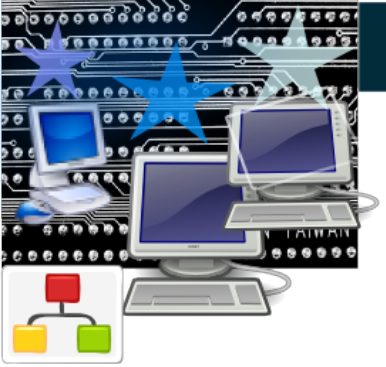


Vol de cycle et chemin critique, Outils pour l'émulation

Christophe Cérin[1]

[1]Université de Paris XIII, CNRS UMR 7030, France

Journée Mao du 3 avril 2008, Lyon

Résumé

ANR SAFESCALE

Contexte : vol de cycle et architectures

- Circuit multi-cœurs (hétérogènes = Cell processor) ;
- Carte multi-circuits (SMP IdKoiff à Grenoble) ;
- Parallélisme à grain fin ;
- Impact de la hiérarchie mémoire sur l'ordonnancement.

Contexte : vol de cycle et tri [Cérin, Traoré, Roch]

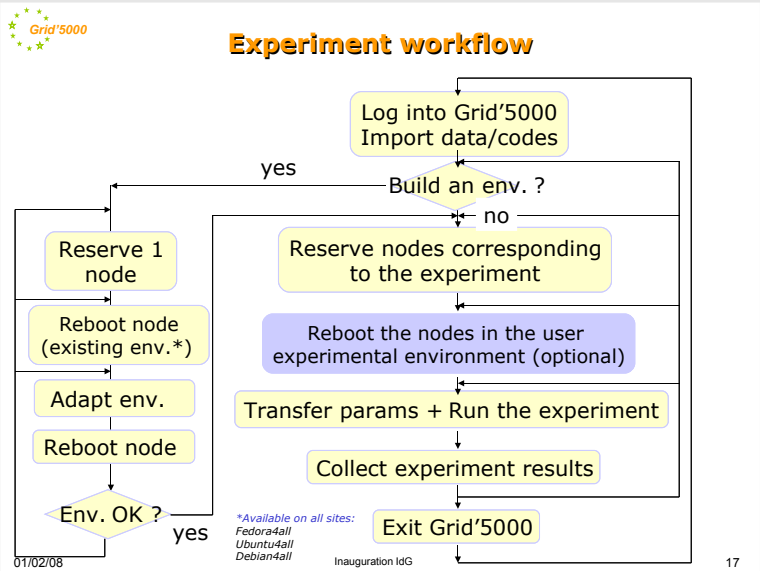
- Couplage d'un alg. parallèle à grain fin (ordonnancé par vol de cycle) et d'un algorithme séquentiel minimisant le nombre d'opérations (travail) ;
- Objectifs méthodologiques ;
- Garantie de performance : avec une grande probabilité,

$$T_p = \frac{W}{p\pi_{ave}} + \mathcal{O}\left(\frac{W_\infty}{\pi_{ave}}\right)$$

- $(W_\infty \ll W \wedge W \simeq W_s \rightarrow \text{optimal})$

Expérimenter : émulation

Grid'5000



Quelles conséquences vis à vis passage à l'échelle ?

- Émulation : expérimentation vraie grandeur
  sauf. . . recourt à un logiciel spécifique, dit émulateur, pour travailler
- Outils de recherche versus stabilité versus production versus exhaustivité des résultats expérimentaux
- Ressources humaines : compétences variées

Résultats connus en vol de cycle hétérogène

Bender et Rabin

- Ordonnancement par vol de cycle ;
- Connus en hétérogène : bornes sur le temps d'exécution (alg. online) :

$$\begin{aligned} T_p &\leq \frac{W_1}{p\pi_{ave}} + \left(\frac{\pi_2}{\pi_1} + \frac{\pi_3}{\pi_2} + \cdots + \frac{\pi_p}{\pi_{p-1}}\right)\frac{W_\infty}{p\pi_{ave}} \\ &\leq \frac{W_1}{p\pi_{ave}} + \frac{p-1}{p}\frac{W_\infty}{\pi_{ave}} \end{aligned}$$

Bender et Rabin : analyse des forces et faiblesses

Forces

- Résultat théorique non trivial
- Contexte hétérogène
- Contexte Online
- Contexte préemptif (quid du nb ?)
- Version selon l'invariant : "if there are $i < p$ ready threads, the fastest idle processor is at most β times faster than the slowest busy processor"

Faiblesses

- Constante π_{ave}
- Scheduler distribué $\Rightarrow$ difficulté d'intégration dans Linux (pas d'implémentation)
- Pas de retour sur les performances effectives (vis à vis du scheduler Linux actuel)
- Pas d'analyse de l'impact de la hiérarchie mémoire sur l'ordonnancement.

