Régulation de charge et adaptation de grain : Athapascan

Thierry Gautier, Frédéric Guinand,
Jean-Louis Roch, Alexandre Vermeerbergen

E-mail : {Thierry.Gautier,Frederic.Guinand,Jean-Louis.Roch,vermeer}@imag.fr

Résumé

Nous présentons les techniques de régulation mises en place dans Athapascan afin de contrôler le grain de parallélisme et l'ordonnancement des calculs d'une application parallèle irrégulière. Ce contrôle est basé sur une connaissance partielle – futur proche – du graphe de précédence de l'application. L'originalité d'Athapascan réside dans la possibilité d'introduire des points de choix au sein de l'application pour permettre d'adapter son grain de parallélisme en fonction de la charge de la machine. Athapascan est expérimenté sur des applications en calcul scientifique (dynamique moléculaire) et en calcul formel (algèbre linéaire).

1 Introduction

Pour un très grand nombre de problèmes de base (et notamment en algèbre linéaire), les recherches menées en algorithmique parallèle ont permis de construire, sur des modèles théoriques généraux (PRAM, Network, etc. [7]), des algorithmes possédant un très fort degré de parallélisme et qui effectuent un nombre d'opérations du même ordre que les meilleurs algorithmes séquentiels.

L'intérêt de tels algorithmes très parallèles réside alors dans leur *portabilité* intrinsèque, *i.e.*, leur aptitude à être implémentés de façon simple sur une machine donnée dans la perspective d'obtenir un programme performant [6, 8]. Pour une machine donnée, le regroupement des calculs de grain trop fin – adaptation de grain – et leur ordonnancement – calcul de la date et du site sur lequel doit être effectué chaque calcul [1, 9, 12] – sont les points clefs pour l'obtention de bonnes performances.

Pour assurer la portabilité d'une application tout en conservant une efficacité maximale, il est donc essentiel d'offrir des mécanismes permettant de décider du grain et de l'ordonnancement à tout niveau de découpe. Il y a alors interaction entre l'application qui propose différentes possibilités de degré de parallélisme et le mécanisme qui décide du grain et de l'ordonnancement.

2 Athapascan : Principe de base et Structuration

En Athapascan, le parallélisme est explicite ; il est exprimé par le découpage d'un calcul en sous-calculs. À toute étape de découpe, un algorithme séquentiel est proposé, qui permet d'adapter le grain de l'exécution du programme en bornant son degré de parallélisme (*poly-algorithmes*). L'exécution du programme (calcul du grain et ordonnancement des sous-calculs) est contrôlée par des mécanismes de régulation de la charge, qui permettent de s'adapter à plusieurs machines, qui diffèrent par leur grain (communication/arithmétique) ou par leur nombre de processeurs.

Le parallélisme est exprimé par des appels de procédure asynchrones n-aires. L'intérêt de cette expression est qu'elle décrit dynamiquement le graphe de précédence de l'exécution de l'application. Elle permet ainsi l'obtention d'une information pertinente pour son ordonnancement.

Athapascan limite donc son cadre d'utilisation à des applications dans lesquelles le comportement et les interactions entre processus sont décrits lors de leur création, ce qui le distingue des modèles à base de processus communiquants ou acteurs, par exemple.

Une application Athapascan-1 est un programme unique. Lors du chargement, le code est chargé sur tous les processeurs cibles (parallélisme SPMD). A toute procédure pouvant être appelée à distance est associé un service, dont le rôle sur le site appelé est la récupération des arguments de la procédure, l'exécution de la procédure séquentielle – qui peut contenir des appels à d'autres services –, et la transmission des résultats à l'appelant.

Athapascan est une bibliothèque écrite en C/C++ construite au dessus d'une couche de communication (PVM 3 / MPI) et d'une couche de processus légers. Cette bibliothèque est structurée en deux couches principales :

Athapascan-0 : ce niveau offre toutes les fonctionnalités nécessaires pour effectuer des appels asynchrones de procédure à distance. Il définit donc le modèle d'exécution de base. La synchronisation entre plusieurs procédures est rendue possible par l'introduction de procédures-barrières (inspirées des multi-procédures de Gothic, IRISA).

Athapascan-1 : ce niveau offre différents opérateurs permettant d'effectuer un ordonnancement dynamique. À chaque appel à effectuer en parallèle est associée une décision de degré de parallélisme et de placement. Des extensions sont en cours pour intégrer une décision de date (logique) d'exécution.

