INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

About effective cache miss penalty on out-of-order superscalar processors

André Seznec, Fabien Lloansi

N° 2726

Novembre 1995

PROGRAMME 1

Rapport de recherche

ISSN 0249-6399

About effective cache miss penalty on out-of-order superscalar processors

André Seznec, Fabien Lloansi *

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet Caps

Rapport de recherche n°2726 — Novembre 1995 — 21 pages

Abstract: For many years, the performance of microprocessors has depended on the miss ratio of L1 caches. The whole processor would stall on a cache miss. The contribution of a cache miss to the execution time was exactly the miss penalty. Limiting the miss ratio on L1 caches has been a major issue for the last ten years. Studies showed that, for current cache sizes, 32 or 64 bytes cache blocks was a good tradeoff.

Today, technology has changed. Most of the newly announced processors implement a very complex superscalar microarchitecture allowing out-of-order execution. On these processors, instruction execution continues while L1 cache misses are serviced by a pipelined L2 cache.

In this paper, we show that, on such superscalar processors, the effective contribution of a cache miss to the execution time is quite distinct of the miss use penalty for the missing data or instruction. We also show that the L2 cache busy time becomes a major bottleneck and that decreasing the demanded throughput on this cache tends to become more important than limiting the L1 miss ratio. This favors the use of short cache block sizes. For current L1 cache sizes and a 16-byte bus, a 16 byte block size is shown to be a good trade-off.

Key-words: Out-of-order execution, Cache block size, effective miss penalty

(Résumé : tsvp)

This work was partially supported by CNRS (inter-PRC project ILIAD)

*seznec@irisa.fr,flloansi@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Á propos de la pénalité sur les défauts de cache sur les processeurs superscalaires éxécutant dans le désordre

Résumé : Dans cet article, nous montrons que, pour les processeurs exécutant les instructions dans le désordre, la contribution effective au temps d'exécution d'un défaut de cache est très différente du temps de service du défaut par le cache secondaire. Nous montrons aussi que l'occupation du cache secondaire devient le goulôt d'étrangement majeur pour les performances plus que le taux de défaut sur les caches. Ceci tend à faire préférer l'usage d'une taille de bloc petite, typiquement de la taille du bus reliant le cache secondaire au cache primaire.

1 Introduction

Many newly announced processors [7, 6, 5] implement a very complex superscalar microarchitecture allowing out-of-order execution. On these processor, instruction execution continues while L1 cache misses are serviced.

Another technological trend is the limited L1 cache size. Two major constraints limit the L1 cache size. First the access time on the cache increases with the cache size [12]. Second, multiple accesses on L1 cache are needed in a single cycle when the superscalar degree increases [13]. Therefore we believe that the size of the L1 caches will remain limited (in the 8-32Kbytes range). For these cache sizes, block sizes of 16 to 64 bytes lead to the best miss ratio [17].

On these processors, the L2 cache transactions are generally pipelined and many transactions may be pending at the same time. The L2 cache is a shared ressource which has to service instruction and data misses, but also writes (either write through requests or write back requests), and maybe also prefetches and external transactions for maintaining memory consistency.

In this paper, we explore the precise performance impact of pipelined L2 caches on the performance of out-of-order execution microprocessors. We show that, on such superscalar processor, the effective contribution of a cache miss to the execution time is quite distinct of the miss use penalty for the missing data or instruction.

A cycle-by-cycle simulation of an aggressive out-of-order execution microprocessor and of its cache hierarchy shows that the L2 cache busy time becomes a major bottleneck for performance. As a result decreasing the demanded throughput on the L2 cache tends to become more important than limiting the L1 miss ratio. This favors the use of short cache block sizes. For current L1 cache sizes and a 16-byte bus, a 16 byte block size is shown to be a good trade-off. Since the L2 cache busy time becomes the major issue, implementing a prefetch mechanism has to be cautiously studied: prefetching may load useless data or instructions, it may waste L2 cache bandwidth and decrease performance.