Bender, Rabin et chemin critique

Dans le même artcile...

- Suppose that the maximum utilisation strategy additionally maintains the invariant that the i-th fastest processor executes the thread that is i-th farthest from the end of the dag. This amounts to putting the fastest processor on the critical path. Then the computation has completion time :

$$T_p \leq \frac{W_1}{p\pi_{ave}} + \left[\frac{\pi_2}{\pi_1} + \frac{2\pi_3}{\pi_1+\pi_2} + \cdots + \frac{(p-1)\pi_p}{\pi_1+\cdots+\pi_{p-1}}\right]\frac{W_\infty}{p\pi_{ave}}$$

- Non constructif : dans le sens où les auteurs ne disent pas comment faire (quels data structures...) : moins raffiné que l'alg. précédent.

Gérer la mémoire

Scheduling Threads for Constructive Cache Sharing on CMPs

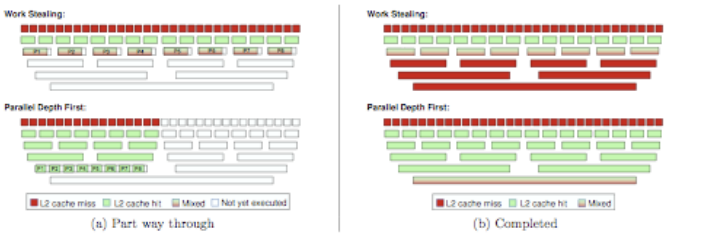


Figure 1: Scheduling parallel Mergesort using WS and PDF: Picturing the misses. Each horizontal box is a sorted array of records, where at each level, pairs of arrays from the previous level are merged until all the records are sorted. The L2 hits and misses are shown for sorting an array of C_P bytes, where C_P is the size of the shared L2 cache, using 8 cores.

Vol de cycle = Anarchie

- Il faut voler « à coté »
- Comment modéliser de la mémoire ?
- Quels algorithmes ?
- Quelles performances ?

Gérer la mémoire

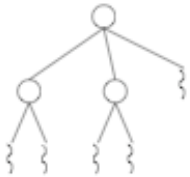
WS versus PDF

- PDF = when a core completes a task, it is assigned the ready-to-execute task that the sequential program would have executed the earliest.
- Note : ceci peut se faire online (Blelloch)
- Modèle de cache : shared + ideal (The ideal cache is fully associative and relies on the optimal replacement policy, i.e., the block that is accessed furthest in the future is replaced in case of a cache miss.)
- Exemple de résultat : avec PDF on peut asurer $Miss_{seq}$ avec un cache de taille au moins $C + P.D$. Alors que pour WS, il faut une taille de cache de $C.P$ pour garantir le même $Miss_{seq}$
- Blelloch : $C_p \leq C_1 + p.W_\infty \Rightarrow M_p \leq M_1$

BubbleSched (dans PM2/Marcel)

Samuel Thibault (Labri)

(a) Représentation graphique.



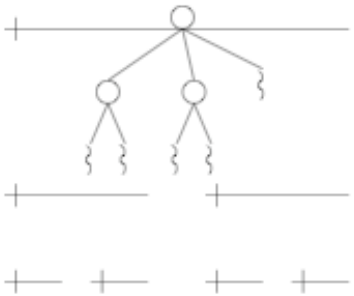
(b) Représentation arborescente.

```
marcel_t cpu_thread[4], comm_thread;
marcel_bubble_t bubbles[2], bubble;

marcel_bubble_insertthread(&bubbles[0],&thread[0]);
marcel_bubble_insertthread(&bubbles[0],&thread[1]);
marcel_bubble_insertthread(&bubbles[1],&thread[2]);
marcel_bubble_insertthread(&bubbles[1],&thread[3]);

marcel_bubble_insertbubble(&bubble,&bubbles[0]);
marcel_bubble_insertbubble(&bubble,&bubbles[1]);
marcel_bubble_insertthread(&bubble,&comm_thread);
```

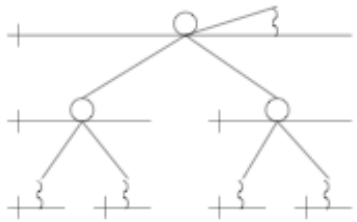
(c) Code source correspondant.

