Two novel cache management mechanisms on CPU-GPU heterogeneous processors

Huijing Yang* and Tingwen Yu
Beijing University of Technology, Beijing, China
{yangkx, ytw412}@emails.bjut.edu.cn

Abstract

Heterogeneous multicore processors that take full advantage of CPUs and GPUs within the same chip raise an emerging challenge for sharing a series of on-chip resources, particularly Last-Level Cache (LLC) resources. Since the GPU core has good parallelism and memory latency tolerance, the majority of the LLC space is utilized by GPU applications. Under the current cache management policies, the LLC sharing of CPU applications can be remarkably decreased due to the existence of GPU workloads, thus seriously affecting the overall performance. To alleviate the unfair contention within CPUs and GPUs for the cache capability, we propose two novel cache supervision mechanisms: static cache partitioning scheme based on adaptive replacement policy (SARP) and dynamic cache partitioning scheme based on GPU missing awareness (DGMA). SARP scheme first uses cache partitioning to split the cache ways between CPUs and GPUs and then uses adaptive cache replacement policy depending on the type of the requested message. DGMA scheme monitors GPU's cache performance metrics at run time and set appropriate threshold to dynamically change the cache ratio of the mutual LLC between various kernels. Experimental results show that SARP mechanism can further increase CPU performance, up to 32.6% and an average increase of 8.4%. And DGMA scheme improves CPU performance under the premise of ensuring that GPU performance is not affected, and achieves a maximum increase of 18.1% and an average increase of 7.7%.

Keywords: heterogeneous, multicore, CPU-GPU, cache partitioning

1 Introduction

With the continuous development of semiconductor industry technology, computer architecture is transitioning from the multicore era into the heterogeneous era. For the heterogeneous multi-core system that the CPU and GPU share the cache, the situation is different [7] , due to the GPU's Single-Instruction Multi-Data (SIMD) model, therefore the traditional LRU or FIFO algorithm cannot adapt to this special storage mode, which will seriously downgrade the performance of the system[14].

Recently, a rich body of state-of-art work has been done on certain applications combining the CPU and GPU[16]. For instance, as the CPU-GPU chip has seriously high energy consumption, e.g., the GPU-accelerated DeepMind pioneering system, renowned for the first time of beating the champion Go player, whose electricity bill is over 150 million each year[12]. Motivated by this, there are also some work focusing on energy efficiency, i.e., dynamic voltage and frequency scaling (DVFS) and dynamic resource sleep (DRS)[1]. Nevertheless, a large amount of GPU applications are memory-bounded and the attributes that will influence their performance are more than processor frequency[9], but also GPU memory management strategy. Moreover, a very material application, termed community detection, which is used to search within the community structures such as social networks or graph data[17].

Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 7, Article No. 1 (June 15, 2021)
DOI:*10.22667/ReBiCTE.2021.06.15.001*
*Corresponding author

Since the huge quantity and complex inner-relationships of structural data, a certain level of parallelism ought to be inevitably considered. However, it should be noted (for example, see [18], [5]) that even well-formed parallelized managements reveal the potential problems and risks of significant low speedups. The state-of-the-art cache replacement policies [8] applied Belady's algorithm to the history of memory accesses to help make better eviction or prefetch decisions. The above cache replacement algorithms are not effective for CPU-GPU heterogeneous architectures, but little cache management work has been done on CPU-GPU architectures[2],[13].

In our research, two novels shared LLC management policies for CPU-GPU heterogeneous system are furnished, including static cache partitioning scheme based on adaptive replacement policy (SARP) and dynamic cache partitioning scheme based on GPU missing awareness (DGMA). We evaluate the experiment results of two proposed scheduling plans, which show that the devised methods achieve higher system performance compared to the traditional LRU policy.

The rest of this paper is organized as follows. Section 2 briefly gives a systematic literature review. Section 3 proposes SARP cache management scheme. Section 4 proposes DGMA cache management scheme. Section 5 evaluates our approach based on experiment results. Section 6 concludes the paper.

