principe de base est de mesurer directement les interférences entre les flux de données. Pour cela, la bande passante normale entre deux machines données est comparée à celle obtenue lorsqu'un autre lien (entre deux autres machines) est saturé. Une variation indique alors que les deux liens partagent des ressources réseau, ce qui constitue une information capitale pour reconstruire la topologie d'interconnexion des machines.

L'avantage principal d'ENV est qu'il offre la vue du réseau telle qu'elle peut être ressentie par les applications en se fondant uniquement sur des expérimentations de niveau applicatif ne nécessitant donc pas de privilèges ou d'outils spéciaux pour être menées. En revanche, ENV ne cherche pas à donner une vue complète du réseau, mais simplement le point de vue d'une machine donnée (maître), ce qui est suffisant dans le paradigme maître/esclave visé par ENV. Cette simplification permet à ENV de mener à bien la cartographie d'une plate-forme d'une vingtaine de machines en quelques minutes. Nous présentons maintenant l'algorithme d'ENV.

2.3.1 Topologie structurelle : cartographie ne dépendant pas du maître

Il s'agit d'une première approximation de la topologie construite à l'aide de traceroute. Son objectif principal est de guider les tests actifs menés dans les phases postérieures. Chaque hôte de la plate-forme indique le chemin réseau entre lui-même et une machine extérieure choisie arbitrairement. Ces chemins sont ensuite combinés. Les hôtes utilisant le même chemin pour sortir du réseau local sont placées dans la même branche de l'arbre. Cela permet d'obtenir une vision arborescente de la grille comme celle présentée dans la Figure 2.1.

2.3.2 Topologie effective : cartographie dépendant du maître

La seconde phase de la collection des données dépend du choix de la machine maître. Les expériences de cette phase peuvent être vues comme des raffinements successifs de la topologie structurelle afin d'obtenir une vision nommée topologie effective et contenant des informations sur les couches inférieures du modèle OSI. La plupart de ces mesures se fondent sur des seuils expérimentaux arbitraires. Leurs valeurs ont un impact important sur les résultats

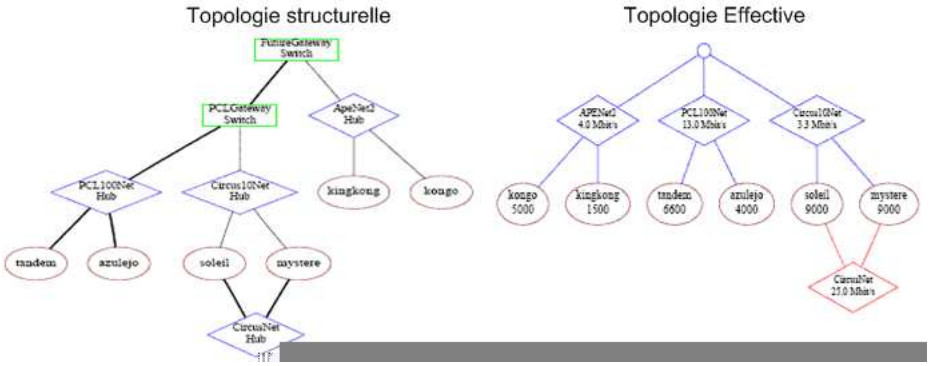
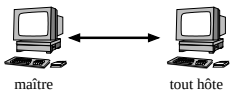


FIG. 2.1 – Exemples de topologie structurelle et topologie effective.

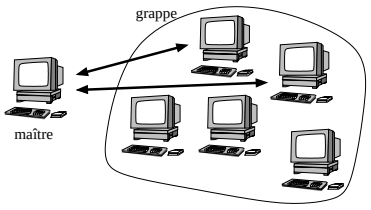
de la cartographie. Ils ont été déterminés expérimentalement par les auteurs d'ENV. Cela permet d'obtenir une vue effective de la grille en incluant les informations de bandes passantes obtenues par les mesures actives réalisées dans cette étape ainsi que la vitesse de processeur de chaque machine (voire Figure 2.1).

Bande passante de bout en bout



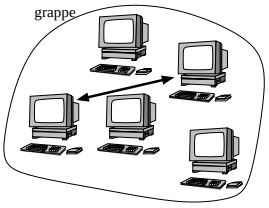
Cette expérience raffine le découpage constitué lors de la première phase de l'algorithme en clustérisant les machines présentant une connectivité comparable avec le maître. La bande passante entre le maître et chaque hôte est mesurée séparément. Si le ratio des bandes passantes entre deux hôtes donnés est supérieur à 3, ils sont placés dans deux sous-groupes séparés.

Bande passante par paire



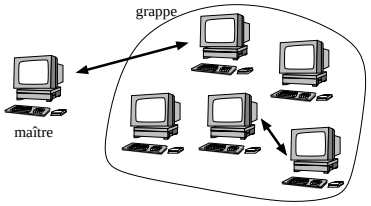
Cette expérience raffine encore le découpage en fonction de la façon dont le lien entre les membres du groupe et le maître est partagé. Pour cela, pour chaque paire de machines A et B dans chaque groupe, les bandes passantes de MA et MB sont mesurées en effectuant les transferts de manière concurrente. Cette mesure est ensuite comparée à celle obtenue lors de l'étape précédente. Si le ratio $\frac{\text{Bande Passante}(MA)}{\text{Bande Passante}_{//(MB)}(MA)}$ est inférieur à 1.25, les deux machines A et B sont déclarées indépendantes et leur groupe est subdivisé afin de les séparer.

Bande passante intra-groupe



Les caractéristiques des communications à l'intérieur de chaque groupe sont obtenues en mesurant la bande passante pour chaque couple de machines du groupe. Cela permet de détecter les groupes ayant une bande passante locale différente de celle obtenue lors de communications avec le maître.

Mesure du partage des liens



Pour chaque groupe, la bande passante avec le maître est mesurée lors d'un transfert entre deux autres membres. Cette expérience est répétée cinq fois et la moyenne du ratio Bande passante/Bande passante partagée est calculée. Si cette valeur est inférieure à 0.7, le groupe est considéré comme étant connecté par un lien partagé de type bus où la bande passante est partagée entre les différents transferts concurrents. Si elle est supérieure à 0.9, ENV considère que le lien interne au groupe est de type switch, ce qui signifie que les transferts concurrents n'ont aucune influence les uns sur les autres. Si le ratio est compris entre 0.7 et 0.9, une erreur est détectée et l'étude de ce groupe cesse car les mesures ne sont pas suffisamment significatives.

2.4 Algorithme ALNeM

INCA [LMQ06] (Interface-Centric Approach) a été développé par Arnaud Legrand, Frederic Mazoit et Martin Quinson de l'Ecole Normale Supérieure de Lyon. INCA ne vise pas la construction du schéma d'interconnexion physique du réseau mais plutôt à extraire une vision pratique pour les applications. Ce qui explique la nature qualitative plutôt que quantitative des informations à capturer lors de la phase de mesure.

En particulier, INCA ne considère que les machines sur lesquelles des constituants de la grille peuvent être exécutés sans privilège particulier. L'information principale qu' INCA doit capturer est donc l'éventuelle interférence entre les flux de données ayant lieu en même temps.

INCA présente cette notion d'interférence entre les flux (AB) et (CD) dans un graphe. Il permet de vérifier si le plus court chemin (en nombre de sauts) entre A et B a une intersection non nulle avec celui entre C et D.

Le graphe généré est non orienté. Il est composé de deux types de sommets : des *nœuds* représentant les machines constituant la grille dont l'ensemble est noté $\mathcal{H}$ et des *séparateurs* représentant les points de contention du réseau. Le routage utilisé est celui du plus court chemin.

