# A Mathematical Model for the Transitional Region Between Cache Hierarchy Levels

Michael Krietemeyer, Daniel Versick, and Djamshid Tavangarian

Chair of Computer Architecture, Department of Computer Science
University of Rostock, Albert-Einstein-Str. 21, D-18059 Rostock, Germany
{michael.krietemeyer,daniel.versick,
djamshid.tavangarian}@informatik.uni-rostock.de

**Abstract.** The knowledge of internal structures of the cache-memory hierarchy and its performance is very important in modern computer systems. Therefor, this paper introduces a mathematical model that describes the transition between Level 1 and Level 2 cache of current processors. The theoretical predictions are proved by measurements for two Intel CPUs and an UltraSparc II system.

## 1 Introduction

As the performance gap between CPU and I/O devices increases rapidly in modern computer systems the understanding of I/O systems is of great interest. In the context of the IPACS project that develops benchmarks for distributed systems a mathematical model was introduced that characterizes some performance issues within the memory subsystem.

Modern computer systems have a hierarchical architecture of storage called the memory hierarchy. Each level of the hierarchy is slower and of larger size than higher levels. Typically the memory hierarchy consists of CPU registers, Level 1 cache, Level 2 cache, main memory and disk storage. Often referenced memory can be stored in the fastest possible hierarchy level to increase the performance of memory access, especially in cache and main memory. This technique can strongly improve the application performance. However it complicates the exact understanding of the processes that influence the performance of memory accesses.

This paper describes a mathematical model that helps to understand the performance within the cache hierarchy levels. It focuses on the performance description of the Level 1 and Level 2 cache. Especially, it describes the performance of accesses to the transitional region between that two cache hierarchy levels. While Section 2 presents some former work Section 3 discusses the measuring method and the mathematical model for the memory performance. Section 4 presents some measured results of three chosen example architectures before Section 5 concludes the paper.

## 2 Related Work

Most of the related works that try to characterise the performance of cache memories, describe methods for implementing memory benchmarks and present measurements for different architectures. Mathematical models that establish a theoretical background for the effects are rare. Saavedra describes in [1] and [2] the characteristics of cache systems. Cache parameters are measured by varying the stride size of memory accesses. As the papers focus on the performance in dependency of the stride size of memory accesses, conclusions about the effective memory bandwidth in different operating ranges are missing. In addition both papers do not give any conclusions about the performance in transition regions between memory hierarchy levels.

Hockney discovers in [3] the phenomenon of a very low bandwidth in the transitional region between cache and main memory. He does not explain it in detail and refers to [4] where the cache memory effect is explained by means of a mathematical model. The model is based on Hockneys linear timing model $(r_\infty, n_{1/2})$ described in [5]. The mathematical model described in [4] uses a very abstract model that does not take cache specifics like associativity into account. As modern computer systems widely use set-associative caches with *least recently used* as replacement strategy, our approach for characterising the cache memory effect will adhere to that. It will introduce a model that can easily be used in most modern computers to predict the performance of consecutive memory accesses for a better optimisation of program behaviour.

Hristea [6] describes some benchmark kernels written in C that measure different performance parameters of the memory hierarchy. The advantage of the suggested benchmarks is that they are completely hardware-independent. To prove the introduced mathematical model a benchmark was written that is based on a benchmark kernel from [6]. Based on that kernel we measured cache memory characteristics on some example architectures to compare it with the expected values acquired by the introduces mathematical model. The comparison between theoretical and measured results gives us the potentials of the suggested theoretical model.

## 3 Measuring Method and Mathematical Model

This section describes with priority the mathematical model and its theoretical fundamentals.

### 3.1 Measuring Method

The mathematical model characterizes the performance of reading different block sizes of consecutive memory. Additionally, it allows to calculate the associativity of caches from bandwidths in the several hierarchy levels. A program measures the time for reading memory blocks with an increasing size. The acquired times for the different vector sizes are used to determine the different bandwidths. The main part of the program is a slightly modified *back-to-back memory latency benchmark kernel*, as described in [6].

To minimize disturbing effects, like task switching or memory swapping within the benchmark kernel, and to make sure that all cacheable memory is located inside the caches, the code for traversing the pointer-list is performed 100 times. The program only reports the best time. This guarantees reliable and reproducible values.

