Dataflow processing for irregular parallelism, Part I
Exploration of manycore architecture

Chair of Computer Architecture and Operating Systems (CAOS) &
Chair of Computer Architecture and Parallel Systems (CAPS)
Technical University of Munich
yicheng.zhang@tum.de

1 Introduction

Irregular parallelism is a critical type of workload like sparse linear algebra and graph analytics, being pervasive in scientific computing and simulation, and machine learning tasks. However, they feature data dependent branch and pointer-based data structure, challenging the traditional computer architecture with frequent synchronization, irregular memory access and dynamic task generation. Manycore architecture has massive parallelism with at least hundreds of cores compared with conventional multicore. Furthermore, as a multiple instruction multiple data machine with higher flexibility, it can inherently handle irregular workloads, being more efficient than single instruction multiple data machines like GPUs. With full programmability, it is more widely applicable than the fixed datapath, i.e. accelerators though at the cost of extra area for instruction processing. Software or hardware level dynamic scheduling by task-based dataflow programming enables the loading balancing of available computing resources, making the most of computing capacity. Overall, manycore architecture is good choice of general purpose solution for a wide range of irregular parallelism workloads balancing performance, programmability, and flexibility.

2 Research directions and questions

- `Microarchitecture of the single core.` Two types of core design, i.e. basic in order core [1]–[5] and the complex core[6]–[8] with SIMD, multithreading, or superscalar, offer a trade-off between the single core performance and area. This brings question that which case could lead to larger aggregated performance of the manycore system. Targeting pointer-based data structure like graph and tree, which can hardly benefited from the data locality offered by cache, some designs [4], [5], [7], [8] completely get rid of the cache and its coherence overhead. Specific optimizations through ISA extension are also widely used to improve the efficiency further in terms of a application domain: non-linear functions[6], flow control, network access and synchronization[8], streaming semantics register extension and floating point repetition extension[2]. Other micro-architectural efforts include pipeline parallelism exploiting [9], [10] and prefetching[11], [12] to improve the latency of irregular memory access, and speculative execution[13], [14] which may speed up ordered irregular parallelism[15].
- `System organization.` In the flat distributed system[4]–[8] each core is independent and has exclusive resources, while in the clustered-based system[1]–[3], several cores are grouped up and they may share resources like router/interconnect, vector unit, and memory which can increase the density, i.e. the number of cores under the same area, at the potential cost the performance due to resource contention. Network-on-chip (NoC) is the key to organize the manycore system, and (custom) 2D Mesh [3]–[6], [8] is the most used one while other topology choices are 2D Torus[1], hierarchical crossbar[2] and island[7].
- `Implementation of dynamic scheduling.` Dynamic scheduling, especially task-based dataflow, is critical to allocate those dynamically generated tasks in the irregular parallel workloads to computing units available for loading balancing. Full software implementation is common ranging from from industry to academia like Intel TBB, Microsoft .NET, Cilk[16], ParSEC[17] and StarPU[18]. Besides, most publications centering hardware also provide a primitive programming model. However, hardware level support like a hardware task scheduler [4], [5] could be included underneath the programming model to efficiently accelerate the scheduling at the cost of extra hardware area which is a valuable trade-off to be balanced.

3 Tentative target and Conduct

- Have a basic understanding of manycore architecture, including its hardware structure (single core, scaling), software infrastructure (compiler, runtime), and application (algorithm, dataset, benchmark suite);
- Study the state of the art open-source full system system simulator, e.g. Dalorex[4]. Then iterative effort may involve: identify the bottleneck in terms of performance, power and area in three directions aforementioned, propose optimizations, implementation the simulator, performance evaluation.
- RTL implementation of full system or core components, physical design, evaluation and report.

Bibliography

[1] J. Vasiljevic and D. Capalija, "Blackhole & tt-metalium: The standalone ai computer and its programming model," in *2024 IEEE Hot Chips 36 Symposium (HCS)*, 2024, pp. 1–30. DOI: 10.1109/HCS61935.2024.10664810.

[2] F. Zaruba, F. Schuiki, and L. Benini, "Manticore: A 4096-core risc-v chiplet architecture for ultraefficient floating-point computing," *IEEE Micro*, vol. 41, no. 2, pp. 36–42, 2021. DOI: 10.1109/MM.2020.3045564.

