Nahalal: Cache Organization for Chip Multiprocessors

Zvika Guz[1], Idit Keidar[2], Avinoam Kolodny[2], Uri C. Weiser[2]
Electrical Engineering Department, Technion, Haifa, Israel
[1]zguz@tx.technion.ac.il [2]{idish, kolodny, uri.weiser}@ee.technion.ac.il

***Abstract*— This paper addresses cache organization in Chip Multiprocessors (CMPs). We show that in CMP systems it is valuable to distinguish between *shared data*, which is accessed by multiple cores, and *private data* accessed by a single core. We introduce Nahalal, an architecture whose novel floorplan topology partitions cached data according to its usage (shared versus private data), and thus enables fast access to shared data for all processors while preserving the vicinity of private data to each processor. Nahalal exhibits significant improvements in cache access latency compared to a traditional cache design.**

I. Introduction

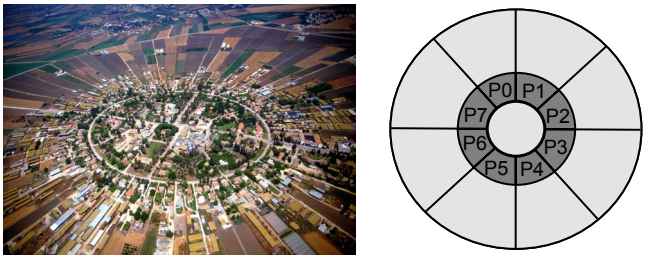
Chip Multiprocessors (CMPs) are rapidly becoming mainstream thanks to their ability to leverage the parallelism of multithreading and multitasking to achieve higher performance within a given power envelope. This paradigm shift towards on-chip mutli-processing environments introduces new computer architecture challenges. In particular, cache data access is often a principal bottleneck in such systems, as multiple threads compete for limited on-die memory resources and accessibility. Hence, CMP-tailored cache architecture, organization, and management are critical for system performance with the CMP paradigm. This paper introduces a new memory subsystem architecture for the emerging CMP environment.

Uniprocessor microarchitectures typically partition the cache based on content (e.g., instructions versus data cache in the Harvard architecture) and hierarchically (e.g., cache levels: L1, L2, etc.). We argue that with the shift to CMP architectures, additional cache dimensions will prove valuable, for example, based on data sharing, data coherency, and other CMP characteristics. In this paper, we develop the CMP data sharing paradigm. We distinguish between *shared data*, which is accessed by multiple cores, and *private data* accessed by a single core. Our study of typical multi-processing applications (Section III) shows that such a distinction is often clear cut, both spatially and temporally, as shared data is mostly shared by many threads throughout the program's entire life span.

Sharing of cached data hampers CMP performance for two main reasons: access contention, and wire delays. As global wire delays become a dominant factor in VLSI design [7], on-chip access time increasingly depends on the distance between the processor and the data, and thus the accesses are *non-uniform* across the chip. Consequently, data should be kept close to a processor that accesses it. In current designs, the L2 cache is typically located as a bulk in one location (e.g., at the center, surrounded by all CMP processors), causing shared data to inevitably reside far from at least some of it sharers, and hence suffer from long access times.

As we further show in Section III, in many multithreaded applications, a substantial fraction of the memory accesses involves cache lines that are shared by many processors. Furthermore, in commercial workloads, a significant fraction of memory accesses involve modified-shared data, which cannot be replicated without performance penalty for ensuring cache coherence. Consequently, accesses to shared data severely penalize the average memory access time and hinder overall performance. These phenomena call for a new cache architecture that explicitly accounts for data sharing.

We propose Nahalal (Section IV), a new CMP cache architecture that partitions the L2 cache according to the programs' data sharing, and can thus offer vicinity of reference to both shared and private data. The topology was inspired by the layout of the cooperative village Nahalal, shown in Fig. 1(a), which is based on urban design ideas from the 19th century [8]. In Nahalal, public buildings (school, administrative offices, warehouses, etc.) are located in an inner core circle, enclosed by a circle of homesteads. Private tracts of land are arranged in outer circles, each in proximity to its owner's house.



(a) Aerial view of Nahalal Village. (b) CMP conceptual layout scheme.

