

Towards Exascale Performance Using The Codelet Model

Stéphane Zuckerman
szuckerm@eecis.udel.edu

Souad Koliaï
koliai@eecis.udel.edu

Guang R. Gao
ggao@capsl.udel.edu

University of Delaware
Electrical and Computer Engineering Department
Computer Architecture and Parallel Systems Laboratory

Introduction

The research challenges associated with Exascale systems have been articulated in numerous reports. The challenges of Exascale computing in execution, parallelism, scalability, resilience, and energy efficiency require a new execution model that enables unprecedented efficiency when used in an Exascale system.

The Codelet Program Execution Model [14, 30] is one of the execution models that have begun to address these challenges. It is a hybrid model that incorporates the advantages of macro-dataflow [12, 16, 20] and the Von Neumann model. The Codelet Model can be used to describe programs in massively parallel systems, including hierarchical or heterogeneous systems.

Proposed Approach

Challenge 1: Parallelism The codelet model relies on explicit data dependence specified between its units of computations, called codelets. Codelets are collections of instructions that can be scheduled “atomically” as a unit of computation which run until completion. Synchronization between codelets is ensured through the signaling of fulfilled events—data or resource availability—on which a specific codelet waits.

Parallelism and more generally asynchronous execution of programs in the codelet model can be achieved through two separate means: loop parallelism and asynchronous procedure invocation, called threaded procedures or TPs [24]. Codelets are linked together to form a codelet graph (CDG). Threaded procedures are containers of CDGs with additional storage space for input, intermediate and output values, as well as meta-data called the TP frame.

The codelet model assumes the presence of an underlying abstract machine model (AMM). The codelet AMM is hierarchical and heterogenous. It is divided in “clusters” of cores, each composed of multiple computation units (CUs) and at least one synchronization unit (SU). The CUs may hold multiple codelet contexts at once, but can only run one at a time. SUs handle the scheduling and synchronization between codelets, as well as out-of-cluster communications. They decide whether a given TP must be run, handle outstanding hardware events and communications, etc.

Challenge 2: OS/Runtime Structure Traditional computer systems tend to use heavy-weight operating systems (OS), which nowadays also embed runtime systems (e.g. POSIX threads are usually implemented at the OS level). Runtime systems (RTS) tend to be thin layers put on top of the OS (e.g. MPI or OpenMP).

To reach sufficient scalability in extreme-scale systems, the role of the operating system will need to be more focused and less all-encompassing. The OS must be restricted to providing APIs to communicate with external devices, memory management, job scheduling, and I/O management (for at least networking and file systems). In contrast, the RTS will see its importance grow for efficient program execution: while a given job is scheduled by the OS on a set of resources, the RTS will manage concurrency on the resources, without help from the OS (e.g. not like “kernel threads + OpenMP”).

We propose that the RTS and the OS work closely together, but that the RTS handle most of the OS-provided resources by itself. The RTS must then decide how to schedule threaded procedures, whether to migrate them from a cluster of cores to another, etc.; it must also decide how to schedule codelets contained in a TP within a single cluster. By putting the burden of concurrency management at the runtime level rather than at the OS level, it allows for more fine-grain and user-driven execution: the user and/or the

compiler can provide “hints” to the runtime on how to schedule specific parts of a given program (e.g. data affinity, locality-driven or communication-driven, etc.).

Challenge 3: Dynamic Environment Exascale systems are expected to use very aggressive techniques to save on power and energy consumption, which are expected to imply more faults, transient or otherwise. Hence, both the OS and the RTS will need to constantly check the state of the underlying hardware for temperature rises, incorrect behaviors, etc. This *self-aware* behavior needs to be present both at the OS and RTS levels.

- At the OS level: any significant environmental change (rise in temperature, hardware failure of some kind, etc.) will need to be communicated to the relevant instance of the codelet RTS. Most of the time, it will be the sole job of the OS: to report changes, and let the RTS handle it.
- At the codelet RTS level: it will have almost total control over the hardware resources it is provided. The SU must then decide how to tune the job currently execututing according to hardware and OS events—for example, clock- or power-gating CUs, shutting down a cluster, etc.