2 Related Work

Due to the CPU and the GPU cores have completely different resource requirements [19], thread-level parallel awareness of the cache management policy (TAP) [20] used core sampling and cache block life cycle normalization to manage the shared cache for the CPU and GPU, the core sampling mechanism employs a simple heuristic method to choose the appropriate replacement policy according to the cache sensitivity of the GPU. The cache block lifecycle normalization technology detects the difference in cache access rates, using this information to make the data access time of cache for CPU and GPU applications approximately equal. The author [10] utilized a and fined-grained cache replacement method to deal with the CPU-GPU access behavior discrepancy among cache sets. Though it added additional hardware cost for the LLC miss counter. The author [4] propose a utility- and fairness-aware cache replacement policy in a SRAM- and STT-RAM-based hybrid LLC, which adjusts the priority and position of GPU cache blocks dynamically based on core sampling and cache occupation monitoring. The cache block lifecycle normalization technology detects the difference in cache access rates, using this information to make the data access time of cache for CPU and GPU applications approximately equal. The author [11] utilized a and fined-grained cache replacement method to deal with the CPU-GPU access behavior discrepancy among cache sets. Though it added additional hardware cost for LLC miss counter.

3 SARP: static cache partitioning scheme based on an adaptive replacement policy

In a heterogeneous environment, there is a certain degree of cache access differences between the CPU and GPU about LLC resources, properly managing and distributing shared LLC resources is critical to the performance of the whole system [6]. To avoid the excessive occupation of cache resources by the GPU and reduce the negative impact of the GPU on the CPU, this chapter proposes a Static Caching Partitioning Scheme Based on Adaptive Replacement Policy (SARP).

3.1 Cache Partitioning

To eliminate the unfair occupation of the shared LLC by the GPU, a cache partitioning mechanism is adopted [3]. By modifying the address mapping algorithm, the CPU and GPU data blocks are mapped

into different cache spaces respectively, so that the CPU and GPU have their own specific cache space, which avoids the mutual interference between the CPU and GPU.

3.2 SARP

The SARP mechanism in this chapter is a fusion of the cache partitioning mechanism and the adaptive cache replacement policy. Based on cache partitioning, according to the type of cache requests, an appropriate cache replacement policy is selected, so that the cache utilization of the CPU and GPU is further improved [15].

For the exact ratio of the LLC between the CPU and GPU, we compare two different split ratios,1:1 and 1:3 to test the effect of various ratios on the performance. Besides, when the LLC cache is full and data blocks need to be removed, we monitor the type of memory request. If it comes from the CPU application, we employ the classic LRU replacement strategy. If the request belongs to the GPU application, we use the improved tree-based pseudo-LRU replacement policy. So that the CPU and GPU requests are being processed in their respective address spaces and use their characteristic-based replacement algorithms.

4 DGMA: dynamic cache partition scheme based on GPU missing awareness

The static partitioning scheme has certain limitations, the partition ratio is set in advance and it will not be changed until the end. Therefore, the partitioning algorithm can be further optimized, and a dynamic partitioning mechanism based on GPU missing awareness is proposed to perform dynamic partitioning. So that CPU performance is increased as much as possible while ensuring that GPU performance is not impacted, which optimizes the overall performance.

According to the state of the GPU application's execution, DGMA dynamically monitors its performance metrics and changes the ratio of LLC based on performance metrics for the CPU and GPU, thereby maximizing the utilization efficiency of the shared LLC.

The workflow of DGMA can be divided into four steps. First, allocates an initial division ratio of L2 at the initial stage, and then performs statistical computations on the access information of the GPU at intervals of certain clock cycles. The cache miss rate is calculated based on the total number of accesses from the GPU and the number of missed memory accesses, and then the next division ratio is determined according to the partition rule. The specific threshold allocation process is as follows:

The specific threshold allocation process is as follows:

a) In the initial state, the CPU and GPU occupy the equal size of the LLC space.

b) If M_i is less than the $Threshold_{low}$, the division ratio of the shared LLC by the CPU and the GPU is 1:7.

c) If the GPU's M_i is calculated to be greater than or equal to the $Threshold_low$ and less than or equal to the $Threshold_{high}$, the CPU and GPU use the intermediate state 1:3 division ratio for the shared LLC.

d) If the M_i is greater than or equal to the $Threshold_{high}$, it indicates that the missing rate of the GPU application is large, GPU performance needs to be improved, and the cache partition ratio is restored to the initial value of 1:1.

e) At the beginning of the next cycle interval, M_i of the GPU application is calculated according to the GPU access information gathered during the current interval, the cache is allocated to CPU and GPU applications according to the rule of cache partitioning.

Through the above algorithm, the partitioning algorithm is executed periodically. At the beginning of the next cycle interval, according to the characteristics of the collected access requests, the best ratio

of the CPU and GPU for the LLC is established.