Fig. 1. Nahalal

We project the same conceptual layout to CMP, as schematically illustrated in Fig. 1(b). (We give a more practical rectangular layout in Section IV.) A fraction of the L2 memory is located in the center of the chip, enclosed by all processors, while the rest of the L2 memory is placed on the outer slices. The inner memory is populated by the hottest shared data, allowing fast access by all processors. The outer slices create a "backyard" for each processor.

Manuscript submitted: 24-Apr-2007. Manuscript accepted: 23-May-2007.
Final manuscript received: 29-May-2007.

We demonstrate the Nahalal concept with a simple implementation example in Section IV. In Section V, we show that this implementation of Nahalal improves L2 cache access times by up to 41% compared to traditional CMP designs. Beyond this particular example, the Nahalal concept of placing "public" data in a location easily accessible by all sharers may be more broadly applicable, and is expected to benefit performance, reduced power and improve available bandwidth. Conclusions and future research directions are outlined in Section VI.

II. Related Work

Recent studies of CMP cache organization usually locate the L2 cache in the center of the chip, surrounding it by all processors [1][2][4][5][9]. We henceforth refer to this layout as Cache-In-the-Middle (CIM).

Beckmann et al. [1][2] have studied memory access patterns in CMP. They identified the imbalance between the number of accesses to shared cache lines and the number of shared cache lines in the working set, and pointed out the importance of shared lines to overall memory performance. Our cache access characterization (Section III) continues this line of work.

Previous works on CMP cache design have recognized the need for shared L2 caches that follow a Non Uniform Cache Architecture (NUCA), where access times vary according to the distance between the data and the client processor [1][2][5][9][12]. In this context, both Beckmann et al. [2] and Huh et al. [9] have studied the applicability of Dynamic NUCA (DNUCA) to CMPs. DNUCA uses line migration in order to move frequently accessed data closer to processors that use it. This policy is intended to reduce access times to the most frequently accessed data, and thus reduce the average access time compared to static line placement. Nevertheless, both works have concluded that access to shared data hinders the effectiveness of DNUCA, since shared data, being pulled by several processors to different directions, ends up in the middle of the chip, far from all of them. We believe that the Nahalal concept of bringing shared data close to all processors can solve this Achilles heel of DNUCA, and may provide a platform where DNUCA can realize its potential [6].

Several works have used line replication to ease the shared data problem [1][5][12]. Such replication, however, reduces the effective cache capacity, further increasing the on-chip capacity pressure. Moreover, line replication is only cost-effective when the shared lines are read-only, since writing entails invalidation of all copies which may impact performance. The work presented in this paper reduces the need for replication by allowing a single copy to reside close to all the processors that share it.

III. Memory Access Characterization

In this section, we characterize memory access patterns occurring in multithreaded workloads. We study sample scientific benchmarks from the Splash-2 and SPEComp kits, as well as three commercial workloads (*apache*, *zeus*, and *SPECjbb*). We use the Pin program analysis tool [10] to profile accesses to each cache line (either L1 or L2). Details regarding system parameters are given in Table I.

TABLE I - System Parameters

Parameter	Value
Number of cores	8
Cache line size	64B
Instruction window. A line is considered shared if multiple processors access it within this window	10,000,000

The profile results, summarized in Table II, lead to several observations.

TABLE II - Cache Line Sharing Characteristics

Sample Benchmarks		Shared lines		Modified shared lines	
		Percentage out of all accesses	Percentage out of all lines	Percentage out of all accesses	Percentage out of all lines
SPEComp	equake	32.05	0.73	2.78	0.40
	fma3d	8.93	0.16	0.37	0.14
Splash2	barnes	15.36	7.07	3.14	0.61
	water	24.90	11.96	17.55	10.85
Commercial	apache	58.25	34.33	47.91	25.26
	zeus	56.85	37.76	41.64	28.16
	specjbb	44.24	13.78	15.39	0.89

First, in many workloads, accesses to shared data comprise a substantial fraction of the total memory accesses (e.g., up to 58.25% in the *apache* workload). Moreover, in commercial workloads, many of these accesses involve shared lines which are modified by at least one of the sharers (e.g., in *apache*, 82% of the memory accesses to shared data are to modified shared lines).

