An Analytical Model to Study Optimal Area Breakdown between Cores and Caches in a Chip Multiprocessor

Taecheol Oh Hyunjin Lee Kiyeon Lee Sangyeun Cho
Department of Computer Science
University of Pittsburgh

Abstract

A key design issue for chip multiprocessors (CMPs) is how to exploit the finite chip area to get the best system throughput. The most dominant area-consuming components in a CMP are processor cores and caches today. There is an important trade-off between the number of cores and the amount of cache in a single CMP chip. If we have too few cores, the system throughput will be limited by the number of threads. If we have too small cache capacity, the system may perform poorly due to frequent cache misses. This paper presents a simple and effective analytical model to study the trade-off of the core count and the cache capacity in a CMP under a finite die area constraint. Our model differentiates shared, private, and hybrid cache organizations. Our work will complement more detailed yet time-consuming simulation approaches by enabling one to quickly study how key chip area allocation parameters affect the system performance.

1. Introduction

The last decade has seen the emergence and proliferation of various general-purpose chip multiprocessor (CMP) architectures. Today, CMPs are deployed in all major market segments including PCs, servers, game consoles, and mobile devices. A typical CMP chip has multiple identical processor cores and large on-chip cache memory. A multitude of threads co-exist at a given time to perform a task in parallel fashion. On-chip caches facilitate fast retrieval of data so that the cores can make good progress with computation.

Historical technology trends and predictions reveal that the chip size of future high-performance CPUs will stay relatively constant [1]. An important question for a CMP architect is then how to break down a fixed chip area to various processor components, with each new technology generation. Given that processor cores and caches are the two most dominant area consumers, the question boils down to: How many cores (or how much cache capacity) shall we integrate on a chip? More detailed questions can follow. How many cache levels should be integrated? How should the caches be organized, shared or private? This paper introduces an analytical model to derive an optimal area breakdown between cores and caches in a CMP design under a finite die area constraint. The model also estimates the trade-off between shared, private, and hybrid cache organizations.

Previously, Zhao et al. [2] introduced a constraint-aware design analysis method and showed how to prune the cache design space. They considered design constraints such as area, bandwidth, and performance. Unlike our work where the focus is to hit the trade-off between the core count and the cache capacity, their architecture has fixed 32 cores. Huh et al. [3] explored the design space of the CMP architecture and studied the area and performance trade-offs. However, they only considered CMP organizations where L1 and L2 caches are coupled to individual cores and there is no inter-processor sharing. Alameldeen [4] derived a simple analytical model that exposes the trade-off between the core count and the cache capacity. Because the goal of their study was to measure the impact of their cache compression techniques, they limited themselves to the shared L2 cache organization in their CMP model and did not consider an L3 cache. Hill and Marty [5] applied Amdahl's Law to a CMP chip and build a cost model to obtain speedups for symmetric, asymmetric, and dynamic CMP chips. They assume that a CMP chip of a given size can have at most N *BCEs (Base Core Equivalents)*, where a BCE implements the baseline core. However, they do not consider chip resources expended on shared caches and do not account for the trade-offs between cores and caches.

The rest of this paper is organized as follows. Section 2 builds a background for the paper. Section 3 and 4 present in detail our analytical model with and without an L3 cache. Section 5 provides validation results. Section 6 gives a case study using our model, followed by conclusions in Section 7.

2. Background

2.1. Chip area trend

ITRS predicts that the Moore's Law rate of on-chip transistor density improvement will be maintained in the near future [1]. It also forecasts that the high-end microprocessor chip size, based on the competitive requirements for affordability and power management, will remain constant ($\sim$310 mm^2). Hence, with each new technology generation, core count and the cache capacity in a microprocessor will double under an idealistic scaling assumption.