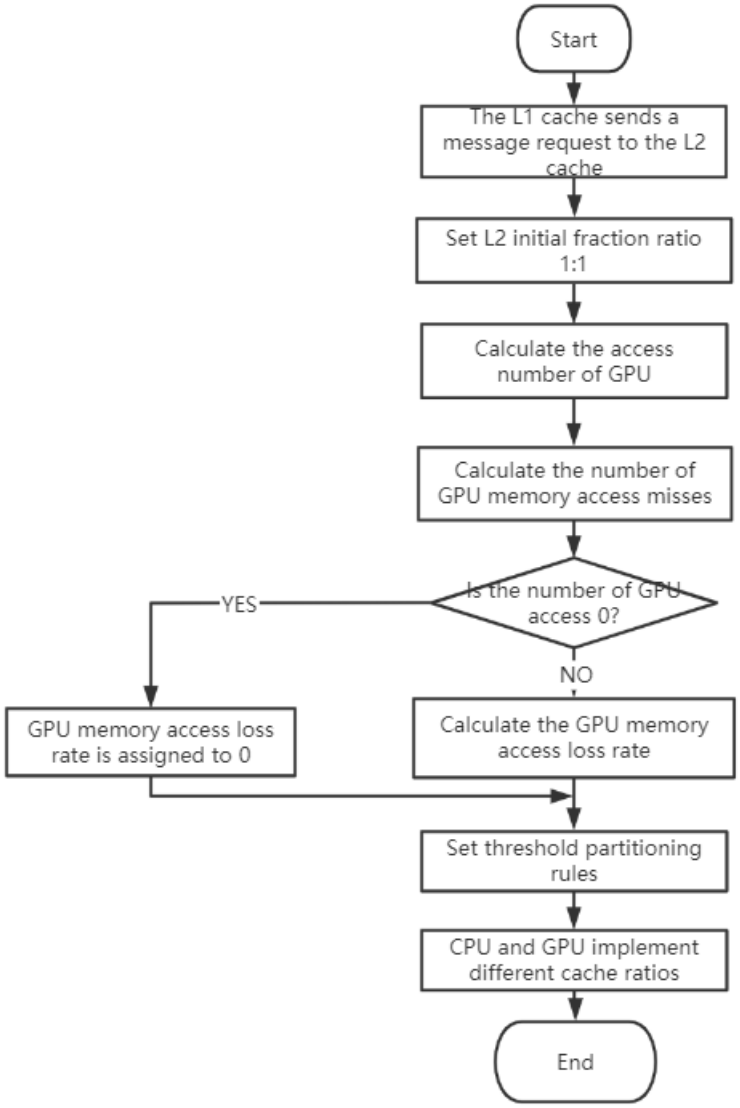

Figure 1: The flow of the execution of dynamic cache partitioning

5 Experiments and Evaluation

5.1 Simulator

We utilize the gem5-gpu to simulate heterogeneous multi-core architecture. This experiment set up two CPU cores, four GPU cores, all the cores share an L2 cache for the experimental standard. We use SPEC CPU2006 as the CPU workload. For GPU applications, Rodinia, Parboil, and NVidia CUDA SDK are chosen.

The experiment uses the geometric mean (Eq.1) as the speedup of each application for the main evaluation metric.

$$speedup = geomean(speedup_{(0...n-1)}) \tag{1}$$

IPC (Instruction per Clock) that is, the number of instructions executed per clock cycle, the calculation formula is shown in Eq.2.

$$IPC = \sum_{i=0}^{n-1} Intructions_i / Cycles \tag{2}$$

5.2 Evaluation

We review the benefit of SARP and DGMA of the shared LLC in a heterogeneous multicore processor. We compare the performance among SARP_1:1, SARP_1:3, DGMA, and LRU. SARP_1:1 represents the SARP algorithm where the CPU and GPU occupy an equal amount of LLC space. SARP_1:3 represents the SARP algorithm where the ratio of the LLC space that the CPU and GPU occupy is 3:1. Fig.1 shows the CPU speedup results compared to the LRU replacement policy on workloads.

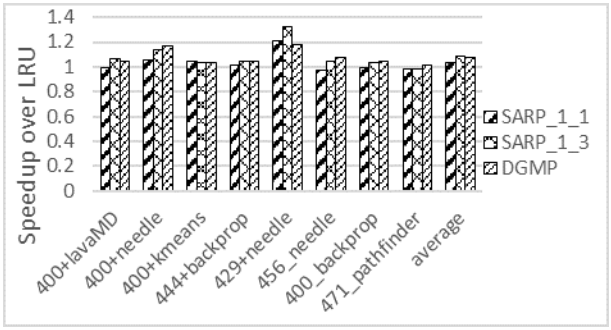

Figure 2: Speedup over LRU of CPU

The experiment results show that the SAR_1_1 partition scheme improves CPU performance over LRU by 3.4% on average, 21.5% on maximum. SARP_1_3 partition scheme improves CPU performance by 8.36% on average, max for 32.6 %, and DGMA algorithm improves 7.68% on average, max for 18.1%. The GPU speedup results are shown in Fig. 2.

