Design and Analysis of Efficient Inter-core Communication in GPUs

Mohamed Ibrahim*, Hongyuan Liu*, Onur Kayiran†, and Adwait Jog*
*College of William & Mary †Advanced Micro Devices, Inc.
Email: {maibrahim,hliu08}@email.wm.edu, onur.kayiran@amd.com, ajog@wm.edu

Graphics Processing Unit (GPU) architectures are becoming an inevitable part of every computing system [1] because of their ability to provide orders of magnitude faster execution. They have become the default choice for accelerating innovations in various fields [2]–[10] such as high-performance computing (HPC), artificial intelligence, deep learning, and virtual/augmented reality. Traditionally, GPUs have relied on bandwidth to achieve high throughput [11]–[14]. The sources of bandwidth have been local/shared caches, scratchpad, and memory. Additionally, high bandwidth interconnect is required to support the data flow between caches/memory and cores.

In this work, we focus on dynamically identifying and exploiting an additional source of bandwidth in GPUs, which we call as *remote-core bandwidth*. As also observed by previous works [15]–[17], the source of this additional bandwidth stems from *inter-core locality*. Specifically, the data required by one of the GPU cores (i.e., L1 misses) can also be found in the local L1 caches of other remote GPU cores. We find that this additional source of bandwidth leads to significant improvement in performance, however, can only be leveraged if an efficient inter-core communication is enabled.

There are several challenges towards designing an efficient inter-core communication, which have not been addressed by prior works. In particular, this work systematically addresses the following research questions: 1) How to determine which data can also be found in remote cores, 2) How to determine which cores have the data of the requester core, and 3) How to get the data as soon as possible without congesting the interconnect bandwidth?

Figure 1 shows the architectural diagram of our proposal. To the best of our knowledge, this is the first work that addresses these questions in a systematic manner. Specifically, this work makes the following contributions:

- We observe a bi-modal distribution of inter-locality across different instructions – some instructions use data that is shared across cores and some do not. We leverage the observation and use the program counter (PC) to predict which L1 misses are likely to be found in the L1 caches of other cores (A).
- We develop a low-overhead mechanism that can locally predict which cores are likely to have the shared data (B). It is based on our key observation that the data required by a core is generally shared across *only a few* cores, which can be detected via sampling a limited number of core replies.
- We develop a novel two-level probing mechanism that searches the identified cores in parallel while considering the network bandwidth consumption (C). This ensures that the searching overhead does not hamper inter-core locality as well as performance.
- Our combined schemes take advantage of the unutilized remote-core bandwidth, leading to 21% improvement (up to 40%) in performance if the data is a priori known to be shared, and 10% (up to 26%) with our PC-based predictor. These results are averaged across 11 diverse GPGPU applications that exhibit inter-core locality and achieved at a modest area overhead of 0.058 mm^2 per core (determined by detailed RTL synthesis). Additionally, our proposed schemes do not affect the performance of other applications that possess low inter-core locality.

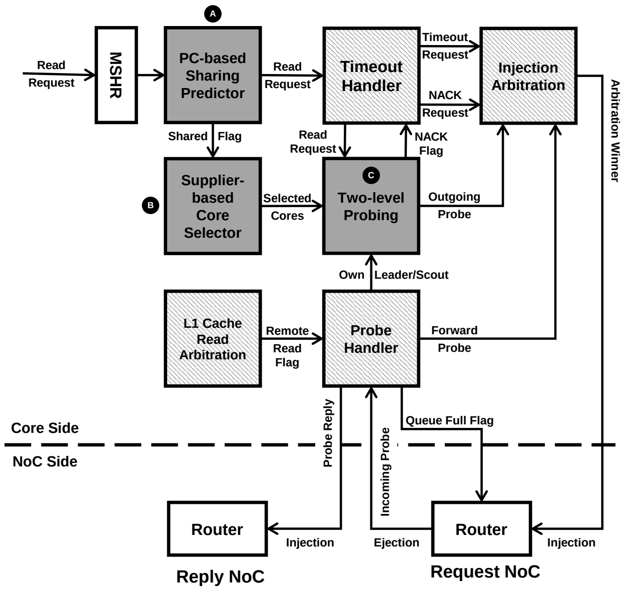

Fig. 1: Hardware organization of our proposal. The shaded components are used for inter-core communication. The gray components are newly added to support our proposal.