In the measured values some straight lines can be identified. Each line represents a memory hierarchy level or a transition between two levels. As described in [3] the asymptotic bandwidth for the sections can be calculated as:

$$B_i = \frac{1}{t_i} \tag{1}$$

$$B_i - Bandwidth\ in\ region\ of\ line\ i$$

$$t_i - slope\ of\ straight\ line\ i$$

For memory vectors that fit into one cache hierarchy level, the time for reading the memory-block increases linear with its length. By increasing the vector size over the size of a cache, the reading time rises faster than in the last level. This is caused by the inevitable cache misses when reading the linear memory area. The change from one hierarchy level to the next is not a simply degradation of the reading time. Because of the cache organization, there is a short transition area between two levels where the reading time per byte rises faster than in both – the faster and in the slower hierarchy level. These effects and their reasons are described in the next section.

## 3.2 Set Associative Caches

In order to understand the arising effects within the transitions between two memory hierarchy levels, it is important to understand the function and organization of cache memory. There are three forms of cache organization:

- direct mapped (1-way set associative),
- n-way set associative, and
- fully associative caches.

The fact that the cache memory is divided into lines with a specific length of e.g. 16 bytes is applicable to all three forms. These cache lines are organized in sets. For n-way set associative caches a set is containing n lines. In the case of a fully associative cache there is only one set containing all cache lines. Each memory block can only be stored in one cache set. *Least recently used* is a commonly used replacement strategy for loading a further memory block if all cache lines in a set are engaged (see [7] for explanation of LRU).

The functionality of the three cache types is explained in [8].

An example for a 2-way set associative cache will be discussed in order to understand the transition effects between two hierarchy levels. Figure 1 shows such a cache. In order to make this example simpler, the block size is exactly the

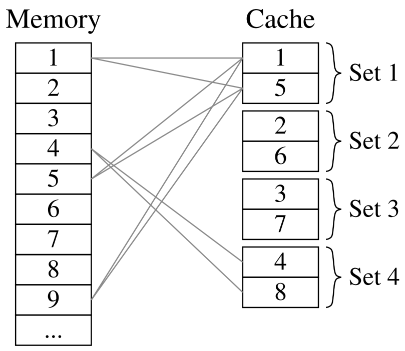

**Fig. 1.** 2-way set associative cache example

size of a vector element. Additionally Figure 1 shows the contents of the cache memory after reading a vector with a size of eight. As this is the size of the cache, no cache misses occur under the condition that a read is performed more than once. If the size is increased to nine, more cache misses will occur and it will be necessary to replace blocks inside the cache. For this purpose the LRU algorithm is used.

When reading a vector with nine elements, cache misses arise only in the first cache set. For a pointer-list with ten elements, misses appear in the first and second set, and so on. Figure 2 shows the cache content after reading areas with a size of nine to thirteen elements (every range was read twice). The more interesting part in the figure is the number of occurred cache misses. The misses only arose in the gray colored cache sets.

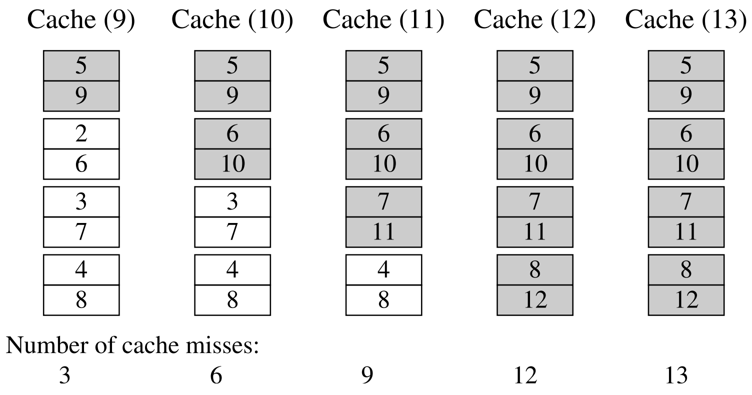

**Fig. 2.** Occurred number of cache misses for rising vector sizes

The transition effects can be explained with these values. For vector sizes between nine and twelve elements the number of cache misses rises by three with every step, against the rising by one for greater vector sizes. The generic equation for the number of cache misses in the transition area is:

$$m(n) = \frac{n - c}{c_L} \cdot (a + 1) \qquad \text{if} \quad n > c \tag{2}$$

$$\begin{aligned} m(n) &- \textit{number of cache misses} \\ n &- \textit{number of read bytes} \\ c &- \textit{cache size of faster memory}[\textit{bytes}] \\ c_L &- \textit{cache line size of faster memory}[\textit{bytes}] \\ a &- \textit{associativity of faster memory} \end{aligned}$$