Let's take Sun Niagara 2 as an example [6]. It features the smallest core among the general-purpose high-end processors built with a 65-nm technology [7–9]. Compared with its predecessor Niagara 1 built with a 90-nm technology,

Gate length	65 nm	32 nm
Core size	18.5 mm^2	5 mm^2
Area for 1MB cache	15 mm^2	4 mm^2

Table 1. Estimates of component sizes for a 32-nm CMP based on the Niagara 2 implementation (65 nm) [10].

the Niagara 2 chip has shrunk in size (from 378 mm^2 to 342 mm^2) and has a larger L2 cache (from 3MB to 4MB). This relatively small increase in L2 cache capacity is due to the addition of more thread contexts (from 32 threads to 64 threads) and a floating-point unit to each core as well as more system components such as network processing units.

Our case study in Section 6 will use the same chip size as Niagara 2 (342 mm^2) and use the component sizes derived from Niagara 2. Without loss of generality we estimate the component sizes (core and cache) of Niagara 2 by measuring the published die photograph. We then convert the values to a 32-nm technology node as shown in Table 1.

Given a die area, the number of cores and cache capacity, the following inequality holds:

$$A \leq N \cdot A_{core} + A_{L2}\ (+\ A_{L3}\ )$$

where A is the chip area, N the core count, A_{core} the core area, A_{L2} the L2 cache area, and A_{L3} the L3 cache area.

2.2. Unit area model

In this work, we use a unit area model similar to *cache byte equivalent area* (CBE) [3, 4]. CBE is the area for one byte of cache memory. We can express chip area, core area, and cache area in terms of CBE. Similarly, we define A_1 as the chip area equivalent to a 1MB cache area and use it as a unit area. Given the definition, the following is derived:

$$\begin{aligned} A_{cache} &= A - N \cdot A_{core} \\ &= m \cdot A_1 - N \cdot c \cdot A_1 = A_1(m - c \cdot N) \end{aligned} \quad (1)$$

where m and c are design parameters. The actual capacity of the L2 cache (in MBs) S_{L2} is given by A_{cache}/A_1 and has the value $(m - c \cdot N)$.

2.3. Throughput model

We use IPC (Instructions Per Cycle) as the metric to report system throughput. To compute IPC, we obtain CPI (Cycles Per Instruction, reciprocal of IPC) of individual processors first. A processor's "ideal" CPI can be obtained with an infinite L2 cache. In reality, the available cache capacity is limited and it results in performance loss [11, 12]. Hence,

$$CPI = CPI_{ideal} + CPI_{finite\ cache\ penalty} \quad (2)$$

$CPI_{finite\ cache\ penalty}$ can be obtained by dividing the aggregate value of CPU cycles lost in handling cache misses with the total number of instructions. (2) can be rewritten:

$$CPI = CPI_{ideal} + mpi(L2size) \cdot mp_{_M} \quad (3)$$

where $mpi(L2size)$ is the number of misses per instruction for a given cache size $L2size$ (in MBs) and $mp_{_M}$ is the average number of cycles needed to access memory and handle an L2 cache miss. Once we obtain individual processor's CPI, the system throughput IPC_{system} can be obtained from the individual CPI values. Because we assume that all the available processors are symmetric, we have $IPC_{system} = N/CPI$.

3. Model without L3 Cache

We first consider a CMP with L2 cache memory but no L3 cache memory. Many CMP systems fall into this category.

3.1. Private L2 Cache

Private L2 cache offers low access latency but may suffer from many cache misses due to its increasingly limited caching capacity as we add more cores [13]. From (2) we derive the CPI for a processor with a private L2 cache.

$$CPI = CPI_{pr} + mpi(S_{L2p}) \cdot mp_{_M} \quad (4)$$

where CPI_{pr} is the CPI with an infinite private L2 cache. The per-core private cache area and size are given by:

$$A_{L2p} = \frac{A_{L2}}{N}, \quad S_{L2p} = \frac{S_{L2}}{N} \quad (5)$$

This work uses the square root rule of thumb [14, 15] to define $mpi()$. Given $mpi(1)$ (misses per instruction with a 1MB cache), the following captures the rule:

$$mpi(S_{L2p}) = mpi(1) \cdot \sqrt{\frac{1}{S_{L2p}}} \quad (6)$$

From equations (4), (5) and (6),

$$CPI = CPI_{pr} + mpi(1) \cdot \sqrt{\frac{A_1 \cdot N}{A_{cache}}} \cdot mp_{_M} \quad (7)$$

