ComPAS'2014 : Parallélisme/ Architecture / Système
Neuchâtel, Suisse, du 23 au 25 avril 2014

Reproductibilité des expériences de l'article "Exécution en parallèle"

Djallal Rahmoune and David Parello and Bernard Goossens

Univ. Perpignan Via Domitia, Digits, Architectures et Logiciels Informatiques, F-66860, Perpignan.
Univ. Montpellier II, Laboratoire d'Informatique Robotique et de Microélectronique de Montpellier, UMR 5506, F-34095, Montpellier.
CNRS, Laboratoire d'Informatique Robotique et de Microélectronique de Montpellier, UMR 5506, F-34095, Montpellier.
prenom.nom@univ-perp.fr

Résumé
Nous décrivons les étapes qui permettent de reproduire les résultats présentés dans l'article compagnon *Exécution en parallèle*, des mêmes auteurs et soumis à Compas'2014.

---

1. Introduction.

L'article compagnon a trois figures correspondant à trois expériences dont nous souhaitons valider la reproductibilité. Les autres figures de l'article sont essentiellement faites "à la main", sans calculs sur machine.
La figure 1 (*L'ILP de 11 benchmarks de la suite PBBS issus d'algorithmes parallèles, appliqués à 8 jeux de données*), située en page 2, est l'histogrammes d'ILP (Instruction-Level Parallelism ou nombre d'instructions exécutées par cycle sur une machine idéale). Cet histogramme a été dessiné par *gnuplot* à partir de données issues du simulateur PerPI appliqué à la suite de *benchmarks* PBBS.
La figure 8 (*L'ILP de 11 benchmarks de la suite PBBS dans le modèle parallèle*), située en page 11, est un autre histogramme obtenu de la même façon. La figure 9 (*L'ILP des 11 benchmarks modifiés*), située en page 12, a été obtenue à partir d'une version modifiée des benchmarks de la suite PBBS.
Nous décrivons les principaux éléments de l'installation, tout étant accessible à l'URL suivant, dans la partie Realis09 :

```
http://perso.univ-perp.fr/david.parello/software/realis-2014/index.html
```

Pour reproduire les résultats de l'article, il faut disposer d'une machine x86_64. Un environnement linux est recommandé, en mode 64 bits (indispensable). Des détails supplémentaires sur certains logiciels sont ajoutés dans la suite (version de gcc, gnuplot ...).

2. Installer Pin.

Le logiciel Pin (outil gratuit Intel) est indispensable. Il est accessible à l'URL :

```
http://software.intel.com/en-us/articles/pintool-downloads
```

Il faut utiliser la version Linux 56759 du 20 janvier 2013.
L'installation de Pin se fait en deux commandes shell : (elle est expliquée dans le fichier téléchargeable *Release notes* ; l'installation décrite dans le manuel de l'utilisateur de Pin n'est plus à jour)

```
//On suppose que le dossier courant est celui où Pin est placé
$
$ cd source/tools/SimpleExamples
$ make dir obj-intel64/opcodemix.so
```

Selon la version de noyau Linux utilisée, il peut être nécessaire (c'est le cas avec les distributions Ubuntu) pour utiliser pin, de modifier une variable du système. Cela se fait ainsi :

```
$ echo 0 > zero
$ sudo cp zero /proc/sys/kernel/yama/ptrace_scope
```

On vérifie que l'installation de Pin est réussie en exécutant un exemple de Pintool qui écrit dans le fichier *opcodemix.out* le nombre d'instructions exécutées pour chaque *opcode*.

```
//On suppose que le dossier courant est
//pin-2.12-56759-gcc.4.4.7-linux (où Pin est placé).
//On effectue le test à partir de la commande /bin/ls appliquée
//au dossier courant.
$ ../../../pin -t obj-intel64/opcodemix.so -- /bin/ls
...
//Le fichier opcodemix.out contient le nombre d'instructions
//machines exécutées par /bin/ls, classées par catégories.
$ cat opcodemix.out
...
```

3. Installer PerPI.

Le logiciel PerPI est dans PerPI.tar.gz.
Après avoir déballé le dossier PerPI, il faut adapter le fichier makefile.inc en fixant les chemins d'accès à PerPI (PATH_TO_PERPI) et à Pin (PATH_TO_PIN).
La construction de PerPI se fait ainsi :

```
//On suppose que le dossier courant est PerPI
$ cd perpi_tools/ilptool-1.0
$ scons
```

Le dossier *obj-intel64* doit contenir quatre exécutables : ilptool-1.0_ilp_inscount, ilptool-1.0_ilp_nolimit, ilptool-1.0_ilp_seq, ilptool-1.0_ilp_speculative_fork.