Entre ces deux niveaux, en vue de permettre à une application de construire sa propre politique de régulation, Athapascan-Toolkit offre des primitives permettant de défini une architecture de régulation (centralisée, distribuée ou hiérarchique), des indicateurs de charge et une politique de diffusion des informations de charge.

Dans l'implémentation en cours, dont la cible privilégiée est un IBM-SP1 (32 processeurs), l'architecture de régulation est hiérarchique avec une stratégie à deux seuils.

3 Ordonnancement dynamique en Athapascan

De manière générale, chaque possibilité d'appel de service à distance (parallélisme potentiel) est précédé d'un appel au régulateur – généralement avec une information de coût approximatif de calcul et de volume de communication –, lequel retourne un choix de grain et un ordonnancement. L'application effectue ensuite l'appel correspondant au grain calculé avec l'ordonnancement imposé. Il existe trois formats distincts d'appels au régulateur, que nous détaillons.

3.1 L'appel de procédure à distance avec choix local

Ici, le choix est soit d'appeler la procédure localement, soit à distance. Dans le cas où la réponse est locale, c'est directement la fonction associée qui est exécutée. Cette possibilité permet la migration d'un calcul vers un autre processeur. Il est alors possible d'alimenter des processus sous-chargés en allégeant les processus sur-chargés.

3.2 L'appel de plusieurs procédures à distance

Très souvent, et notamment dans les algorithmes de type « diviser pour régner », plusieurs appels de procédures sont à effectuer en parallèle.

En Athapascan, il est possible de spécifier en une instruction plusieurs appels de procédures (on parle de *multi-procédure*). Le régulateur dispose ainsi de plus d'informations pour calculer une répartition. La déclaration d'un groupe de *job*s généralise celle utilisée pour un seul. Il faut noter que l'attribution d'un coût, même grossier, à chaque *job* peut permettre au régulateur d'effectuer des choix judicieux si les travaux sont hétérogènes [10].

3.3 Appel d'une alternative de procédures à distance

Une alternative d'appels de procédures peut être exprimée en Athapascan. Le but de cette construction est double : d'une part, elle exprime plusieurs niveaux de découpes possibles pour un même calcul parallèle : cela peut aider le régulateur à choisir plus justement le *grain* du parallélisme. D'autre part, il existe en général non pas un, mais plusieurs algorithmes parallèles pour résoudre un problème donné. La complexité de ces algorithmes dépend au moins de la taille du problème, mais aussi du nombre de processeurs « disponibles » et des caractéristiques de la machine. Pour le régulateur, l'alternative permet donc, en principe, de choisir les meilleurs algorithmes.

Les alternatives ne peuvent être employées qu'avec des régulateurs adaptés : il faut en particulier pouvoir attribuer un *coût* à chaque alternative, qu'il soit relatif ou absolu. La possibilité d'employer des coûts portables en utilisant les fonctions de complexité des algorithmes a été étudiée dans [10, 11]. Le portage de cette possibilité à Athapascan-1 et son expérimentation est en cours.

Un cas spécifique de poly-algorithme est celui où le calcul peut être découpé en k sous-calculs, avec $1 \leq k \leq M$, M étant la borne sur le degré de parallélisme potentiel du calcul. En parallèle, en supposant que le calcul est effectué sur k processeurs, le temps de calcul séquentiel T_S est alors divisé par k, avec une surcharge en calcul T_O (due à la partition en sous-calculs et à la fusion des résultats obtenus) multipliée par k. Une autre surcharge provient du volume de communication nécessaire pour effectuer les k sous-calculs en parallèle. Ce volume est lui-aussi séparé en deux parties : le volume V_D des données distribuées (i.e., sans réplication) aux k sous-calculs et donc divisé par k, et le volume V_B des données diffusées (i.e., répliquées pour chacun des sous-calculs) aux k sous-calculs. Un tel appel s'exprime sous la forme : `MapJobs`$(1, M, T_S, T_O, V_D, V_B)$.

En conséquence, l'application récupère le degré K de parallélisme le mieux adapté et l'ordonnancement associé.

Grâce à cette primitive, l'algorithme de partition et de fusion des résultats peut facilement être écrit de manière générique, notamment dans le cas de calculs en algèbre linéaire. L'hypothèse de découpe linéaire du temps de calcul correspond au cas d'algorithmes parallèles optimaux.

Remarque 1 : Si aucune information de coût ne peut être connue sur les temps de calcul ou de communication, la décision de grain est prise uniquement en fonction des seuils de charge, et l'ordonnancement est effectué de manière aléatoire sur les machines les moins chargées.