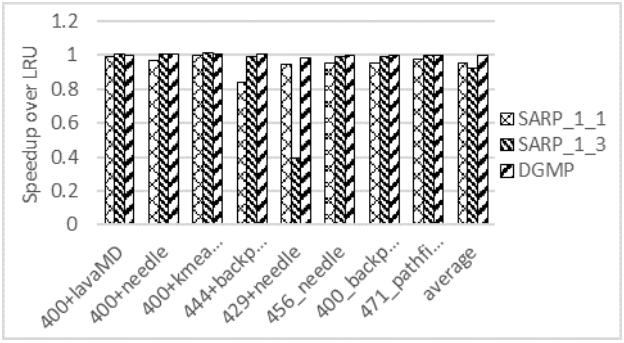

Figure 3: Speedup over LRU of GPU

As described above, DGMA aims to guarantee GPU performance. As we can see from the picture, for the SARP_1_1 partition, GPU performance is decreased by 4.6% on average, for the SARP_1_3 partition, GPU performance is reduced by 7.5% on average. But, GPU performance is lowered by only 0.1% when using the DGMA algorithm.

The result of combining the speedup of the CPU and GPU for all workloads is shown in Fig. 3.

The figure shows that DGMA outperforms compared with the other two partition schemes, on average, the DGMA improves system performance by 5% over LRU, 11% on maximum.

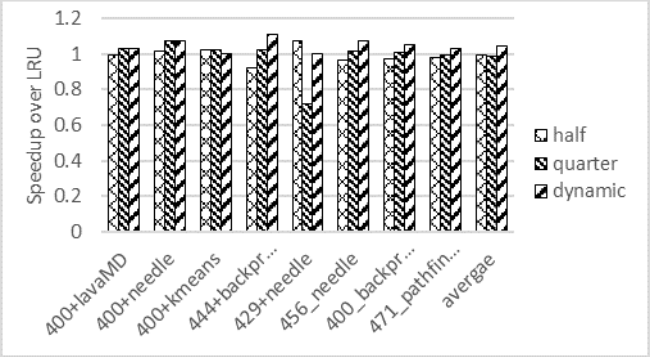

Figure 4: Speedup over LRU of CPU

6 Conclusion and Future works

In this paper, we introduced two algorithms for determining the partitioning and replacement policies of the last-level cache jointly used by CPU and GPU cores in a heterogeneous processor. The main novelty of the SAPR algorithm is partitioning with adaptive replacement policy, partitioning can avoid any interference with the performance of the application at runtime. DGMA dynamically monitors the GPU's miss rate during program execution and changes the space of LLC for the CPU and GPU based on performance metrics, the empirical results show that DGMA outperforms SARP with the same system, it can achieve better performance for both the CPU and GPU.

Funding

This research was funded by Beijing Natural Science Foundation, grant number 4192007 and the National Natural Science Foundation of China (61202076).

Acknowledgments

This work is supported by Beijing Natural Science Foundation (4192007), and supported by the National Natural Science Foundation of China (61202076), along with other government sponsors. The authors would like to thank the reviewers for their efforts and for providing helpful suggestions that have led to several important improvements in our work. We would also like to thank all teachers and students in our laboratory for helpful discussions.

References

[1] N. Antoniadis and A. Sifaleras. A hybrid CPU-GPU parallelization scheme of variable neighborhood search for inventory optimization problems. *Electronic Notes in Discrete Mathematics*, 58:47–54, 2017.

[2] O. S. Dong, G. B. Kim, J. M. Kim, and C. H. Kim. Cache reuse aware replacement policy for improving gpu cache performance. In *Proc. of the 2017 International Conference on IT Convergence and Security (ICITCS'17), Seoul, South Korea*, volume 450 of *Lecture Notes in Electrical Engineering*. Springer, Singapore, September 2017.

[3] P. Faldu, J. Diamond, and B. Grot. Domain-specialized cache management for graph analytics. In *Proc. of the 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA'20), San Diego, CA, USA*, pages 234–248. IEEE, February 2020.