2.4.1 Collecte des informations

Les auteurs de INCA proposent deux algorithmes de collecte des informations d'interférence : une façon intuitive et une autre optimisée.

Algorithme intuitif

Cet algorithme utilise la façon la plus simple pour collecter ces informations. Il est réalisé alors en $|\mathcal{H}|^4$ étapes. Chaque étape est associée à un quadruplet $(a, b, c, d) \in \mathcal{H}^4$ et consiste à :

1. Mesurer la bande passante de (ab) (notée $bw(ab)$)
2. Mesurer la bande passante de (ab) lorsque le lien (cd) est saturé (notée $bw_{/\!/cd}(ab)$)
3. Calculer le ratio

$$\frac{bw_{/\!/cd}(ab)}{bw(ab)}$$

Si le ratio est :
- **inférieur à 0.7 :** alors il y a une interférence entre (ab) et (cd). On la note $(ab) \curlywedge_{mes} (cd)$
- **supérieur à 0.9 :** alors il n'y a pas une interférence entre (ab) et (cd). On la note $(ab) /\!\!/_{mes} (cd)$
- **entre 0.7 et 0.9 :** alors il n'y a une erreur de mesure, nécessitant de réaliser l'expérience une nouvelle fois

Les étapes (1) et (2) doivent être suffisamment longues pour permettre au réseau de se stabiliser. Chaque étape doit attendre que l'expérience de saturation précédente se termine réellement pour éviter toute perturbation tandis qu'il est nécessaire d'attendre que le lien (cd) soit saturé avant l'étape (2).

Optimisations

Afin d'accélérer le processus, INCA mène les expériences en parallèle en tirant parti de l'existence de liens indépendants. Puisqu'ils n'interfèrent pas les uns sur les autres, il est possible de saturer plusieurs de ces liens en parallèle. Ainsi, lorsque INCA teste un nouveau lien. Ce test ne se fait pas par rapport à un seul lien d'un seul autre lien, mais vis-à-vis de tous ceux actuellement saturés.

2.4.2 Algorithme de reconstruction

L'idée principale de la méthode de reconstruction d'INCA est la suivante : pour deux machines a et b, si tout flux sortant de a interfère avec tout flux sortant de b, c'est que ces flux passent par un même lien réseau. a et b sont donc séparés des autres machines par lien. Ces nœuds sont donc placés dans un sous-arbre dans le graphe construit.

Dans [LMQ06], les auteurs démontrent un algorithme permettant de traiter les forêts d'arbres dont les racines sont interconnectées par une clique. Ils proposent également une extension permettant de traiter d'autres formes de cliques.

2.5 Algorithmes issus de la théorie des graphes

D'autres algorithmes issus de la théorie de graphe peut être considéré dans notre contexte de la découverte de la topologie.

2.5.1 Algorithme Clustering de latence

Cette approche est une méthode classique de clustering qui consiste à regrouper les machines proches en terme de latence dans les mêmes clusters. Parmi les travaux qui le met en pratique dans le cadre de la découverte de la topologie pour les grilles de calcul et les systèmes pair à pair, le travail réalisé par Luiz Angelo Barchet-Estefanel et Grégory Mounié dans le projet Apache au sein du Laboratoire ID-IMAG [BEM04].

C'est une plate-forme dédiée à la découverte automatique de la topologie du réseau. Elle est utilisée dans l'objectif d'optimiser la modélisation des communications collectives adaptées à l'environnement hétérogène des grilles de calcul.

Cette approche de découverte de topologie se déroule en deux étapes : la première étape est responsable de la collecte des informations de connectivité des différents réseaux ; la deuxième

étape sert à identifier les clusters des machines homogènes et à compléter la matrice par d'autres mesures.

Méthode de Mesure

Plusieurs travaux d'optimisation des communications collectives pour les environnements de grille considèrent que les clusters sont définis par leurs localités (IP), et que toutes les machines à l'intérieur d'un cluster sont homogènes. Mais, cette hypothèse de localité n'est pas conforme à la réalité de ces systèmes, qui peuvent contenir des machines hétérogènes avec des performances de calcul et de communication différentes. Par conséquent, le choix de la topologie du réseau doit se baser sur les aspects opérationnels qui reflètent la performance réelle des machines et de leurs liaisons.

Les informations de latence seront utilisées pour identifier les clusters de machines homogènes. Un avantage de cette méthode est qu'à cette étape on n'a pas besoin de toutes les interconnexions entre toutes les machines. En effet, l'objectif de cette première partie est de découvrir les hétérogénéités cachées à l'intérieur de chaque cluster. De plus, chacun de ces clusters peut utiliser son propre système de surveillance sans avoir besoin de contacter les autres. Ainsi, cette approche permet la réduction du trafic inutile sur le réseau.

Les données ainsi collectées des différentes grappes sont réunies dans une unique matrice de latence, ce qui sera fourni pour la deuxième étape de clustering.

Clustering

À partir de la matrice de latence obtenue dans la première partie, et en utilisant des algorithmes de Clustering, on peut regrouper les machines en des clusters logiques différents. Un des algorithmes qui peut être utilisé, est l'algorithme de Lowekamp [BEM04]. Chaque sous réseau garde la valeur du plus petit arc à l'intérieur du groupe et chaque nouvel arc est comparé avec cette valeur minimum.

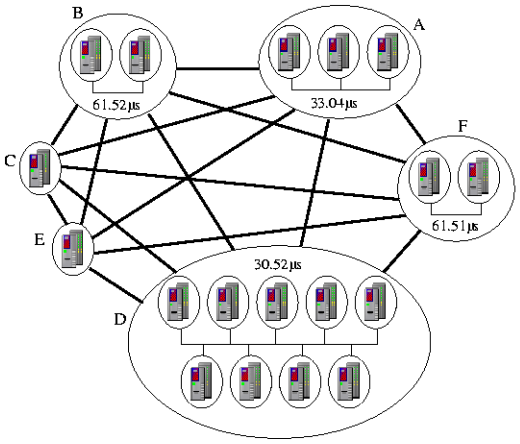
Dans le cadre de la découverte de la topologie et pour ne pas induire l'algorithme de clustering en erreur, le travail [BEM04] ajoute une procédure de détection des machines SMP. Il termine par l'acquisition des paramètres PLogP pour compléter les mesures entre les clusters qui ont été ignoré dans la première étape.

Exemple

L'exemple de la Figure 2.2 montre l'application de cette approche sur 20 machines. Les informations inscrite dans le tableau associé montrent les latences entre les clusters obtenus. Il est intéressant de mentionner l'apport de cette approche dans le cas d'un modèle à 2 niveaux ou multi-niveaux. Par exemple, il devient préférable dans ce cas de passer par D pour aller de C vers E que de se connecter directement.

2.5.2 Clustering Bande Passante

Par équivalence à l'algorithme de clustering de latence, nous pouvons fournir à l'étape 2 de ce dernier une matrice de bande passante, et on aura alors un algorithme qui regroupe les machines proches en terme de bande passante dans des clusters homogènes.



	A	B	C	D	E	F
A	0	61.53	105.25	37.99	103.45	69.96
B	61.53	0	224.98	61.52	139.04	137.52
C	105.25	224.98	0	61.49	207.98	129.45
D	37.99	61.52	61.49	0	66.49	61.51
E	103.45	139.04	207.98	66.49	0	123.97
F	69.96	137.52	129.45	61.51	123.97	0

latence entre les clusters (µs)

FIG. 2.2 – clustering du réseau et matrice de latence.

