Impact of the Memory Controller on the Performance of Parallel Workloads

Crispín Gómez Requena

Departamento de Sistemas Informáticos,
Universidad de Castilla–La Mancha, Albacete, Spain

Abstract. Multicore processors have become de facto the typical processors being implemented in almost all microprocessor-based systems ranging from embedded devices to large-scale data centers.

Technology advances allow the integration of a larger amount of cores with each new microprocessor generation. Consequently, the number of memory requests competing for memory rises, so increasing the already huge memory latencies. A straightforward solution to deal with this problem is to increase the number of memory controllers so spreading memory requests among them. However, this solution is not feasible because it would be too costly and it is also limited by technological constraints since core size shrinks at a higher pace than the memory subsystem components.

This paper explores the impact of the number of memory controllers for a medium to large range (manycore) of number of cores with the aim of analyzing the best tradeoff between performance and cost. Results are shown for parallel workloads, which are typically targeted to these processors.

1 Introduction

Chip-multiprocessors (CMPs) have dominated a wide spectrum of the microprocessor market, ranging from embedded to supercomputers, since a few years ago. The on-chip core counts in CMPs have rapidly grown and it is expected to reach several hundreds of cores in the following years [Bell et al.(2008)Bell, Edwards, and Amann, et al]. This trend will even grow in a more significant way in computation-intensive environments like servers, where even we can find current products with tens of cores [Held(2010)]. This means that manycore processors are in the near horizon.

This large number of cores jointly with the tend to implement aggressive out-of-order execution cores contribute to increase contention in the memory subsystem which in turns increases even more the high latency of accessing the main memory. A way to hide this huge memory latencies observed by cores and increase the memory bandwidth is based on using several memory controllers in order to increase the parallelism in the memory access. Having several memory controllers helps to the system performance but at expenses of important energy consumption, which means that the selection of the number of memory controllers in a system must be properly balanced to provide competitive

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 423–432, 2014.

products in the market achieving the best performance with the least power consumption.

Due to this fact, this paper analyzes the impact of the memory controller design in CMPs. The idea is to provide insights on how many memory controllers are required depending on the core aggressiveness. In particular, as cores are more powerful and a larger number of them are present in the processors, the memory subsystem becomes the main bottleneck, and therefore in this paper we explore the memory subsystem performance, as a measure of the system performance, by analyzing the impact of the number of memory controllers on the latency observed by cores.

In this paper, we assume that a CMP is composed by two main subsystems: core and main memory. Actually, the system performance depends on the interaction among them. On one hand, the cores are the elements that can produce the memory requests and the memory is the one that receives and answers these requests. However, a deep and through study of all the interactions among them is out of the scope of this paper.

From the applications point of view, while it is true that serial workloads still make up a strong segment of the market today, many-core processor designs usually assume parallel workload as recently claimed by James Reinders, Director of Intel Corporation [Jam(2012)]. Thus, in this paper we use parallel workloads to perform this analysis as the workloads that must be considered in highly parallel systems.

Experimental results show that increasing the number of memory controllers in the system helps the machine to reduce the overall memory latency observed by cores which depends on the number of cores and the core aggressiveness and is translated to lower execution times for the benchmarks. Thus, in this paper we claim that a holistic system design must be done considering the features of the different components in order to obtain a balanced system, obtaining the highest benefits of each component.

The remainder of this paper is organized as follows. Section 2 presents the baseline system architecture and discusses pros an cons on the overall performance from the two major subsystems points of view: core and memory. Section 3 highlights the reasons of the the holistic approach by focusing on the benefits of considering interactions among subsystems. Section 4 introduces the experimental framework used in this work. Section 5 presents and analyzes the experimental results. Finally, Section 6 presents some concluding remarks.

2 Major Subsystems: Cores and Main Memory

This section describes the baseline system and discusses the impact of the two major subsystems on performance.

2.1 Baseline Architecture

This paper focuses on high-performance multicore processors. For instance, the Intel SCC prototype consists of 48 cores [Jainschigg(2012)]. This processor

consists of 4 memory controllers and 4 memory domains, that is, each MC is used by the closest 12 cores. This scheme, where memory domains are distributed among several MC is the typical expected approach for future systems to reduce memory latency by reducing contention in the MCs (a subset of cores access each MC) and network latency (less hops are required to reach the nearest MCs).

