Limitations of the PlayStation 3 for High Performance Cluster Computing

Alfredo Buttari, Jack Dongarra, Jakub Kurzak

July 2007

En novembre 2006 est sortie la PlayStation 3 [1] de Sony, l'une des toutes premières architectures basées sur le processeur Cell Broadband Engine [2] de Sony, Toshiba et IBM. La particularité de cette architecture est d'être rapide et puissante en terme de calcul tout en restant bon marché. De plus, elle répond aux besoins actuels en terme de consommation et de dissipation de chaleur.

L'article de Buttari et al. fait une étude des possibilités offertes par l'architecture de la PS3 pour la réalisation d'une grille de calcul : se prête-t-elle à du calcul rapide et performant ? Nous verrons que, outre le fait que la programmation ne soit pas aisée sur ce système, il présente également de nombreuses limitations en terme de rapidité et de mémoire. Pour cela, nous testerons les possibilités de la grille de calcul sur le produit de matrices.

1 Architecture de la grille de calcul

On réalise une grille de calcul composée de *np* PS3, elles-mêmes dotées d'un processeur Cell BE (voir Fig. 1). Nous allons voir quelles sont les caractéristiques techniques à chaque niveau (processeur Cell, PS3 et grille) qui permettent de comprendre les caractéristiques de la grille de calcul. Nous ne nous attacherons car une version réduite des architectures, centralisée sur le sujet qui nous intéresse, à savoir l'étude des performances du système.

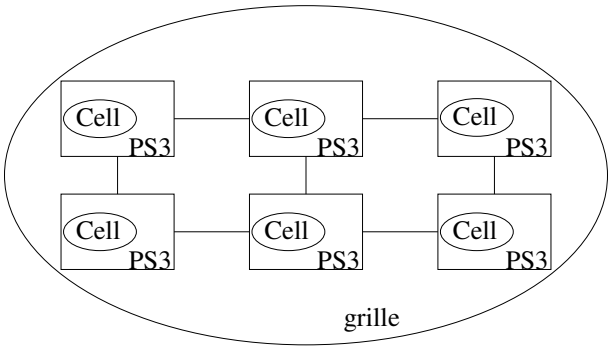

Fig. 1 – *Grille de calcul*

1.1 Processeur Cell

1.1.1 Architecture

Le processeur Cell BE est un processeur multicœur hétérogène composé d'un PowerPC, de huit unités de calcul et d'une mémoire partagée, le tout relié à un bus (voir FIG. 2).

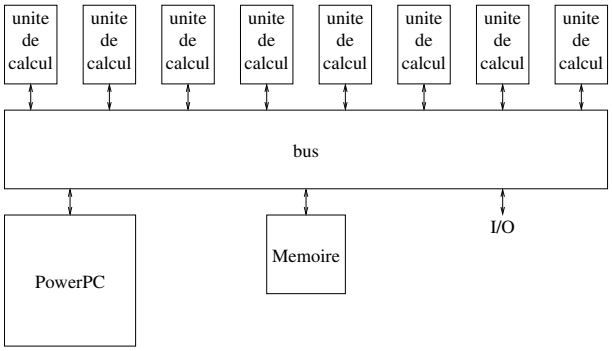

FIG. 2 – *Architecture simplifiée d'un processeur Cell*

PowerPC Le PowerPC est un processeur 64-bit qui sert à contrôler le flot d'exécution pour les unités de calcul.

Unités de calcul Les unités de calcul sont des processeurs vectoriels permettant d'effectuer des opérations en simple ou en double précision. Ils ont chacun, en simple précision, une vitesse d'exécution maximale de 25,6 Gflop/s. Cependant, les instructions en double précision n'étant pas totalement pipelinées, la vitesse d'exécution maximale en double précision n'est que de 1,8 Gflop/s.

Elle possèdent chacune une mémoire locale de 256 Ko et 128 registres de 128 bits chacun.

Bus La bande passante maximale du bus est de 204,8 Go/s.

1.1.2 Points forts

On constate que l'organisation de la mémoire permet 128 transferts simultanés entre les unités de calcul et la mémoire partagée, ce qui est plutôt rapide. De plus, la présence du PowerPC pour le contrôle des opérations donne une grande fréquence à l'ensemble.

