A Fine-Grained Cache with Adaptive Merged Block

Chao Zhang
Lehigh University
Email: chz616@lehigh.edu

Yuan Zeng
Lehigh University
Email: yuz615@lehigh.edu

Xiaochen Guo
Lehigh University
Email: xig515@lehigh.edu

I. Background and Motivation

Conventional cache moves a block of contiguous data and allocates space for them together, which works well for applications with good spatial locality. However, many applications have poor spatial locality. In the multi-core and multi-program system, contentions at each level of the memory hierarchy can cause early cache block eviction and reduce the benefit of locality. When running these applications, conventional cache suffers from poor effective cache capacity and high data movement overhead. Storing data that will not be used before eviction reduces the effective capacity and degrades performance. In addition, data over-fetching from main memory to cache wastes bandwidth and energy.

Fine-grained caches [1], which use small cache block size and only fetch useful data to cache, have been proposed to improve effective capacity. However, the size of the metadata needs to be increased to support data identification, state tracking, and replacement at a smaller data granularity. To reduce the metadata overhead, one approach is to share the common tag bits among multiple fine-grained blocks within one merged block [2] . However, in order to share the common tag bits, data that can be stored in the same merged block has constraints. A merged block may not be completely filled and therefore can waste cache space.

II. Design Challenges and Key Ideas of the Proposed Design

To improve the effective capacity, an ideal fine-grained cache should have 1) low metadata overhead, 2) high data utilization (useful data/fetched data), and 3) high block utilization (fetched data/allocated space). The proposed work takes a systematic approach to maximize the effective capacity and to improve data movement efficiency by balancing these three factors.

Reduce metadata overhead: Sharing the common bits in the tag of multiple fine-grained blocks can reduce the metadata overhead. The fine-grained blocks that can be merged together must have the same shared tag and index. Each of the fine-grained blocks has their unique private tag. The proposed work maximized the length of the shared tag to reduce metadata overhead. To further reduce the metadata overhead, the proposed design adaptively increases the merged block size to share common bits among as many fine-grained blocks as possible.

Improve block utilization: Prior works [2] keep a fixed sized of merged block and a static private tag position. The length and the position of the private tag add constraints on which fine-grained blocks can be merged together. However, prior works [2] simply select the LSB of the address bits as the private tag without considering the application characteristics. Therefore, many merged blocks is not full because not enough number of fine-grained blocks can be merged together. This work proposes private tag selection to increase the possibility of merging more fine-grained blocks.

Improve data utilization: The unused data wastes both cache space and bandwidth. In the proposed design, a spatial pattern predictor is incorporated and optimized to eliminate unused data transferring.

Reduce control overhead: The total control overhead of writing the fine-grained block into the cache is higher as compared to the conventional cache design. In the proposed design, multiple fine-grained blocks are merged together at the memory controller. The control overhead can be reduced by using the same set of control signals to write multiple fine-grained blocks.

III. An Overview of the Proposed Fine-Grained Cache

Supporting merged blocks with variable length can improve the effective capacity and thereby reduce the miss rate. However, additional logic and tag bits should be added, which increases the cache access time. Therefore, in the proposed design, the L1 cache has a fixed merged block size to maintain low access delay; whereas the L2 cache has merged blocks with variable length to improve effective capacity and reduce the miss rate.

An overview of the proposed design is shown in the Figure 1. Both L1 and L2 caches can store merged blocks. In the 4-way L1 cache, each set stores 32 words in four fixed 64B merged block. For an 8-way L2 cache, the number of the merged blocks in one cache set can vary from one to sixteen. To identify each merged block, 16 shared tags are required to support the maximum number of merged blocks, which leads to a large metadata overhead. Moreover, to lookup for a merged block in a set, sixteen comparators are required if all of the merged blocks are 32B. To reduce the overhead, the proposed fine-grained cache limits the number of merged blocks such that the number of comparators for the shared tag remains the same as it is in the conventional cache. The merged blockIDs are used to indicate which shared tag should be used for each 32B space (minimum merged block granularity) in the data array. This mechanism allows non-contiguous cache space to be allocated to one merged block. Data transferring between L1 and L2 is chosen to be physical block granularity, which can improve the data movement efficiency. A spatial pattern predictor is located at the memory controller and trained by the