[4] L. Gao, R. Wang, Y. Xu, H. Yang, Z. Luan, D. Qian, H. Zhang, and J. Cai. SRAM- and stt-ram-based hybrid, shared last-level cache for on-chip CPU-GPU heterogeneous architectures. *Journal of Supercomputing*, 74(7):3388–3414, 2018.

[5] V. Garcia, J. Gómez-Luna, T. Grass, A. Rico, E. Ayguadé, and A. J. Peña. Evaluating the effect of last-level cache sharing on integrated GPU-CPU systems with heterogeneous applications. In *Proc. of the 2016 IEEE International Symposium on Workload Characterization (IISWC'16), Providence, RI, USA*, pages 168–177. IEEE, September 2016.

[6] M. Gowanlock and B. Karsin. A hybrid CPU/GPU approach for optimizing sorting throughput. *Parallel Computing*, 85:45–55, 2019.

[7] M. Hofmann, R. Kiesel, D. Leichsenring, and G. Rünger. A hybrid CPU/GPU implementation of computationally intensive particle simulations using opencl. In *Proc. of the 17th International Symposium on Parallel and Distributed Computing (ISPDC'18), Geneva, Switzerland*, pages 9–16. IEEE, June 2018.

[8] A. Holey, V. Mekkat, P. C. Yew, and A. Zhai. Performance-energy considerations for shared cache management in a heterogeneous multicore processor. *ACM Transactions on Architecture & Code Optimization*, 12(1):1–29, 2015.

[9] A. Jain and C. Lin. Back to the future: Leveraging belady's algorithm for improved cache replacement. In *Proc. of the 43rd ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'16), Seoul, South Korea*, pages 78–89. IEEE, June 2016.

[10] Z. Li, L. Ju, H. Dai, X. Li, M. Zhao, and Z. Jia. Set variation-aware shared LLC management for CPU-GPU heterogeneous architecture. In J. Madsen and A. K. Coskun, editors, *Proc. of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE'18), Dresden, Germany*, pages 79–84. IEEE, March 2018.

[11] J. Ma, L. Yu, T. Chen, and M. Wu. Analyzing memory access on cpu-gpgpu shared llc architecture. In *Proc. of the 14th International Symposium on Parallel and Distributed Computing (ISPDC'15), Limassol, Cyprus*, June-July 2015.

[12] X. Mei and X. Chu. Dissecting GPU memory hierarchy through microbenchmarking. *IEEE Transactions on Parallel and Distributed Systems*, 28(1):72–86, 2017.

[13] J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood. gem5-gpu: A heterogeneous CPU-GPU simulator. *IEEE Computer Architecture Letters*, 14(1):34–36, 2015.

[14] S. Qiao, N. Han, Y. Gao, R. Li, J. Huang, J. Guo, L. A. Gutierrez, and X. Wu. A fast parallel community discovery model on complex networks through approximate optimization. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1638–1651, 2018.

[15] Z. Shi, X. Huang, A. Jain, and C. Lin. Applying deep learning to the cache replacement problem. In *Proc. of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'19), Columbus, OH, USA, October 12-16, 2019*, pages 413–425. ACM, October 2019.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[17] S. Souravlas, A. Sifaleras, and S. Katsavounis. Hybrid CPU-GPU community detection in weighted networks. *IEEE Access*, 8:57527–57551, 2020.

[18] E. Teran, Z. Wang, and D. A. Jiménez. Perceptron learning for reuse prediction. In *Proc. of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'16), Taipei, Taiwan*, pages 2:1–2:12. IEEE, October 2016.

[19] Q. Wang and X. Chu. GPGPU performance estimation with core and memory frequency scaling. In *Proc. of the 24th IEEE International Conference on Parallel and Distributed Systems (ICPADS'18), Singapore*, pages 417–424. IEEE, December 2018.

[20] L. Xiao, S. Wang, and G. Mei. Efficient parallel algorithm for detecting influential nodes in large biological networks on the graphics processing unit. *Future Generation Computer Systems*, 106:1–13, 2020.

Author Biography

Huijing Yang received the B.S. degree from Zhoukou Normal University, Zhoukou, China in 2018. And she is currently a Ph.D. student at Beijing University of Technology, Beijing, China. She is a student member of CCF. Her main research direction is computer architecture.

Tingwen Yu received a bachelor's degree from Beijing University of Technology in 2018 and is currently a graduate student at Beijing University of Technology. She is a student member of CCF and her main research field is computer architecture.