FIG. 1 – Expression des relations entre threads.

FIG. 2 – Modélisation d'un machine très hiérarchique.



(a) Bonne utilisation processeur, affinités non prises en compte.



(b) Utilisation distribuée des processeurs, affinités correctement prises en compte.

FIG. 3 – Distributions possibles des threads et bulles sur la machine

Résumé des transparents précédents

en hétérogène

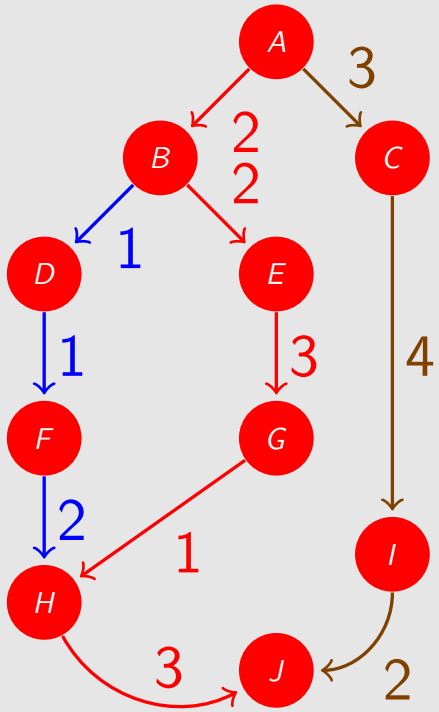
- Résultats théoriques non triviaux
- Contexte Online
- Implémentations non triviales à réaliser
- Pas de résultats intégrant des problématiques de gestion mémoire

en homogène

- Résultats théoriques
- Online
- Analyses de l'impact de la hiérarchie mémoire sur l'ordonnancement.
- Pas vraiment de mesures expérimentales très fines

Vol de cycles

Vol de cycle et chemin critique (mémoire, hétérogénéité)



$$T_p \leq \frac{\sum_{i=1}^{n} W_i + S_i}{\sum_{i=1}^{p} \pi_i}, \text{for } n > p;$$

Cérin & Koskas [2008]

In the permanent regime, a processor...

α : becomes idle because it has finished its 'job' on a path ;

β : must interrupt its work because it needs to wait for a task to complete.

The protocol is then as follows for the α, β cases :

α : the processor looks at the array, marks the 'finish ?' field with true and considers the first path which is either not yet started or, if it has been started by a slower processor, it steals the work. A processor which has been stolen by another one has the behaviour of a processor that is idle (its job is just terminated) ;

β : the processor interrupts its work on its path. Let N be the node requiring such synchronisation point. We consider the paths belonging to the parents of N and not yet terminated.

If the paths have not yet been started, we start one of them on the interrupted processor ;

notre approche : forces et faiblesses

Forces

- Résultat théorique sur T_p
- Contexte hétérogène
- Contexte préemptif (quid du nb de vols ?)
- Scheduler centralisé $\Rightarrow$ moins de difficulté d'intégration dans Linux

Faiblesses

- Plus de constante π_{ave} ... mais la borne est grossière
- Contexte Offline
- Pas de retour sur les performances effectives (vis à vis du scheduler Linux actuel)

- http ://www-lipn.univ-paris13.fr/~cerin/documents/

CPU burning

- À partir de l'idée de http ://wreka-voc.gforge.inria.fr/
- Fixe la charge d'un des cœurs pendant x secondes
- On doit être root (sched_setschedule)
- Quid des interactions cpufreq ?
- Question : workload de programmes multithreadés ?

PERFCTR (M. Pettersson)

- Nouveau driver $\Rightarrow$ nouvelle API
- Danger : *"The kernel driver restricts sampling_interval to 10000 usec or more, mainly to avoid excessive overheads. The limit is old and could perhaps be lowered nowadays. The sampling_interval is mapped to a timeout value to be used in a kernel timer : that timeout must be at least one kernel clock period in length, which limits it to at most 100/250/1000 HZ depending on the kernel's configuration"*

Vol de cycle et chemin critique, Outils pour l'émulation

Christophe Cérin[1]

[1]Université de Paris XIII, CNRS UMR 7030, France

Journée Mao du 3 avril 2008, Lyon