The first part is the number of cache lines of the vector that exceeded the size of the cache. This number of lines must be reloaded $a + 1$ times when reading the vector. Increasing the associativity by one is necessary because the first cache line has to be loaded twice. The size of the transition area depends on the associativity of the cache:

$$size_t = \frac{\frac{c}{c_L}}{a} \cdot c_L = \frac{c}{a} \tag{3}$$

$$size_t - \textit{size of the transition area}$$

*Summary:* For vectors that fit into the maximum number of cache lines of the faster hierarchy level, the number of cache misses is zero (the vector size is less or equal to the cache size). Vectors that are greater than the cache, need a reload of memory blocks from the slower level into the assigned cache lines. For vector lengths that do not require reloads in all cache sets, the number of misses, and therefore the reading time rises faster than for vectors that need reloads in all sets. In the first case the vector size is between $c$ and $c + size_t$.

## 3.3 Mathematical Model

The following section explains the mathematical model that describes the memory access performance in the L1 and L2 cache of uniprocessor systems with multiple cache hierarchy levels. Additionally we present a method to calculate the cache associativity from the bandwidths for n-way set associative caches.

The time of every memory access to $n$ consecutive bytes consists of the time that is necessary to read $m(n)$ bytes from the slower memory hierarchy level and the time that is needed to read $n - m(n)$ bytes from the faster level:

$$t(n) = t_s \cdot m(n) + t_f \cdot (n - m(n)) \tag{4}$$

$$\begin{aligned} t_f &- \textit{access time per byte in faster memory hierarchy level} \\ t_s &- \textit{access time per byte in slower memory hierarchy level} \end{aligned}$$

**The Fast Hierarchy Level ($n \leq c$):** In case the number of read bytes is smaller than the size of the faster memory no cache misses ($m(n) = 0$) will occur. Applying this in conjunction with Equation (4) results in:

$$\begin{aligned} t(n) &= t_f \cdot n \qquad \text{if} \qquad n \leq c \\ t_1 &= t_f \end{aligned} \tag{5}$$

Thus the measurable slope of the straight line in the fast hierarchy level is equal to the access time per byte in that level.

**The Slow Hierarchy Level ($n > c + \frac{c}{a}$):** Consecutive memory accesses in the slower memory level are more complicated, as the number of cache misses in the slower level depends on the number of read bytes and the cache line size in the faster level. As stated in Section 3 an access to a new memory block with the size of one cache line causes exactly one cache miss in the slower hierarchy level. Other accesses to this memory block can be served by the faster memory. Therefor, the number of cache misses $m(n)$ is the quotient of the number of accesses and the cache line size. Referring to Equation (4) this leads to:

$$\begin{aligned} m(n) &= \frac{n}{c_L} \qquad \text{if} \quad n > c + \frac{c}{a} \\ t(n) &= t_s \cdot \frac{n}{c_L} + t_f \cdot \left( n - \frac{n}{c_L} \right) \\ &= \left( \frac{t_s - t_f}{c_L} + t_f \right) \cdot n \\ &= t_3 \cdot n \end{aligned}$$

Thus one can see there also is a linear interrelation between the read time and the number of read bytes in the slower memory hierarchy level. The slope of that function is the measurable slope $t_3$. It is possible to calculate the real access time to the slower memory $t_s$ using this value:

$$\begin{aligned} t_3 &= \frac{t_s - t_f}{c_L} + t_f \\ t_s &= \left( t_3 + \frac{t_f}{c_L} - t_f \right) \cdot c_L \\ &= t_3 \cdot c_L - t_f \cdot (c_L - 1) \end{aligned}$$

Hence, the access time $t_s$ can be seen as a complex dependency between the measurable slopes in the memory hierarchy levels and the cache line size in the faster hierarchy level.