The remainder of the paper is organized as follows. In Section 2, we explain why the effective contribution of a miss to the execution time is very difficult to modelize on out-of-order execution microprocessor. In Section 3, we present different phenomena happening in the L2 cache. These phenomena tend to favor the use of small block sizes. In Section 4, we describe the processor simulator, the benchmarks and the simulated memory hierarchy. Performance results are presented and analyzed in Section 5. Section 6 summarizes this study.

Related work

Numerous studies have investigated cache behaviors and many hardware mechanisms have been proposed to limit the miss ratio and the impact of misses on performance. Fewer studies have been done on non-blocking caches (also called lock-up free caches [14]) and pipelined accesses toL2 cache.

Kroft [14] was the first one to consider a processor which does not stall on I-cache misses.

Sohi and Franklin [19] investigated the organisation of a multiport non-blocking L1 cache. They mainly focussed on the L1 organisation and showed that a L1 cache may be build with interleaved cache banks. However they did not completely simulate the superscalar processsor (no branch prediction, ..) and the whole memory hierarchy (instruction cache, write requests,..).

Farkas and Jouppi [2] explored the alternative implementations of non-blocking caches.

The usefulness of non-blocking data loads in multiple-issue processors has been studied by Farkas et al [3]. The study concludes that non-blocking loads and stream buffers are very efficient to enhance performance, nevertheless the model used for the external memory system (L2 cache or main memory) is very poor: contentions on the L2 cache are not simulated.

2 Effective miss penalty for on-chip L1 caches

Many newly announced processors (e.g. HP8000, Intel P6, MIPS R10000) implement a very complex superscalar microarchitecture featuring out-of-order execution. Instruction execution continues while L1 cache misses are serviced by a pipelined L2 cache. This L2 cache may be off-chip (e.g. MIPS R10000 or Intel P6) or on-chip (e.g DEC 21164).

We explain here why the delay needed for servicing a miss on such out-of-order execution processors do not necessarily result in a similar loss in execution time.

2.1 Miss penalty for synchronous blocking caches

On most microprocessors introduced before 1994, a miss on the instruction or data cache was resulting in a complete stall of the processor pipeline.

On these processors, the penalty paid on execution time for a L1 cache miss is quite simple to modelize. The time for servicing a miss consists in a latency L for accessing the first word from the missing block plus an extra penalty r paid for each extra word in the block. Let K be the number of words in a cache block, Formula 1 represents the time for servicing the miss.

$$L \; + \; (K-1)*r \quad cycles \tag{1}$$

L includes the delay for checking the miss, driving the address pins for accessing the external L2 cache (or main memory), reading the L2 cache and getting the data in the processor, and resume the execution. For instance, on 60 Mhz Pentium systems, minimum L is 5 cycles and minimum r is one cycle.

In order to limit the penalty on execution time, in many implementations, the missing word is returned first. The execution can then be resumed as soon as the missing word is present, L cycles after the beginning of the miss servicing. In this case, for each individual miss, the lost execution time is at minimum L cycles; the average penalty on execution time is slightly higher than L cycles, because a cache miss servicing may be delayed by the end of a previous cache miss servicing or a write update.

Notation: For convenience, in the remainder of the paper, r the access time of the L2 cache RAMs will be called the L2-cycle.

2.2 Miss penalty versus use penalty on out-of-order execution microprocessors

On recently announced microprocessors [7, 5, 6], aggressive out-of-order execution is implemented and/or pipelined L2 caches are used. We explain here, why the execution time on these processors is merely less affected by each individual cache miss than in traditional synchronous processors.

First let us define the two different notions, the effectively miss penalty and the use penalty:

Definition 2.1 *The use penalty is the delay between the reference to data (or instruction) in the L1 cache and the time the data (or instruction) is available for use.*

On a miss, there is a minimum use penalty L for using the data.

Definition 2.2 *The effective miss penalty is the real contribution to the execution time of a L1 cache miss.*

On a out-of-order execution microprocessors, the effective miss penalty associated with a specific cache miss depends on its whole context: an isolated cache miss may be traded by the out-of-order execution, while a rapid succession of cache misses will lead to some processor stalls. In the remainder of the paper, we will use the average effective miss penalty.