2.5.3 Arbre recouvrant minimal (ARM) vis à vis des latences

Un arbre recouvrant est tout arbre qui atteigne tous les nœuds d'un réseau dans l'intérêt de construire une topologie sans boucle. Cet arbre est dit minimal si la somme des poids de ses arêtes est le minimum de tous les autres arbres de recouvrement. Etant toujours dépendant d'une source, cet algorithme commence par définir la racine. Ensuite, l'arbre peut être construit en partant de la racine et en allant vers chaque nœud par le chemin le plus court en terme de latence (voir Figure 2.3).

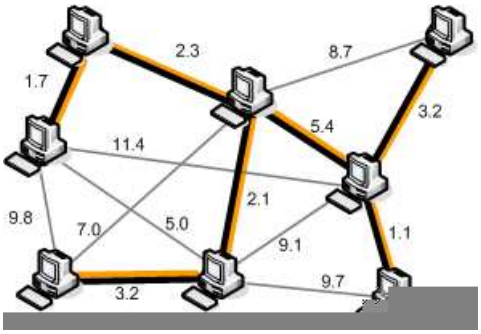


FIG. 2.3 – Arbre recouvrant minimal vis à vis des latences en ms.

Cette approche prend donc comme entrée un graphe du réseau réel pondéré par les latences entre les nœuds et la racine initiale et rend comme sortie l'arbre recouvrant minimal associé à cette racine.

2.5.4 Arbre recouvrant maximal (ARM) vis à vis des bandes passantes

Cet algorithme cherche à avoir un arbre recouvrant avec le même principe du précédant. Il cherche à déterminer une arbre qui maximise la bande passante. Il récupère comme paramètre d'entrée un graphe du réseau réel pondéré par les bandes passantes entre nœuds et

rend comme résultat l'arbre recouvrant maximal associé à cette racine.

2.6 Comparaison des algorithmes

Le Tableau 2.1 décrit les avantages et les inconvénients des algorithmes de découverte de la topologie. Les notations **(+)**, **(-)**, **(mes)** et **(cons)** désignent respectivement les avantages, les inconvénients, la phase mesure ou de collecte de donnée et la phase de reconstruction du graphe de la topologie.

Les algorithmes de découverte issus de la théorie de graphe tels que le clustering de latences (clustering Latence), le clustering de bande passante (clustering B.P.), l'arbre recouvrant minimal de latence (ARM latence), et l'arbre recouvrant maximal de bande passante (ARM B.P.) ont généralement la même partie de collecte de données. Cette partie consiste à fournir une matrice de latence ou de bande passante de dimension N (Nombre de machine de réseau). Les mesures se font en $O(N^2)$. Ceci représente un avantage par rapport à INCA qui fait ses mesures en un temps de l'ordre de $O(N^4)$ (méthode intuitive).

La phase d'acquisition des données des algorithmes INCA et ENV présente aussi un avantage considérable. En effet, cette phase ne nécessite aucun privilège d'exécution particulier.

INCA et ENV utilisent la saturation des liens de réseau pour tester les transferts parallèles. Ce qui perturbe énormément le réseau et peut rendre ce dernier inutilisable tout au long de la phase de mesures.

Une particularité de ENV qui peut être considérée à la fois comme avantage et inconvénient est que cet algorithme se limite à une vue client/serveur. Cette hypothèse fait accélérer le temps de réalisation des mesures. Néanmoins, cela rend les mesures utilisables que par le serveur choisi dès le début de cette phase et ne permette pas d'avoir une vue globale du réseau.

Le principal avantage d' INCA et ENV par rapport aux autres est que ces deux algorithmes cherchent les interférences possibles dans le cas des transferts parallèles. Une fois ces interférences détectées, elles offrent aux applications une bonne utilisation du réseau et un bon ordonnancement des tâches.

Un inconvénient majeur d'INCA et ENV est qu'ils supposent que les routes du réseau sont symétriques et ne différencient donc pas les routes (AB) et (BA). Cette supposition ne tient malheureusement pas sur Internet où il est relativement commun de trouver des routes asymétriques du fait de problèmes de configuration par exemple.

Enfin, INCA génère plusieurs topologies comme les arbres, les cliques. Il propose aussi une extension pour les cycles qui présentent encore quelques problèmes. Alors que ENV et les arbres de recouvrement se limitent aux topologies de type arbre. Nous allons présenter dans les chapitres suivants une solution pour comparer quantitativement ces topologies.

Conclusion

Dans ce chapitre, nous avons étudié les principaux outils de découverte de la topologie dans le cadre des réseaux et plus particulièrement les algorithmes offrant une topologie applicative pour les grilles de calcul. Ces algorithmes commencent généralement par la collecte

<table>
<tr><th></th><th>ENV</th><th>ALNEM</th><th>clustering Latence</th><th>ARM Latence</th><th>Clustering B.P.</th><th>ARM B.P.</th></tr>
<tr><td>+ mes</td><td>– Mesure sans privilège</td><td>– Mesure sans privilège</td><td colspan="4">– Génère peu de trafic inutile
– Mesures en $O(N^2)$</td></tr>
<tr><td>- mes</td><td>– Vue client/serveur : mesures exploitées que par le serveur
– Mesures actives : Génère énormement du tafic (saturation du réseau)</td><td>– Temps d'exécution des mesures avec la méthode intuitive en $O(N^4)$
– Mesures active : Génère énormement du tafic (saturation du réseau)</td><td colspan="4"></td></tr>
<tr><td>+ cons</td><td>– Cherche les interférences</td><td>– Cherche les interférences
– Génère les topologies arbre, clique et une extension pour les cycles</td><td></td><td></td><td></td><td></td></tr>
<tr><td>- cons</td><td>– Suppose la symétrie des routes
– Ne génère que la topologie de type arbre : vue client/serveur</td><td>– Suppose la symétrie des routes
– problèmes pour certaines topologies de type cycle</td><td colspan="4">———————
– Ne traite pas les interférences
– Ne génère que la topologie de type arbre</td></tr>
</table>

TAB. 2.1 – Comparaison des algorithmes de découverte de la topologie

des informations de réseau (latences, bande passante, etc). Ensuite, ils reconstituent une topologie découverte pondérée par des informations qualitative et quantitative (bande passante, interférence des transferts parallèles, etc). À la fin de ce chapitre, nous avons effectué des comparaisons qualitatives de ces algorithmes en montrant les avantages et les inconvénients de chacun d'entre eux.

Chapitre 3

Méthodologie de Comparaison

Dans ce chapitre, nous décrivons notre méthodologie de comparaison qui va nous permettre de comparer qualitativement les différents algorithmes présentés dans le chapitre précédant. Nous précisons aussi l'ensemble de critères choisi pour bien mener cette comparaison.

3.1 Critères de comparaison

Pour pouvoir comparer les algorithmes (ENV, INCA, etc) et juger l'efficacité d'une approche par rapport aux autres, nous avons été amenés à quantifier les résultats obtenus par chaque algorithme par rapport à plusieurs critères comme la dynamicité, la rapidité de l'exécution pour assurer le passage à l'échelle, ou la meilleure topologie applicative générée.

1. Tolérance à la dynamicité de la plate-forme : Les plates-formes à très large échelle sont souvent confrontées à des problèmes de disponibilité de leurs ressources puisque des nœuds peuvent quitter, s'ajouter ou tomber en panne. Cela rend le critère de tolérance à la dynamicité un critère important et doit être assuré par ces algorithmes pour offrir des solutions réelles et adaptées au caractéristiques des plates-formes traitées.
2. Rapidité de l'exécution : Une des caractéristiques que doivent assurer les algorithmes de cartographie des réseaux classiques est la rapidité de l'exécution. Ce critère est important dans notre contexte. En effet, les algorithmes de la découverte de la topologie de la grille doivent être très rapides pour pouvoir passager à l'échelle. Le meilleur algorithme par rapport à ce critère est celui qui assure les mêmes fonctions dans un temps plus court que les autres.
3. Meilleure topologie : Chaque algorithme génère à la fin de son exécution une topologie applicative de la plate-forme réelle. Le problème est de comparer ces topologies. La comparaison de ces topologies constitue un critère majeur dans la détermination du meilleur algorithme.