3.2. Shared L2 Cache

Uniform Cache Architecture (UCA). The CPI of one core in a CMP with N processor cores sharing an L2 cache [15] is

$$CPI = CPI_{sh} + mpi(S_{L2sh}) \cdot mp_{_M} \quad (8)$$

where CPI_{sh} is the CPI with an infinite shared L2 cache and S_{L2sh} is the effective cache capacity ($<S_{L2}$) seen by each core. CPI_{sh} is usually larger than CPI_{pr} because private caches have a lower latency. S_{L2sh} is likely larger than S_{L2p} because there are cache blocks being shared by multiple cores and a core may borrow caching space from another core that does not require a large cache capacity. If a cache block is shared by N_{sh} cores on average [4], we have

$$S_{L2sh} = \frac{S_{L2}}{N - N_{sh} + 1} = \frac{A_{cache}}{A_1 \cdot (N - N_{sh} + 1)} \quad (9)$$

One can set N_{sh} to be a constant value or a more general function (e.g., $\lambda \cdot N$) based on system characterization.

With an UCA design processor cores experience the same latency to the L2 caches. On the other hand, they may see contentions in the network and the cache ports, especially when there are many cores. We introduce this contention factor in our CPI calculation. For instance, the time spent on contention can be expressed as a linear function $\beta \cdot (N-1)$. β is computed from the ratio of memory operations to instructions, the L1 cache miss rate, and the probability of having to pay the bandwidth penalty. With this assumption and equations (6), (8) and (9) we have

$$CPI = CPI_{sh} + \beta \cdot (N-1) + mpi(S_{L2sh}) \cdot mp_{_M} \quad (10)$$

Non-Uniform Cache Architecture (NUCA). The NUCA approach relaxes the uniform latency constraint in multi-bank cache designs and typically employs a scalable switched network such as the 2-D mesh [16]. With R rows and C columns, the maximum hop distance in a 2-D mesh network is $(R + C - 2)$ and the average hop distance is $\frac{1}{3}(R + C)$. Assuming $R = C$, the average hop distance becomes $\frac{2}{3}\sqrt{N}$ where N is the core count. If t_r is the single-hop network traversal latency, the expected on-chip network traverse latency in a NUCA chip will be $\frac{2}{3}\sqrt{N} \cdot t_r$. Assuming that this network traversal latency is dominant over the network bandwidth penalty in a NUCA design:

$$CPI = CPI_{sh} + \gamma \cdot \frac{2}{3}\sqrt{N} \cdot t_r + mpi(S_{L2sh}) \cdot mp_{_M} \quad (11)$$

where γ is computed from the ratio of memory operations to instructions and the L1 cache miss rate. Notice the similarity in (10) and (11); their difference comes mainly from the assumptions about the property of the network.

3.3. Hybrid L2 Cache

A hybrid cache tries to capture the strengths of a private cache (smaller hit latency) and a shared cache (larger capacity) [16]. We model a hybrid L2 cache by computing the latency penalty offered by a private cache and any latency savings by additional capacity borrowed from other caches on chip. As in the private caching scheme, each core gets a private cache whose size is S_{L2p}. When a cache miss occurs in the private cache, one may find a cache block in some other cache slices on chip. Effectively, each core gets a "larger" cache than S_{L2p}. We introduce a parameter σ to specify this effective cache size relative to S_{L2p}. Importantly, we have two disparate cache hit latencies now; on a remote cache hit, we pay a higher latency penalty than the local private cache. Assuming the same 2-D mesh network we used in the shared NUCA cache discussion.

$$\begin{aligned} CPI &= CPI_{pr} + mpi(S_{L2p}) \cdot \frac{2}{3}\sqrt{N} \cdot t_r \\ &\quad + mpi(\sigma \cdot S_{L2p}) \cdot mp_{_M} \end{aligned} \quad (12)$$

The actual value of σ is determined by the hybrid cache scheme and the workload used. For instance, Zhang and Asanović [16] show that their hybrid scheme can often cut the off-chip miss rate of the private caching scheme by half.

