Exploring BSP styles of computing for near-data processing and processing-in-memory

Holger Fröning[†], Günther Schindler[†], Andreas Kugel[†], Jeffrey Young[‡]
[†] Ruprecht-Karls University of Heidelberg, Germany [‡] Georgia Institute of Technology, US
[†] {holger.froening,guenther.schindler,andreas.kugel}@ziti.uni-heidelberg.de
[‡] jyoung9@gatech.edu

1. SUMMARY

Given current technology constraints and the implications of the end of Dennard scaling, it seems mandatory to explore computing concepts that minimize the distance between processor and memory for several reasons: first, an increasing distance contributes to an increasing latency due to the limited signal propagation speed. Second, and less obvious, an increasing distance generally results in a linear increase of energy consumption (with some exceptions that increase consumption like dielectric loss, skin effect, and issues with signal-to-noise ratio due to amplification). Otherwise, energy would grow exponentially with distance, which in practice usually happens as soon as a certain distance is reached. Third, technology constraints like pin count increasingly limit throughput when traversing component boundaries like die-to-package or package-to-board.

Near-data processing (NDP) and processing-in-memory (PIM) provide promising directions for reducing processor-memory distance. These new archictures are also amenable for use with Bulk-Synchronous Parallel (BSP) styles of computing for data intensive problems, in which parallel slackness is exploited to tolerate latency by scheduling and pipelining. This work shortly summarizes our motivation and current observations.

2. RESEARCH QUESTIONS

Styles of computing: we need a fundamental understanding of the differences between GPUs and FPGAs. GPUs and FPGAs share many architectural features, including high concurrency at reduced frequency, a flat memory hierarchy, manually controlled scratchpad memory, and programming using data-parallel kernels in an off-load model. However, while GPUs are currently prime examples of an BSP execution model and can exploit parallel slackness for latency tolerance [5], FPGA software currently poorly supports this, even though it seems that the architectural similarity should allow for it.

Programming models for reconfigurable logic: recent vendor efforts have introduced BSP programming models like OpenCL for FPGAs. While OpenCL can be compiled to execute correctly on reconfigurable logic, it remains unclear to which extent performance depends on parameters like kernel launch configuration and workgroup size. Furthermore, we see synergies with our work on automated workload partitioning [1].

Implications of memory access patterns: as the internal architecture of stacked memory can be complex, there exists a set of implications towards memory access patterns and data layout. In recent work, we performed a set of studies describing the implications on time and energy for stacked memory accesses [3].

Power and thermal constraints: in particular PIM is strongly constrained by power consumption and thermal effects. Thus, we are highly interested in applying multi-physics libraries like KitFox [4] to model and explore implications of control and data placement.

Workload characteristics: we are most interested in workloads including graph computations, deep neural networks (in particular inference), and sparse linear algebra. We are in particular exploring the use of binarization and related forms of quantization, which have shown promising results on ARM processors and FPGAs [2].

3. FUTURE DIRECTIONS

We are currently pursuing these research questions synergistically with two related research projects: the DeepChip project focuses on deep learning for embedded systems using reduced precision, sparsity and automated mappings to reconfigurable logic. A joint project with SAP (Graphite) explores graph processing engines for concurrent queries.

4. ACKNOWLEDGEMENTS

We acknowledge the sponsoring we received from Micron in form of equipment grants (Micron AC-510 system).

5. REFERENCES

[1] A. Matz, M. Hummel, and H. Fröning. Exploring llvm infrastructure for simplified multi-gpu programming. In *MULTIPROG workshop, collacted with HiPEAC 2016.*

[2] G. Schindler, M. Mücke, and H. Fröning. Linking application description with efficient simd code generation for low-precision signed-integer gemm. In *UCHPC Workshop, collocated with Euro-PAR 2017.*

[3] J. Schmidt, H. Fröning, and U. Brüning. Exploring time and energy for complex accesses to a hybrid memory cube. In *2nd International Symposium on Memory Systems (MEMSYS)*, 2016.

[4] W. J. Song, S. Mukhopadhyay, and S. Yalamanchili. Kitfox: Multi-physics libraries for integrated power, thermal, and reliability simulations of multicore microarchitecture. In *IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 5, no. 11*, 2015.

[5] L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990.