Performance, Area and Bandwidth Implications on Large-scale CMP Cache Design

Li Zhao, Ravi Iyer, Srihari Makineni, Jaideep Moses, Ramesh Illikkal, Donald Newell
System Technology Lab, Intel Corporation
Contact authors: {li.zhao, ravishankar.iyer}@intel.com

Abstract

Large-scale CMP (LCMP) platforms that consist of 10s of cores for throughput computing will soon become reality. The performance and scalability of these architectures is highly dependent on the design of the cache hierarchy. In this paper, our goal is to explore the cache design space for LCMP platforms. We approach this exploration problem by developing a constraint-aware analysis methodology (CAAM). CAAM first considers two important constraints and limitations that the LCMP cache design needs to account for -- area constraints and on-die / off-die bandwidth limitations. Based on the approximate area constraints, we determine a viable range of cache hierarchy options. We then estimate the bandwidth requirements for these cache hierarchy options by running server workload traces on our LCMP performance model. Based on allowable bandwidth constraints, we narrow the design space further to highlight a few cache options that are indeed viable for LCMP platforms. We then compare these options based on performance and make specific recommendations for future LCMP cache hierarchies.

1. INTRODUCTION

The momentum behind CMP architectures [10] is pushing architects and designers to consider integrating more and more cores on the die. Within this decade, we expect that large-scale CMP (LCMP) architectures with 10s of cores and several 10s to 100s of threads on the die will be a reality. As compared to traditional single-core multi-socket platforms, LCMP single-socket platforms seem to be an attractive choice for throughput computing [7] because of the potential for low latency and high bandwidth communication on the die. However, for LCMP architectures to be scalable, it is critical that the on-die cache/memory hierarchy be designed to support many cores / threads efficiently. In this paper, our focus is on exploring the cache hierarchy design for LCMP platforms.

When investigating cache hierarchy design for LCMP platforms, there are several important factors to consider. One such factor is the implication of die area constraints. While a significant fraction of the die is devoted to cache area in single-core and dual-core processors, the addition of more cores may limit the amount of die space that can be devoted to cache. Another factor is the amount of on-die and off-die bandwidth that is available in the platform. Since the on-die interconnect will potentially carry the communication between two levels of the cache hierarchy, the amount of bandwidth it can support plays a critical role in sizing the two levels of the caches. Off-die memory bandwidth also plays a critical role in determining cache design. If significant amounts of memory bandwidth can be provided and long memory latencies can be tolerated, then the cache size can be moderate. However, platform constraints (pin count, power, packaging etc) tend to limit the amount of memory bandwidth that can be supported. In this scenario, it is important that sufficient cache space be enabled on the die as a last line of defense [15] against the memory bandwidth wall. In addition, the power consumption of the caches [3] also plays a critical role in determining the cache hierarchy and more specifically in the policies that govern the operating modes of the caches. Last but not least, the overall platform performance is an indicator of the effectiveness of the cache hierarchy in supporting the many simultaneous threads of execution. In this paper, we focus on understanding the implications of three of these vectors: area constraints, bandwidth constraints and overall performance / scalability of the platform.

Previous studies on cache design space exploration have largely been focused on performance [6, 20, 23], with few that have considered the implications of power and/or area [3, 21]. This paper proposes a methodology to study area, bandwidth and performance implications on cache design space exploration in the context of LCMP platforms. In this paper, we attempt to answer the following key questions:

- How do we prune the LCMP cache design space? What methodology needs to be put in place?
- How should the cache be sized at each level and shared at each level in the hierarchy?
- How much on-die/off-die bandwidth is required?

We start by proposing a constraints-aware analysis methodology to analyze the cache design space. We employ existing tools for estimating the area required for a given cache size. We develop a detailed performance simulator to simulate the LCMP architecture with specific emphasis on cache hierarchy and coherence protocols. We conduct extensive sets of experiments running commercial server workloads. Based on the resulting data, we prune the cache hierarchy design space and make key recommendations for future LCMP platforms.

2. CACHE HIERARCHY FOR LCMPS

In this section, we introduce the LCMP architecture and discuss the cache design considerations in more detail.

