Rapport sur le manuscrit de Thèse présenté par

Sébastien Miquée

pour l'obtention du Doctorat de l'Université de Franche-Comté

Rapporteur : Stéphane GENAUD,
Maître de conférences habilité à diriger les recherches,
Université de Strasbourg.

Titre du document évalué :

Exécution d'applications parallèles en environnements hétérogènes et volatils : déploiement et virtualisation

Le document présenté par Sébastien Miquée décrit les recherches qu'il a menées dans le cadre de sa thèse sous la direction de Raphaël Couturier, Professeur à l'Université de Franche-Comté, et de David Laiymani, Maître de Conférences à l'Université de Franche-Comté, au sein de l'équipe AND à l'IUT de Belfort-Montbélliard.

Contexte de l'étude

Le cadre général du travail est celui de la conception d'intergiciels capables de prendre en charge des programmes parallèles et de les exécuter sur des ressources de calcul dont le nombre, la nature, et la disponibilité sont incertaines et variables. Ce sujet a pris une grande importance durant la dernière décennie. L'objectif général est de fédérer des ordinateurs individuels dont la puissance de calcul est largement sous-exploitée et mobiliser leurs capacités réunies sur une application de calcul parallèle. Les difficultés communes à toutes les approches de ce type sont connues : elles sont liées à l'hétérogénéité, à la volatilité et à la sécurité.

On peut néanmoins distinguer les efforts de recherche sur ce sujet selon le type d'application qui peut être pris en charge par l'intergiciel. Les problèmes qui peuvent être découpés en un grand nombre de tâches indépendantes de calcul se prêtent particulièrement bien à cette approche, et des standards se sont dégagés comme le projet BOINC. En revanche, quand il s'agit de déployer un programme parallèle dont les tâches ont besoin de communiquer, les difficultés sont bien plus grandes. Il est en particulier beaucoup plus difficile de proposer des mécanismes de tolérance aux pannes pour remédier à la volatilité des ressources. D'autre part, les synchronisations entre les tâches entrainent généralement de très grandes amplitudes dans les performances de l'application, en fonction de la façon dont les tâches sont placées sur les ressources dont la puissance et l'interconnexion sont hétérogènes.

Le travail proposé dans cette thèse concerne la prise en charge de cette dernière catégorie de programmes. Le travail prolonge l'expérience acquise par l'équipe avec la plateforme Jace depuis 2003, et revisite la problématique de la tolérance aux pannes en utilisant une nouvelle technologie arrivée à maturité : la virtualisation.

Analyse du document présenté

Le document est organisé en trois parties comptant 7 chapitres au total.

La première partie est introductive. Son premier chapitre *Contexte Scientifique* décrit les applications visées. Partant du champ des méthodes numériques, le contexte se focalise sur les méthodes numériques itératives parallélisées, et plus particulièrement sur celles dont on peut tirer une version asynchrone (Itérations et asynchrones et communications asynchrones). Ce type particulier d'algorithme se prête bien aux environnements hétérogènes et volatils. Suit dans le même chapitre une introduction à la tolérance aux pannes donnant les grands principes qui peuvent être mis en œuvre.

Le deuxième chapitre d'introduction concerne les plateformes et les environnements d'exécution. Il y est fait un panorama rapide des catégories de matériels aujourd'hui envisageables pour du calcul parallèle. Le chapitre introduit ensuite plusieurs environnements logiciels permettant le parallélisme à passage de messages, avec un accent sur les évolutions de l'environnement *Jace* développé par l'équipe. On peut regretter que la description faite de l'évolution historique et des objectifs des utilisateurs de chaque classe de matériel soit plutôt naïve, alors qu'on a maintenant un recul suffisant pour comprendre à quels types d'usage peuvent être réservés ces différentes architectures. La partie concernant les environnements d'exécution aurait aussi pu faire référence (même de manière brève) à de plus nombreux travaux relatifs internationaux, afin de situer la catégorie particulière dans laquelle se situe la proposition de la thèse. Hormis MPI, incontournable, seuls les environnements français XtremWeb (et XtremWeb-CH) et ProActive sur lesquels M. Miquée a travaillé sont mentionnés. En revanche, les environnements décrits semblent parfaitement compris et maîtrisés, M. Miquée les ayant utilisés pour expérimenter la faisabilité de certaines idées préalablement à sa thèse. Dans ce travail de thèse, il a également modifié l'environnement Jace de façon à supprimer les éléments centralisés fragiles en cas de panne.