When the miss results in a complete stall of the processor, the use penalty and the miss penalty are equal. But the average effective miss penalty may be far less than the use penalty when out-of-order execution is used.

Data caches On an out-of-order execution microprocessor, a data cache miss does not stall the whole processor. On a load miss at cycle T, dependent subsequent instructions cannot be executed before cycle $T + Use\_Penalty$, while independent instructions continue to progress in the pipeline. Subsequent load/store instructions may even be executed when a lock-up free (or non-blocking) cache is implemented [14]. On recent microprocessors, several cache misses may be pending while the execution of load/store instructions continue.

When the functional units in the processor find a sufficient number of independent instructions to execute, the use penalty is hidden. Aggressive compiler technology may schedule in advance load instructions which are suspected to miss ([2, 1]), thus allowing to farther reduce the miss penalty.

Moreover, on these microprocessors, accesses to the L2 cache are pipelined and more than one miss may be serviced at the same time resulting in an average effective miss penalty significantly lower than the use penalty.

Instruction cache On out-of-order execution superscalar microprocessors, instructions are generally read at a high rate. Due to data dependencies and resource conflicts, the execution units cannot sustain the same execution rate. Therefore some advance on instruction fetching may be obtained, this is particularly true when a sequence without instruction cache miss or branch misprediction is encountered. While servicing an instruction miss, instructions already issued may be selected for execution. In some cases, if enough instructions were issued in advance, an instruction miss would have no impact on the execution time.

Instruction accesses may also benefit from a pipelined L2 cache. Instructions generally exhibits high spatial locality. In order to limit instruction miss penalty, systematic prefetching of the fall-through blocks after an instruction cache miss may be used as on the DEC 21164. This technique may allow to see a miss penalty only the first miss when the L2-cycle and the processor cycle are equal.

2.3 Bus width is limited

A solution for limiting the miss penalty might consist in using a very wide bus (for example the width of a cache block). Unfortunately technological and economical factors limits this width. When the L2 cache is on-chip (as for the DEC 21164), the power consumption is a major limitation; the wider the bus will be, the higher be the power consumption. When the L2 cache is off-chip, pin-out is a serious limitation, the minimum cost of the L2 cache is also a major concern.

On current processor generation, using a 16-byte bus width is the dominant trend.

3 Out-of-order execution favors small cache blocks

In this section, we present some phenomena which explains why the optimal block size is smaller for out-of-order execution microprocessors than for synchronous microprocessors.

3.1 Synchronous blocking caches

When using synchronous blocking caches, for cache sizes in the 8K-32Kbyte range, and for a realistic latency L (5-30 cycles) and memory (or L2 cache) access time r (1-3 cycles), the optimal size of the cache block between 32 and 64 bytes [17, 8].

3.2 Cache line size and out-of-order execution

On recently announced microprocessors [7, 5, 6], aggressive out-of-order execution is implemented and/or pipelined L2 caches are used. We explain here, why this tends to favor the use of small cache blocks.

High throughput L2 caches Specific L2 caches associated with the newly announced processors [7, 18, 5] have a very high bandwidth: 16 bytes per cycle on DEC 21164 or the MIPS R10000 or 8 bytes per cycle on Intel P6 for instance.

The access time to the L2 cache is also very short: on a data cache miss, the load data may be used 5 cycles after the load reference on the DEC 21164 and 6 cycles after on the Intel P6 or MIPS R10000.

Data caches When accessess to the L2 cache are pipelined and more than one miss may be serviced at a time resulting in an average visible penalty significantly lower than the time really spent for servicing the miss.

Figure 1 illustrates a situation where the use of a small cache line leads to a lower execution time. In this example, three loads are considered: load A and load B result in L1 misses and load C results in a L1 hit[1]. Two cases are envisaged, (a) a cache line width equal to the bus width and (b) a cache line width four times larger than the bus width.

In this simple example, we have assumed that a single access port is implemented on the L1 cache. More complex L1 cache designs may be considered: multiple cache accesses by the processor, dedicated Write port is used for updating the cache on a miss, ...

---

[1] In a real program, the sequences of misses would be different when using different blocks sizes . This example is only given to illustrate phenomena which can decrease performance when using large block size.