4. Model with On-Chip L2 and L3 Caches

For a conventional inclusive cache hierarchy, there needs to be enough area given to the L3 cache–at least 2× or more than the L2 cache [2]. We introduce a parameter α to divide the available cache area between the L2 and L3 caches ($A_{L2} = \alpha \cdot A_{cache}$, $A_{L3} = (1-\alpha) \cdot A_{cache}$).

4.1. Private L2 cache and shared L3 cache

In order to expose the effect of on-chip L3 cache, we split the finite cache CPI penalty in (4) into L2 and L3 components.

$$\begin{aligned} CPI &= CPI_{pr} + mpi(S_{L2p}) \cdot mp_{_{L3}} \\ &\quad + mpi(S_{L3sh}) \cdot mp_{_M} \end{aligned} \quad (13)$$

where S_{L2p} is now $\alpha \cdot A_{cache}/A_1$ and S_{L3sh} is the effective per-core capacity of the L3 cache. Thus we have

$$S_{L3sh} = \frac{(1-\alpha) \cdot A_{cache}}{A_1 \cdot (N - N_{sh_{L3}} + 1)} \quad (14)$$

where $N_{sh_{L3}}$ is the average number of sharers per L3 block.

4.2. Shared L2 Cache and shared L3 cache

Uniform Cache Architecture (UCA). We re-write (10) as

$$\begin{aligned} CPI &= CPI_{sh} + \beta \cdot (N-1) \\ &\quad + mpi(S_{L2sh}) \cdot mp_{_{L3}} \\ &\quad + mpi(S_{L3sh}) \cdot mp_{_M} \end{aligned} \quad (15)$$

In the above S_{L2sh} is the value in (9) scaled by α and S_{L3sh} is same as (14).

Non-Uniform Cache Architecture (NUCA). We re-write (11) as

$$\begin{aligned} CPI &= CPI_{sh} + \gamma \cdot \frac{2}{3}\sqrt{N} \cdot t_r \\ &\quad + mpi(S_{L2sh}) \cdot mp_{_{L3}} \\ &\quad + mpi(S_{L3sh}) \cdot mp_{_M} \end{aligned} \quad (16)$$

where S_{L2sh} and S_{L3sh} are identical to those of UCA.

4.3. Hybrid L2 cache and shared L3 cache

After taking into account the effect of L3 cache and extending equation (12) we have

$$\begin{aligned} CPI &= CPI_{pr} \\ &\quad + mpi(S_{L2p}) \cdot \frac{2}{3}\sqrt{N} \cdot t_r \\ &\quad + mpi(\sigma \cdot S_{L2p}) \cdot mp_{_{L3}} \\ &\quad + mpi(S_{L3sh}) \cdot mp_{_M} \end{aligned} \quad (17)$$

where S_{L2p} and S_{L3sh} are defined as in Section 4.1.

4.4. Off-chip L3 cache

Our discussions so far have been limited to processor designs with and without on-chip L3 caches and have not considered off-chip L3 cache. However, our model can be easily

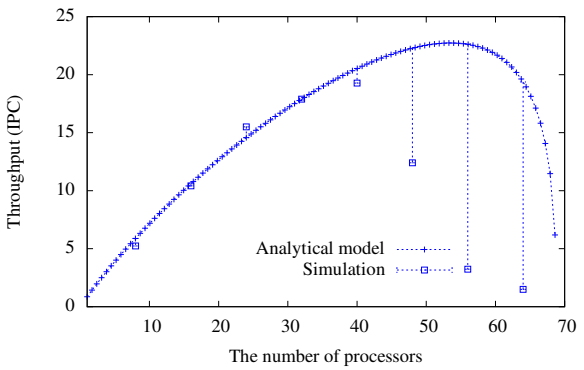

Figure 1. Comparing the outcomes of the proposed model (NUCA) and the simulation. The collapse point was between 40 and 48.

extended to handle the case. Because the off-chip L3 cache capacity is not dependent on the core count or the L2 cache capacity of the processor chip, we can simply introduce a new parameter to directly specify the off-chip L3 cache capacity and factor in its impact on the latency penalty. Assuming a private on-chip L2 cache scheme, we have

$$CPI = CPI_{pr} + mpi(S_{L2p}) \cdot mp_{L3} + mpi(S_{L3}) \cdot mp_{M} \quad (18)$$

where S_{L2p} is simply S_{L2}/N as in (5). We do not show the formulas for the shared and hybrid schemes here.