Nous nous concentrons dans notre étude sur les deux derniers critères, à savoir la rapidité de l'exécution et la recherche de la meilleure topologie. Nous envisageons de traiter la dynamicité dans une extension de notre travail.

La rapidité de l'exécution devrait être aisée une fois nous que aurons implémenté chaque algorithme de découverte. Mais le problème qui persiste est comment comparer et déterminer la meilleure topologie découverte ?

Les différents algorithmes décrivent leurs topologies découvertes avec des graphes très différents par rapport à la signification de leurs composants et par rapport aux hypothèses considérées dès le début de l'algorithme : par exemple, certains considèrent l'asymétrie des routes et d'autres l'ignorent. INCA considère les nœuds du graphe comme étant des machines susceptibles d'exécuter l'application alors que l'algorithme de latence représente les nœuds par les clusters homogènes, etc. Ceci rend la comparaison de ces graphes très difficile.

Supposons que nous arrivons à comparer ces graphes, nous serons confronté à d'autres problèmes puisque chaque algorithme présente ses propres paramètres d'action (latences seulement, latence et bande passante, interférence, etc) d'une façon différente et que la méthode de mesure de ces paramètres varie d'une approche à une autre. À titre d'exemple nous pouvons citer la différence entre les latences mesurées dans [BEM04]. D'une part, cette approche utilise NWS pour construire sa matrice de latence. D'autre part, elle collecte les paramètres PlogP pour augmenter la précision de son modèle et compléter cette matrice. Mais les deux latences ne signifient pas la même chose puisque la première considère le RTT (Temps d'aller retour) complet alors que l'autre élimine le temps de traitement du trafic de test dans le poste distant et elle ne considère ainsi que le temps d'aller retour d'une seule unité de mesure [BEM04], ce qui rend leur comparaison non significative.

3.2 Méthodologie de comparaison des algorithmes

Les divergences de ces graphes et de leurs composants nous ont poussé à définir une méthodologie de test ayant comme entrée une variété de topologies réelles et des *Testcases* et comme sortie le meilleur algorithme de découverte. L'idée est de comparer le comportement de plusieurs applications distribuées classiques, *Testcases*, sur les différents graphes reconstruits à celui obtenu sur la plate-forme d'origine.Ces *Testcases* sont caractérisées chacune par des facteurs limitants variés (bande passante, nombre de communication inter machines,etc). Ceci va nous permettre d'examiner une variété de caractéristiques pour chaque algorithme. Cette approche consiste à exécuter les étapes suivantes :

1. ***Génération des topologies découvertes :*** elle consiste à considérer une variété de topologie réelle comme une topologie en arbre, un graphe complet, un cluster complet, un cluster d'arbre, une topologie aléatoire, etc. Pour chacune de ces topologies réelles nous exécutons les algorithmes de découverte et nous générons les topologies découvertes (voir Figure 3.1).
2. ***Exécution des Testcases :*** pour chaque algorithme, nous exécutons les *Testcases* sur les différentes topologies découvertes. Nous notons alors le temps d'exécution du *Testcases* 1 sur la topologie découverte d'un réseau réel de type arbre en utilisant l'algorithme d'INCA par $T_{Tc1}(arbre/alnem)$.
3. ***Comparaison :*** avant de comparer le temps d'exécution de chaque algorithme, il faut tout d'abord déterminer le temps d'exécution des *Testcases* sur les topologies réelles. Nous notons par exemple la durée du *Testcases* 1 sur la topologie réelle de type arbre par $T_{Tc1}(arbre/rel)$.

   Après avoir déterminé ce temps nous calculons le rapport :

$$R_{Algo1} = \frac{T_{Tc1}(arbre/Algo1)}{T_{Tc1}(arbre/rel)}$$

Cette expression représente le rapport du temps d'exécution d'un *Testcases* sur une plate-forme découverte en utilisant l'algorithme1 sur le temps d'exécution de ce même TEST-CASE sur la topologie réelle.

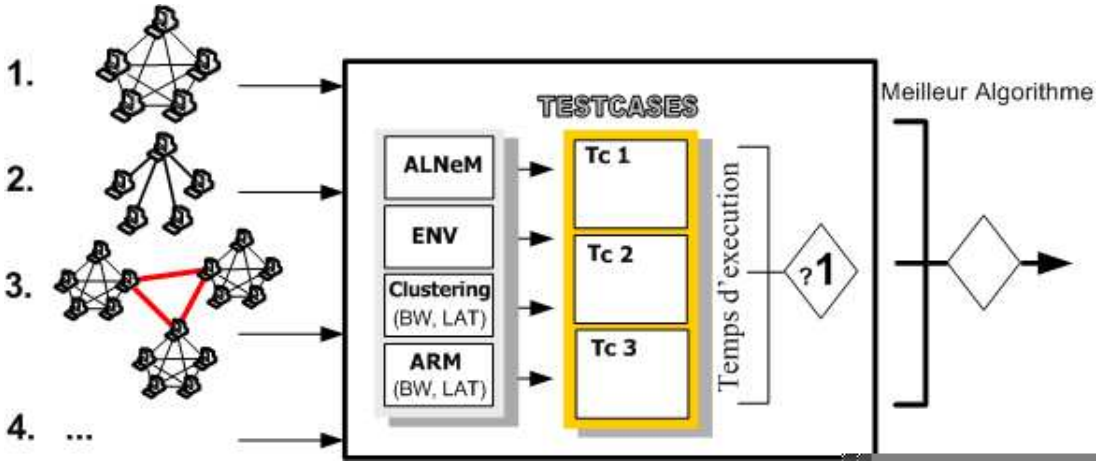


FIG. 3.1 – Méthodologie de compraison pour différentes topologies réelles.

L'algorithme qui réalise le rapport qui tend plus vers 1 par valeur positive est considéré comme le meilleur algorithme pour ce *Testcase* et par conséquent il résiste le mieux aux facteurs limitants de ce *Testcases* (voir Figure 3.2).

3.3 Les *Testcases*

3.3.1 Présentation

Les *Testcases* sont des applications distribuées classiques caractérisées chacune par des facteurs limitant variés à savoir la bande passante, le nombre de communication inter machines, etc. Ces *Testcases* vont être exécutés sur les plates-formes découvertes et puis nous comparons le temps d'exécution sur ces plates-formes par rapport au temps d'exécution sur la plate-forme réelle. Le choix de ces *Testcases* doit se faire d'une façon judicieuse pour couvrir une large bande de caractéristiques. Chaque TESTCASE doit avoir alors des facteurs limitant pour tester la résistance des algorithmes de découvertes vis à vis de ces facteurs (le nombre de messages envoyés, la bande passante, etc). Nous choisissons plusieurs *Testcases* qui sont le *token-ring*, la diffusion, le *all-to-all*, et le produit de matrices parallèles.

Token-ring

Le *Testcases token-ring* ou anneau à jeton est une application dans laquelle une machine doit disposer d'un jeton avant d'émettre ses paquets sur le réseau. Ce jeton logique circulant d'une machine à une autre, a pour but d'éviter que deux machines émettent simultanément sur le réseau et provoquent une collision. Pour permettre la bonne circulation du jeton, le réseau a schématiquement une forme circulaire, où le jeton circule de proche en proche.

