Evaluation des performances de benchmarks MPI sur des architectures multiprocesseur de type CC-DSM

Meriem Zidouni

Bull S.A.
Platforms Hardware R&D
Architecture & Verification
Rue Jean Jaurès
78340 Les Clayes Sous Bois, FRANCE
Meriem.Zidouni@bull.net

1. Introduction

Dans le cadre de son offre de serveurs haut de gamme, Bull[1] conçoit des multiprocesseurs à mémoire distribuée partagée avec un protocole de cohérence de caches CC-DSM (*Cache-Coherent Distributed Shared Memory*) et fournit une implémentation de la bibliothèque de programmation parallèle MPI (*Message Passing Interface*). Le but de notre recherche est de fournir une méthode et ses outils associés, permettant l'évaluation et l'optimisation de cette implémentation en fonction de l'architecture matérielle : topologie d'interconnexion et protocole de cohérence de caches. Ceci permettra, d'une part, de faire les bons choix d'architecture matérielle et d'implémentation logicielle au moment de la conception, et d'autre part, fournira des éléments d'analyse nécessaires pour comprendre les mesures faites au moment de la validation de la machine réelle. Nous définissons et expérimentons une telle méthode, à travers l'étude de cas d'un benchmark de MPI (*ping-pong*). Notre méthode de modélisation est basée sur la théorie des IMC (*Interactive Markov Chains*), implémentée dans la boîte à outils CADP [1]. Elle consiste à modéliser le système en langage LOTOS [2], puis à vérifier formellement sa correction fonctionnelle, et finalement à évaluer ses performances après l'avoir augmenté avec des informations quantitatives (latences).

2. Modélisation et vérification du benchmark ping-pong

Le benchmark ping-pong consiste en des envois alternés de messages entre processus via les primitives *send* et *receive*. Celui considéré dans notre étude est la composition parallèle de deux processus P_0 et P_1, qui s'envoient alternativement un seul message à la fois :

$$\texttt{ping\_pong}(P_0,P_1) =$$
$$\texttt{<send}(P_0 \rightarrow P_1)\texttt{; receive}(P_0 \leftarrow P_1)\texttt{>}$$
$$||$$
$$\texttt{<receive}(P_1 \leftarrow P_0)\texttt{; send}(P_1 \rightarrow P_0)\texttt{>}$$

Dans une implémentation MPICH pour une architecture CC-DSM, les primitives *send* et *receive* utilisent des tampons de messages et éventuellement des verrous pour gérer l'exclusion d'accès aux tampons, en tant que mécanisme de communication. Ces structures de communication résident dans la mémoire distribuée et leurs accès sont régis par le protocole de cohérence de caches qui assure l'intégrité des données, et implique l'emplacement de leur valeur valide à un moment donné.
Comme le programme contient des processus qui s'exécutent en parallèle et accèdent à des variables partagées, un accès d'un processus aura une latence qui ne peut pas être déterminée statiquement, puisqu'elle dépend de l'entrelacement des actions qui auront précédé cet accès. Par conséquent, l'accès à une variable peut avoir différentes latences possibles correspondant aux éventuels transferts entre l'emplacement de la donnée valide (mémoire locale ou distante, ou cache d'un processeur distant) par rapport au processeur demandeur. La latence de ces transferts détermine la performance des communications et des synchronisations inter-processus.
Nous avons modélisé de façon indépendante, en langage LOTOS, les 3 aspects qui définissent les performances d'une implémentation de MPI dans une architecture CC-DSM :

1. la succession d'accès définie par l'algorithme des primitives de communication : dans notre cas de ping-pong, il s'agit des primitives *send* et *receive* qui correspondent à des suites d'accès aux variables pour des opérations de lecture (*load*) et d'écriture (*store*) ;
2. le protocole de cohérence de caches qui régit les changements des états des caches (suivant le protocole MESI classique), et les transferts entre caches et mémoires, et entre caches ;
3. la topologie d'interconnexion de l'architecture multiprocesseur, qui détermine la latence des transferts suivant le niveau de distance entre les mémoires et les processeurs et entre les processeurs.

