IEICE Electronics Express, Vol.9, No.18, 1461–1466

Analytical memory bandwidth model for many-core processor based systems

Hyuk-Jun Lee[1], **Woo-Cheol Cho**[1], **and Eui-Young Chung**[2a)]

[1] *Dept. of Computer Science and Engineering, Sogang University*
#1, Sinsu-dong, Mapo-gu, Seoul 121–742, Korea

[2] *School of Electrical and Electronic Engineering, Yonsei University*
134 Sinchon-dong, Seodaemun-gu, Seoul 120–749, Korea

a) *eychung@yonsei.ac.kr*

Abstract: Many-core processor based systems gain popularity in high-performance parallel embedded applications. Estimating memory bandwidth requirement, i.e. external memory bandwidth, given various cache size for target parallel applications requires a prohibitively large simulation time. In this work, we propose an analytical model to quickly estimate the memory bandwidth for a given cache size and help exploring trade-offs between cache sizes and memory bandwidth requirement. We model the stochastic behavior of cache misses for a single cache as a random process. Using central limit theorems for identically or non-identically distributed random processes, we accurately estimate the collective cache misses from hundreds of processor cores and thus the total memory bandwidth requirement for the whole system. The results show that our model improves a speed of simulation time up to 200.4 times for 200 cores whereas its estimated results achieve less than 0.01% difference from the simulated ones for 200 cores in terms of accuracy.

Keywords: many-core, cache miss, memory bandwidth, central limit theorem

Classification: Electron devices, circuits, and systems

References

[1] J. Manferdelli, N. Govindaraju, and C. Crall, "Challenges and oppertunities in many-core computing," *Proc. IEEE*, vol. 96, no. 5, May 2008.
[2] [Online] http://www.cisco.com/en/US/prod/collateral/routers/ps9343/solution_overview_c22-448936.html
[3] C. Yu and P. Petrov, "Off-chip memory bandwidth minimization through cache partitioning for multi-core platforms," *Design Automation Conference (DAC)*, pp. 132–137, June 2010.
[4] P. Billingsley, *Probability and Measure*, Third ed., John Wiley & sons, 1995.
[5] [Online] http://www.spec.org/cpu2006/

DOI: 10.1587/elex.9.1461
Received August 11, 2012
Accepted August 27, 2012
Published September 19, 2012

[6] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," *WWC-4'01*, pp. 3–14, Dec. 2001.
[7] [Online] http://www.simplescalar.com/

1 Introduction

Many-core processor based systems consist of tens to hundreds of processor cores and widely gain popularity in high-performance embedded systems running parallel applications such as multi-media, networking, bio-applications, scientific applications [1, 2]. Each processor core typically has cache and cache is connected to the external memories directly or via multi-level caches. Two conflicting resources in many-core processor based systems are the die size for on-chip memories and the bandwidth requirement of the on-chip or off-chip memories. Determining the application-specific optimal cache size and memory bandwidth for a many-core processor based system via a simulator could require a prohibitively large simulation time. Previous works on cache misses for multi-core or many-core based systems focused on how to reduce the miss rate [3]. To the best of knowledge, no works discussed estimating memory bandwidth and exploring large design space of many-core based system. In this paper, we propose an analytic model which can be used to find the optimal trade-off between the cache size and memory bandwidth. This model makes use of stochastic behavior of cache misses, which are application specific, and uses the central limit theorem. This technique can be used to rapidly estimate optimal cache size and memory bandwidth for many-core based system running identical or non-identical parallel applications.

2 Memory hierarchy in target many-core processor architecture

The target processor architecture contains tens to hundreds of cores. Each processor core has its own cache and the cache is connected to the lower-level memory, which can be either lower-level caches or external memories. Our model can be applied to any memory hierarchy where many higher-level memories, i.e. first-level caches, share a lower-level memory, i.e. a lower-level cache or external memories, provided that the sharing ratio is high.