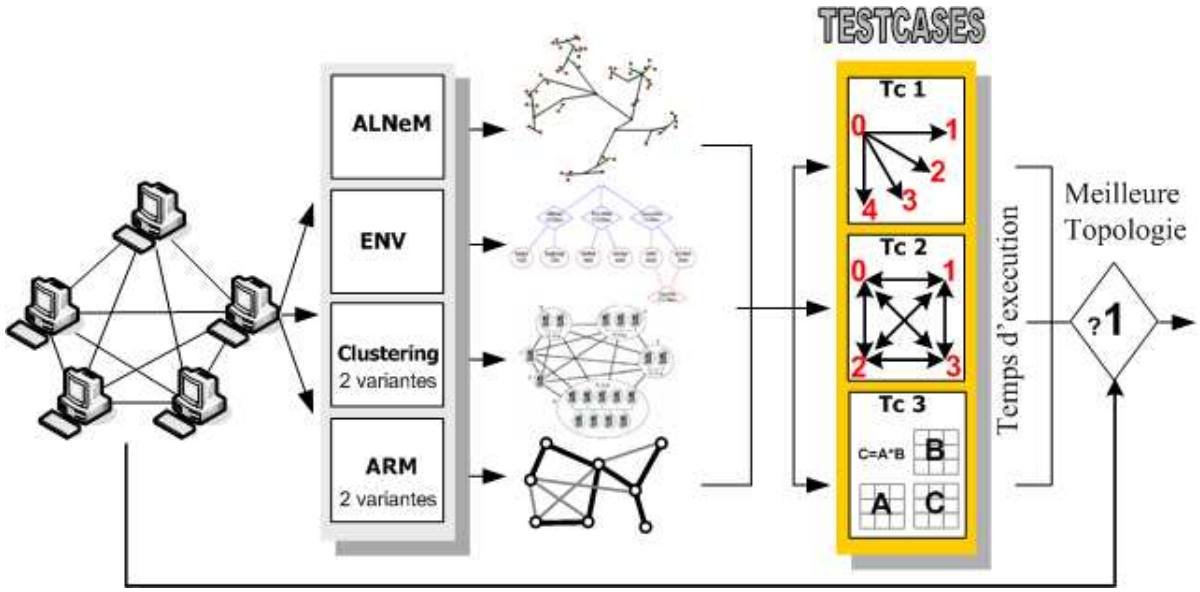


FIG. 3.2 – Méthodologie de comparaison pour une plate-forme de type clique.

Lors du fonctionnement du réseau, une machine souhaitant émettre va recevoir le jeton de la machine voisine précédante sur le réseau, émettre les données souhaitées. Puis, à la fin de sa transmission (ou quand le temps d'émission lui étant accordé est écoulé), elle va faire passer le jeton à la machine lui succédant sur l'anneau.

Diffusion

La *Testcases* Diffusion est une des applications la plus connue des communications collectives. Initialement, seul le processus racine détient le message qui doit être diffusé. À la fin de l'opération, une copie de ce message est déposée dans chaque processus du groupe. Cependant, l'environnement de type grille est normalement caractérisé par un grand nombre de processus communicants résultants de l'association des différentes grappes de calcul. Dans ce cas, la bande passante au niveau serveur représente un facteur limitant au niveau de l'application diffusion.

All to All

Le *all-to-all* est l'une des opérations les plus générales et l'une des plus complexes. En effet, tous les processus s'engagent dans un processus d'échange de données où les messages sont diffèrents selon le destinataire. Dans le cadre des applications distribuées à large échelle, cette opération présente une limite très importante qui est la congestion. En effet, le nombre de messages transmis sur le réseau est de l'ordre de N^2 où N est le nombre de machines dans le réseau. Ce nombre de messages est capable de saturer plusieurs branches sur le réseau surtout dans un environnement hétérogène comme les réseaux de grille.

Produit matriciel parallèle

Le produit des matrices parallèle est l'un des grands classiques des applications de grille de calcul. L'algorithme se déroule sur une grille de calcul avec un maestro et plusieurs calculateurs. Dans notre cas, nous allons utiliser le produit matriciel par double diffusion. Nous effectuons le produit de deux matrices carrées A et B. C est la matrice résultante. Nous notons par N la dimension de ces matrices. Notons aussi qu'il y a M calculateurs, avec M un entier tel que N est un multiple de M. Le processeur P(i,j) est responsable du calcul du bloc de matrice C_{ij}.

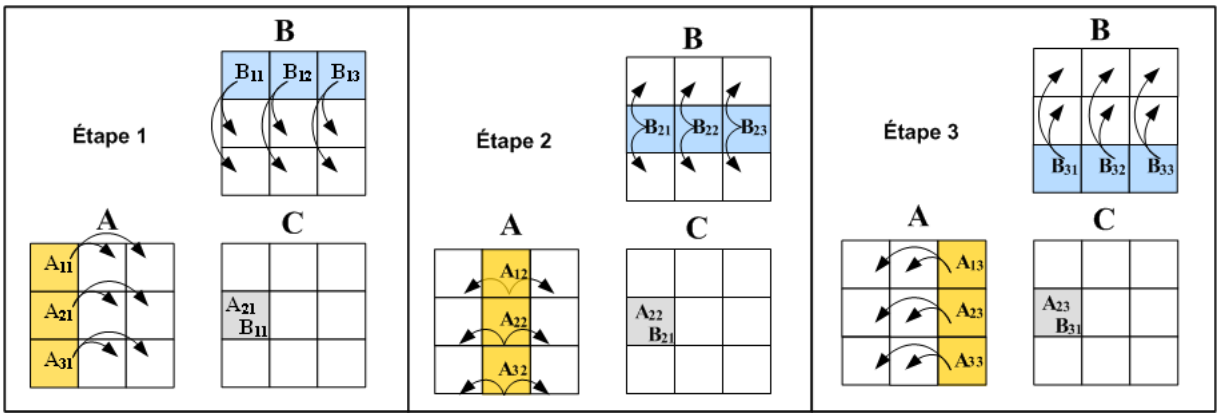


FIG. 3.3 – Produit des matrices parallèle par double diffusion : phase de multiplication.

L'algorithme comporte trois phases principales :

1. ***Scatter :*** l'algorithme commence par une étape d'initialisation dans laquelle le maestro envoit les deux valeurs de A_{ij} et B_{ij} au processeur . L'envoi se fait en séquentiel.
2. ***La multiplication :*** Chaque calculateur, possédant les valeurs de A et B correspondant au bloc de matrice C dont il est responsable, diffuse ces paramètres aux calculateurs de sa ligne et de sa colonne comme suit : À l'étape k, les calculateurs de la k ième colonne de C diffusent leurs valeurs A_{ij} horizontalement vers tous les calculateurs de la même ligne. Les calculateurs la k ième ligne de C diffusent leurs valeurs B_{ij} verticalement vers tous les calculateurs de la même colonne. Les calculateurs non concernés par l'envoi de leurs paramètres se mettent à l'écoute du réseau pour recevoir les nouvelles valeurs envoyées. À la fin de chaque étape, chaque calculateur possède un nouveau A_{ik} et B_{kj}. Il calcule alors leur produit et l'ajoute à C_{ij}. À la N ième étape, chaque P(i,j) possède la valeur finale de C_{ij}. Il envoi alors un message de terminaison au maestro (voir Figure 3.3).
3. ***Gather :*** c'est l'opération inverse du scatter. Une fois le maestro a reçu tous les messages de terminaison, il commence la collecte des C_{ij} et obtient ainsi la matrice produit.

