Compiler Optimizations for generating bounded Schedules on Task-based Runtimes

Yuhan Peng
Rice University
yp10@rice.edu

Martin Kong
Rice University
mkong@rice.edu

Vivek Sarkar
Rice University
vsarkar@rice.edu

In recent years, there has been a proliferation of parallel programming models and runtime systems [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. These runtimes provide a number of abstractions and features such as isolation and atomicity, seamless execution on shared and distributed-memory platforms, dependence aware scheduling, load balancing, work-stealing and pushing, and usually, one or more runtime scheduling policies. Furthermore, it is widely accepted that the parallel software for the upcoming and future generation of computing applications will be massively parallel and asynchronous [13, 14], while trying to achieve new performance limits under stringent power budgets on extremely heterogeneous platforms. Targeting these runtimes has also been the subject of more recent research efforts [15, 16, 17, 18, 19, 20, 21] which aim to automatically generate (optimized) code for the OpenStream runtime [2], the Open Community RunTime (OCR) [1] or any of the different Concurrent Collections implementations (e.g., [12, 4]).

In this talk, we will discuss the problem of generating *bounded task-parallel schedules* via compiler transformations. Our definition of *bounded* refers to the number of live tasks (created, but not necessarily in a `run` state) at a given point in time. Our approach builds on the notion of *Degrees of Freedom (DoF)* and runtime *policies*. Specifically, we define two DoF, and combine them to generate four different runtime scheduling policies. The benefits of our techniques are multi-fold. First, it addresses the limitation of several runtimes that create the program's dynamic DAG (at runtime) in a serial fashion. Second, it bounds the number of tasks in flight, as not all tasks are created right at the original task creation points identified by the programmer. This has the secondary effect of reducing runtime bookkeeping bottlenecks, which translates into potentially higher performance when using a large number of processors and smaller amounts of memory per core. Thirdly, using the graph structure of the program to drive the prescription and scheduling policy, some locality improvements are also possible. Lastly, our approach does not require modifying the runtime's underlying scheduler, since it is designed to be implemented as a compiler optimization pass. To the best of our knowledge, our work is the first to use compiler optimizations to bound the number of tasks created in task-parallel runtime systems.

IMPACT 2017
Seventh International Workshop on Polyhedral Compilation Techniques
Jan 23, 2017, Stockholm, Sweden
In conjunction with HiPEAC 2017.
http://impact.gforge.inria.fr/impact2017

The goal of our work is to show that by leveraging the dynamic task spawning abilities of a runtime such as Intel's CnC, we can improve the consumption of critical resources such as memory, without degrading performance. For instance, in an implementation of the Johnson matrix multiply algorithm in our framework, for an 8000^3 problem size using single precision, we can reduce its dynamic memory consumption by up to 43% (in a runtime which implements the dynamic single assignment rule, DSA) or improve the execution time without utilizing additional memory resources.

To better understand the problem that we attempt to solve, as well as our tentative solution, the first part of the talk will present and explain the Concurrent Collections execution model. Next, we will give a high-level overview of the PIPES compiler and the representation of the task graph used in it. Then, we will introduce the notion of degrees of freedom (DoF) and scheduling policies and describe their effect on the task graph as well as on the execution of the program. Finally, we will show the impact of the derived scheduling policies in terms of execution time, bookkeeping and memory consumption.

1. REFERENCES

[1] Tim Mattson, R Cledat, Zoran Budimlic, Vincent Cave, Sanjay Chatterjee, B Seshasayee, R van der Wijngaart, and Vivek Sarkar. Ocr: The open community runtime interface. Technical report, Tech. Rep., June 2015.[Online]. Available: https://xstack. exascale-tech. com/git/public,

2015.

[2] Antoniu Pop and Albert Cohen. Openstream: Expressiveness and data-flow compilation of openmp streaming programs. *ACM Trans. Archit. Code Optim.*, 9(4):53:1–53:25, January 2013.

[3] Zoran Budimlić, Michael Burke, Vincent Cavé, Kathleen Knobe, Geoff Lowney, Ryan Newton, Jens Palsberg, David Peixotto, Vivek Sarkar, Frank Schlimbach, and SaÄ§nak TaÅ§Äśrlar. Concurrent collections. *Scientific Programming*, 18(3-4):203–217, 2010.

[4] Frank Schlimbach, James C Brodman, and Kath Knobe. Concurrent collections on distributed memory theory put into practice. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 225–232. IEEE, Feb 2013.