Nous avons vérifié la correction fonctionnelle du comportement donné par la spécification LOTOS

[1] http ://www.bull.com

du ping-pong avec les outils de CADP. Pour établir la correction de l'algorithme ping-pong, nous avons employé la vérification visuelle (*visual checking*) en utilisant les outils BCG_MIN et BCG_EDIT. Pour analyser le protocole de cohérence de caches et la gestion des verrous, nous avons utilisé la vérification par logiques temporelles (*model checking*) à l'aide de l'outil EVALUATOR.

3. Evaluation des performances du benchmark ping-pong

Le but de notre étude est d'évaluer les performances du benchmark ping-pong, ce qui consiste à calculer la latence d'un échange de message (un *send* suivi d'un *receive*) et le nombre de *miss* effectués sur chaque variable de l'algorithme pendant cet échange. Pour cela, nous avons adopté l'approche proposée en [3] basée sur la théorie des IMC. Elle consiste à enrichir la spécification LOTOS, dont les propriétés fonctionnelles ont été déjà vérifiées, par des informations quantitatives appelées *délais Markoviens*. Dans notre cas, ces délais Markoviens correspondent aux latences des transferts d'accès aux variables par les opérations de lecture et d'écriture, évaluées durant la génération du modèle du ping-pong en LOTOS. Ensuite, il faut vérifier, d'une part, que cette insertion est sémantiquement correcte, en s'assurant par exemple qu'elle préserve la concurrence des processus parallèles dans le temps, et d'autre part, qu'elle ne perturbe pas le comportement fonctionnel de la spécification originelle. Nous avons appliqué l'outil BCG_STEADY de CADP pour le calcul des performances sur le modèle Markovien du ping-pong. C'est un outil qui permet de calculer, de manière itérative, à l'état d'équilibre la distribution de probabilité à long terme (de ce fait, on obtient la latence d'un échange de message), et de fournir des mesures de débit pour chacune des étiquettes de transitions (on obtient le nombre de *miss* effectués sur chaque variable pendant l'échange d'un message).

4. Résultats

Cette approche nous a permis d'obtenir des latences moyennes d'échange d'un message confortées par les mesures expérimentales, et de les analyser avec le nombre de *miss* effectué sur les différentes variables pendant cet échange.
Nous avons également évalué et analysé les performances du ping-pong en faisant varier des aspects matériels et logiciels :

- la distance entre les processeurs dans le réseau d'interconnexion de l'architecture : on considère trois niveaux différents de distance entre les processeurs sur lesquels s'exécute le ping-pong ;
- le protocole de cohérence de caches : on étudie deux protocoles de cohérence de caches dont la différence réside dans le changement des états des caches ;
- les primitives *send* et *receive* : on considère deux sortes de primitives : celles qui utilisent des listes chaînées avec des verrous, et celles qui utilisent des tampons sans verrous.

5. Conclusion et perspectives

Nous envisageons de poursuivre nos recherches suivant plusieurs directions. Premièrement, nous allons focaliser nos efforts sur la modélisation en LOTOS de plusieurs variantes de protocoles de cohérence de caches, ainsi que d'autres primitives de communication et de synchronisation de MPI. Ces modèles, groupés dans des bibliothèques réutilisables et paramétrées par la topologie d'interconnexion des processeurs, permettraient de modéliser facilement des configurations différentes de machines. Deuxièmement, nous souhaitons mettre en oeuvre une traduction automatique entre la description de l'algorithme distribué et des primitives de communication et la modélisation LOTOS correspondante, augmentée de latences. Cela permettrait de disposer d'une chaîne complète de modélisation et évaluation de performances qui assisterait de manière utile les concepteurs de logiciels distribués sur les machines multiprocesseurs.

Bibliographie

1. H. Garavel, F. Lang, R. Mateescu, and W. Serwe. – CADP 2006 : A Toolbox for the Construction and Analysis of Distributed Processes. – Proceedings of the 19th International Conference on Computer Aided Verification CAV'2007 (Berlin, Germany), vol. 4590 of Lecture Notes in Computer Science, pp. 158-163. Springer Verlag, July 2007.
2. ISO/IEC. – LOTOS : A Formal Description Technique based on the Temporal Ordering of Observational Behaviour. – International Standard ISO/IEC 8807, 1989.
3. H. Garavel and H. Hermanns. – On Combining Functional Verification and Performance Evaluation using CADP. – Proceedings of the 11th International Symposium on Formal Methods Europe FME'2002 (Copenhague, Denmark), vol. 2391 of Lecture Notes in Computer Science, pp. 410-429. Springer Verlag, July 2002.