Ce *Testcase* présente deux aspects importants. Le premier est le calcul qui n'existe pas dans les deux premiers *Testcases*. Le second est que les transferts se font en plusieurs étapes successives.

3.3.2 Comparaison des *Testcases*

La Figure 3.4 montre un récapitulatif et une comparaison des différents *Testcases* qui sont le *token-ring* (TR), la **diffusion**, le ***all-to-all*** pour 5 nœuds ($All2All_5$), le ***all-to-all*** pour 32 nœuds ($All2All_{32}$), le produit de matrice parallèle pour 500 nœuds (PMM_{500}) et le produit de matrice parallèle pour 5000 nœuds (PMM_{5000}).

	TR	$Diffusion$	$All2All_5$	$All2All_{32}$	PMM_{500}	PMM_{5000}
transfert en parallèle	Non	Non	Oui	Oui	Oui	Oui
contention	Non	Non	Non	Oui	Non	Oui
nombre d'étapes	1	1	1	1	N	N
topologie	Anneau	Arbre	Clique	Clique	Hypercube	Hypercube

FIG. 3.4 – Comparaison des *Testcases*.

Ces *Testcases* sont comparés suivant leurs caractéristiques d'execution à savoir le transfert parallèle, la contention, le nombre d'étapes et la topologie associée. Nous visons ainsi à observer l'impact de ces caractéristiques sur l'exécution sur les plates-formes découvertes.

Conclusion

Dans ce chapitre, nous avons présenté notre méthodologie de comparaison qualitative des algorithmes de découverte de la topologie. Nous avons défini les critères de comparaison et le choix des *Testcases* qui vont être exécutés sur les plates-formes découvertes pour déterminer le meilleur algorithme de découverte de la topologie.

Chapitre 4

Expérimentation

Nous avons définit dans le chapitre précédent notre méthodologie de comparaison des algorithmes de la découverte. Ce chapitre présente les résultats des expérimentations de cette méthodologie. Nous commençons par définir les outils utilisés pour l'implémentation des algorithmes. Ensuite, nous présentons et nous analysons les résultats de ces expérimentations.

4.1 Outils Utilisés

Notre objectif est de déterminer le meilleur algorithme de découverte de la topologie de la grille. Nous avons donc proposé une infrastructure logiciel permettant de mener à bien des experiences de façon reproductible. Nous avons utilisé pour cela l'environnement GRAS du simulateur SIMGRID.

4.1.1 Simgrid

SimGrid [CLM03, CM02, CAS] est un logiciel développé par Henri Casanova, Arnaud Legrand et Martin Quinson. Il s'agit d'un simulateur modulaire écrit en C permettant de simuler des applications distribuées dans un environnement hétérogène où les décisions d'ordonnancement peuvent être prises par différentes entités. Le point fort de ce simulateur réside dans sa capacité à importer et simuler des plates-formes réalistes de très large échelle. Il est basé sur une approche **guidée par la trace**, c'est-à-dire des enregistrements de différents paramètres d'une plate-forme pour obtenir un comportement réaliste. SIMGRID modélise les liens réseau ou les processeurs d'une plate-forme comme une collection de ressources. Chaque ressource est modélisée par une latence L (en secondes) et un débit D (la bande passante en octets/seconde ou la puissance en flop/seconde).

4.1.2 GRAS

SIMGRID est formé par plusieurs Environnements. L'environnement GRAS [gra06] (Grid Reality And Simulation) est conçu par les auteurs de SIMGRID et Rich Wolski de l'Université de California, Santa Barbara. La modélisation de l'application est laissée à la charge de l'utilisateur. Les implémentations en SIMGRID nécessitent une reécriture complète pour une utilisation grandeur nature. L'environnement GRAS résout ce problème en émulant une plate-forme à l'aide de SIMGRID. GRAS permet de développer une application de type "infrastructure de

service distribuée", et donc destinée à s'exécuter sur une plate-forme de calcul distribuée hétérogène, dans l'environnement bien plus contrôlé et plus rapide qu'offre le simulateur. L'infrastructure peut alors être déployée en taille réelle sans avoir à apporter la moindre modification.

4.2 Implémentation

GRAS permet, grâce à deux implémentations spécifiques de la même interface, l'exécution du même code à la fois dans le simulateur SIMGRID et sur des plates-formes réelles. Ce qui permet de bénéficier des avantages de la simulation (modélisation simplifiée, preuve rapide de certaines hypothèses, etc) et de l'expérimentation (validation des résultats, etc).

4.2.1 Implémentation des algorithmes

La premiere étape de l'implémentation concerne les algorithmes de découverte de la topologie à savoir l'ARM , le clustering, ENV et INCA. Ces algorithmes sont composés généralement de deux parties : la première pour l'acquisition des mesures et la deuxième pour la construction du graphe de la topologie découverte. Le développement de ces deux parties a fait l'objet d'une collaboration avec l'équipe MESCAL du Laboratoire Informatique et Distribution de l'ENSIMAG. Notre tâche consiste à développer les fonctions nécessaires pour la partie d'acquisition de données à savoir la génération d'une matrice de latence ou de bande passante à partir d'une plate-forme donnée et la détermination des mesures d'interférence dans le cas des algorithmes de ENV et INCA. La partie de reconstruction était à la charge de l'autre équipe.

Cette première partie nous fournit comme sortie les descriptions des plates-formes découvertes de chaque algorithme dans un format pris en charge par SIMGRID.

4.2.2 Implémentation des *Testcases*

La deuxième partie de l'implémentation consiste à développer les *Testcases* comme le *token-ring*, le *all-to-all*, etc. Ces algorithmes prennent comme entrée les descriptions des plates-formes découvertes lors de la premiere partie de l'implémentation et un fichier de déploiement généré automatiquement. Ils rendent comme résultat le temps d'execution sur chaque plate-forme. Ces résultats sont comparés par rapport au temps d'execution de ce même *Testcases* sur la plate-forme réelle.

4.3 Résultats et Analyse

Avant de commencer la présentation de résultats, il convient de rappeler que le choix des topologies réelles est fait de façon à avoir une très grande variété de topologie comme le clique, l'arbre binaire, etc. Les résultats finaux sont les rapports de temps d'exécution des *Testcases* sur les plates-formes découvertes divisé par le temps d'exécution de ce même *Testcase* sur la plate-forme réelle. L'algorithme qui réalise le rapport le plus proche de 1 par valeur positive est considéré comme le meilleur pour cette expérience. La moyenne de ces rapports d'un algorithme donné détermine la meilleure topologie découverte ayant ainsi un comportement très proche de la réalité.

Faute de temps, nous n'avons pas pu faire les expériences associées aux algorithmes d'INCA et de ENV. Les résultats présentés dans ces premières expériences compare les algorithmes d'arbre recouvrant minimale vis-à-vis des latences, d'arbre recouvrant maximale vis-à-vis des bandes passantes, de clustering de latence et le clustering de bande passante. Nous espérons terminer ces expériences dans les jours qui viennent.

La première expérience est réalisée sur une topologie générée automatiquement, créé avec un générateur de topologie tiers. Nous avons exécuté les algorithmes sur la topologie pour déterminer les plates-formes découvertes. Le premier *Testcase* exécuté sur cette plate-forme est le *token-ring*.

