Aide à la décision pour le choix et le paramétrage de protocoles de cohérence des données

Safae Dahmani, Loïc Cudennec, Guy Gogniat

Aide à la décision pour le choix et le paramétrage de protocoles de cohérence des données.

Safae Dahmani[1,2], Loïc Cudennec[1], Guy Gogniat[2]

[1] CEA, LIST
Saclay, France
{prénom.nom}@cea.fr

[2] Université de Bretagne-Sud
Lorient, France
{guy.gogniat}@univ-ubs.fr

Mots-cl : *Cohérence des données, compilation itérative, optimisation de protocoles.*

1 Contexte général

L'introduction des systèmes massivement parallèles manycores promet de très hautes performances à basse consommation d'énergie. Afin de mieux profiter de telles performances de calcul, il est important de garantir des techniques de parallélisation efficaces. Parmi les modèles de programmation dédiés aux contextes hautement parallèles, le modèle à mémoire partagée a l'avantage de faciliter la parallélisation via l'accès à un espace d'adressage global. Le choix d'un tel paradigme de programmation repose sur l'efficacité du système à gérer les accès aux données partagées. Dans un tel système, les règles de gestion de la synchronisation, la localité et le transfert des données sont mises en œuvre par des protocoles de cohérences. Le choix de ces derniers représente une partie de la capacité du système à améliorer les performances de l'application à l'exécution. De précédents travaux ont montré qu'il n'y a pas un unique protocole adapté aux différents contextes. Nous considérons que le choix d'un protocole adapté doit prendre en compte les caractéristiques de l'application mais aussi les propriétés de la plate-forme cible. Pour répondre à la question sur le choix des protocoles de cohérence, nous proposons un modèle de plate-forme à protocoles multiples. Ce modèle permet d'affecter à chaque accès à une variable partagée un protocole approprié. Nous nous intéressons dans ce travail à la problématique de prise de décision pour un choix de protocole efficace et adapté au contexte de l'application. La définition et la configuration des protocoles choisis sont réalisées à travers un processus de compilation itérative. Ce processus est constitué de plusieurs étapes allant de la caractérisation de l'application à la définition des paramètres des protocoles choisis. Le travail présenté met en avant les deux phases liées aux choix de protocoles et à la définition de leurs paramètres. La phase d'évaluation des protocoles paramétrés est basée sur un processus analytique haut niveau nous permettant d'esquisser le comportement général du protocole à un bas coût de calcul et avec un niveau de précision suffisant. Dans les travaux que nous présentons, le choix du protocole et sa configuration sont effectués en deux phases différentes utilisant des techniques de *Recherche opérationnelle*.

2 La cohérence de données pour les manycores

Dans un modèle de programmation parallèle à mémoire partagée, l'accès aux données se fait de manière directe via un espace d'adressage commun. Cette approche permet de faciliter à l'utilisateur les accès aux données partagées. Toutefois, la tâche de la gestion efficace de ces accès est déléguée au système, qui doit garantir la disponibilité de la donnée et la fraîcheur de

sa valeur retournée par chaque opération de lecture. Pour répondre à ceci, les protocoles de cohérence sont en charge de gérer les synchronisations entre les accès concurrentiels, la localité et le transfert des données sur la puce. Avec l'augmentation du nombre de cœurs intégrés sur une même puce et les exigences en mémoire des applications hautement parallèles, l'efficacité et le passage à l'échelle des protocoles de cohérence deviennent de plus en plus importants pour l'amélioration des performances de calcul. Il existe dans la littérature, différentes techniques de gestion de la cohérence. L'une des techniques qui passe bien à l'échelle est la gestion distribuée de la cohérence [1] où chaque cœur est en charge de la traçabilité d'un ensemble de données. Ce cœur est consulté par tout autre cœur désirant accéder à la donnée, afin d'avoir l'information sur sa localité, et son état. La gestion distribuée des données se trouve heurtée à de nombreux problèmes, notamment ceux liés à la mémoire. Étant limitée à la fois par la surface du Silicon et le surcoût d'accès à la mémoire externe, il devient crucial d'optimiser le stockage des données sur la puce. Le protocole de glissement de données [3] a été proposé, par des travaux précédents, afin de réduire le taux déjection des données hors puce en améliorant l'utilisation des caches privés des cœurs. Il est basé sur une approche à caches coopératifs où chaque cœur peut transférer ses données les plus fréquemment sollicitées à ses voisins les plus proches. Les cœurs coopératifs envoient leurs données, à leur tour, dans leurs voisinages proches. Ainsi, la propagation s'enchaîne sur la puce (de voisinage en voisinage) jusqu'à atteindre une zone moins stressée. Des dérivées de ce protocole ont été étudiées [4] permettant d'adapter le rayon de glissement des données à la charge mémoire des zones stressées, mais aussi à leur distribution sur la puce. A travers ces différentes études, il est clair qu'il n'existe pas de protocole de cohérence compatible à tout contexte applicatif, et à toute architecture cible. Le choix des protocoles de cohérence doit donc prendre en compte les propriétés de l'application et le contexte de performances souhaitées par l'utilisateur ainsi que les contraintes matérielles à l'exécution. La section suivante explique la chaîne de prise de décision, allant de la caractérisation de l'application à la configuration des protocoles choisis.