Approach Assessment

The proposed approach is based on the following dimensions:

- **Challenge Addressed** Integration of the runtime and the operating system. Scalability is addressed by the asynchronous execution of the programs with the codelet model.
- **Maturity** We experimented the relation and symbiotic relation between OS and RTS on peta-scale systems through system software design, implementation and final deployment for the many-core chip based peta-scale Cyclops architecture/system in the past 7 years [6, 5, 9, 29, 11, 10, 28, 3]. The codelet execution model extends previous successful work performed with the EARTH system [15, 21, 2, 24, 26, 25, 27, 19]. It is the result of 2 years of work under the DARPA UHPC project [4]. This led to different implementations of the PXM, such as ETI’s SWARM [18], TIDeFlow [22], or DAR³TS from University of Delaware, have begun to address the exascale challenges.
- **Uniqueness** A joined OS-runtime work/effort for a flexible resource management and task scheduling.
- **Novelty** A correlation/integration of the runtime, and the operating system.
- **Applicability** Applicable over general purpose parallel systems. Tested on linear algebra and Graph500 benchmarks.
- **Effort** A 3-year effort to build a software technology that handles the requirements to run efficiently on extreme-scale systems.

Related Work

The exascale computing challenges create an important need for new execution models. Different fine-grain models have been developed to address these challenges. Works have already been done to exploit the codelet model in the ETI’s SWift Adaptive Runtime Machine (SWARM) [18] and TIDeFlow [22].

Fresh Breeze [7, 8] has been developed as a fine-grain PXM for massively parallel computing. It relies on a globally shared address space where memory where data is not updated but represented as a cycle-free heap. The FreshBreeze system ensures recycling outdated memory chunks.

ParalleX [13, 17, 23] is an experimental execution model that embodies a set of principles to exploit runtime knowledge and control to achieve dynamic adaptive resource management, task scheduling, and exploitation of inherent medium grain parallelism untapped by conventional practices.

The Habanero execution model [1] is an execution model derived from the X10 [5] programming model where programs are expressed as a collection of asynchronous tasks.

References

- [1] R. Barik, Z. Budimlic, V. Cave, S. Chatterjee, Y. Guo, D. Peixotto, R. Raman, J. Shirako, S. Tacsirlar, Y. Yan, Y. Zhao, and V. Sarkar. The habanero multicore software research project. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, OOPSLA '09, pages 735–736, New York, NY, USA, 2009. ACM.
- [2] H. Cai. Dynamic load balancing on the EARTH-SP system. Master's thesis, McGill, May 1997.
- [3] C. Chen, J. B. Manzano, G. Gan, G. R. Gao, and V. Sarkar. A study of a software cache implementation of the openmp memory model for multicore and manycore architectures. In *Euro-Par (2)*, pages 341–352, 2010.
- [4] DARPA. Uhpc: Ubiquitous high performance computing.
- [5] J. del Cuvillo, W. Zhu, and G. Gao. Landing openmp on cyclops-64: an efficient mapping of openmp to a many-core system-on-a-chip. In *Proceedings of the 3rd conference on Computing frontiers*, CF '06, pages 41–50, New York, NY, USA, 2006. ACM.
- [6] J. del Cuvillo, W. Zhu, Z. Hu, and G. R. Gao. Tiny threads: A thread virtual machine for the cyclops64 cellular architecture. In *IPDPS*, 2005.
- [7] J. Dennis. A parallel program execution model supporting modular software construction. In *Proceedings of the Conference on Massively Parallel Programming Models*, MPPM '97, pages 50–, Washington, DC, USA, 1997. IEEE Computer Society.
- [8] J. B. Dennis. Fresh breeze: a multiprocessor chip architecture guided by modular programming principles. *SIGARCH Comput. Archit. News*, 31(1):7–15, Mar. 2003.
- [9] G. Gan, Z. Hu, J. del Cuvillo, and G. R. Gao. Exploring a multithreaded methodology to implement a network communication protocol on the cyclops-64 multithreaded architecture. In *IPDPS*, pages 1–8, 2007.
- [10] G. Gan and J. Manzano. Tl-dae: Thread-level decoupled access/execution for openmp on the cyclops-64 many-core processor. In *LCPC*, pages 80–94, 2009.
- [11] G. Gan, X. Wang, J. Manzano, and G. R. Gao. Tile percolation: An openmp tile aware parallelization technique for the cyclops-64 multicore processor. In *Euro-Par*, pages 839–850, 2009.
- [12] G. Gao, H. Hum, and J.-M. Monti. Towards an efficient hybrid dataflow architecture model. In E. Aarts, J. van Leeuwen, and M. Rem, editors, *PARLE '91 Parallel Architectures and Languages Europe*, volume 505 of *Lecture Notes in Computer Science*, pages 355–371. Springer Berlin / Heidelberg, 1991. 10.1007/BFb0035115.
- [13] G. Gao, T. Sterling, R. Stevens, M. Hereld, and W. Zhu. Paralex: A study of a new parallel computation model. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–6, march 2007.
- [14] G. R. Gao, J. Suetterlein, and S. Zuckerman. Toward an Execution Model for Extreme-Scale Systems - Runnemedede and Beyond. Technical Memo, April 2011.
- [15] H. H. J. Hum, O. Maquelin, K. B. Theobald, X. Tian, G. R. Gao, and L. J. Hendren. A study of the EARTH-MANNA multithreaded system. *Parallel Programming*, 24(4):319–347, Aug. 1996.
- [16] R. Iannucci. Toward a dataflow/von neumann hybrid architecture. In *Computer Architecture, 1988. Conference Proceedings. 15th Annual International Symposium on*, pages 131–140, may-2 jun 1988.
- [17] H. Kaiser, M. Brodowicz, and T. Sterling. Paralex an advanced parallel execution model for scaling-impaired applications. In *Parallel Processing Workshops, 2009. ICPPW '09. International Conference on*, pages 394–401, sept. 2009.