Placement des tâches La partie 2 *Placement des tâches* aborde le cœur du travail. Le premier chapitre de cette partie introduit le contexte et la problématique à travers un exemple introductif qui montre l'intérêt de prendre des bonnes décisions de placement sur une architecture hétérogène. Les définitions introduites à la suite sont importantes car elles indiquent quels choix sont faits pour modéliser les applications et l'architecture. Les critères définis comme le degré d'hétérogénéité semblent pertinents et utilisables dans la pratique. Le choix de modéliser les communications internes des applications visées par des TIG est conforme à ce qui se fait dans la littérature.

Le chapitre 4 décrit la première contribution de la thèse. L'objectif est de proposer un algorithme de placement efficace des tâches pour le contexte choisi. A partir de deux types d'algorithmes proposés dans la littérature, M. Miquée propose un algorithme, baptisé *Maheve*, qui combine les deux algorithmes. Le premier algorithme est basé sur l'algorithme de Farhat, dont l'objectif est de décomposer un domaine en sous-domaines en minimisant les communications entre sous-domaines. Le deuxième algorithme a pour objectif principal d'utiliser les ressources les plus puissantes, sans considération des communications. L'algorithme est modifié par M. Miquée pour tenir compte des communications comme un objectif secondaire. L'expérimentation sur un cas précis introduit l'idée que, l'agorithme tenant compte principalement des communications entre sous-domaines est le plus performant quand le degré d'hétérogénéité est faible. Inversement, celui qui privilégie la puissance totale est plus performant dans le cas contraire. Ce résultat n'est pas surprenant puisque seules les communications importeraient dans un système totalement homogène. L'algorithme Maheve proposé, adopte le comportement de l'un ou l'autre des algorithmes en fonction de l'hétérogénéité mesurée. Il aurait été intéressant d'aller plus loin dans la caractérisation de l'hétérogénéité : comme le suggère l'environnement d'expérimentation choisi, Grid5000, composé de clusters, l'hétérogénéité n'existe qu'entre les différents clusters et une valeur réelle globale unique pour la plateforme capture mal cet aspect. On note également que le contexte d'exécution visé s'est restreint à partir de ce chapitre aux fédérations de clusters, ce que ne laissait pas supposer l'introduction (bas de la page 20) —pour généraliser l'emploi de Maheve à des machines hors cluster, on pourrait argumenter qu'une classification des

machines permettrait de retrouver des clusters.

L'impression globale qu'on peut retirer du chapitre est mitigée. Les idées intuitives qui se cachent derrière les algorithmes sont bien décrites dans le document, mais malheureusement, l'ensemble ne permet pas d'être totalement convaincu. En cause, plusieurs imprécisions dans l'écriture des algorithmes gênent la compréhension. Ainsi, le classement des clusters (algorithme 5) prend en entrée le dégré d'hétérogénéité de la plateforme alors que ce sont les degrés d'hétérogénité des clusters individuels qui permettent de les ordonner. Le deuxième reproche concerne la justification des réglages des paramètres qui gouvernent l'algorithme. Les notes, attribuées aux clusters et aux tâches constituent l'essentiel de l'heuristique. On comprend qu'à puissance cumulée égale on privilégie un cluster homogène à un cluster hétérogène. En revanche, le calcul de la note d'une tâche mêle des métriques de différente nature, pondérées par le degré d'hétérogénéité. Les explications données ne permettent pas d'appréhender la complexité du modèle proposé, ni d'être convaincu de l'efficacité de l'heuristique. Le facteur correctif ajouté au chapitre suivant pour prendre en compte le comportement de l'application ajoute à cette complexité. Le comportement des algorithmes itératifs asynchrones est difficilement modélisable, et c'est pourquoi sans doute, le document ne présente pas de comparaison avec des cas simples dont on serait capable de calculer l'optimal. Cependant rien n'est dit sur la méthodologie envisagée pour apprécier la performance. D'autres références bibliographiques étaient également pertinentes et auraient pu fournir d'autres éléments de comparaison.

Un autre objectif important dans la modification de ces algorithmes, ou de la proposition Maheve, est la prise en compte de la tolérance aux pannes. Les algorithmes proposés incorporent la réservervation de noeuds supplémentaires pour y placer des réplicas. L'influence des choix faits sont bien montrés dans la section expérimentale.

Le dernier chapitre de cette partie rapporte les résultats d'expérimentations. La plateforme est Grid5000, et deux applications sont testées. Il faut noter que les configurations de plateforme testées, jusqu'à 5 clusters dans 4 sites sont complexes à mettre en œuvre d'un point de vue expérimental. Déployer des environnements réels sur de telles architectures est remarquable et les observations recueillies sont importantes. On pourrait légitimement souhaiter un panel plus large d'applications mais la difficulté de mener à bien les expériences permet de comprendre le choix, d'autant que ces applications choisies ne sont pas triviales. Les résultats montrent globalement les gains obtenus en performance avec les trois algorithmes par rapport à un placement aléatoire des tâches. Ces gains sont manifestes sur l'application de gradient conjugué, mais beaucoup plus modestes avec l'advection-diffusion. Dans cette dernière application, les tâches communiquent davantage et peuvent avoir des charges de calcul variables au cours du temps. Les résultats en présence de pannes accentuent cette tendance, et montrent davantage l'intérêt du placement.