5. Validation

We validated our analytical model by comparing its estimated IPC predictions with elaborate simulation results. We used the TPTS simulator [17] to model a multicore processor chip with in-order cores. We construct synthetic parallel workloads by running multiple copies of a single-thread program and treating some of the memory pages as "shared" pages between threads. Benchmarks were drawn from the SPEC2k CPU suite, including *art, gcc, facerec,* and *ammp*. After obtaining the basic parameters (L1 cache miss rate and base CPI) from simulation, we used the analytical model to estimate the IPC. Simulations were then repeated with different core counts to generate sample points for comparison. Figure 1 shows the result for the benchmark *ammp* and the NUCA cache type. Overall, we observe that our model is in good agreement with the simulation. We found that after a specific core count, however, the agreement ends and the result of simulation drops down much more drastically than that of our model. The "collapse point" was at about 48 cores for NUCA and UCA, and about 40 cores for Private and Hybrid. This occurs because the square root rule of thumb [14, 15] is not accurate if the cache becomes smaller than a certain size. We do not pursue to develop a more accurate cache miss model, which is beyond the scope of this paper. The average RMS error before the collapse point was found to be small: 5.2 % (NUCA), 4.2 % (UCA), 5.5 % (Private) and 6.5 % (Hybrid).

Parameter	Description	Value
CPI_{pr}	CPI with infinite private L2 cache	1.36
CPI_{sh}	CPI with infinite shared L2 cache	1.54
mp_{M}	Memory access penalty	400
mp_{L3}	L3 cache access penalty (on-chip/off-chip)	15/80
$mpi(1)$	Misses per instruction (1MB cache)	0.006
t_r	Network traversal latency per hop	4
σ	Cache expansion factor (hybrid cache)	1.2
α	Cache area division (L2 vs. L3) factor	<1
β	Bandwidth penalty factor (UCA)	0.024
γ	Bandwidth penalty factor (NUCA)	0.03
λ	Degree of cache block sharing	0.5

Table 2. Model parameters.

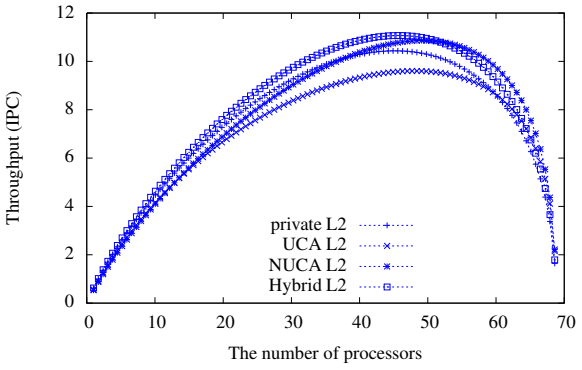

Figure 2. IPC with on-chip L2 cache (no L3 cache).

6. Case Study

In this section, we present a case study that uses the presented analytical model. We employ a hypothetical benchmark to clearly reveal the properties of different cache organizations and the capability of our model. Table 2 shows our base parameters obtained experimentally from the SPEC2k CPU benchmark suite. We study L2 cache configurations with and without an L3 cache, both on-chip and off-chip. The main metric is system IPC (throughput). In all experiments in this section, we change the number of processor cores and show how that affects the throughput. With the given chip area (342 mm^2), core size (5 mm^2), and the 1MB cache area (4 mm^2), a CMP chip can hold at most 68 cores (with no caches) and at most 86 MB of cache capacity (with no cores).