2.1. Architecture Overview

Today's server platforms employ multiple processor sockets, each with one or two multithreaded cores. The LCMP architecture employs many cores per socket and fewer sockets per platform. Figure 1 illustrates the LCMP platform architecture with a single socket. The on-die architecture consists of several nodes (each with some number of multi-threaded cores and a shared node cache), an on-die fabric that interconnects the nodes, potentially a shared last-level cache (L3), integrated memory controllers and other external interfaces. Several companies [1, 12] are already designing or have announced products that resemble the LCMP architecture.

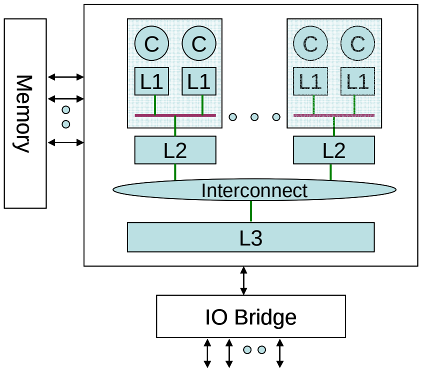


Figure 1: LCMP Architecture Overview:

As process technology advances, each new technology generation is expected to provide a minimum of 0.6x feature size scaling and an increase of ~2X in transistor density [5]. As a result, we expect that LCMP architectures with 16 and 32 light weight cores on the die will be a reality within the end of this decade. Assuming 4 threads per core, this enables as many as 64 and 128 threads per socket in the near future. This study focuses on cache hierarchy design of LCMP architectures with 32 quad-threaded light weight cores. The scalar performance of each core is assumed to be low, but many cores packed together can provide a throughput computing advantage.

2.2. LCMP Cache Design Considerations

There are several design considerations to account for when exploring cache hierarchy design for LCMP platforms. Some of the key considerations are: (a) Area constraints, (b) Bandwidth Constraints, (c) Power Implications and (d) Performance of the cache hierarchy and overall platform. In this paper, we study three of the above four considerations (excluding power implications).

Area Constraints: The design of a microprocessor has to adhere to a certain area budget [4]. Server processors tend to have large dies in the order of 400 to 500 mm^2. Given that the die will contain 32 cores, an interconnect, the integrated memory controller, external interfaces and other glue logic, only a fraction of space (~40 to 60%) may be available to the cache hierarchy to occupy. For example, the Sun Niagara die is about ~380 mm^2 [14] and only 40%of it appears to be cache space (which allows only 3MB of L2 for 32 executing threads). As a result, it is important to take a hard look at the area constraints before embarking upon designing a cache hierarchy. We assume that the first level cache is an integral part of the core and hence will limit our studies to mid and last level caches.

Bandwidth Constraints: The cache hierarchy is designed to be a defense against memory latency and bandwidth limitations. While enabling multithreading on top of multiple cores allows for tolerating memory latency, the sheer number of requests generated by these threads may place a significant bandwidth demand on the memory subsystem. The rate at which memory bandwidth increases is far slower than the rate at which the compute power (CPU frequency coupled with the increase in the number of cores/threads) increases. With LCMP, it also becomes important to consider the bandwidth available on the on-die interconnect. The on-die interconnect is expected to provide several 100 GB/s and perhaps up to a TB/s. However, it is also possible that the communication between L2 and L3 will be significant because of the sheer number of cores/threads and the cache sizes. Therefore, it is important to look at on-die interconnect bandwidth into account when designing the cache hierarchy.

Application Performance: After considering the constraints, the ultimate factor that influences the cache hierarchy design is the level of performance it provides to the applications.

2.3. Constraints-Aware Analysis Methodology

In order to prune the cache design space, we propose a constraints-aware analysis methodology (CAAM). This methodology assumes that the area constraints and the off-die bandwidth constraints are known. The CAAM methodology consists of three major steps:

(1) <u>Area-Constrained Options:</u> This step essentially attempts to prune the design space by the area constraints. We first estimate the area required for L2, and then apply the overall area constraints to this cache. All options that exceed the area constraints are immediately discarded. For the options that have more area available than consumed by L2, L3 may be considered. For traditional inclusive cache hierarchies, it should be noted that there actually needs to be enough area available to allow at least 2x or more of L2 in L3. This is required since having a cache at the next level that is less than 2x of the cache size of the previous level does not perform well if inclusion is required [2]. The same process is repeated for each level until the desired number of levels of cache has been covered.