In memory hierarchy, tradeoffs among the size of memories, memory bandwidth, and performance are often observed. For instance, a small higher-level cache increases the bandwidth requirement of the lower-level memory. To support high bandwidth, lower-level caches or memories either increase banks or ports in case of on-chip caches or the number of pins in case of external memories, all of which are costly. If sufficient bandwidth is not guaranteed by the lower-level memories, buffering memory accesses is required and that could lead to performance loss due to either excessive buffering or processor stalls caused by buffer overflows. To avoid this, the optimal higher-level memory size and lower-level memory bandwidth should be determined.

DOI: 10.1587/elex.9.1461
Received August 11, 2012
Accepted August 27, 2012
Published September 19, 2012

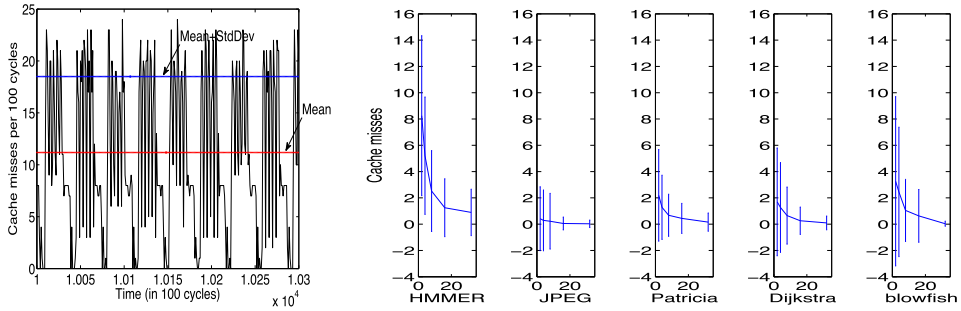

Fig. 1. (a) Cache miss behavior of blowfish for 2 KB cache size (left) (b) Mean and standard deviation of cache misses of the cache with respect to the cache size in KB (right).

3 Stochastic modeling of cache misses and memory bandwidth

One of the key ideas in this work is to identify the cache miss as a random process which varies over time. The random processes of cache miss for different applications have different characteristics, e.g. mean and variance. They also vary depending on the cache size. In Fig 1 (a), blowfish from Mibench suite shows very choppy cache miss behavior over time while its mean and standard deviation are 11.18 and 7.31 respectively. Fig 1 (b) shows the mean and standard deviation of the random process for cache misses sampled every 100 cycles with respect to different cache sizes and applications. In our proposed scheme, the number of cache misses during a sampling period is expressed as a random process, $M(t)$, which is application specific. In a many-core processor environment where each core runs an identical program independently, the collective behavior of individual cache misses can be expressed as a sum of random processes. That is, a random process for the collective miss, $CM(t) = \sum_{i=1}^{n} M_i(t)$ where n is the number of cores in the processor and $M_i(t)$ is a random process for cache misses from core i. The distribution of $CM(t)$ can be approximated via Central Limit Theorem (CLT) [4] as a Gaussian distribution as shown in Eq. (1).

$$Pr(X) = \frac{1}{\sqrt{2n\pi}\sigma} \exp^{\frac{-(X-n\mu)^2}{2n\sigma^2}} \tag{1}$$

where μ and σ are a mean and a standard deviation of $M(t)$ respectively and n is the number of cores in the processor. CLT assumes the number of processor cores is reasonably large and cache misses from different processor cores are independent and identically distributed (i.i.d.) random processes, which dictates that cores are running identical programs.

A same application with different input sets or different applications exhibit the behavior of non-identically distributed random processes. We use the Lyapunov central limit theorem to cover this case [4]. Eq. (2) shows the condition to be satisfied in order to apply the Lyapunov central limit theorem.

DOI: 10.1587/elex.9.1461
Received August 11, 2012
Accepted August 27, 2012
Published September 19, 2012

$$\lim_{n\to\infty} \frac{1}{\sigma_n^{2+\delta}} \sum_{i=1}^{n} E[|M_i(t) - \mu_i|^{2+\delta}] = 0 \tag{2}$$

where $M_i(t)$ is a random process for the cache miss of an application running on core i, μ_i and σ_i are the mean and standard deviation of $M_i(t)$ respectively, and σ_n is $\sqrt{\sum_{i=1}^{n} \sigma_i^2}$. If the condition is met, the sum of random processes for cache misses of non-identical applications converges to the Gaussian distribution. The equation implies that the condition is satisfied for the random process in which the growth of high order moments are limited. This can be easily seen in our case because the maximum number of cache misses per 100 cycles is limited to 100 in the worst case. The distribution of a random process for collective cache misses, $CM(t)$, in this case is slightly modified from Eq. (1). The new equation is shown in Eq. (3).