In Figure 1 we show a block diagram of the modeled system. A typical tiled multicore processor is assumed, each core has its private L1 cache and L2 cache, and coherence is supported by using a MOESI-like [et al.(2003)] coherence protocol. Several memory controllers are supported to enhance memory access parallelism, and at the same time, to shorten the average distance from cores to memory controllers.

The design of such a system must address both performance and costs since both are the main concerns. Overall performance depends on the combined performance of the core and memory subsystems. The idea is to achieve a good tradeoff between performance and power.

2.2 Core

System designers have traditionally focused on the processor since it provides the ever increasing demanding computational power of the machine. This computational power increase is being accomplished following three major trends: i) improving the computational power of individual cores, ii) increasing the number or cores, or iii) combining both actions together.

Most processors manufacturers moved to multicore processors with the aim of providing a good tradeoff between performance and power. To this end, first they included several simple (e.g. in-order execution) cores. However, recent

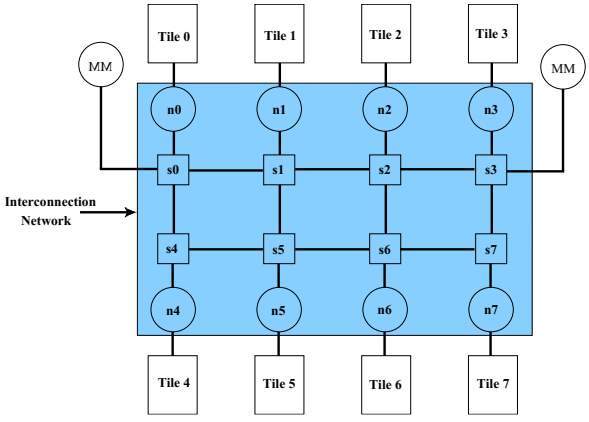

Fig. 1. Baseline processor: cores, on-chip network (NoC) and caches, and MCs. A single memory domain is considererd.

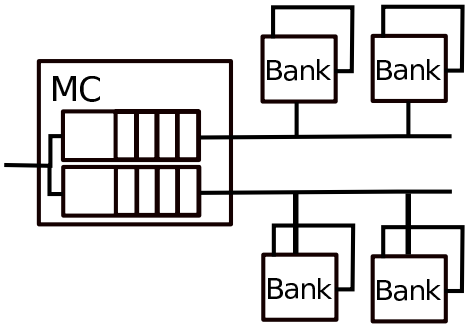

Fig. 2. Scheme of a MC with two memory channels, two banks per channel and two ranks per bank

advances in both technology and microarchitectural techniques, have allowed manufacturers to deploy aggressive out-of-order execution cores in CMPs.

Nowadays even in high performance processors, the power consumption is a concern. So processors are designed both from the power-aware perspective and high-performance point of view. From a high-level point of view, two main features characterize the processor, the number of cores and its aggressiveness. In this work, we explore the impact on performance of both features in a multicore processor as a part within the envisioned baseline system. Different aggressiveness have been explored by varying the decode, issue, and commit width of the computational cores.

2.3 Main Memory

The main memory subsystem has become the major design concern in high-performance processors because it represents a major performance bottleneck, which is even more aggravated with the increasing core counts. This is caused by the large number of memory requests delivered by multiple cores that compete among them for accessing a given memory module attached to a memory controller. This access is performed through the NoC that usually implements a mesh topology in CMPs. Figure 2 depicts a block diagram of a modern memory architecture consisting of two memory channels, two banks per channel and two ranks per bank.

Memory requests issued at the cores that do not hit in the cache hierarchy are enqueued at the memory controller. It represents the interface with main memory modules (e.g. DRAM). It translates memory requests to memory modules commands. Memory controllers usually deploy several queues and handles them according to specific priority policies. The most simple and widely-used is the First Come - First Served (FCFS) in which memory requests are served in arrival order.

Apart from having several memory controllers, with the aim of alleviating the huge memory access latency, the memory is organized in channels, banks and

ranks to enhance memory access parallelism. Memory channels are the mean that interconnects the memory controller to the memory chips, that is, a memory controller can issue as many commands concurrently to main memory as number of memory channels. A memory bank holds a set of *r*ows also referred to as memory pages, which expand through multiple devices that are accessed at the same time (not shown for clarification purposes).