**The Transitional Region ($c < n \leq c + \frac{c}{a}$):** The former acquired equations are used to determine the interrelations in the transition area. Therefor, the calculated number of cache misses in the transitional region from Equation (2) can be used in Equation (4):

$$t(n) = \left( t_3 + \frac{t_f}{c_L} - t_f \right) \cdot c_L \cdot \frac{n - c}{c_L} \cdot (a + 1)$$

$$
\begin{aligned}
&+t_f \cdot \left(n - \frac{n-c}{c_L}(a+1)\right) \\
&= ((t_3 - t_f) \cdot a + t_3) \cdot n + (t_f - t_3) \cdot (ac + c) \\
&\quad \text{if} \quad c < n \leq c + \frac{c}{a}
\end{aligned}
$$

Thus a linear dependency between $n$ and the time $t(n)$ is shown. The slope $t_2$ of this straight line is of great interest because it can easily be measured and depends on the associativity of the faster memory hierarchy:

$$
\begin{aligned}
t_2 &= (t_3 - t_f) \cdot a + t_3 \\
a &= \frac{t_2 - t_3}{t_3 - t_f} = \frac{t_2 - t_3}{t_3 - t_1} \qquad \text{because } t_f = t_1 \text{ from (5)}
\end{aligned}
$$

The associativity of the faster memory hierarchy level can now be calculated from the measurable slopes of the three regions. The following bandwidth equation is equivalent to the last one:

$$a = \frac{(B_3 - B_2) \cdot B_1}{(B_1 - B_3) \cdot B_2} \tag{6}$$

$$\Longrightarrow B_1 > B_3 > B_2 \quad \text{for} \quad a > 0$$

As one can see in this equation, the bandwidth in the transition area must be smaller than the bandwidths in both memory hierarchy levels to get a positive result for the associativity.

Comprising the following equation describes the reading time for consecutive bytes from the two cache hierarchy levels and their transitional region:

$$t(n) = \begin{cases} t_1 \cdot n & \text{if} \quad n \leq c \\ ((t_3 - t_1) \cdot a + t_3) \cdot n + (t_1 - t_3) \cdot (ac + c) & \text{if} \quad c < n \leq c + \frac{c}{a} \\ t_3 \cdot n & \text{if} \quad n > c + \frac{c}{a} \end{cases} \tag{7}$$

## 4 Experimental Results

To verify the model described in Section 3, some measurements were done on three different example architectures.

The first example architecture is an Intel Xeon 2.4 GHz with a L1 cache size of 8 kB. The 1st level cache on this architecture is organized 4-way set associative.

The used Pentium III system has a clock frequency of 1.4 GHz and cache sizes of 16 kB for the 1st level data cache and 512 kB for the 2nd level cache. The 1st level cache of this CPU is organized 4-way set associative and the 2nd level cache 8-way set associative. More detailed information about the architecture can be found in [9].

The third architecture used for these measurements was a Sun UltraSPARC-II with 300 MHz clock frequency. This CPU has a direct mapped 1st level cache of size 16 kB. Some more information can be found in [10].