Comparing private, shared, and hybrid caches. Figure 2 presents the result for the private cache scheme, the shared cache scheme (UCA and NUCA), and the hybrid cache scheme without an L3 cache. Their performance peaks at the core count of 45 (private), 48 (UCA), 50 (NUCA), and 47 (hybrid), respectively. It is shown that the shared scheme can exploit more cores as it provides relatively more caching capacity than the private and the hybrid scheme. Given the parameter set, however, the hybrid cache exhibits the best

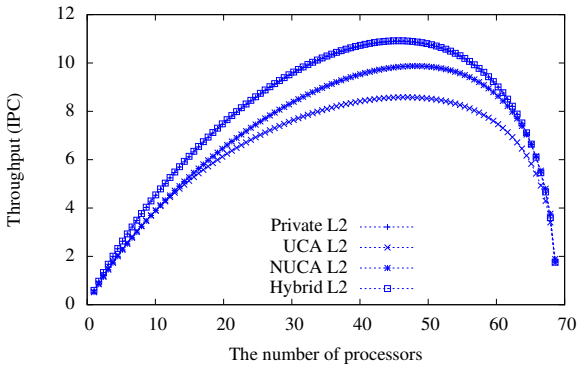

Figure 3. IPC with on-chip L2, L3 cache ($\alpha = 0.2$).

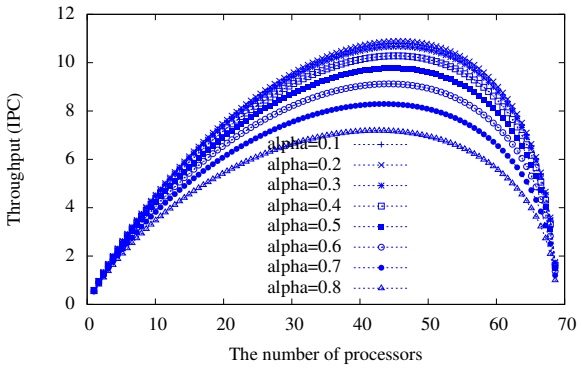

Figure 4. Effect of changing α, private L2 and shared L3.

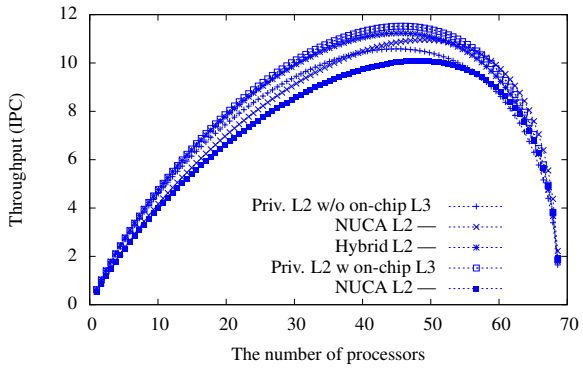

Figure 5. Comparison of cache organizations with and without on-chip L3 cache ($\alpha = 0.2$).

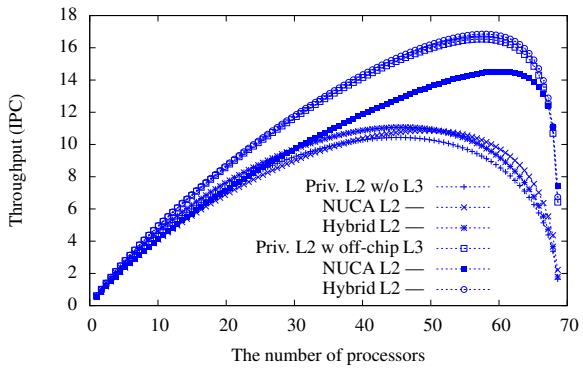

Figure 6. IPC with and without off-chip L3 cache.

performance among all the schemes. The hybrid cache benefited from low local hit latency (same as that of the private cache) and more on-chip cache hits in remote cache slices. The peak performance of each scheme was: 11.05 (hybrid), 10.85 (NUCA), 10.43 (private), and 9.60 (UCA) in the decreasing order. After reaching the peak, the throughput of the CMP drops quickly as we add more cores. This is because the caching space given to each core is reduced and the performance benefit of adding more cores is quickly offset by the increase in cache misses. Due to the similarity in their IPC formula, the private scheme and the hybrid scheme have a similar performance curve. By the same token, the UCA and the NUCA schemes show a similar curve as well.