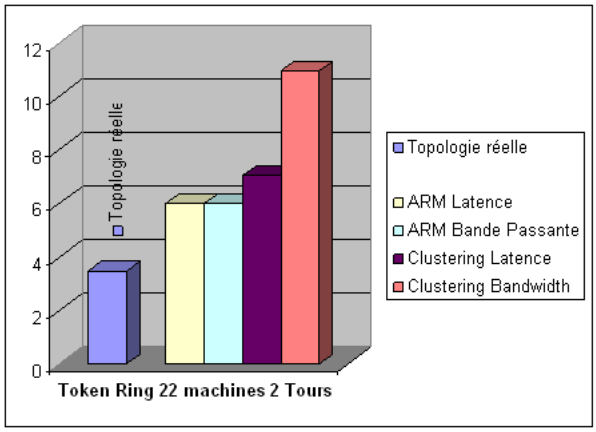


FIG. 4.1 – Exécution de *token-ring* sur les plates-formes découvertes.

La Figure 4.1 montre que les deux algorithmes d'arbre recouvrant offrent les meilleurs résultats avec un rapport de 1.72 alors que les deux autres algorithmes de clustering réalisent des rapports largement supérieurs à 2.

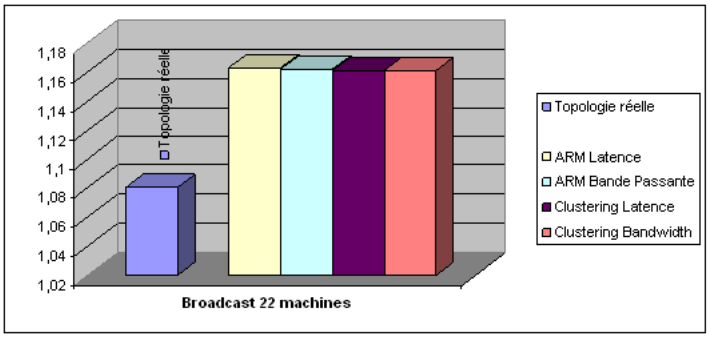


FIG. 4.2 – Exécution de la diffusion sur les plates-formes découvertes.

Pour le *Testcase* de diffusion, la Figure 4.2 montre que les quatre algorithmes réalisent un temps d'exécution très proche de temps réalisé sur la plate-forme réelle. Les rapports sont de l'ordre de 1.07. Les résultats trouvés sont donc très satisfaisants, et montrent que ses algorithmes sont très adaptés à l'algorithme de diffusion.

Le *Testcase* suivant est le *all-to-all*, nous nous limitons dans une première expérience à l'exécution de ce *Testcase* sur 5 machines de la plate-forme. Les résultats trouvés montrent que seul

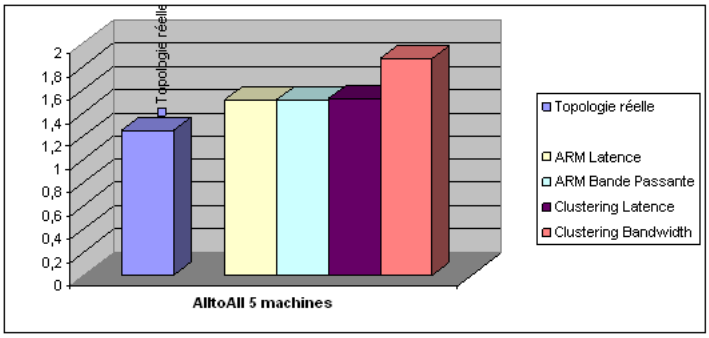


FIG. 4.3 – Exécution de *all-to-all* sur les plates-formes découvertes (cas de 5 machines).

le clustering bande passante réalise un temps d'exécution supérieur aux autres (voir la Figure 4.3).

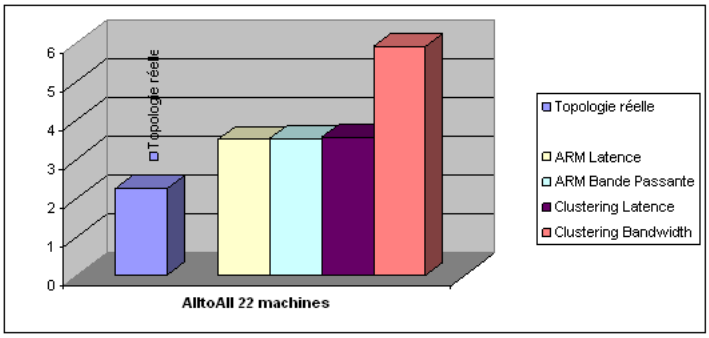


FIG. 4.4 – Exécution de *all-to-all* sur les plates-formes découvertes (cas de 22 machines).

La Figure 4.4 montre que l'exécution de ce même *Testcase* sur toute la plate-forme ne fait qu'accentuer l'écart entre le temps d'exécution l'algorithme de clustering de bande passante et celui de la plate-forme réelle avec un rapport que arrive à 2.62.

Le *Testcase all-to-all* présente deux facteurs limitants très importants qui sont le très grand nombre de messages circulants sur le réseau et le transfert parallèle. Les deux dernières expériences ont dévoilé une première observation qui montre que l'algorithme de clustering bande passante ne résiste pas trop à ces deux facteurs.

Nous répétons les expériences pour une topologie de type clique. La Figure 4.5 montre l'exécution des *Testcases* diffusion et *token-ring*. Ces expériences montrent que les algorithmes d'arbre recouvrant minimale vis-à-vis des latentes et le clustering de latences s'adaptent très bien à ce genre de topologie en clique et donnent des rapports de l'ordre de 1.1. Les rapports des deux autres algorithmes restent valables pour le *Testcase* diffusion mais pour l'exécution de *token-ring* les rapports s'écartent de plus en plus de 1.

La Figure 4.6 montre les courbes des algorithmes en fonction des *Testcases*. Les valeurs mentionnées pour la le *Testcase* de *token-ring* sont très proches de temps d'exécution sur la plate-forme réel et s'éloignent de plus en plus avec les autres *Testcases*.

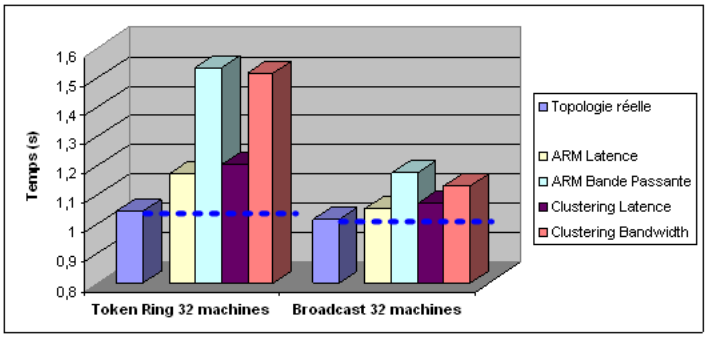


FIG. 4.5 – Exécution de *token-ring* et la diffusion sur une topologie Clique.

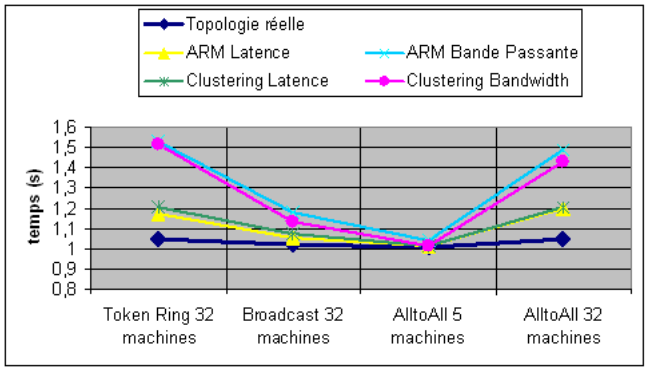


FIG. 4.6 – Courbes des *Testcases* pour une topologie clique.