(2) <u>Bandwidth-Constrained Options:</u> This step attempts to further prune the options of those already pruned by area constrained as above by applying the on-die and off-die bandwidth constraints. This requires estimation of the

number of requests generated by the caches at each level and as a result depends on core performance and cache performance for a given workload. Let us consider the architecture described in Figure 1. An approach to bandwidth estimation is to start by simulating each node. Once the bandwidth demand for each is derived, the on-die and off-die bandwidth constraint can be used to prune the design options that require more bandwidth. In fact, it is prudent to discard options that exceed more than 50% or 60% of the bandwidth constraint since it is preferable for the memory utilization to be in this range.

(3) Overall Performance: Once the area and bandwidth constraints are applied, we have a pruned set of design options that are viable. The performance of these options is then compared to determine the top two or three design choices. Note that during the initial phases of the architecture/design process, it is more likely that the area and bandwidth constraints are a range as opposed to a fixed value. If this is the case, then it is important to conduct sensitivity studies. For example, when applying the area constraint, it may appear that only 10MB of last level cache can be provided in a certain design option. It is important, however, to measure and compare the performance per unit area of options that range from 8M to 16M. If increasing the cache size to 16M increase the area by a modest amount, but provides a significant boost in performance/area, then it may emerge as a potential design choice at the expense of additional die area.

3. EVALUATION TOOLS AND WORKLOADS

In this section, we describe an overview of the simulation environment, area estimation tools and the workloads used.

3.1. LSIM Simulation Environment

We develop an in-house platform simulator called LSIM to allow evaluation for varying degrees of fidelity. The LSIM core simulation mimics the execution profiles present in the traces and injects memory events into the interconnect/cache subsystem. The cache models a detailed invalidation-based coherence protocol. The cache hierarchy is modeled to be inclusive by modeling the back-invalidation messages required to evict L2 copies of a line that is replaced in L3.The operating frequency (of the core, interconnect, etc), queue sizes (interconnect interface and cache controller structures), bandwidths (interconnect, cache and memory) and latencies (delays between L2 and L3s, etc) are all configurable in LSIM and allows us to explore the design space sufficiently.

3.2. Workloads & Traces

As our focus in this study is on LCMP server platform architecture and performance, we picked a few important commercial server workloads: OLTP, SAP and SPECjbb. For representing OLTP, we used traces of a TPC-C [22] workload, which is an online-transaction processing benchmark that simulates a complete computing environment where a population of users executes transactions against a database. For representing SAP workloads, we used traces of a SAP SD 2-tier benchmark [17], which is a sales and distribution benchmark to represent enterprise resource planning (ERP) transactions. For Java-based server benchmark, we use SPECjbb2005 [19] that models a warehouse company with warehouses that serve a number of districts (much like TPC-C).

For all of these workloads, we collected long instruction traces on real systems. Wherever sufficient number of instruction traces is not available, we replicate the execution profiles appropriately to feed the remaining cores/threads. When replicating traces, we make sure that the code memory accesses are shared, whereas data accesses are privatized in order to not artificially inject any incorrect data sharing. Based on detailed understanding of the workloads as well as measurements to validate them, we already know that SAP and SPECjbb have negligible data sharing. TPC-C is known to have significant data sharing (which we do not simulate sufficiently well due to the nature of our tracing/simulation environment), but newer databases seem to be trending towards reduced data sharing to avoid synchronization penalties.

The workload characteristics described and/or traces collected were not audited and the data presented in this paper should not be misused to represent benchmark performance of the architecture under evaluation.

3.3. Area Estimation Tools

For area estimation, we used CACTI (version 3.2), an integrated cache access time, cycle time, area, aspect ratio, and power model [18]. The parameters we held constant in our evaluation is the line size at 64 bytes. We vary the cache size, the number of banks and the associativity depending on L2 or L3 caches. We assume that a distributed shared L3 cache (somewhat like in NUCA [11]) with independent controllers. All cache area estimates are based on a 45nm process as our intention is to look at architectures around the end of the decade.