[5] Kathleen Knobe and Carl D Offner. Tstreams: A model of parallel computation (preliminary report). Technical report, Technical Report HPL-2004-78, HP Labs, 2004.

[6] Stephen T. Heumann, Vikram S. Adve, and Shengjie Wang. The tasks with effects model for safe concurrency. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '13, pages 239–250, New York, NY, USA, 2013. ACM.

[7] Vincent Cavé, Jisheng Zhao, Jun Shirako, and Vivek Sarkar. Habanero-java: The new adventures of old x10. In *Proceedings of the 9th International Conference on Principles and Practice of Programming in Java*, PPPJ '11, pages 51–61, New York, NY, USA, 2011. ACM.

[8] Rajkishore Barik, Zoran Budimlic, Vincent Cavè, Sanjay Chatterjee, Yi Guo, David Peixotto, Raghavan Raman, Jun Shirako, Sagnak Tasirlar, Yonghong Yan, Yisheng Zhao, and Vivek 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.

[9] Zoran Budimlic, Aparna Chandramowlishwaran, Kathleen Knobe, Geoff Lowney, Vivek Sarkar, and Leo Treggiari. Multi-core implementations of the concurrent collections programming model. In *14th International Workshop on Compilers for Parallel Computers (CPC)*, 2009.

[10] Andi Drebes, Karine Heydemann, Nathalie Drach, Antoniu Pop, and Albert Cohen. Topology-aware and dependence-aware scheduling and memory allocation for task-parallel languages. *ACM Trans. Archit. Code Optim.*, 11(3):30:1–30:25, August 2014.

[11] Dounia Khaldi, Pierre Jouvelot, Corinne Ancourt, and François Irigoin. *Task Parallelism and Data Distribution: An Overview of Explicit Parallel Programming Languages*, pages 174–189. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[12] S. Chatterjee, S. Tasirlar, Z. Budimlic, V. CavÃl', M. Chabbi, M. Grossman, V. Sarkar, and Y. Yan. Integrating asynchronous task parallelism with mpi. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 712–725, May 2013.

[13] Vivek Sarkar, William Harrod, and Allan E Snavely. Software challenges in extreme scale systems. In *Journal of Physics: Conference Series*, volume 180, page 012045. IOP Publishing, 2009.

[14] Saman Amarasinghe, Mary Hall, Richard Lethin, Keshav Pingali, Dan Quinlan, Vivek Sarkar, John Shalf, Robert Lucas, Katherine Yelick, P Balanji, et al. Exascale programming challenges. In *Proceedings of the Workshop on Exascale Programming Challenges, Marina del Rey, CA, USA. US Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR)*, 2011.

[15] Benoit Meister, Muthu Baskaran, Benoit Pradelle, Thomas Henretty, and Richard Lethin. Efficient compilation to event-driven task programs. *arXiv preprint arXiv:1601.05458*, 2016.

[16] Nicolas Vasilache, Muthu Baskaran, Tom Henretty, Benoit Meister, M Harper Langston, Sanket Tavarageri, and Richard Lethin. A tale of three runtimes. *arXiv preprint arXiv:1409.1914*, 2014.

[17] Martin Kong, Antoniu Pop, Louis-Noël Pouchet, R. Govindarajan, Albert Cohen, and P. Sadayappan. Compiler/runtime framework for dynamic dataflow parallelization of tiled programs. *ACM Trans. Archit. Code Optim.*, 11(4):61:1–61:30, January 2015.

[18] Roshan Dathathri, Ravi Teja Mullapudi, and Uday Bondhugula. Compiling affine loop nests for a dynamic scheduling runtime on shared and distributed memory. *ACM Trans. Parallel Comput.*, 3(2):12:1–12:28, July 2016.

[19] Martin Kong, Louis-Noël Pouchet, P Sadayappan, and Vivek Sarkar. Pipes: A language and compiler for distributed memory task parallelism. SC '16. IEEE, 2016.

[20] Alina Sbirlea, Louis-Noel Pouchet, and Vivek Sarkar. Dfgr an intermediate graph representation for macro-dataflow programs. In *Data-Flow Execution Models for Extreme Scale Computing (DFM), 2014 Fourth Workshop on*, pages 38–45. IEEE, 2014.

[21] Alina Sbîrlea, Jun Shirako, Louis-Noël Pouchet, and Vivek Sarkar. *Polyhedral Optimizations for a Data-Flow Graph Language*, pages 57–72. Springer International Publishing, Cham, 2016.