On the arrival of a command through the channel to the corresponding bank, the bank is accessed and the channel is released until the data is available at the row buffer (bank output). Thus, assuming that a channel transaction lasts one cycle, the memory controller can issue two distinct requests through the same channel in two consecutive bus (channel) cycles if they target different memory banks (i.e. no bank conflict arises). If two requests target the same memory bank, the memory controller will stop the second one until the first one is completed. Finally, banks in a channel can be organized in different ranks. Each rank corresponds to an independent set of devices. Thus banks in different ranks can be accessed in parallel. Thus increasing the number of ranks increases the potential bank level parallelism.

3 Dependencies among the Two Major Subsystems

The system performance and energy consumption depend not only on the individual performance of each subsystem, but on how subsystems interact among them.

The number of cores in each memory domain is, in general, a key design issue, since it affects costs, energy, and performance. Performance of the memory controller can be expressed in terms of latency and throughput. Increasing the number of memory controllers reduces the number of cores in each memory domain, thus the number of access conflicts due to accesses to the same main memory bank is also reduced. In other words, a higher number of memory controllers supports by design a higher level of memory access parallelism. This means that the larger the core counts, the higher the number of memory controllers required for a given memory controller performance. On the other hand, the main memory cost represents an important fraction of the overall system cost [Hennessy and Patterson(2012)].

In order to design a balanced system addressing performance, energy, and cost, a compromise must be met with the number and aggressiveness of cores and the memory subsystem design (number of MCs, number of banks, etc). This work pursues to provide high-performance designers insights on such issues.

4 Experimental Framework

This section presents the simulation environment used for the design exploration study. An extended version of the Multi2sim simulation framework [Ubal et al.(2012)Ubal, Jang, Mistry, Schaa, and Kaeli] has been used to model the devised future manycore embedded system. Contention at the different points of

the system has been modeled: link and injection buffer contention at NoC, channel, bank, and queue contention at MC, as well as stalls at the cores due to resource availability and stalls due to memory operations, mispredictions, and so on. Table 1 summarizes the main architectural parameters for the three studied subsystems

As aforementioned, it is expected that manycore processors would be used to executed parallel loads. Therefore, we took some applications from the SPLASH2 benchmark suite [Woo et al.(1995)Woo, Ohara, Torrie, Singh, and Gupta], which characterize the parallel behavior of near future applications. No sampling technique or kernels are used to collect the results, the benchmarks have been executed completely.

5 Experimental Evaluation

Nowadays, as cores are more powerful and more cores are deployed in CMPs, the pressure on the memory memory is greater. This results into memory latency is even more a major design concern since it limits the performance achieved by the CMP. In this section, we explore the system performance depending on the memory subsystem performance as we change the core aggressiveness and the number of memory controllers.

Table 1. Machine parameters

Microprocessor core	
Issue policy	Out of order
Fetch, issue, commit width	2/4/8 instructions/cycle
ROB size (entries)	256
L1 inst. cache	64B-line, 2-way, 32KB, 2-cycle
L1 data cache	64B-line, 2-way, 32KB, 2-cycle
L2 unified cache	128B-line, 16-way, 1MB, 10-cycle
NoC	
Topology	Mesh
Virtual channels	No
Routing algorithm	X-Y
Switch buffer size	2 entries
Memory subsystem	
Memory controller	single queue 64-entry single queue FCFS
Channels	2
Banks / ranks	4 / 1
DRAM access time	300-cycle

5.1 Impact of the Number of Memory Controllers on Performance

Systems with a relatively high number of cores, such as 64-core systems, are heavily constrained by main memory latencies, as depicted in Figure 3. The 64-core system configurations with just a single memory controller show a huge memory latency for all the evaluated benchmarks. Despite the memory access time is 300 cycles, the latency reached by all the benchmarks and configurations increases over 900 cycles, due to the pressure done by the high number cores on the memory subsystem which results in an extreme congestion in the MC and a huge number of bank and channel conflicts.

This congestion turns in unacceptable memory subsystem performance, which will fill the ROB due to pending memory requests and therefore, the processor will be blocked hurting the performance. Considering core aggressiveness, it can be seen that it has a small but increasing influence for FFT and Ocean, but it is markedly stronger in Radix, in which memory latency is increased from 1931 cycles to 2315 cycles when core width goes from 2 to 8 ways. As core width grows, more memory requests can be sent at the same time from the cores to main memory which leads to increase the congestion in the NoC and in the memory itself.