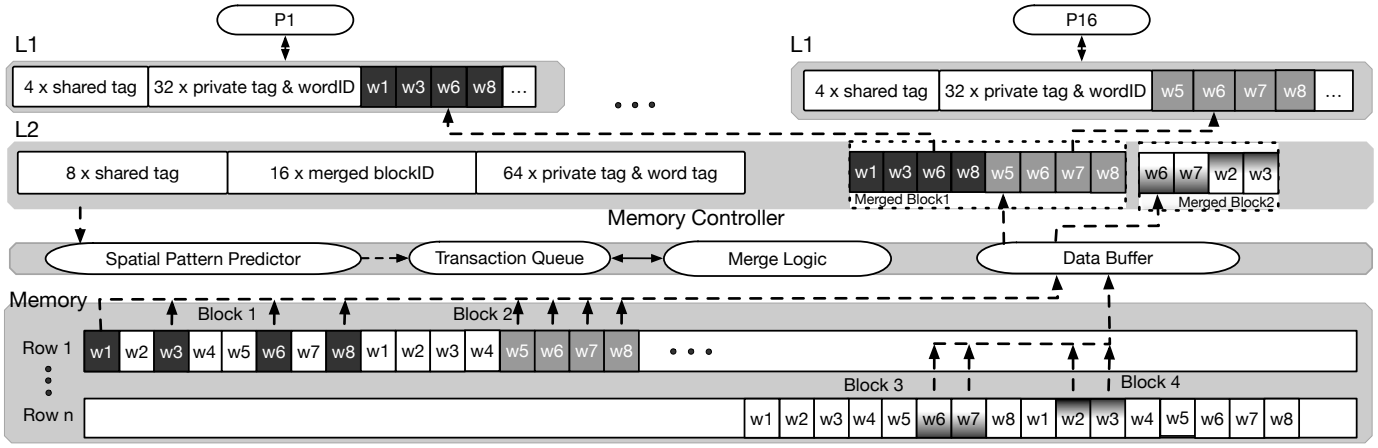

Fig. 1. The proposed design with 16 private 4-way L1s and a shared 8-way L2 cache.

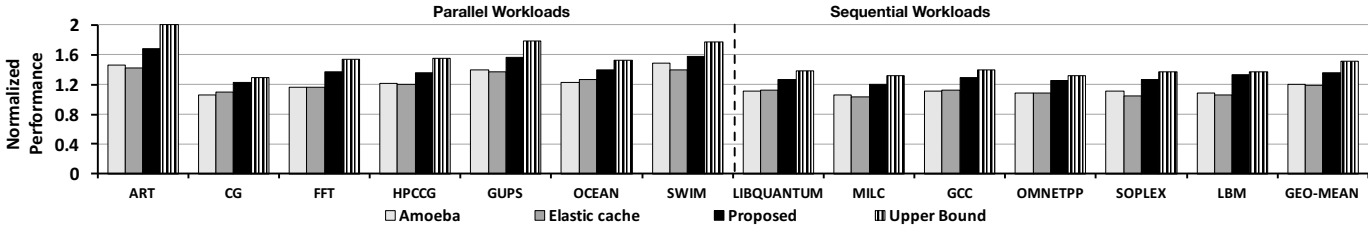

Fig. 2. Normalized performance. The proposed fine-grained cache has a fixed 64B merged block size for L1 and a variable merged block size (32-512B) for L2. The upper bound is estimated by increasing the cache capacity to have 100% of data utilization and space utilization based on the tested applications, which also has a compressed tag array using C-PACK+Z [3].

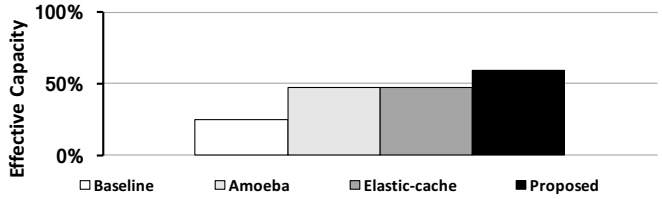

Fig. 3. Normalized effective capacity (GEO-MEAN).

eviction information. The merge logic in the memory controller determines which partial blocks can be merged together and the sizes of each merged block.

IV. Novelty and Contributions

This work makes the following contributions:

Merging fine-grained blocks with private tag selection. Sharing tag bits among fine-grained blocks reduce metadata overhead. Prior works use a long private tag to improve merging efficiency. Instead of increasing the length of the private tag, carefully selecting private tag can increase the merging possibility. This scheme reduces the metadata overhead by increasing the length of the shared tag and improving the block utilization.

Supporting variable length merged block. A large merged block can reduce the metadata overhead by sharing the tag bits with more fine-grained blocks; whereas a small merged block can improve the block utilization when there are not many fine-grained blocks that can be merged. Hence, this work supports variable length merged block to minimize the metadata overhead and improve block utilization.

Grouped line-fill to reduce control overhead. Multiple fine-grained blocks can share the same set of control signals if they can be merged and refilled together. A new merging logic is added at the memory controller to merge fine-grained blocks before accessing the DRAM. Hence, the fine-grained blocks within the same merged block are grouped before refilled to the last level cache. This scheme can improve both performance and bandwidth efficiency.

V. Main Results

This work evaluates the proposed design using HPCC, SPEC-OMP, SPLASH-2, Mantevo, and SPEC2006 benchmarks and is compared with Amoeba [1] and Elastic caches [2] on a 16-core system with 64KB private L1 caches and an 8MB shared L2 cache. As shown in the Fig. 2, the proposed design performs significantly better as compared to the baseline and other fine-grained caches. This is because the proposed design can achieve the highest effective cache capacity (Fig. 3).

References

[1] S. Kumar, H. Zhao, A. Shriraman, E. Matthews, S. Dwarkadas, and L. Shannon, "Amoeba-cache: Adaptive blocks for eliminating waste in the memory hierarchy," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-45. Washington, DC, USA: IEEE Computer Society, 2012, pp. 376–388. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2012.42

[2] B. Li, J. Sun, M. Annavaram, and N. S. Kim, "Elastic-cache: Gpu cache architecture for efficient fine-and coarse-grained cache-line management," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2017, pp. 82–91.

[3] X. Chen, L. Yang, R. P. Dick, L. Shang, and H. Lekatsas, "C-pack: A high-performance microprocessor cache compression algorithm," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 8, pp. 1196–1208, 2010.