Second, there is a clear discrepancy between the number of accesses to shared lines and the number of shared lines in the working set: a small number of cache lines, shared by many processors, accounts for a significant fraction of the total accesses to memory [1][2]. We dub this phenomenon the *shared hot lines* effect. Furthermore, we observe that a small number of shared lines- some very hot lines- are more popular than others, accounting for most of the accesses to shared data. This phenomenon is shown in Fig. 2.

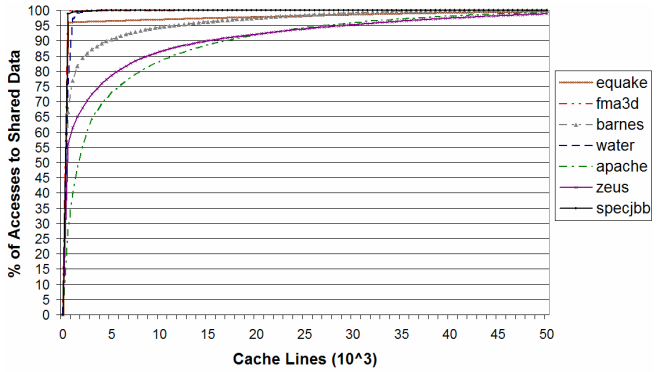


Fig. 2. Access distribution of shared cache lines. X-axis marks the number of shared lines (in thousand), and Y-axis marks the percent of accesses out of the total accesses to shared data that targeted these lines.

We have also found that typically the same data is shared throughout the program's lifetime; and that shared data is

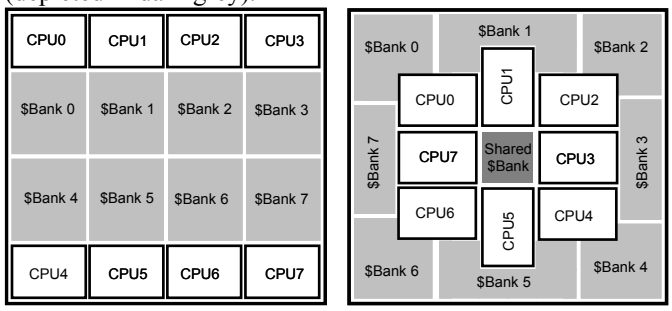
typically shared by many processors. Together, our observations indicate that reducing the access latency to a modest number of shared data lines can have a significant impact on CMP performance; and that a small subset of the memory capacity suffices for holding the shared lines to which the majority of accesses are made.

IV. Example Cache Organization and Management

We now discuss a possible realization of the Nahalal concept in the context of an 8-way CMP.

A. Nahalal Layout

The main concept in Nahalal is placing shared data in a relatively small area in the middle of the chip, surrounded by the processors, and locating private data in the periphery. This solution is feasible thanks to the *shared hot lines* phenomenon, which suggests that a relatively small structure is sufficient for serving the majority of shared data accesses. Fig. 3(a) depicts a general organization for an 8-way CMP where the L2 cache is located in the middle (CIM) and Fig. 3(b) portrays an alternative layout based on the Nahalal concept. In both designs, each processor has a private L1 cache, and all processors share an L2 cache. The L2 cache capacity is partitioned among the processors such that each processor has one cache bank in its proximity (depicted in light grey). In Nahalal, some of the total L2 cache capacity is designated for shared data only, and is located at the center (depicted in dark grey).



(a) CIM layout. (b) Nahalal layout.

Fig. 3. Two cache organizations for an 8-way CMP.

B. Nahalal Cache Management

The Nahalal scheme can be implement via a broad range of potential cache management strategies. Following, we present one possible solution.

1) Placement and Migration

In both implementations shown in Fig. 3, each address can be located in any of the banks. Thus, cache management needs to decide where to place the line when it is fetched, and subsequently, if and when to migrate a line from its current bank, and to where (another bank or be evicted from the cache).

In both CIM and Nahalal implementations, on a first line fetch, the line is placed in the bank adjacent to the processor that made the request. In our example of CIM, the line remains in its initial location as long as it is in the cache. In contrast, in Nahalal, we use migration to steer *shared-hot-lines* to the center. In order to prevent pollution of the shared cache area, a line is migrated to the shared structure only after N accesses (for some threshold N) from different processors. Although an adaptive threshold may be of merit, for the sake of simplicity, we use a fixed threshold of 8 accesses (3 bits per cache line) in our simulations.