Table 1: LCMP Configurations and Parameters

Parameters	Values
Core	4GHz, In-order, 4 threads, abstract model (core CPI varied)
L1 I/D cache	32 Kbytes, 4-way
L2 cache	128K-4M bytes, 8-way
L2 cache hit time	10 cycles (varied)
MSHR size	16
L3 cache	8 ~ 32M bytes, 16-way, 64-byte, banked (1, 2, 4M) organization
L3 cache hit time	50 cycles (varied)
Interconnect BW	128GB/s ~ 512GB/s
Memory access time	400 cycles
Memory bandwidth	32GB/s ~ 128GB/s

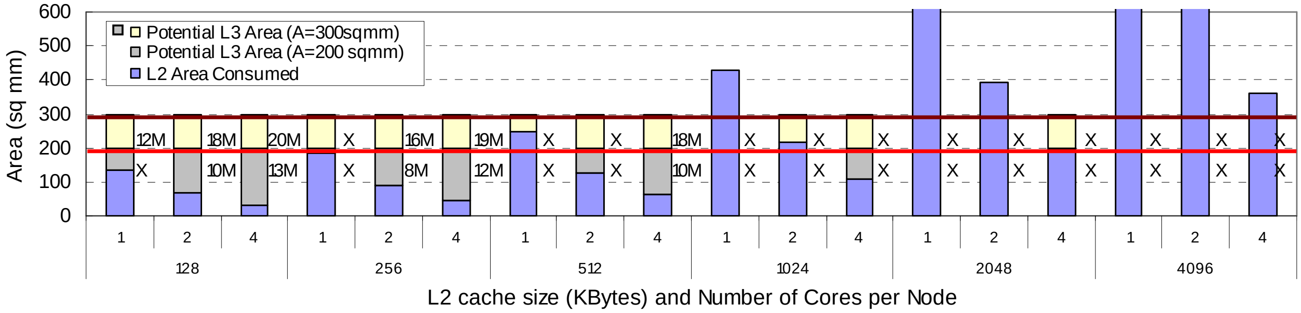


Figure 2. Area Considerations for a 32-Core LCMP Cache Design.

3.4. Baseline Configurations & Assumptions

The baseline architecture and associated simulation configurations that we evaluate and the range of values used is presented in Table 1. The simulated architecture consists of 32 cores (with 4 threads each) at a frequency of 4GHz. The on-die architecture is made up of several nodes. Each node may consist of 1, 2 or 4 cores. In LSIM, we simulate L2 cache size per node that varies from 128K to 4M. The L2 cache may be configured as either private per core or shared between all of the cores in the node. For configurations where an L3 cache appeared to be viable (based on area constraints), we simulated an L3 cache with size varied from 8M to 32M to a perfect L3 cache.

4. Area and Bandwidth Implications

In this section, we present our evaluation of the LCMP cache hierarchy design space based on the constraints-aware analysis methodology.

4.1. Implications of Area Constraints

We start applying the CAAM methodology to 32-core LCMP cache design space exploration by first considering area constraints. In Figure 2, the bottom bar summarizes the L2 cache area estimates as a function of the L2 cache size per node (128K to 4M) and the number of cores per node (1 to 4). Note that as the number of cores per node increases, the total number of nodes (which is the same as the number of L2 caches) deceases, thus the total L2 area decreases. We can see that as the cache size increases from 128K to 512K, the space consumed by the cache does not increase linearly. However, as the cache size increases past 512K, the cache area starts showing closer to a linear increase. Also note that as the cache size per core goes to 1M and beyond, the area consumed by the cache space is about 400 mm^2 or higher.

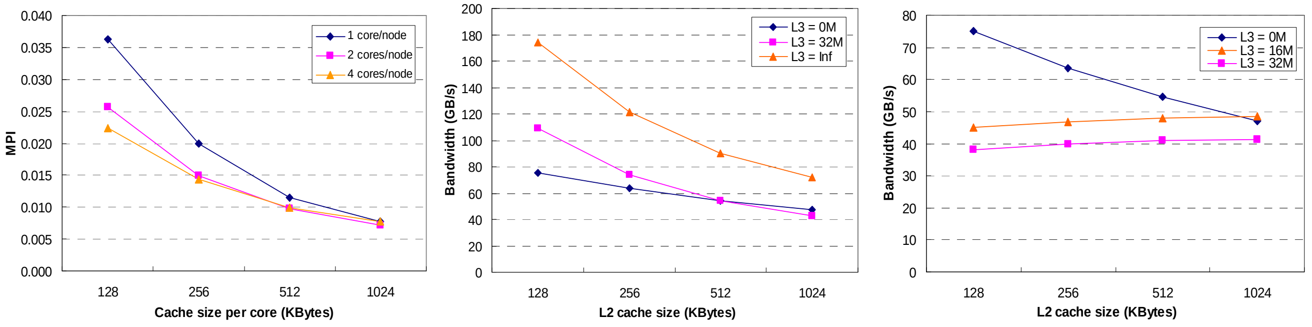
Due to manufacturing costs as well as form factor limitations, it is important to keep the die size of the processor as low as possible. Server chips are larger than desktop processor chips and have been generally less than 400 mm^2. The largest die in production today is an Itanium 2 processor (estimated to be around 432 mm^2 [15]). In order to keep the die area under 400 mm^2, it is important to keep the cache area to a reasonable fraction of the overall area. In this paper we study the effect of constraining the cache space to 50% (200 mm^2) or 75% (300 mm^2). Figure 3 shows the constraint with the two horizontal lines that represent the 200mm2 and 300mm^2 constraints. It can be observed that if the LCMP die is assumed to only possess L2 cache (no L3 cache), then all configurations with up to 512K L2 per core seem to stay within the constraint.