$$Pr(X) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp^{\frac{-(X-\sum_{i=1}^{n}\mu_i)^2}{2\sigma_n^2}} \tag{3}$$

Memory bandwidth can be viewed as another random process, $B(t)$, which is a function of $CM(t)$ as shown in Eq. (4).

$$B(t) = CM(t) \times Cache\ Line\ Size \tag{4}$$

A conventional engineering method used to estimate the memory bandwidth requirement is to measure the peak memory bandwidth for a single program and multiplying that with the number of processor cores. This is, however, too pessimistic. On the other hand, the use of miss rate only gives an average value and cannot give an accurate estimation on the peak memory bandwidth requirement. From Fig. 1 (a), we can easily see that peak value for the cache misses can be quite different from the average value. To estimate the bandwidth requirement accurately, we consider both mean and variance of cache misses. CLT gives accurate distribution of collective cache misses (thus the memory bandwidth requirement) and provides the probability of overflow in buffers between higher-level and lower-level memory for given memory bandwidth. If the overflow happens, it backpressures all processor cores to stall, which could severely degrade performance. The Gaussian distribution from CLT implies that the probability of overflow decreases exponentially as we increase the supported bandwidth.

4 Results

To verify the accuracy of our proposed analytical model, we modify Simplescalar [7]. We add routines to collect the statistics of cache misses and create traces for the cache misses. Traces of first-level cache misses are collected from cores running either identical applications or non-identical applications. These traces are combined to create a trace for the collective cache misses from a many-core system. This combined trace is used to create traffic for the lower memory. The statistics for the combined trace (simulation result) is compared with the estimated collective cache misses based on our

DOI: 10.1587/elex.9.1461
Received August 11, 2012
Accepted August 27, 2012
Published September 19, 2012

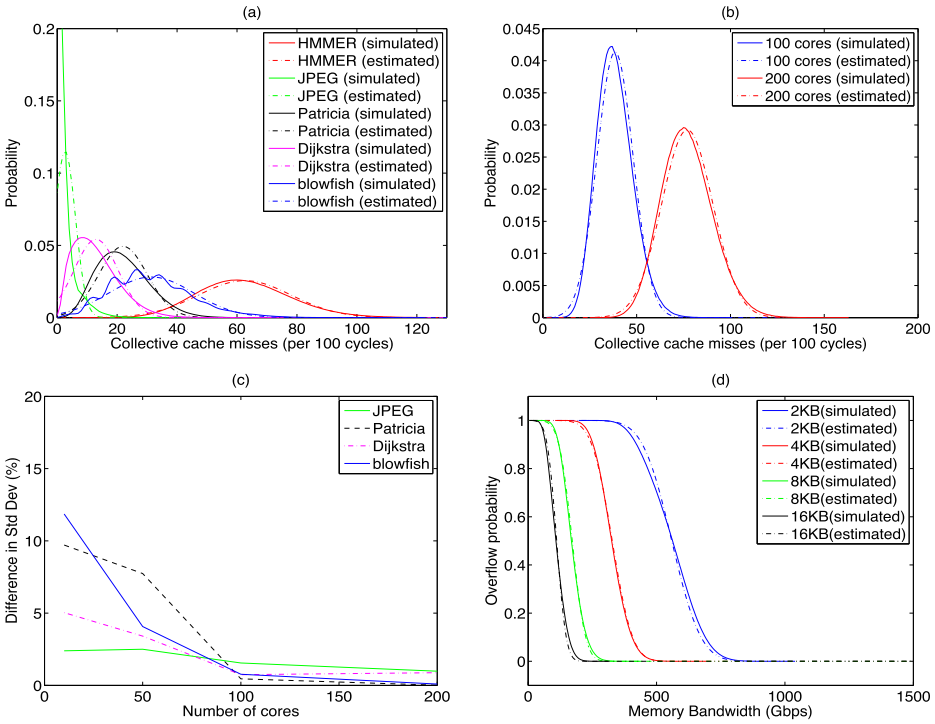

Fig. 2. Distribution of collective cache misses from simulation and estimation for (a) identical applications with 16 KB cache and (b) non-identical applications with 32 KB cache. (c) Difference in standard deviation between estimated and simulated cache misses. (d) Overflow probability for memory bandwidth with respect to different cache sizes in 100 cores running Patricia.