4. Récupérer les *benchmarks* PBBS

Les *benchmarks* PBBS utilisés dans les expériences sont dans l'archive compressée PBBS.tar.gz. Cette archive contient également les jeux de données et les scripts pour construire les figures 1, 8 et 9.

4.1. Configurer l'installation

Le script bash *configurer* rend tous les scripts bash exécutables. Il doit lui-même être rendu exécutable :

```
//On suppose que le dossier courant est celui
//où est installé PBBS (où se trouve le script configurer)
$ chmod u+x configurer
$ ./configurer
```

4.2. Compilation des *benchmarks* PBBS et exécution de PerPI

Les mesures de PerPI sont basées sur le code machine exécuté. Le compilateur utilisé, la version, les options, le niveau d'optimisation impactent le code produit, donc le nombre d'instructions machines exécutées et leurs dépendances, c'est-à-dire le graphe d'ordonnancement partiel des exécutions. Dans les expériences rapportées dans l'article compagnon, les *benchmarks* ont été compilés en niveau d'optimisation -O3 pour la figure 1 et en niveau -O1, option -fno-inline pour les figures 8 et 9.
Pour la figure 9, certains *benchmarks* sont fournis en assembleur. Leur traduction nécessite une version récente de *as* (égale ou postérieure à la version 2.23.52).
Le script bash *faire* compile les benchmarks (en -O1 et en -O3) puis lance les exécutions de PerPI pour calculer tous les ILP (ceux reproduits par les trois figures). Après avoir adapté les chemins PIN_HOME et PerPI_HOME dans le fichier *faire*, on peut exécuter le script (l'exécution est longue : plusieurs heures) :

```
//On suppose que le dossier courant est celui
//où est installé PBBS (où se trouve le script faire)
//adapter les chemins PIN_HOME et PerPI_HOME
$ ./faire
```

L'exécution est terminée quand tous les ILP ont été produits, soit 264 (11 benchmarks, 8 jeux de données, pour trois figures : 3*8*11 = 264). On peut le vérifier ainsi :

```
//On suppose que le dossier courant est celui
//où est installé PBBS (où se trouve le script faire)
$ grep ILP */*/_ilp* | wc -l
//La valeur retournée doit être 264
//(attendre tant que la valeur est inférieure)
```

5. Obtention des figures

L'outil PerPI permet de simuler différents modèles de processeurs. Dans l'article compagnon, deux simulateurs sont utilisés : ilp_seq et ilp_speculative_fork. Le premier simule le comportement d'un cœur actuel disposant de ressources étendues (prédicteur de saut parfait, renommage illimité des registres, nombre illimité d'opérateurs, latence d'un cycle de toutes les opérations). Le second simule le modèle de processeur à beaucoup de cœurs que nous proposons dans l'article. La figure 1 vient d'ILP calculés par le simulateur ilp_seq. Les figures 8 et 9 viennent d'ILP calculés par le simulateur ilp_speculative_fork.
Le script bash *dessiner* produit les trois figures au format eps (la version de gnuplot doit être récente (4.6) et disposer du term eps).

```
//On suppose que le dossier courant est celui
//où est installé PBBS (où se trouve le script dessiner)
//adapter les chemins PIN_HOME et PerPI_HOME
$ ./dessiner
```

Les trois figures sont dans les fichiers figure_1_rahmoune.eps, figure_8_rahmoune.eps et figure_9_rahmoune.eps. Pour passer du format eps au format pdf on peut utiliser epstopdf :

```
//Pour convertir le fichier "fichier.eps" au format pdf
$ epstopdf --outfile=fichier.pdf fichier.eps
```

6. Conclusion

La figure 1 doit faire la démonstration que l'ILP de programmes parallèles, sur un processeur actuel, est inférieur à 10 (dans l'article, la figure montre qu'il est entre 4 et 7). Il varie peu avec la taille de la donnée.
La figure 8 doit montrer que l'ILP des mêmes programmes, sur un processeur appliquant le modèle d'exécution proposé, est élevé (>100 pour les données numéro 8) et surtout croît avec la taille de la donnée. Le benchmark numéro 11 fait exception.
La figure 9 montre que l'ILP des programmes modifiés (pour les adapter au parallélisme du modèle de calcul sans changer l'algorithme) est encore meilleur (>500 pour les données numéro 8).