Pour développer des applications sur une architecture Cell, l'utilisateur possède un contrôle total sur le matériel, et peut donc réaliser une implantation efficace, pourvu qu'il respecte certaines règles de programmation :

- utiliser des opérations vectorielles ;
- aligner les données en mémoire ;
- masquer le coût des communications entre la mémoire partagée et les unités de calcul par le coût des calculs ;
- réutiliser au maximum les données du cache ;
- dérouler les boucles ;
- réduire le nombre de branchements.

1.1.3 Points faibles

La distribution de la mémoire est telle qu'il y a de très nombreux transferts entre la mémoire partagée et les unités de calcul : même si ces transferts peuvent se faire rapidement, il faut donc mieux essayer de les cacher par le calcul.

Ainsi, si cette architecture peut s'adapter à l'algèbre linéaire dense, qui demande beaucoup de calculs et des accès assez réguliers à la mémoire, elle ne convient pas du tout pour l'algèbre linéaire creuse : les accès à la mémoire sont très particuliers et une mauvaise implantation peut les rendre lents.

1.2 PlayStation 3

1.2.1 Architecture

La PS3 se compose donc d'un processeur Cell BE, auquel on ajoute :
- une mémoire de 256 Mo ;
- une carte Gigabit Ethernet ;
- une carte graphique NVIDIA avec 256 Mo de mémoire RAM.

On ne peut pas accéder au matériel directement : un hyperviseur, appelé Game OS, installé sur l'une des unités de calcul du processeur Cell, sert à relier le matériel et le logiciel. On ne peut programmer la PS3 sans passer par celui-ci, ce qui pose de nombreux problèmes :
- l'accès au réseau, qui doit se faire à travers l'hyperviseur, s'en trouve ralenti ;
- la carte graphique, et notamment sa mémoire RAM, est totalement inaccessible ;
- le Game OS occupe l'une des unités de calcul à laquelle on ne peut plus accéder.

De plus, pour les besoins de la programmation de bibliothèques de calcul, un système d'exploitation Linux est installé sur une autre des unités de calcul, ce qui l'empêche également de participer au calcul distribué.

1.2.2 Point(s) fort(s)

La PS3 possède une assez bonne carte réseau, ce qui permettra d'augmenter la vitesse des communications dans la grille de PS3.

1.2.3 Points faibles

Nous avons déjà évoqué les problèmes liés à la présence du Game OS et au fait que deux des unités de calculs sont occupées.

Un problème se pose également au niveau de la mémoire : une mémoire de 256 Mo ne permet pas de faire du calcul sur de grandes matrices ! Cela limite énormément les possibilités de calcul, surtout si on utilise de la double précision.

1.3 Grille de calcul

1.3.1 Architecture

On connecte les np processeurs via un switch. Pour ne pas perdre de performance au niveau du réseau, il faut utiliser un switch de bonne qualité.

1.3.2 Protocole

L'article teste trois protocoles MPI, afin de choisir celui qui donnera les meilleures performances. Pour cela, les auteurs utilisent une bibliothèque de test de protocoles MPI pour comparer MPICH1, MPICH2 et OpenMPI en termes de bande passante et de latence.

On constate que MPICH1 est moins performant que les deux autres protocoles. De plus, tous ces protocoles voient leurs performances très diminuées par la présence de l'hyperviseur qui ralentit les communications : au maximum, la bande passante n'est qu'à 60% de sa valeur théorique.

L'article ne mentionne pas quelle bibliothèque est utilisée au final, ce qui empêche la reproductibilité de l'expérience décrite ci-dessous.

2 Test de performances

Pour tester les performances de cette grille de calcul, on étudie le produit de matrices denses distribué. Ceci est en effet un bon exemple :

- comme vu précédemment, les caractéristiques du système le rendent plus propice à l'algèbre linéaire dense ;
- on connaît pour ce produit des algorithmes performants et faciles à implanter, et qui utilisent des opérations de bases que l'on sait réaliser très rapidement ;
- c'est un exemple qui demande beaucoup de calculs, et permet donc d'utiliser la méthode de programmation consistant à cacher le coût des communications par le coût des calculs.