analytical model using the Eq. (1) and (3). Fig. 2 (a) shows the distribution for both simulated and estimated cache misses for various applications running on a many-core processor with 50 cores. Applications include gene search (Hmmer) from SPEC benchmark [5] and multimedia(JPEG), networking (Dijkstra, Patricia), security (blowfish) from Mibench suite [6]. In this experiment, many copies of the same application are running on 50 cores and the distribution for the simulated cache misses from 50 processor cores is quite close to the estimated one, which proves that our model is relatively accurate in estimating cache misses with only 50 cores. The mean values for estimated and simulated cache misses differ by less than 0.01%. The differences in standard deviation for simulated and estimated cache misses for 10 cores through 200 cores are shown in Fig. 2 (c). They decrease to 1∼2% as the number of cores increases beyond 100. Fig. 2 (b) shows the distribution for both simulated and estimated cache misses for a mixture of applications running on 100 or 200 processor cores. In this experiment, 30%, 30%, and 40% of processor cores run blowfish, Dijkstra, and Patricia respectively. The results show that estimation via Lyaponov CLT again accurately predicts collective cache misses for a mixture of applications. The differences in the mean of simulated and estimated collective cache misses are only 0.05% and

DOI: 10.1587/elex.9.1461
Received August 11, 2012
Accepted August 27, 2012
Published September 19, 2012

Table I. Simulation time with varying number of cores for various applications

Applications	1 core	10 core	50 cores	100 cores	200 cores
Patricia(small input)	1m15s	12m54s	63m38s	126m36s	250m27s
JPEG(small input)	27s	4m53s	22m33s	42m33s	86m48s
Hmmer	45m41s	456m50s	N/A	N/A	N/A

0.01% for 100 and 200 cores. Fig. 2 (d) shows the probability of overflow for each supported memory bandwidth in a processor consisting of 100 cores running Patricia. In other words, each curve (1-cumulative density function) shows the probability of the temporal peak memory bandwidth required by 100 copies of Patricia being greater than the available memory bandwidth. The exponential decay implies that the probability decreases rapidly as we increase the available memory bandwidth. In addition, as we increase the cache size, the required memory bandwidth decreases. This plot can be used to optimize the cache size and memory bandwidth. Again, our estimated distributions closely track the simulated ones.

We measure simulation time on 2.4 GHz Intel Xeon Dual-core Processor and show them in Table I. The simulation consists of two major steps: running identical or non-identical programs on processor cores to generates cache miss trace files and merging trace files to generate collective cache misses. The first step dominates the simulation time and simulating independent programs increases the simulation time roughly in proportion to the number of cores. For this reason, the speedup in simulation time for Patricia and JPEG are 200.4 and 192.9 respectively for 200 cores. In Hmmer, the simulation times larger than a day are omitted because it takes too long to perform any meaningful architecture evaluation. From Table I, it is clear that our analytical model can significantly reduce architecture evaluation time.

5 Conclusion

In this paper, we propose an analytical model to rapidly estimate the required lower-level memory bandwidth of many-core processor based system given first-level cache size and statistical behavior of programs. We can use this result to study the architectural trade-offs between the cache size of processor cores and the lower level memory bandwidth. While simulating many-core processor takes a prohibitively large time which makes it impossible to study the various architectural alternatives, our estimation scheme can provide an accurate and essential tool for many-core processor based SoC design.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No.2011-0023798, No.2010-0025423, and No.2010-0026822) and by the Sogang University Research Grant of 2011 (No.201110026).

DOI: 10.1587/elex.9.1461
Received August 11, 2012
Accepted August 27, 2012
Published September 19, 2012