The straightforward solution to alleviate this performance problem is to deploy multiple memory controllers. Nevertheless, the ideal number of memory controllers to achieve a good trade-off between performance and cost is still an open research problem. The aim of this section is to provide insights into this concern. To this end, this section evaluates the impact on performance of adding more memory controllers to a 64-core system. For evaluation purposes, we assume that all the memory controllers are uniformly distributed on the border of the system, except for the 64-memory controller configurations where each core has a MC attached to its switch. We consider the configuration with 64 MCs in spite of being impossible to implement to show an upper bound of the performance that can be obtained by only adding more memory controllers.

Figure 3 shows that, in general, the memory access time linearly decreases with the number of MCs in the system. For instance, in FFT it drops from 2094 cycles with 1 MC to 1788, 1388, 846, and 382 cycles for 2, 8, 16, and 64 memory controllers, respectively. This is the expected outcome, since when deploying more memory controllers the number of bank/channel conflicts is enormously reduced. Furthermore, the NoC utilization is spread all over the different MCs equally distributed along its border which reduces network congestion, and as a side-effect memory access time. As it can be seen, the more MCs that are in the system, the better latency results that are obtained.

However, in order to fully analyze the effect of adding more MCs to the system is not enough to study the memory latency reduction, but its impact on the total execution time of the applications, which is plotted in Figure 4. Having 2 or more MCs clearly has a positive influence on the execution time. Again for FFT and 2 ways, execution time is reduced to 74%, 56%, 42%, and 28% of the execution time with 1 MC. Reductions in execution time from 2 to 64 MCs are linearly consistent in each application, thus again results show that as many MCs there are in the system, the better.

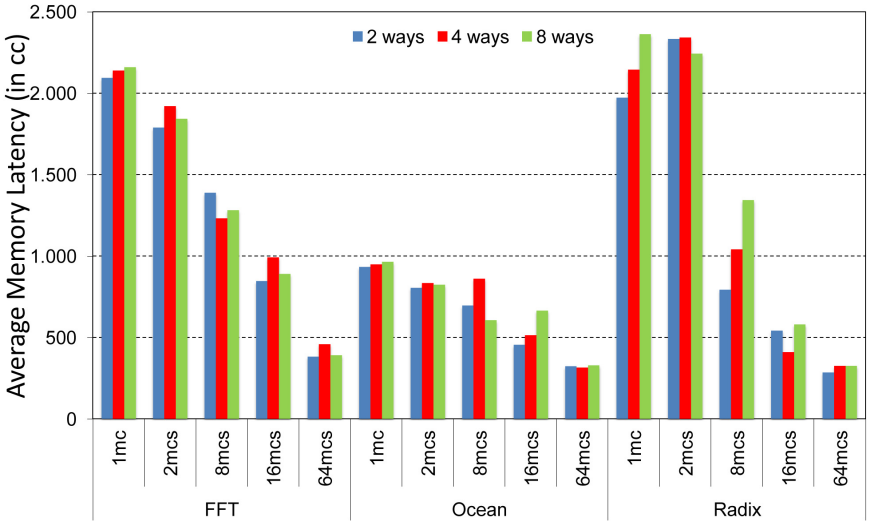

Fig. 3. Average memory latency measured in core cycles for different applications varying the number of MCs and core aggresiveness in 64-core systems