2.1 Choix des caractéristiques de la grille de calcul

On a les caractéristiques suivantes :

Donnée	Notation	Valeur
Nombre de processeurs	np	4
Taille d'un vecteur (en octets)	$fpsize$	4
Bande passante du réseau	bw	600 Mb/s
Performance maximale d'une PS3	$peak$	$25,6 \times 6 = 153,6$ Gflop/s

Pour la deuxième ligne, le fait que les vecteurs aient pour longueur 4 octets se traduit par le fait que l'on est en simple précision.

Pour la dernière ligne, chaque unité de calcul a en effet une performance maximale de 25,6 Gflop/s, et seules six unités sont disponibles.

2.2 Choix de l'algorithme

L'algorithme retenu car correspondant le mieux à l'architecture dont on dispose est l'algorithme SUMMA avec un stockage par blocs acyclique des matrices : si on a np processeurs et une matrice de taille $m \times n$, chaque processeur possède $\frac{m}{np} \times \frac{n}{np}$ coefficients de la matrices qui forment un bloc.

On veut multiplier $A \in \mathcal{M}_{m,n}(\mathbb{R})$ et $B \in \mathcal{M}_{n,p}(\mathbb{R})$:

- A est divisée en blocs de taille $\frac{m}{np} \times \frac{n}{np}$, et on notera A_i la i^{e} colonne de blocs de A ;

– B est divisée en blocs de taille $\frac{n}{np} \times \frac{p}{np}$, et on notera B_i la i^{e} ligne de blocs de B ;

On stocke le résultat dans $C \in \mathcal{M}_{m,p}(\mathbb{R})$. L'algorithme utilisé est le suivant :

1. Pour i de 1 à n/nb faire
2. Si on possède un bloc de A_i alors
3. copier ce bloc dans buf_1 ;
4. envoyer buf_1 à tous les processeurs de sa ligne
5. FinSi
6. Si on possède un bloc de B_i alors
7. copier ce bloc dans buf_2 ;
8. envoyer buf_2 à tous les processeurs de sa colonne
9. FinSi
10. $C \leftarrow C + buf_1 \times buf_2$
11. FinPour

2.3 Analyse de l'algorithme

On se place sous les trois hypothèses suivantes :
– si plusieurs communications ont lieu en même temps, on a la même performance que lorsqu'elles ont lieu séparément ;
– envoyer un message à n processeurs a le même coût qu'envoyer un message de longueur n à 1 processeur ;
– les unités de calculs et la bande passante sont utilisées avec leur performance maximale.

On considère également le cas (plus simple) où $m = n = p$.

Temps des communications Pour un i donné, les processeurs qui mettent le plus de temps pour les communications sont les processeurs qui possèdent les diagonales des matrices, et qui doivent donc faire les deux envois. Le nombre de messages à envoyer est alors :

$$nmsg = 2(\sqrt{np} - 1)$$

Chaque message a pour taille :

$$msgsize = \frac{n}{\sqrt{np}} \times nb \times fpsize \text{ où } nb = \frac{n}{np}$$

On a donc :

$$t_{com} = \frac{nmsg \times msgsize}{bw}$$

Temps des calculs Pour un i donné, le nombre d'opérations virgule flottante est :

$$ops = 2 \times \frac{n}{\sqrt{np}} \times \frac{n}{\sqrt{np}} \times nb$$

On a donc :

$$t_{cal} = \frac{ops}{peak}$$

Temps total Pour un pas de calcul :

– si on ne superpose pas communications et calculs, $t = t_{com} + t_{cal}$;

– si on superpose communications et calculs, $t = \text{Max}\{t_{com}, t_{cal}\}$.

Il est assez simple, dans l'architecture que l'on a, de superposer communications et calculs : il suffit en effet de réaliser les communications par les PowerPC pendant que les unités de calcul calculent.

On obtient le modèle de la FIG. 3.