3 Prise de décision sur les protocoles de cohérence

Le processus de définition des protocoles de cohérence fait partie de la chaîne de compilation globale. Une première phase dans la chaîne de compilation permet de définir un graphe d'accès aux données partagées à partir de l'analyse statique du code. La caractérisation du comportement de l'application se fait à travers la reconnaissance de patrons d'accès aux données au sein de ce graphe descriptif (ex : producteur/consommateur). Cette analyse fournit également des informations sur la fréquence d'accès à chacune des données partagée, ainsi que les dépendances entre ces différents accès. En plus du comportement de l'application, l'utilisateur peut définir le contexte applicatif ainsi que le profil de performance qui en découle (ex : temps réel, basse énergie). Nous définissons une bibliothèque de protocoles de cohérence dans laquelle les différents protocoles sont classés par profil de performances. Par exemple, grâce au principe de préchargement des motifs d'accès, le protocole Cocca [2] permet d'accélérer les accès aux données, il est donc plus adapté pour un profil haute performance. Un protocole coopératif tel que le glissement de données [3] permet de réduire les pics de charge dans les zones les plus stressées, il est ainsi plus adapté à un profil d'équilibrage de consommation énergétique sur la puce. Le raffinement des choix de protocoles se fait lors de la deuxième phase de paramétrage considérant les caractéristiques de la plate-forme d'exécution.

4 Algorithme d'optimisation des protocoles choisis

Chaque instance de protocole est caractérisée par un ou plusieurs paramètres définissant son comportement (ex : la longueur du motif de préchargement pour le protocole Cocca). On appelle une configuration toute combinaison d'instances de protocoles affectée à une application. Le choix des paramètres dans une configuration permet d'ajuster son comportement selon les besoins du contexte applicatif telle qu'une variation de la charge de la mémoire sur puce. La

combinaison de plusieurs instances (différents protocoles, différents paramètres) permet aussi une adaptation dynamique aux variations en ligne du comportement de l'application. Ainsi, la complexité d'optimisation de l'ensemble des protocoles augmente en fonction du nombre de phases caractérisant l'application et de la plage des valeurs des différents paramètres à configurer. Un paramètre tel que le rayon de glissement des données pour un protocole coopératif est limité par la taille de la puce ce qui permet de borner la plage des solutions . Tandis qu'il est moins évident de couvrir un espace de solutions plus large comme le cas d'une combinaison de plusieurs protocoles. Dans la solution proposée, les protocoles sont d'abords instanciés de manière aléatoire. Ensuite, un processus d'optimisation itératif va permettre de rechercher les paramètres optimaux dans un contexte donné. Ce processus d'optimisation est inspiré d'un algorithme génétique (voir figure 1). Les protocoles de cohérence choisis sont initialisés par des paramètres arbitraires. Ces paramètres sont variés, par la suite, de manière à balayer leur intervalle de définition. Pour chaque nouvelle configuration (nouveaux paramètres), on effectue une évaluation des performances, et seuls les paramètres qui permettent de s'approcher de la fonction objective définie sont gardés. Le processus de prise de décision est répété jusqu'à atteindre une solution stable satisfaisant les propriétés définies.