The next step is to identify design options where there is provision for a shared L3 cache. Figure 3 highlights the configurations where there is a potential for a L3 cache. Since we are considering traditional inclusive cache hierarchies, it is important that the area available to an L3 cache allow for at least twice the size of the L2 cache area. By applying this criteria, we show L3 cache size estimates (numbers to the right of the bars) for both area constraints (200 mm^2 and 300 mm^2 in a 45nm process). For example, the very first bar in the figure shows that with the configuration of 128K per node, 1 core per node and a total of 32 cores, we cannot employ a suitable L3 cache if the area constraint is 200 mm^2. This is because the amount of area available cannot accommodate an inclusive L3 cache that is equal to or larger than twice the size of the L2 cache size (which is 4M = 128K*32 in this case). However, if the area constraint is relaxed to 300 mm2, then a L3 cache that is roughly 12 MB in size can be accommodated.

4.2. Implications of Bandwidth Constraints

Another crucial step in the CAAM methodology is to apply on-die and off-die bandwidth limitations to the cache design space exploration. Note that unlike area constraints which can be applied independent of the workload running on the platform, the bandwidth constraints need to be considered along with a representative set of workloads that place bandwidth demand on the platform. For this exercise, we chose TPC-C as an example.

To understand the bandwidth demand on the on-die interconnect (between L2 and L3), we first measured the number of L2 misses per instruction (MPI). Figures 3(a)



(a) TPC-C L2 MPI (b) TPC-C On-Die Bandwidth (c) TPC-C Off-Die Bandwidth

Figure 3. MPI and Bandwidth Characteristics of LCMP Server Platforms

shows the respective L2 MPI data as a function of L2 cache size per core. Note that there is a single L2 cache per node and all of the cores share the cache. For example, the data point corresponding to 128K L2 per core and 2 cores/node represents the configuration that has a 256K cache shared by the 2 cores in the node.

The data in the figures clearly shows that it is best to share the cache space across 4 cores in the node as opposed to having private caches per core. This is because at these small to moderate cache sizes, a large fraction of the cache is occupied by code which is shared by many threads. Replicating the code in private caches obviously wastes space; hence, shared caches provide significant performance/area benefit for CMP architectures [16]. In addition, it is worth noting that the L2 MPI reduces significantly when going from 256K to 512K. Therefore, the 512K cache size appears to be a sweet spot for this workload in such a configuration.

Figures 5 (b) shows the on-die bandwidth demand with 8 nodes and 4 cores per node (since this had the lowest MPI). We estimated the bandwidth demand for three cases: (i) with no L3, (ii) with a 32M L3 and (iii) with a perfect L3. The use of a perfect cache points to the maximum demanded on-die bandwidth. We can see that the maximum bandwidth demand for TPCC appears to be ~180 GB/s. With a large L3 cache, the bandwidth demands reduce significantly to the range of 50 to 100 GB/s. Since it would be preferable that the interconnect utilization is low (avoiding high queuing delays or saturation), it is clear that an on-die interconnect with 200 GB/s sustainable data bandwidth or more would be sufficient for the LCMP architecture.

Off-die memory bandwidth is also a key consideration when determining the cache hierarchy. For example, if sufficient interconnect bandwidth is available, but the memory bandwidth is meager, it is desirable to allocate more space to the L3 as opposed to the L2. Figure 5(c) shows the memory bandwidth demands for three configurations: (i) with no L3 cache, (ii) with a 16M L3, and (ii) with a 32M L3 cache. As we can see, TPC-C memory bandwidth demands range between 50GB/s and 75 GB/s. With a 32 MB L3 cache, the memory bandwidth demands reduce down to 40 GB/s.