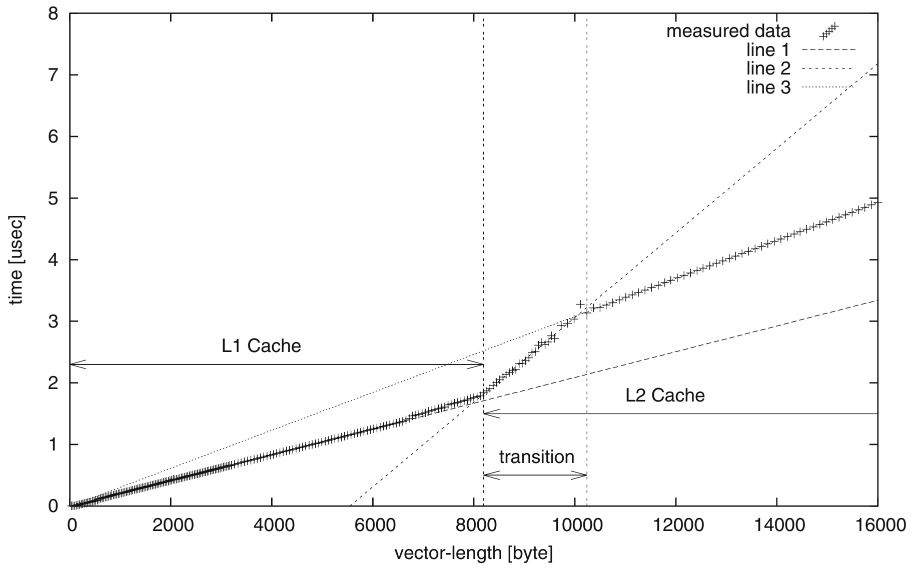

**Fig. 3.** Reading time for different vector lengths up to 16 kB on Intel Xeon

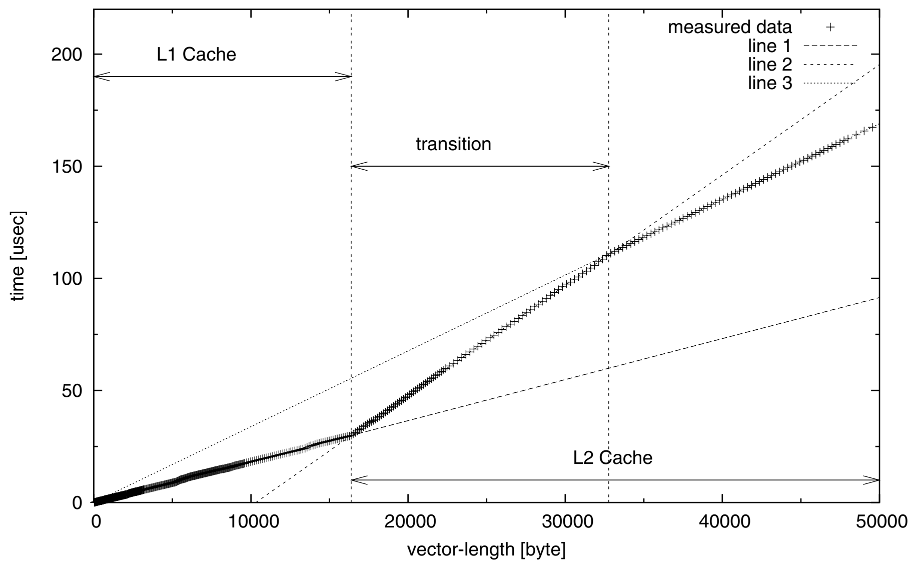

**Fig. 4.** Reading time for different vector lengths up to 50 kB on SPARC

In all cases the time for reading a number of $n$ consecutive bytes is measured. Figures 3 and 4 show the reproducible results for the Xeon and SPARC CPU. The results for the Pentium III CPU are only given below in Table 1 since the

**Table 1.** Asymptotic bandwidths on example architectures and associativities calculated by means of mathematical model

| region | straight line $[\mu s]$ | bandwidth $[MB/s]$ |
|---|---|---|
| | Intel Xeon | |
| 1st level cache | $t(n) = 5.44 \cdot 10^{-4} \cdot n - 3.18 \cdot 10^{-3}$ | 4561.28 |
| transition zone | $t(n) = 1.07 \cdot 10^{-3} \cdot n - 3.81$ | 1387.94 |
| 2nd level cache | $t(n) = 6.52 \cdot 10^{-4} \cdot n - 2.89 \cdot 10^{-3}$ | 3095.04 |
| associativity | calculated: 3.83 expected: 4 | |
| | Pentium III | |
| 1st level cache | $t(n) = 2.09 \cdot 10^{-4} \cdot n + 7.22 \cdot 10^{-3}$ | 1752.74 |
| transition zone | $t(n) = 6.87 \cdot 10^{-4} \cdot n - 8.62$ | 888.88 |
| 2nd level cache | $t(n) = 3.08 \cdot 10^{-4} \cdot n + 1.00 \cdot 10^{-2}$ | 1463.15 |
| associativity | calculated: 3.91 expected: 4 | |
| | SPARC | |
| 1st level cache | $t(n) = 1.82 \cdot 10^{-3} \cdot n + 6.39 \cdot 10^{-2}$ | 521.28 |
| transition zone | $t(n) = 4.92 \cdot 10^{-3} \cdot n - 50.77$ | 193.73 |
| 2nd level cache | $t(n) = 3.38 \cdot 10^{-3} \cdot n + 5.37 \cdot 10^{-2}$ | 282.22 |
| associativity | calculated: 0.996 expected: 1 | |

results look much like the ones obtained from the Xeon system. One can see that every curve consists of three straight lines. The first line is equivalent to the region of the 1st level cache. The 2nd line is the transition zone between the first and the 2nd level cache and the 3rd line is equivalent to the 2nd level cache. It can be seen that the slope of the curve in the transition region is greater than in the other regions. Since the reciprocal value of the slope is the asymptotic bandwidth $B_i$ in that region, it is even less than in the 2nd level cache. It should be a design goal to minimize this bandwidth breaking in the transitional region.

Table 1 shows the cache bandwidths on the example architectures. They were calculated from the slopes of the three curves that are determined by means of least-square fits. The shown figures also demonstrate that the transition region on the Xeon architecture is smaller than on the SPARC architecture. Equation (3) points out that this length is reciprocally proportional to the associativity of the first level cache. The smaller the associativity of the cache, the larger the transition region. Since the SPARC architecture has a direct mapped 1st level cache, the transition region is as large as the 1st level cache itself.