[3] D. C. Jung, M. Ruttenberg, P. Gao, *et al.*, "Scalable, programmable and dense: The hammerblade open-source risc-v manycore," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 770–784. DOI: 10.1109/ISCA59077.2024.00061.

[4] M. Orenes-Vera, E. Tureci, D. Wentzlaff, and M. Martonosi, "Dalorex: A data-local program execution and architecture for memory-bound applications," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 718–730. DOI: 10.1109/HPCA56546.2023.10071089.

[5] M. Orenes-Vera, E. Tureci, M. Martonosi, and D. Wentzlaff, "Muchisim: A simulation framework for design exploration of multi-chip manycore systems," in *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2024, pp. 48–60. DOI: 10.1109/ISPASS61541.2024.00015.

[6] S. Lie, "Wafer-scale ai: Gpu impossible performance," in *2024 IEEE Hot Chips 36 Symposium (HCS)*, 2024, pp. 1–71. DOI: 10.1109/HCS61935.2024.10664673.

[7] S. Knowles, "Graphcore," in *2021 IEEE Hot Chips 33 Symposium (HCS)*, 2021, pp. 1–25. DOI: 10.1109/HCS52781.2021.9567075.

[8] E. Talpes, D. Williams, and D. D. Sarma, "Dojo: The microarchitecture of tesla's exa-scale computer," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–28. DOI: 10.1109/HCS55958.2022.9895534.

[9] Q. M. Nguyen and D. Sanchez, "Pipette: Improving core utilization on irregular applications through intra-core pipeline parallelism," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 596–608. DOI: 10.1109/MICRO50266.2020.00056.

[10] Q. M. Nguyen and D. Sanchez, "Phloem: Automatic acceleration of irregular applications with fine-grain pipeline parallelism," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 1262–1274. DOI: 10.1109/HPCA56546.2023.10071026.

[11] M. Orenes-Vera, A. Manocha, J. Balkind, *et al.*, "Tiny but mighty: Designing and realizing scalable latency tolerance for manycore socs," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22, New York, New York: Association for Computing Machinery, 2022, pp. 817–830, ISBN: 9781450386104. DOI: 10.1145/3470496.3527400. [Online]. Available: https://doi.org/10.1145/3470496.3527400.

[12] N. Talati, K. May, A. Behroozi, *et al.*, "Prodigy: Improving the memory latency of data-indirect irregular workloads using hardware-software co-design," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 654–667. DOI: 10.1109/HPCA51647.2021.00061.

[13] M. C. Jeffrey, S. Subramanian, C. Yan, J. Emer, and D. Sanchez, "A scalable architecture for ordered parallelism," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2015, pp. 228–241. DOI: 10.1145/2830772.2830777.

[14] M. Abeydeera and D. Sanchez, "Chronos: Efficient speculative parallelism for accelerators," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20, Lausanne, Switzerland: Association for Computing Machinery, 2020, pp. 1247–1262, ISBN: 9781450371025. DOI: 10.1145/3373376.3378454. [Online]. Available: https://doi.org/10.1145/3373376.3378454.

[15] K. Pingali, M. Kulkarni, D. Nguyen, *et al.*, "Amorphous data-parallelism in irregular algorithms," *The University of Texas at Austin, Department of Computer Sciences, Austin, TX, USA*, 2009.

[16] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou, "Cilk: An efficient multithreaded runtime system," in *Proceedings of the Fifth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPOPP '95, Santa Barbara, California, USA: Association for Computing Machinery, 1995, pp. 207–216, ISBN: 0897917006. DOI: 10.1145/209936.209958. [Online]. Available: https://doi.org/10.1145/209936.209958.

[17] R. Hoque, T. Herault, G. Bosilca, and J. Dongarra, "Dynamic task discovery in parsec: A data-flow task-based runtime," in *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ser. ScalA '17, Denver, Colorado: Association for Computing Machinery, 2017, ISBN: 9781450351256. DOI: 10.1145/3148226.3148233. [Online]. Available: https://doi.org/10.1145/3148226.3148233.

[18] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "Starpu: A unified platform for task scheduling on heterogeneous multicore architectures," in *Euro-Par 2009 Parallel Processing*, H. Sips, D. Epema, and H.-X. Lin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 863–874, ISBN: 978-3-642-03869-3.