Remarque 2 : Les algorithmes qui utilisent le parallélisme itératif, qu'on rencontre fréquemment en calcul scientifique (ex. : élimination de Gauss) s'expriment à l'aide de procédures barrières.

4 Conclusion

Athapascan-0 [3] et Athapascan-Toolkit sont opérationnels. En ce qui concerne Athapascan-1, un régulateur dynamique décidant du grain à partir d'informations sur la longueur de la file d'attente CPU et de coûts sur les appels est développé. Ce régulateur ne délivre actuellement que les sites sur lesquels doivent être exécutés les appels. Une extension délivrant des dates (logiques, essentiellement une date « plus tard » qui permet de mettre des sous-calculs en réserve pour les placer ultérieurement) est en cours d'étude.

Athapascan-1 est utilisé dans un logiciel de calcul formel PAC++. Les résult ats obtenus sur des programmes de relativement gros grain, mais déséquilibrés – notamment calcul du polynôme caractéristique d'une matrice creuse à coefficients rationnels [4] – montrent l'efficacité d'Athapascan pour la résolution de ce type de problè mes.

D'autres applications sont en cours d'étude, en particulier au sein du projet national STRATAGÈME en collaboration avec le LABRI à Bordeaux (résolution de systèmes linéaires creux)

et le laboratoire PRISM à Versailles (optimisation combinatoire) [2].

L'intérêt d'Athapascan-1 est que l'exécution du programme décrit directement son graphe de précédence. Lorsque le problème est régulier, c'est à dire qu'il est possible d'obtenir une description du graphe de précédence par une exécution symbolique (effectuée à partir de la connaissance de la taille des données en entrée de l'application et non de leurs valeurs), l'interprétation abstraite du programme Athapascan est possible (une directive permet de masquer les calculs sur les valeurs en entrée). Un travail est en cours pour calculer un ordonnancement statique du programme (c'est à dire affecter à toutes les activités qui seront créées lors de l'exécution un site d'exécution) en utilisant un algorithme de liste [5]. L'objectif poursuivi ici est de pouvoir utiliser dynamiquement un algorithme de liste lorsque le graphe de précédence est progressivement découvert au cours de l'exécution d'une application irrégulière.

Références

[1] I. Ahmad, A. Ghafoor, and G.C. Fox. Hierarchical Scheduling of Dynamic Parallel Computations on Hypercube Multicomputers. *J. Parallel and Distributed Computing*, 20:317–329, 1994.

[2] C. Roucairol et al. *Conception et Analyse des Algorithmes Parallèles.* Hermès à paraître, 1995.

[3] M. Christaller. Athapascan-0a sur PVM 3. Définition et mode d'emploi. Technical Report 11, LMC-IMAG, 1994.

[4] T. Gautier and J.L. Roch. PAC++ System and Parallel Algebraic Numbers Computat ion. In Hoon Hong, editor, *First International Symposium on Parallel Symbolic Computation (PASCO'94)*, volume 5 of *Lecture Notes Series in Computing*, page 145, 1994.

[5] F. Guinand. *Ordonnancement avec communications et modèle d'exécution* Pré-rapport de thèse. PhD thesis, INPG, 1995.

[6] K.T. Herley and G. Bilardi. Deterministic simulations of prams on bounded degree networks. In *Proc. Twenty-Sixth Annual Allerton Conference on Communication, Control and Computation*, pages 1084–1093, 1988.

[7] J. Jájá. *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.

[8] A. Karlin and E. Upfal. Parallel hashing - an efficient implementation of shared-memory. *SIAM J. Computing*, 35(4):876–892, 1988.

[9] B. Kruarachue and T. Lewis. Grain size determination for parallel processing. *IEEE Software*, pages 876–892, January 1988.

[10] J. L. Roch, A. Vermeerbergen, and G. Villard. A new load-prediction scheme based on algorithmic cost functions. In Springer-Verlag, editor, *comptes-rendus de CONPAR 94-VAPP VI, LNCS*, Sep. 1994.

[11] A. Vermeerbergen. Les poly-algorithmes et la prévision de coûts pour une expression portable et extensible du parallélisme. In L. Bougé, editor, *Actes de RenPar'6, ENS Lyon, France*, pages 51–54. Ecole Normale Supérieure de Lyon, Jun 1994.

[12] T. Yang and A. Gerasoulis. A parallel programming tool for scheduling on distributed memory multiprocessors. In *Proc. of SHPCC'92*, pages 330–343, 1990.