4.4 Récapitulatif

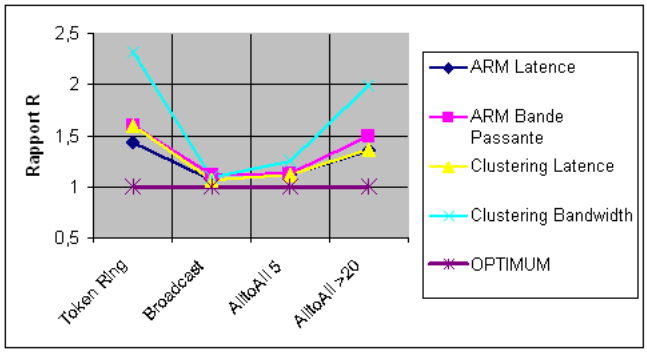


FIG. 4.7 – Courbes des rapports moyens de temps d'exécution.

Pour déterminer le meilleur algorithme de découverte de la topologie, nous calculons les moyennes des rapports des différentes expériences réalisées. La Figure 4.7 montre les courbes d'évolution de ces rapports en fonction des *Testcases* par rapport à un optimum de valeur 1. Les courbes montrent bien l'avantage de l'algorithme d'arbre recouvrant minimale vis-à-vis des latences par rapport aux autres. Le comportement des *Testcases* sur les plates-formes découvertes par cet algorithme semble le plus proche du comportement des ces *Testcases* sur les plates-formes réelles. Toutefois, ces résultats restent non définitifs dans l'attente des résultats des algorithmes ENV et INCA et l'ajout du *Testcase* du produit des matrices parallèles. Ceci permet de déterminer les comportements de ces algorithmes vis-à-vis du facteur de calcul non traité par les autres *Testcases*.

Conclusion

Nous avons présenté dans ce chapitre les résultats d'expérimentation des algorithmes de découvertes de la topologie des grilles de calcul. Nous avons déduit à l'aide de notre méthodologie de comparaison le meilleur algorithme parmi ceux qui ont été déjà développés. les résultats de ces expériences montrent que l'algorithme d'arbre recouvrant vis-à-vis des latences est celui qui donne les meilleurs résultats. Nous espérons terminer les expériences des algorithmes de ENV et INCA dans les jours qui suivent et l'intégrer dans le processus de comparaison.

Chapitre 5

Conclusion et Perspectives

Les travaux présentés dans ce stage portent sur la découverte de la topologie applicative de la grille de calcul. Nous avons détaillé les différents algorithmes existants de découverte de la topologie à savoir InCA, ENV, arbre de recouvrement et clustering. Ces algorithmes sont généralement composés de deux étapes. La première partie consiste à la collecte des mesures quant à la deuxième, elle reconstruit un graphe représentant la topologie découverte.

Nous avons également présenté une comparaison quantitative de ces algorithmes par rapport à l'ordre des mesures effectuées, les interférences et les types des topologies prises en charge. La comparaison de l'efficacité de la deuxième partie des algorithmes revient à comparer les graphes des plates-formes générés. Toutefois cette comparaison reste une tâche difficile vu la diversité de ces graphes et la différence de significations de leurs composantes. Ainsi, nous étions amené à proposer une méthodologie permettant la comparaison qualitative de ces algorithmes. L'idée est de comparer le comportement de plusieurs applications distribuées classiques, *Testcases*, sur les différents graphes reconstruits à celui obtenu sur la plate-forme d'origine. Nous avons fini par la mise en pratique de notre méthodologie pour plusieurs topologies et pour les algorithmes d'arbre recouvrant et de clustering. Faute de temps, nous n'avons pas pu faire les expériences associées aux algorithmes d'InCA et de ENV. Nous espérons terminer ces expériences dans les jours qui viennent. Les résultats des premières expériences ont montré que l'algorithme d'arbre recouvrant minimale vis à vis des latences est celui qui donne la meilleure plate-forme découverte. Le comportement des *Testcases* sur les plateformes produit par cet algorithme est plus proche du comportement de ces *Testcases* sur les plates-formes d'origines.

Le travail présenté dans ce mémoire pourra être étendu pour ajouter le critère de tolérance à la dynamicité de la plate-forme dans notre méthodologie de comparaison. En effet, la grille de calcul est caractérisée par l'indisponibilité de ses ressources. Il devient obligatoire de savoir l'algorithme de découverte qui s'adapte le plus à ce critère. Il est aussi intéressant d'ajouter des nouveaux *Testcases* plus complexes pour avoir une bibliothèque complète de test.

D'un point de vue expérimental, il est nécessaire d'exécuter ses algorithmes sur des plates-formes grandeur nature. Il est cependant possible d'exécuter nos expériences sur la plate-forme Grid5000 [PG05] accessible à partir de notre Laboratoire.

Au niveau des algorithmes de cartographie applicative, et comme toute autre étude comparative, cette étude nous a permis de dégager les avantages et les inconvénients de chaque algorithme. Nous pouvons alors améliorer un de ces algorithmes en s'inspirant des avantages

des autres. Nous pouvons aussi proposer un nouvel algorithme. Une amélioration est déjà prévue. Il s'agit de l'utilisation de la méthode de mesure tomographique cité dans [RNC04] pour la phase de collecte des données et l'algorithme de reconstruction de INCA pour la phase de génération de la topologie découverte.

Bibliographie

[BEM04] Luiz Angelo Barchet-Estefanel and Gregory Mounie. Identifying logical homogeneous clusters for efficient wide-area communications. *Springer-Verlag,2004*, pages 319–326, janvier 2004.

[CAS] Henri CASANOVA. SimGrid : A Toolkit for the Simulation of Application Scheduling. In *1st International Symposium on Cluster Computing and the Grid*, pages 430–441.

[CLM03] H. Casanova, A. Legand, and L. Marchal. Scheduling Distributed Applications : the SimGrid Simulation Framework. In *Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, may 2003.

[CM02] Henri Casanova and Loris Marchal. A Network Model for Simulation of Grid Application. In *Rapport de Recherche INRIA RR-2002-40*, Octobre 2002.

[ENV04] *Automatic Deployment of the Network Weather Service Using the Effective Network View.*, 2004.

[FK99] Ian Foster and Karl Kesselman. *The grid : Blueprint for a new computing infrastructure*. Morgan kaufmann Publishers, 1999.

[gra06] *GRAS : a research and development framework for grid services.*, janvier 2006.

[JLM89] Van Jacobson, Craig Leres, and S. McCanne. The Tcpdump Manual Page. Lawrence Berkeley Laboratory. *http ://ee.lbl.gov/*, juin 1989.

[KMK+01] Ken Keys, David Moore, Ryan Koga, Edouard Lagache, Michael Tesch, and k claffy. The architecture of the coralreef internet traffic monitoring software suite. *PAM'2001*, 2001.

[LMQ06] Arnaud Legrand, Frederic Mazoit, and Martin Quinson. An application-level network mapper. *Rapport de Recherche INRIA RR-5792*, janvier 2006.

[MCC04] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia distributed monitoring system : Design, implementation and experience. *Parallel Computing*, 30(5-6) :817–840, Août 2004.

[PG05] 2005. Projet GRID5000. http ://www.grid5000.fr. 2005.

[RNC04] Michael Rabbat, Robert Nowak, and Mark Coates. Multiple source, multiple destination network tomography. 2004.

[SBW99] Gary Shao, Francine Berman, and Richard Wolski. Using effective network views to promote distributed application performance. 1999.

[Sob03] Marcel Soberman. *Les grilles informatiques*. Hermes science publications, 2003.

[WSH99] R. Wolski, N. T. Spring, and J. Hayes. The Network Weather Service : A distributed resource performance forecasting service for metacomputing. *Future Generation Computing Systems, Metacomputing Issue*, 15(5–6) :757–768, Oct 1999.