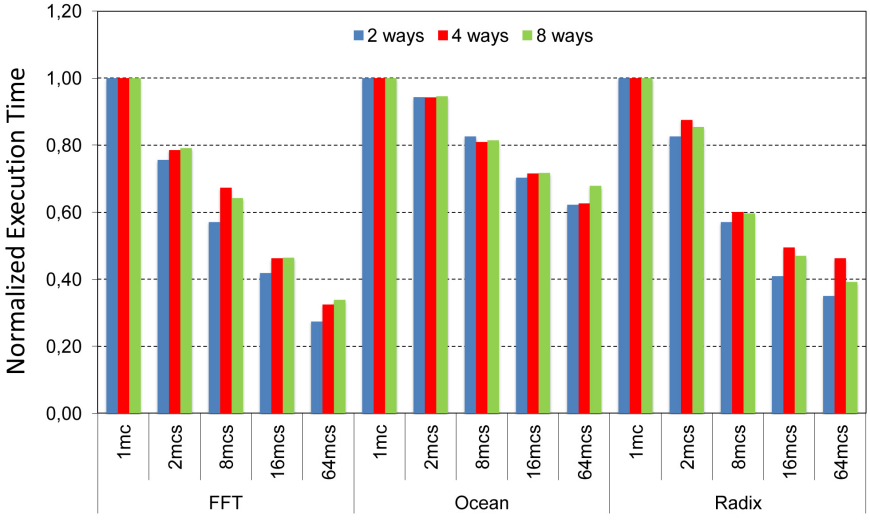

Fig. 4. Normalized execution time for different applications varying the number of MCs and core aggresiveness in 64-core systems

However, cost should also be considered, since in FFT to almost halve execution time from the configurations with 16 MCs to the configuration with 64 MCs, cost is multiplied by 4. So the optimal trade-off between cost and performance lies between 8 MCs and 16 MCs, where performance is increased linearly with the increase in cost of adding more MCs to the system.

6 Conclusions

Multicore/manycore systems have become the fastest growing segment in the microprocessor market. Nowadays power and also as traditionally performance are key design concerns of CMP systems. The two major subsystems that contribute to the system performance are cores and the memory subsystem. In this paper we explore the design space for future high-performance manycore by focusing on the analysis of interactions among the aforementioned two major subsystems. In particular, we explore the memory subsystem performance as a main contributor of the system performance, as a result of the interactions with the other subsystems.

Experimental results show that, increasing the number of memory controllers in the system helps the machine to reduce the overall memory access time and execution time of the applications. A optimum number of cores per MC to achieve a good performance-cost tradeoff is between 8 and 16 cores. Thus, one can conclude that interactions among the two major subsystems should be considered in order to obtain a well balanced global system, and obtaining the most benefits of each component, and therefore obtaining a good performance-cost tradeoff.

Acknowledgments. This work was supported by the Universitat Ppolitècnica de València with tthe grant Primeros Proyectos de Investigacin (PAID-06-10), with reference number 2370, and jointly supported by the Spanish MINECO and European Commission under the project TIN2012-38341-C04-04."

References

[Bell et al.(2008)Bell, Edwards, and Amann, et al] Bell, S., Edwards, B., Amann, J., et al.: TILE64^{TM} processor: A 64-core SoC with mesh interconnect, pp. 88–598

[Held(2010)] Held, J.: "Single-chip cloud computer", an IA tera-scale research processor. In: Guarracino, M.R., Vivien, F., Träff, J.L., Cannatoro, M., Danelutto, M., Hast, A., Perla, F., Knüpfer, A., Di Martino, B., Alexander, M. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 85–85. Springer, Heidelberg (2011)

[Jam(2012)] `http://goparallel.sourceforge.net/ask-james-reinders-multicore-vs-manycore/` (2012)

[Jainschigg(2012)] Jainschigg, J.: (2012), `http://goparallel.sourceforge.net/manycore-mobiles/`

[et al.(2003)] Martin, M.M.K., et al.: Protocol Specifications and Tables for Four Comparable MOESI Coherence Protocols: Token Coherence, Snooping, Directory, and Hammer (2003), `http://www.cs.wisc.edu/`

[Hennessy and Patterson(2012)] Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach, 5th edn. Morgan Kaufmann Publishers Inc., San Francisco (2012); Appendix E by T. Conte

[Ubal et al.(2012)Ubal, Jang, Mistry, Schaa, and Kaeli] Ubal, R., Jang, B., Mistry, P., Schaa, D., Kaeli, D.: Multi2sim: a simulation framework for cpu-gpu computing. In: Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques, PACT 2012, pp. 335–344. ACM, New York (2012)

[Woo et al.(1995)Woo, Ohara, Torrie, Singh, and Gupta] Woo, S.C., Ohara, M., Torrie, E., Singh, J.P., Gupta, A.: The splash-2 programs: characterization and methodological considerations. SIGARCH Comput. Archit. News 23, 24–36 (1995)

[Bailey(1989)] Bailey, D.H.: FFTs in External or Hierarchical Memory. In: Proceedings of the ACM/IEEE Conference on Supercomputing (1989)