- [18] C. Lauderdale and R. Khan. Position paper: Toward a codelet-based runtime for exascale computing. In *Proceedings of the 2nd International Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era*, EXADAPT '12, London, UK, 2012. ACM.
- [19] C. J. Morrone. An EARTH runtime system for multi-processor/multi-node Beowulf clusters. Master's thesis, Delaware, Newark,, May 2001.
- [20] W. A. Najjar, E. A. Lee, and G. R. Gao. Advances in the dataflow computational model. *Parallel Comput.*, 25:1907–1929, December 1999.
- [21] S. S. Nemawarkar and G. R. Gao. Measurement and modeling of EARTH-MANNA multithreaded architecture. In *of the Fourth on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 109–114, San Jose,, Feb. 1–3, 1996. TCCA and TCS.
- [22] D. Orozco, E. Garcia, R. Pavel, R. Khan, and G. Gao. Tideflow: The time iterated dependency flow execution model. *Proceedings of Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM 2011)*, 2011.
- [23] A. Tabbal, M. Anderson, M. Brodowicz, H. Kaiser, and T. Sterling. Preliminary design examination of the parallex system from a software and hardware perspective. *SIGMETRICS Perform. Eval. Rev.*, 38:81–87, March 2011.
- [24] K. B. Theobald. *EARTH: an efficient architecture for running threads*. PhD thesis, McGill University, Montreal, Que., Canada, Canada, May 1999. AAINQ50269.
- [25] K. B. Theobald, G. Agrawal, R. Kumar, G. Heber, G. R. Gao, P. Stodghill, and K. Pingali. Landing CG on EARTH: A case study of fine-grained multithreading on an evolutionary path. In *of SC2000: High Performance Networking and Computing*, Dallas,, Nov. 4–10, 2000. ACM SIGARCH and. URL <http://www.supercomp.org/sc2000/proceedings/>.
- [26] K. B. Theobald, R. Kumar, G. Agrawal, G. Heber, R. K. Thulasiram, and G. R. Gao. Developing a communication intensive application on the EARTH multithreaded architecture. In A. Bode, T. Ludwig, W. Karl, and R. Wismüller, editors, *of the 6th Euro-Par*, number 1900, pages 625–637, Munich, Germany, Aug. 29–Sept. 1, 2000. Springer-Verlag.
- [27] K. B. Theobald, R. Kumar, G. Agrawal, G. Heber, R. K. Thulasiram, and G. R. Gao. Implementation and evaluation of a communication intensive application on the EARTH multithreaded system. *Concurrency and Computation: Practice and Experience*, 14(3):183–201, Mar. 2002.
- [28] X. Wang, G. Gan, J. Manzano, D. Fan, and S. Guo. A quantitative study of the on-chip network and memory hierarchy design for many-core processor. In *ICPADS*, pages 689–696, 2008.
- [29] W. Zhu, Z. Hu, and G. R. Gao. On the role of deterministic fine-grain data synchronization for scientific applications: A revisit in the emerging many-core era. In *IPDPS*, pages 1–8, 2007.
- [30] S. Zuckerman, J. Suetterlein, R. Knauerhase, and G. R. Gao. Using a "codelet" program execution model for exascale machines: position paper. In *Proceedings of the 1st International Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era*, EXADAPT '11, pages 64–69, New York, NY, USA, 2011. ACM.