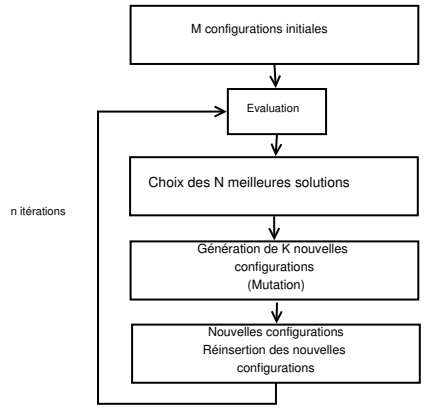

FIG. 1 – Algorithme d'optimisation

5 Étude de cas

Dans cette section, nous nous intéressons au cas particulier du paramétrage des protocoles de cohérence par glissement de données. Comme précisé dans des travaux précédents [4], le choix des paramètres d'un tel protocole détermine son efficacité à gérer le stockage des données. Le rayon de glissement de données représente dans ce cas le principal paramètre de ce protocole car il définit la durée de vie d'une donnée sur la puce. Le rayon de migration d'une donnée doit permettre un compromis entre la distance séparant le cœur propriétaire et ses données délocalisées, et l'équilibrage de la charge mémoire sur la puce. Le coût d'accès aux données est, en effet, à prendre en compte lorsque l'on décide de stocker la donnée dans un cœur plus éloigné ou l'éjecter hors puce. Dans la phase de définition des paramètres de glissement des données partagées, l'objectif est de minimiser le taux d'éjection hors-puce des données tout en garantissant de petites distances de migration afin de minimiser le coût d'accès aux données. En plus de la configuration des instances de protocoles pour optimiser les performances des protocoles d'autres facteurs, liés au processus d'optimisation, sont à prendre en compte. Plusieurs scénario d'affectation des protocoles de cohérence sont autorisés par la plate-forme multi-protocolaire proposée. Dans le cas où une seule instance de protocole de glissement est assigné à l'ensemble de l'application, la complexité de recherche du paramètre optimal ne dépendra que de la plage des valeurs que peut prendre ce paramètre (i.e Rayon de glissement). Par ailleurs, lorsqu'on distingue différentes phases dans l'application, plusieurs instances de protocoles sont assignées à l'application selon la caractérisation de chaque phase. Le processus

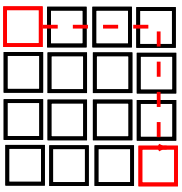

FIG. 2 – La distance maximale de glissement dans un mesh 2D à 16 cœurs

de paramétrage de chaque instance de protocole est plus complexe car il dépend, en plus de la plage des paramètres, du nombre de phases détectées dans l'application. En outre, le modèle de la plate-forme multi-protocolaire permet d'associer un protocole de cohérence à chaque données partagée. Ce niveau de granularité garantit une meilleure adaptabilité du protocole au comportement de l'application à un très grand niveau de précision. Or, la complexité liée à la définition des paramètres des instances de protocoles va augmenter proportionnellement au nombre de données partagées de l'application. Pour un protocole de glissement de données, l'espace des solutions d'une telle configuration est fonction du nombre de données, de la plage des valeurs que peut prendre le rayon de glissement et du nombre de phases caractérisant l'application. Le rayon maximal de glissement, limité par la taille de la puce, correspond à la distance entre les deux cœurs les plus éloignés de la puce (voir exemple : figure 2). Une recherche exhaustive dans l'espace des solutions permettrait de définir de manière exacte la solution optimale. Or, le coût de la solution augmente proportionnellement au nombre des données partagées. L'utilisation de l'algorithme génétique décrit dans ce travail permet donc de trouver une solution stable dans un temps réduit. La précision de la solution est fortement dépendante du nombre d'itérations effectuées dans le processus de recherche des solutions. La complexité du processus de prise de décision dépend de bien d'autres facteurs tels que la fréquence de changement de phase dans une application et le nombre de protocoles assignés qui en découle. De futurs travaux vont porter sur cette question afin de raffiner l'algorithme proposé.

6 Conclusion

L'efficacité des phases de prise de décision et de paramétrage des protocoles dépend des algorithmes d'optimisation utilisés. Nous avons proposé un algorithme génétique itératif afin de parcourir un grand nombre de solutions. Dans des travaux futurs, nous nous intéressons à d'autres pistes d'optimisation des deux problèmes de prise de décision définis auparavant. Afin de répondre à la question du coût d'évaluation de l'ensemble des solutions, l'utilisation d'heuristiques ou d'algorithmes adaptés peut réduire la complexité de la résolution tout en garantissant une bonne qualité de la solution.

Références

[1] Kai Li and Paul Hudak. Memory coherence in shared virtual memory systems. *ACM Trans. Comput. Syst.*

[2] Jussara Marandola, Stéphane Louise, Loïc Cudennec, Jean-Thomas Acquaviva, and David Bader. Enhancing Cache Coherent Architectures with Access Patterns for Embedded Manycore Systems. IEEE, 2012.

[3] S.Dahmani, L.Cudennec, and G.Gogniat. Introducing a data sliding mechanism for cooperative caching in manycore architectures. *Proceedings of the 18th International Workshop on High-Level Parallel Programming Models and Supportive Environments*, 2013.

[4] S.Dahmani, L.Cudennec, S.Louise, and G.Gogniat. Using the spring physical model to extend a cooperative caching protocol for many-core processors. In *Embedded Multicore/Manycore SoCs (MCSoc)*. IEEE, 2014.