Whenever a cache line needs to be evicted from the center (shared cache) in order to allow for a new shared line to move in, the least recently used line (LRU) of all cache lines in the center is evicted. (The victim line is not evicted from the cache, but merely the two lines switch locations.) Since saving usage time statistics for all the lines in the center may be unrealistic in terms of hardware complexity, we organize the shared area as an 8-way cache structure, tracking LRU statistics over each set. Such a structure is more feasible in terms of hardware complexity, while still keeping the most used shared lines in the center.

2) Search

In Nahalal, L2 cache lines are likely to be served either from the center (for shared data) or from the local cache structure (for private data). This is contrary to CIM, where shared lines can be located in any of the different banks with equal probability. (Thus, on average, shared lines are fetched from distant cache banks in CIM.) For simplicity, in our simulations we use a parallel search for both the CIM and Nahalal implementation. Such a search favors CIM, as it mitigates the delay of fetching a line from remote banks.

TABLE III - System Parameters

Parameter	Value
L1,L2 line size	64B, 64B
Private L1 caches size, ways, access	32KB, 2-way, 3 cycle
L2 cache size, ways (per bank), bank access	16MB, 8-way, 15 cycles
Connectivity delay between adjacent banks	5 cycles
Main memory access	300 cycles

V. Results

A. Methodology

To demonstrate the potential performance gain of the Nahalal topology (Fig. 3b) over a CIM topology (Fig. 3a), we consider an 8-processor CMP, where each processor has a private 32KB L1 cache, and all processors share a 16MB L2 cache. For the CIM layout, we divide the L2 cache capacity between the eight cores such that every core has 2MB in its proximity. For the Nahalal topology, every processor has 1.875MB of capacity in its "private yard" (bank), and all processors share a 1MB designated shared bank. More details regarding system parameters are given in Table III. Our evaluation uses a full-system simulation in Simics [11], using the x86 in-order processor model as a building block.

B. Results

1) Cache delay/Relative distance

Fig. 4 presents average L2 cache access times for Nahalal and CIM in a number of benchmarks. Nahalal achieves superior results in all benchmarks, reducing the average L2 cache access time by 27.41% on average over CIM, to the extent of 41.1% for the *apache* benchmark.

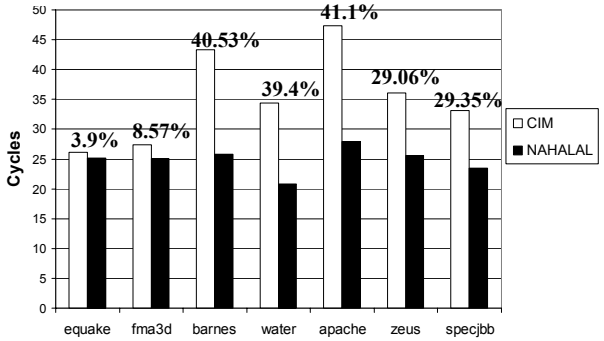


Fig. 4. Average L2 cache access time. (Labels indicate the relative reduction in L2 hit time of Nahalal over CIM.)

Nahalal's gain stems from faster access to shared data, as shown in Fig. 5. In both layouts, most of the private data is located in the local banks of each processor. But while Nahalal is able to serve most of the accesses to shared data from the shared designated structure, in the CIM design most of the accesses to shared data are served from remote banks, thus suffering long access times. Consequently, Nahalal is able to provide a local-like average access time even for benchmarks with many accesses to share data.

Nahalal's gain over traditional CMP cache design increases with the frequency of access to shared data. The exact extent of this gain in each benchmark (Fig. 4) depends on the percentage of accesses it makes to shared data among L2 accesses, which we do not show here for space limitations.