In order to have a low memory utilization (< 50%), it is important that a 32-core LCMP memory subsystem (with a L3 cache) provide a sustainable bandwidth of over 100 GB/s. Based on DDR/FBD memory trends [8, 9], we expect that towards the end of the decade (when 45nm is available), the peak memory bandwidth will be around 64 to 128 GB/s. Applying a 64GB/s constraint essentially shows that the "no L3" options are not viable for TPC-C.

4.3. Summary of Cache Hierarchy Options

Based on the area and bandwidth constraints, we were able to prune the design space sufficiently and summarize a smaller set of configurations as listed in Table 2. The major factors that affected the pruning process are:

- Applying area constraints showed that around 128K to 256K per core seems viable.
- Applying area constraints resulted in L3 sizes ranging from 8M to about 18M depending on the configuration being considered.
- Applying bandwidth constraints essentially showed that configurations without L3 cache were not viable (due to memory bandwidth constraints).

Table 2: LCMP Cache Options Summary

Cores per node	Number of nodes	L2 cache per node	L3 Cache size
1	32	128K	~ 12M
2	16	256K – 512K	8M – 16M
4	8	512K – 1M	10M – 18M

5. LCMP Cache Hierarchy Performance

In this section, we study the performance of the LCMP cache hierarchy options summarized in Table 2. The metrics used in this section are both performance and performance/area, although area constraints have already been applied to prune the design space sufficiently.