Table 1 also shows the associativities of the first level caches calculated by means of Equation (6). It is shown that the calculated values for the associativity are approximately the expected values described in the architectural manuals of the three CPUs. Xeon and Pentium III have a 4-way set associative 1st level cache and SPARC a direct mapped 1st level cache. The bandwidths in the transition regions on the architectures are interesting. On the Pentium III the bandwidth in the transition area between the 1st and 2nd level cache is about 50 percent smaller than in the 1st level cache and about 40 percent smaller as in the 2nd level cache. On SPARC the bandwidth in that area is about 62 percent smaller than in the 1st level cache but only 30 percent smaller as in the 2nd level cache. On

Xeon systems the difference is even larger than on SPARC. There the bandwidth in the transitional region is about 70 percent smaller than the bandwidth in the 1st level cache and about 55 percent smaller than in the 2nd level cache.

The calculation of the axis intercepts of the lines that describe the transitional regions by means of the mathematical model also resulted in the measured values. As shown in Equation (7) the axis intercepts of the lines in the 1st and 2nd level caches are zero. Approximately this applies to the measured lines. Only the straight line of the transitional region has a negative axis intercept as the mathematical model predicts.

## 5 Conclusion and Further Work

This paper introduced a mathematical model that describes the transition between two cache hierarchy levels in widespread memory architectures that use set-associative caches and LRU or FIFO as replacement algorithms within the sets. The model allows the calculation of the associativity of the L1 cache and the performance prediction of consecutive memory accesses within the transitional region between two cache level hierarchies in case that the bandwidths in the hierarchy levels are known. The model was proved by means of measurements on three hardware architectures with different memory system designs. It was possible to show that the presented model creates reasonable results.

The sample measurements of Section 4 were acquired on UNIX like operating systems (Solaris 5.8 for the SPARC system and Linux 2.4.x on the Intel systems). One problem of these operating systems is that it is impossible to get large blocks of consecutive memory in user space. Hence, at the moment it is only possible to prove that the introduced model describes the performance behaviour of the first and second level cache. Further work will include the extension of the model to other memory hierarchy levels since it should also be applicable to the transitional region between cache and main memory. Besides performance measurements will be done on more than the current three architectures.

## Acknowledgement

This research was sponsored by the German Federal Ministry of Education and Research grant 01IRB03E in the context of the IPACS project [11]. We thank the sponsors and the project partners Fraunhofer Institute for Techno- and Business Mathematics, T-Systems, National Energy Research Scientific Computing Center and the University of Mannheim for their support.

## References

1. Saavedra, R.H., Gaines, R.S., Carlton, M.J.: Characterizing the performance space of shared memory computers using micro-benchmarks. In: Proceedings of Hot Interconnects. (1993) 3.3.1–3.3.5

2. Saavedra, R.H., Smith, A.J.: Measuring Cache and TLB Performance and Their Effect on Benchmark Runtimes. IEEE Transactions on Computers **44** (1995) 1223–1235
3. Hockney, R.W.: The Science of Computer Benchmarking. Society for Industrial and Applied Mathematics, Philadelphia (1996)
4. Getov, V.S.: Performance characterisation of the cache memory effect. Supercomputer **11** (1995) 31–49
5. Hockney, R.W.: Performance parameters and benchmarking of supercomputers. In Dongarra, J.J., Gentzsch, W., eds.: Computer benchmarks, Amsterdam, Elsevier Science Publishers B.V. (1993) 41–63
6. Hristea, C., Lenoski, D., Keen, J.: Measuring memory hierarchy performance of cache-coherent multiprocessors using micro benchmarks. In: Proceedings of the ACM/IEEE SC97 Conference (SC'97). (1997) 1–12
7. Belady, L.A.: A study of replacement algorithms for a virtual-storage computer. IBM Systems Journal **5** (1966) 78–101
8. Smith, A.J.: Cache memories. ACM Comput. Surv. **14** (1982) 473–530
9. Intel Corporation: Pentium Processor Family, Developer's Manual – Volume 3: Architecture and Programming Manual (1996)
10. SPARC International Inc.: The SPARC Architecture Manual, Version 9 (2000)
11. IPACS: IPACS Benchmark. http://www.ipacs-benchmark.org (2004)