Acknowledgment

The authors thank the anonymous reviewers and members of the Insight Computer Architecture Lab at the College of William and Mary for their feedback. This material is based upon work supported by the National Science Foundation (NSF) grants (#1657336 and #1750667). This work was performed in part using computing facilities at the College of William and Mary which were provided by contributions from the NSF, the Commonwealth of Virginia Equipment Trust Fund and the Office of Naval Research. AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

References

[1] "Top500 Supercomputer Sites - June 2015," *http://www.top500.org/lists/2015/06/*.

[2] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte, "Medical image processing on the gpu-past, present and future," *Medical Image Analysis*, 2013.

[3] G. Pratx and L. Xing, "Gpu computing in medical physics: A review," *Medical physics*, vol. 38, 2011.

[4] S. S. Stone, J. P. Haldar, S. C. Tsao, W. mei W. Hwu, B. P. Sutton, and Z.-P. Liang, "Accelerating advanced MRI reconstructions on GPUs," *J. Parallel Distrib. Comput.*, vol. 68, 2008.

[5] NVIDIA, "How to harness big data for improving public health," *http://www.govhealthit.com/news/how-harness-big-data-improving-public-health*.

[6] I. Schmerken, "Wall street accelerates options analysis with gpu technology," *2008-11-07)[2009-11-02]. http://wallstreetandtech.com/technology-risk-management/showArticle.jhtml*, 2009.

[7] NVIDIA, "Jp morgan speeds risk calculations with nvidia gpus," 2011.

[8] NVIDIA, "Computational finance," *http://www.nvidia.com/object/ computational_finance.html*.

[9] NVIDIA, "Researchers deploy gpus to build world's largest artificial neural network," *http: //nvidianews.nvidia.com/Releases/Researchers-Deploy-GPUs-to-Build-World-s-Largest-Artificial-Neural-Network-9c7.aspx*.

[10] S. I. Park, S. P. Ponce, J. Huang, Y. Cao, and F. Quek, "Low-cost, high-speed computer vision using nvidia's cuda architecture," in *Applied Imagery Pattern Recognition Workshop, 2008. AIPR'08. 37th IEEE*. IEEE, 2008.

[11] A. Jog, O. Kayiran, N. C. Nachiappan, A. K. Mishra, M. T. Kandemir, O. Mutlu, R. Iyer, and C. R. Das, "OWL: Cooperative Thread Array Aware Scheduling Techniques for Improving GPGPU Performance," in *ASPLOS*, 2013.

[12] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, April 2009.

[13] A. Jog, E. Bolotin, Z. Guz, M. Parker, S. W. Keckler, M. T. Kandemir, and C. R. Das, "Application-aware Memory System for Fair and Efficient Execution of Concurrent GPGPU Applications," in *GPGPU*, 2014.

[14] A. Jog, O. Kayiran, T. Kesten, A. Pattnaik, E. Bolotin, N. Chatterjee, S. Keckler, M. T. Kandemir, and C. R. Das, "Anatomy of GPU Memory System for Multi-Application Execution," in *MEMSYS*, 2015.

[15] G. Koo, H. Jeon, and M. Annavaram, "Revealing critical loads and hidden data locality in gpgpu applications," in *2015 IEEE International Symposium on Workload Characterization (IISWC)*, Oct 2015.

[16] D. Li and T. M. Aamodt, "Inter-core locality aware memory scheduling," *IEEE Computer Architecture Letters*, vol. 15, Jan 2016.

[17] S. Dublish, V. Nagarajan, and N. Topham, "Cooperative caching for gpus," *ACM Trans. Archit. Code Optim.*, vol. 13, Dec. 2016. [Online]. Available: http://doi.acm.org/10.1145/3001589