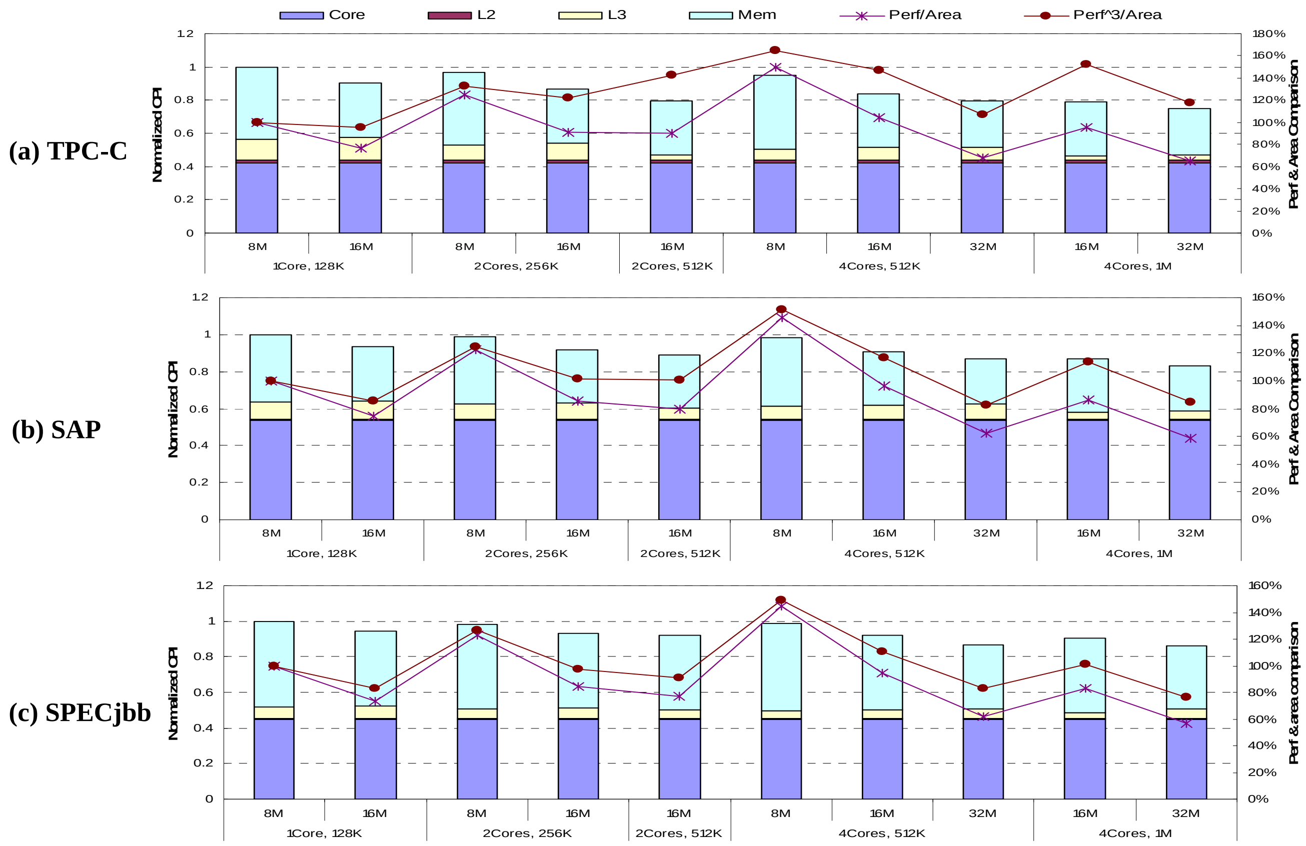


Figure 4. Detailed Comparison of the 32-core LCMP Cache Hierarchy Options

5.1. Performance of LCMP Cache Options

Figure 4 shows the performance data that we collected for various cache hierarchy options. The variation in the number of cores per node, the L2 cache size per node and the L3 cache size are shown on the x-axis. The on-die bandwidth is 512 GB/s and the maximum sustainable memory bandwidth is 64 GB/s. The vertical bars in Figure 6 show the CPI broken down into the time spent in the core, between the core and L2, between the L2 and L3 and finally in the memory subsystem. A simple observation is that the dominating factors are the performance of the core and the performance of the memory subsystem as the time spent in L2 & L3 subsystems are fairly low. It should be noted that this observation may change if the interconnect and cache bandwidth levels in the platform are modified. This will be discussed in a subsequent section.

From a performance (CPI) perspective, it is not surprising that the configuration that performs the best is the one to the far right (with 4 cores per node, 1M L2 per node and 32 M of L3 cache). It should be noted that in all configurations except those with 32M, the area consumed remains between 170 mm^2 and $350mm^2$. If we exclude the options with 32M L3 cache, then the high performance option is the 4-core node configuration with 16M L3 cache and either 1M or even 512K of L2 cache per node.

However, this performance comes at the expense of additional area. In order to comprehend performance and area together, we first looked at performance per unit area (shown as a line with values on the 2^{nd} y-axis). Placing equal emphasis on performance and area shows that the configuration with the least amount of cache area (4 cores per node, 512K per node, 8M L3) turns out to be the best configuration. However, this is primarily due to the fact that the difference in area dominates the comparison. In order to emphasize performance more than area, we then looked at $performance^3$/area as a potential metric. The behavior of the two metrics (performance/area and $performance^3$/area) is significantly different for TPC-C, but not as much for other two workloads. The reason for minimal change with other workloads is their insensitivity to cache size beyond 8M and minimal performance impact as a result. However, TPC-C is very memory-intensive (the former being more sensitive to cache size and latency, and the latter being very sensitive to memory bandwidth). As a result, the performance (CPI) is affected significantly for these workloads, thus placing less emphasis on area becomes important for these workloads. Overall, since the area constraints have already been applied (except for options with 32M L3 cache), the design option that provides good performance with low area overhead consists of 4 cores per node, 512K to 1M of L2 cache and 16M L3 cache.

5.2. Summary of Recommendations

Based on the constraints-aware analysis methodology and the results presented in this section as well as the previous one, our recommendation for LCMP architecture would be to design a 3-level cache hierarchy with 512K to 1M of L2 cache per node, where each node consists of four cores. The L3 cache size is recommended to be a minimum of 16M in order to minimize memory stall time as well as reduce the memory bandwidth pressure. We also did extensive sensitivity studies by varying core performance and bandwidth availability and data are not shown here due to space limitation. Based on the sensitivity studies, we recommend that the platform support at least 64GB/s of memory bandwidth and 512GB/s of interconnect bandwidth as shown by the sensitivity studies presented in the previous subsection.

6. Conclusions and Future Work

In this paper, we performed the first study of performance, area and bandwidth implications on LCMP cache design exploration. We introduced a constraints-aware analysis methodology for exploring the LCMP cache hierarchy options. We applied this methodology to a 32-core LCMP architecture and showed how the pruning process works. Applying these constraints quickly narrowed down the design space to a small subset of viable options and helped us focus our attention on these. We then conducted an in-depth performance/area evaluation of these options to summarize a set of recommendations for architecting efficient LCMP platforms.