Effect of on-chip L3 cache. We have previously observed that the hybrid scheme, taking advantage of low cache hit latency and larger effective caching capacity, performed better than the private and the shared scheme. How would the picture change if we have an on-chip L3 cache? Figure 3 presents the performance curves of the same cache organizations, this time with an L3 cache. It is shown that the distance between performance peaks has narrowed. Their performance peaks at the core count of 46 (private), 47 (UCA), 48 (NUCA), and 46 (hybrid), respectively. Moreover, the private cache scheme and the hybrid cache scheme noticeably outperform the shared schemes. This is because the relatively high miss rate of the private and hybrid cache scheme is compensated by the L3 cache and its on-chip cache hit rate matches that of the shared schemes. Under the given parameter set, the performance difference of the private and the hybrid schemes was negligible. Because the hybrid cache management is more complex than that of the private cache, the private cache becomes more favorable in the presence of the on-chip L3 cache. How much area we allocate to L2 and L3 caches becomes an interesting design question now. Figure 4 depicts how α, the parameter that governs the area allocation between the L2 and L3 caches, changes processor throughput. In this experiment, we use the private scheme at L2. It is shown that among the values we assigned to α, 0.2 produces the best throughput. The throughput continuously improves as we change α from 0.8 down to 0.2. Below $\alpha = 0.2$ (at $\alpha = 0.1$), however, the performance starts to degrade. At the bottom line, it is suggested that we need to allocate much larger area for L3 (80% of A_{cache}) than L2 to hit the right point in the trade-off between more L2 hits (fast access) and more L3 hits (higher on-chip hit rate). Figure 5 presents the throughput of a few selected cache schemes with and without on-chip L3 cache. Among all the schemes we examined, the (private L2 + shared L3) scheme performed the best, followed by the hybrid L2 cache

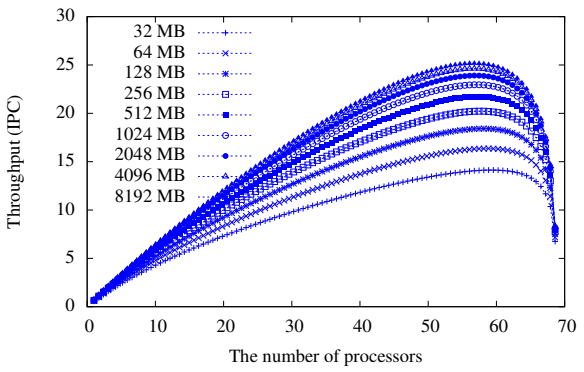

Figure 7. Impact of the off-chip L3 cache size. The private scheme is used in the on-chip L2 cache.

scheme. In the case of the shared scheme (NUCA), adding an L3 cache degrades the performance. Again, what is revealed in this comparison is that taking the best of the private and the shared strategies results in high performance. We observe that the performance benefit of having the two levels of cache memory is limited; the hybrid L2 cache scheme performed close to the (private L2 + shared L3) scheme. One reason for this limitation is the *inclusiveness* we maintain between the L2 and L3 caches. Even though we allocate a large capacity at L3, much of the space is used to keep the duplicate copies of data in the L2 caches. Given this, a possible optimization for the two-level on-chip cache hierarchy is to introduce *non-inclusiveness* between L2 and L3 caches. In fact, Dorsey et al. [8] describe such a non-inclusive cache design. We leave exploring such a design as a future work.