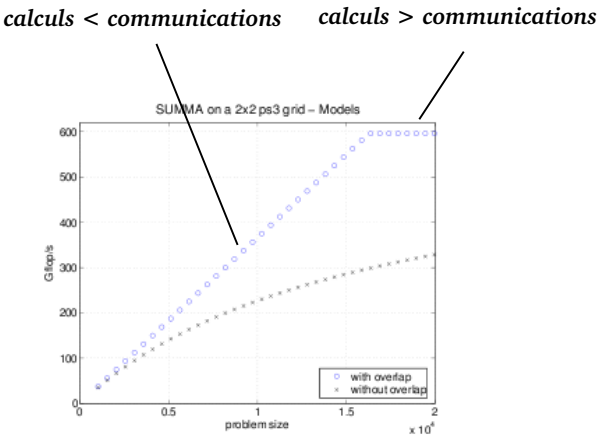

FIG. 3 – *Modèle*

On constate que le facteur d'accélération devient constant lorsque les calculs recouvrent les communications, c'est-à-dire lorsque la taille du problème est suffisamment grand : c'est ce qu'on appelle l'effet "surface-to-volume".

2.4 Résultats expérimentaux

On utilise le recouvrement des communications et des calculs, et on compare les performances lorsqu'on utilise une seule ou les six unités de calcul (voir FIG. 4).

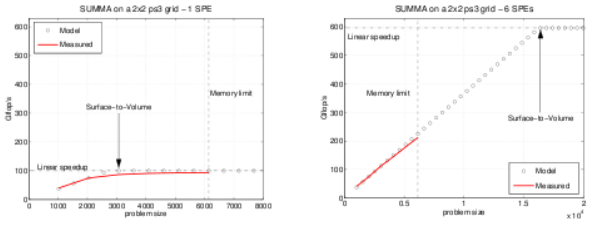

FIG. 4 – *Résultats expérimentaux*

On constate que la courbe est très proche du modèle, ce qui montre que celui-ci est plutôt bon. On utilise donc bien les composants avec leurs performances maximales.

Cependant, comme annoncé au paragraphe 1.2.3, les limitations mémoires des PS3 sont telles que l'on ne peut multiplier que des matrices relativement petites. Ainsi, on constate sur le deuxième graphe que l'on ne peut pas profiter du recouvrement des communications par les calculs.

3 Remarques

L'article est très détaillé, mais cependant quelques points importants manquent :

- certains choix ne sont pas justifiés, comme les choix du matériel pour les tests. En particulier, pourquoi se restreindre à une grille à quatre PS3 ? Il semble en effet que cela ne parallélise pas beaucoup...
- comme mentionné au paragraphe 1.3.2, le choix de la bibliothèque MPI finalement retenue n'est pas précisé, ce qui empêche de reproduire l'expérimentation dans les mêmes conditions ;
- il n'y a aucune comparaison avec une grille de calcul identique mais utilisant une autre architecture que la PS3 à chaque nœud. Comment savoir alors si cette architecture est performante ou pas, ou pour quels genres de problèmes il est intéressant de l'utiliser ? Je pense qu'une comparaison approfondie serait indispensable.

Conclusion Cet article est très intéressant car il fournit une excellente base pour des recherches futures qui seraient menées en utilisant une grille de calcul de PS3 (voir [3, 4]). Les personnes souhaitant s'en servir connaissent ainsi toutes les caractéristiques techniques, les méthodes de programmation à utiliser et les performances qui en découleront.

Cependant, pour qu'il soit vraiment utilisable, l'article devrait comporter une comparaison avec d'autres architectures, afin de savoir quelle architecture utiliser lorsqu'on a tel ou tel problème. On peut cependant penser que cet article très récent sera suivi de comparaisons et d'autres tests que le produit de matrices.

Références

[1] Playstation 3. `http://fr.playstation.com/ps3`.

[2] IBM : Cell broadband engine architecture, version 1.0, August 2005.

[3] J. Panetta, P. R.P. de Souza Filho, C. da Cunha Filho, F. Roxo da Motta, S. Sinedino Pinheiro, I. Pedrosa Junior, A. Romanelli Rosa, L. Monnerat, L. Carneiro et C. de Albrecht : Computational characteristics of production seismic migration and its performance on novel processor architectures. *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on*, pages 11–18, October 2007.

[4] D. Göddeke, R. Strzodka, J. Mohd-Yusof, P. McCormick, S. Buijssen, M. Grajewski et S. Turek : Exploring weak scalability for FEM calculations on a GPU-enhanced cluster. *Parallel Computing*, 33(10–11):685–699, September 2007.

[5] A. Buttari, J. Dongarra et J. Kurzak : Limitations of the playstation 3 for high performance cluster computing. *University of Tennessee Computer Science*, July 2007.