A summary of the observations made in our LCMP cache design space study is as follows:

- Applying area constraints showed that around 128K per core seems viable for 32-core LCMP, whereas between 128K and 256K L2 per core seems viable for 16-core configurations
- Applying area constraints resulted in L3 sizes ranging from 8M to about 20M depending on the configuration being considered.
- Applying bandwidth constraints essentially showed that configurations without L3 cache were not viable (due to memory bandwidth constraints)
- A deeper study of the performance/area comparison of the area/bandwidth-viable options shows that the best performance option that does not exceed the area constraints is that of 4 cores per node, 512K to 1M cache per node and 16M of L3 cache.

References

[1] Azul Compute Appliance," Azul Systems, can be found at http://www.azulsystems.com/products/cpools_cappliance.html

[2] J. Baer, and W. Wang, "On the inclusion properties for multi-level cache hierarchies," 15th Annual international Symposium on Computer Architecture (ISCA), 1988.

[3] R.I. Bahar, G. Albera, and S. Manne. "Power and performance tradeoffs using various caching strategies", In Proc. of Int'l Symposium on Low-PowerElectronics and Design, 1998.

[4] D. Bhandarkar. "Billion Transistor Chips in Mainstream Enterprise Platforms of the Future," Keynote Speech, 9th International Symposium on High-Performance Computer Architecture (HPCA'03), Feb 8-12, 2003.

[5] M. Bohr, "Intel's 90nm technology: Moore's law and more," Intel Developer's Forum, available at ftp://download.intel.com/technology/silicon/Bohr_IDF_0902.pdf

[6] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Optimizing Replication, Communication, and Capacity Allocation in CMPs," In Proceedings of the 32nd International Sympoisum on Computer Architecture (ISCA), June 2005.

[7] S. Choudhary, P. Caprioli, et al. "High Performance Throughput Computing," IEEE Micro 2005.

[8] R. Faramarzi, "High Speed Trends in Memory," available at http://www.jedex.org/images/pdf/reza_hynix_keynote.pdf

[9] "FBDIMMS – A Revolutionary New Approach to Memory Modules," available at http://www.micron.com/products/modules/ddr2sdram/fbdimm.html

[10] Intel Corporation. "Intel Dual-Core Processors -- The First in the Multi-core Revolution," http://www.intel.com/technology/computing/dual-core/

[11] C. Kim, D. Burger, S. W. Keckler, "Nonuniform Cache Architectures for Wire-Delay Dominated On-Chip Caches," IEEE Micro 23(6): 99-107 (2003)

[12] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-Way Multithreaded Sparc Processor," IEEE Micro 25, 21-29, Mar. 2005

[13] K. Krewell, "Best Servers of 2004: Where Multicore is Norm," Microprocessor Report, Jan 2005.

[14] J. Laudon, "Performance/Watt: The New Server Focus," 1st Workshop on Design, Architecture and Simulation of CMP (dasCMP), Nov 2005.

[15] C. Liu, A. Sivasubramaniam, and M. Kandemir, "Organizing the Last Line of Defense before Hitting the Memory Wall for CMPs," 10th IEEE Symposium on High-Performance Computer Architecture, Feb. 2004.

[16] K. Olukotun, B. A. Nayfeh , et. al., "The case for a single-chip multiprocessor," Proceedings of the 7th International Conference on Architectural support for Programming Languages and Operating Systems, October 01-04, 1996.

[17] Sap America Inc., "SAP Standard Benchmarks," http://www.sap.com/solutions/benchmark/index.epx

[18] P. Shivakumar and N. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power,and Area Model," WRL Research Report 2001/2, August 2001.

[19] SPECjbb2005, http://www.spec.org/jbb2005/

[20] E. Speight, H. Shafi, L. Zhang, R. Rajamony, "Adaptive Mechanisms and Policies for Managing Cache Hierarchies in Chip Multiprocessors," 32nd International Sympoisum on Computer Architecture (ISCA), June 2005.

[21] C. Su and A. M. Despain "Cache design trade-offs for power and performance optimization: a case study," In Proceedings of the 1995 international Symposium on Low Power Design (ISLPED '95), April, 1995.

[22] "TPC-C Design Document", http://www.tpc.org/tpcc/

[23] M. Zhang and K. Asanovic, "Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors," 32rd International Symposium on Computer Architecture (ISCA-32), Madison, 2005.