Effect of off-chip L3 cache. Figure 6 shows the same curves with and without an off-chip L3 cache of 128 MB. The private and the hybrid schemes benefit from the off-chip L3 cache the most and their maximum throughput improvement amounts to 52%. The core count at the performance peak is much larger, close to 60 in all the schemes. As with the on-chip L3 cache, the private and the hybrid cache schemes are favored over the shared cache scheme. Finally, Figure 7 shows how the off-chip L3 cache size affects the system throughput. It is observed that at a large core count ($>$40) the L3 cache size has a large impact on the system throughput. Doubling the cache size from 32MB boosts the throughput by at least 10% until the cache size reaches 256 MB. Beyond 256 MB, however, the throughput gain is diminishing quickly. Considering the cost, 0.5$\sim$2 GB of L3 cache capacity gives a reasonable system throughput.

7. Conclusion

In this paper, we presented a simple and effective analytical model to study the trade-off between the core count and the cache capacity in a CMP under a finite die area constraint. Our model differentiates shared, private, and hybrid cache organizations. The presented model enables one to quickly study how key chip area allocation parameters affect the system performance. The model input includes: CMP chip area, core size, core count, miss rate for the 1MB cache, and L2 cache miss penalty. To evaluate the effectiveness of our approach, we performed a case study where we use a set of parameter values derived from a hypothetical benchmark. The result shows that different cache organizations have different optimal core and cache area breakdown points. For instance, the shared cache organization leads to more cores at its peak performance point than the private and hybrid cache. Another finding is that when there is a shared L3 cache, the private and the hybrid schemes produce more performance than the shared schemes.

As future work we plan to incorporate accurate area models for various on-chip networks and a CMP power model. We plan to extend our model to study asymmetric CMPs.

Acknowledgment

This work was supported in part by NSF grant CCF-0702236 and an A. Richard Newton Graduate Scholarship from the 45th Design Automation Conf. (DAC), 2008. Authors thank the anonymous reviewers for their constructive comments.

References

[1] ITRS (International Technology Roadmap for Semiconductors), http://www.itrs.net/reports.html.

[2] L. Zhao, R. Iyer, S. Makineni, J. Moses, R. Illikkal, and D. Newell, "Performance, area and bandwidth implications on large-scale cmp cache design," HPCA, 2007.

[3] J. Huh, D. Burger, and S. W. Keckler, "Exploring the design space of future cmps," PACT, 2001.

[4] A. R. Alameldeen, *Using compression to improve chip multiprocessor performance*. PhD thesis, 2006.

[5] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, pp. 33–38, July 2008.

[6] U. M. Nawathe, "An 8-core 64-thread 64bit power-efficient sparc soc," ISSCC, 2007.

[7] J. Friedrich, "Design of the power6 microprocessor," ISSCC, 2007.

[8] J. Dorsey, "An integrated quad-core opteron processor," ISSCC, 2007.

[9] N. Sakran, "The implementation of the 65nm dual-core 64b merom processor," ISSCC, 2007.

[10] A. B. Kahng, "Design challenges at 65nm and beyond," DATE, 2007.

[11] P. G. Emma, "Understanding some simple processor-performance limits," *IBM J. Res. Dev.*, vol. 41, no. 3, pp. 215–232, 1997.

[12] R. E. Matick, T. J. Heller, and M. Ignatowski, "Analytical analysis of finite cache penalty and cycles per instruction of a multiprocessor memory hierarchy using miss rates and queuing theory," *IBM J. Res. Dev.*, vol. 45, no. 6, pp. 819–842, 2001.

[13] X. Zhao, K. Sammut, F. He, and S. Qin, "Split private and shared l2 cache architecture for snooping-based cmp," *ICIS*, pp. 900–905, 2007.

[14] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 3rd ed., 2003.

[15] K. A. Bowman, A. R. Alameldeen, S. T. Srinivasan, and C. B. Wilkerson, "Impact of die-to-die and within-die parameter variations on the throughput distribution of multi-core processors," ISLPED, 2007.

[16] M. Zhang and K. Asanovic, "Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors," ISCA, 2005.

[17] S. Cho, S. Demetriades, S. Evans, L. Jin, H. Lee, K. Lee, and M. Moeng, "TPTS: A novel framework for very fast manycore processor architecture simulation.," ICPP, 2008.