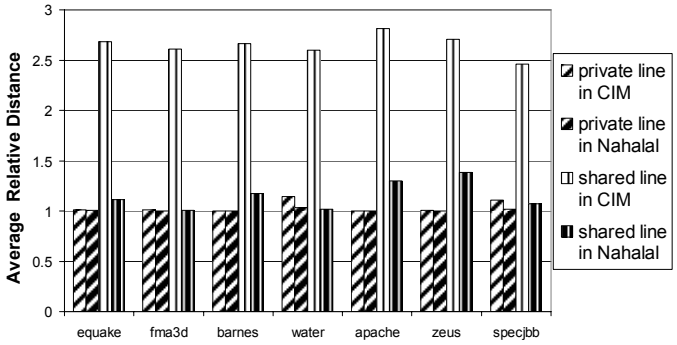


Fig. 5. Average relative distance to shared versus private lines for CIM and Nahalal. Relative distance is the distance in banks between the data and the processor (a relative distance of 1 is defined as an access to the most adjacent bank, or to the shared bank in Nahalal).

2) Overall performance

The average speedup in total execution time of Nahalal over CIM is 5.84% across the different benchmarks, up to 9.32% and 12.65% in *zeus* and *apache* benchmarks respectively. Nahalal is most advantageous when there is a substantial number of accesses to shared data in L2. Thus, for benchmarks with low L2 access rate (e.g., *barnes*) where L2 is not the bottleneck, or for benchmarks with almost no sharing (e.g. *fma3d*), Nahalal's superiority is not effectively realized in overall speedup. In the future, one can expect CMP applications to have larger memory demands and exhibit more sharing, while the growing in wire delays will increase the importance of locality of reference. Therefore, the benefits of Nahalal will become more significant.

VI. Summary and Future Directions

The shift towards Chip Multi Processors makes the on-chip memory system a primary performance bottleneck. This shift calls for a new approach to cache design, in which cache architecture will be tailored and optimized for the multiprocessing environment. We have proposed partitioning the cache in CMPs according to the level of data sharing. This approach is motivated by the observation that, in many multithreaded applications, a small set of shared cache lines accounts for a significant portion of the memory accesses. We have presented Nahalal – a novel CMP cache architecture and floorplan that exhibits shorter access distances to shared data compared to the conventional CMP with cache-in-the-middle (CIM) architecture. We have shown that Nahalal improves average L2 cache access times by up to 41%.

Handling shared data is only one of numerous challenges unique to CMP systems. We believe that the CMP era calls for a paradigm change in many aspects, including accesses to memory, memory sharing, cache affinity, interrupt handling, and interconnects. Consequently, the Nahalal scheme can be combined with other CMP-tailored solutions. In particular, we believe that many of the suggested CMP cache designs may benefit from the Nahalal concept; we are now investigating implementations of Nahalal in caches with more aggressive line migration policies (e.g., CMP-DNUCA [2]) and caches built upon state-of-the-art Networks-on-Chip interconnects[3].

Acknowledgment

We thank Avi Mendelson, Ronny Ronen, and Rachel Sebba for their insightful comments; and Evgeny Bolotin, Tomer Morad, and Isask'har Walter for their help.

References

[1] B. M. Beckmann, M. R. Marty, and D. A. Wood, "ASR: Adaptive Selective Replication for CMP Caches," MICRO 39, December 2006

[2] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip multiprocessor caches," MICRO 37, pages 319-330, Dec. 2004

[3] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS Architecture and Design Process for Network on Chip," Special issue on Networks on Chip, The Journal of Systems Architecture, 2004

[4] J. Chang and G. S. Sohi. "Cooperative Caching for Chip Multiprocessors," ISCA-33, June 2006

[5] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Optimizing Replication, Communication, and Capacity Allocation in CMPs," ISCA32, pages: 357 - 368, 2005

[6] Z. Guz, I. Keidar, A. Kolodny, and U. C. Weiser, "Nahalal: Memory Organization for Chip Multiprocessors," Technical Report CCIT 600, Technion Department of Electrical Engineering, September 2006

[7] R. Ho, K. Mai, and M. Horowitz, "The future of wires," Proceedings of IEEE,89(4), April 2001.

[8] E. Howard, "Garden Cities of To-Morrow," London: Swan Sonnenschein & Co. Ltd, 1902

[9] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S.W. Keckler, "A NUCA substrate for Flexible CMP Cache Sharing," ICS 05, June, 2005

[10] C.K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, and K. Hazelwood, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," PLDI 2005.

[11] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, and G. Hallberg, "Simics: A full system simulation platform," IEEE Computer, 35(2):50–58, Feb. 2002.

[12] M. Zhang and K. Asanovic, "Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors" ISCA32, 2005.