Virtualisation La troisième partie décrit l'autre partie importante du travail de thèse. Elle a pour objectif d'introduire les techniques liées à la virtualisation dans une plateforme nommée *HpcVm*, qui n'est plus spécialisée dans les applications itératives asynchrones. L'objectif du projet est de mettre à disposition une plateforme de calcul parallèle simple et peu onéreuse pour les utilisateurs non spécialistes du HPC. Un chapitre introductif à la partie présente les différentes approches de la virtualisation, ainsi que deux projets de recherche utilisant la virtualisation pour construire des plateformes de calcul. Cette introduction est pédagogique et amène correctement les choix faits par la suite.

Le chapitre suivant décrit la proposition *HpcVm*. L'idée principale est d'utiliser les mécanismes génériques applicables aux machines virtuelles pour implémenter la tolérance aux pannes dans un environnement à passage de messages. Ces machines virtuelles peuvent être stoppées, sauvegardées, migrées, redémarrées. Il faut noter que la plateforme *HpcVm* est un développement complètement nouveau. A

ce titre, il ne bénéficie pas des efforts faits sur JaceP2P-V2 (et décrits au début du document), pour décentraliser complètement l'infrastructure. On peut imaginer la difficulté à repartir de JaceP2P-V2 pour peu que le code gérant l'infrastucture intergicielle ne soit pas bien isolé, mais il eût été intéressant d'en indiquer brièvement les raisons. Le développement d'une telle plateforme est en soi un très gros effort.

D'un point de vue fondamental, le mécanisme de tolérance aux pannes proposé dans *HpcVm* est un *checkpoint* coordonné. Ceci signifie qu'un processus central est chargé de synchroniser les autres processus. Il faut en effet attendre que que tous les messages "en vol" soient arrivés pour assurer un état gloval cohérent de reprise. Cette approche est la plus coûteuse, et a été améliorée largement par l'introduction de *message logging*, qui consiste à mémoriser entre deux checkpoints les messages envoyés et l'ordre d'envoi/réception pour les rejouer en cas de reprise après panne. Il est dommage que le document ne discute pas de ces options, même si on comprend aisément les difficultés techniques et le temps nécessaire à une telle réalisation technique. Les expérimentations qui sont présentées donnent une bonne idée des surcoûts liés à la virtualisation et à la tolérance au panne. On peut discuter de ce coût et je n'ai pas la même analyse que M. Miquée, qui conclut que ce mécanisme de tolérance est efficace. La différence entre une exécution sur machine virtuelle et une exécution tolérante aux pannes subissant une panne est d'environ 20 minutes dans les deux applications, ce temps incluant un checkpoint nécessaire et une reprise. Cette durée semble essentiellement liée à la taille de l'image et à la performance du réseau. Il faut donc des durées d'exécution assez longues pour amortir le surcoût de chaque panne ou sauvegarde. Ceci apparaît compatible avec l'objectif fixé pour la plateforme, qui vise des applications d'une durée de plusieurs jours.

Conclusion

Ce document de thèse rapporte un travail conséquent, sur deux environnements distincts, mais qui ont le même objectif global de prise en charge de programme parallèles à passage de message. La première contribution concernant un algorithme de placement des tâches d'un programme itératif asynchrone a été intégrée dans l'environnement Jace-P2PV2 développé dans l'équipe. M. Miquée a eu le souci de rendre ce travail modulaire en le proposant comme une bilbiothèque. La deuxième contribution est le développement d'un nouvel environnement, HpcVm, qui s'appuie sur des techniques de virtualisation pour assurer la tolérance aux pannes.

Il faut souligner le travail très important de développement mené dans cette thèse, qui a manifestement donné à M. Miquée une grande maîtrise des technologies utilisées. Le travail démarré avec HpcVm constitue très probablement une très bonne base pour poursuivre des travaux expérimentaux d'utilisation de la virtualisation.

En revanche, la description de la méthodologie du travail est souvent incomplète. Si les idées générales sont énoncées clairement, la justification des choix et les raisonnements amenant les solutions proposées ne sont pas assez rigoureux.

C'est néanmoins un travail ayant abouti à des réalisations intéressantes sur lesquelles d'autres travaux pourront s'appuyer. Considérant tous ces éléments, j'émets un avis favorable à la soutenance de la thèse de Sébastien Miquée en vue de l'obtention du titre de Docteur en informatique de l'université de Franche-Comté.

fait à Strasbourg, le 18 novembre 2011.