Abstract:

To make use of parallel resources on modern multicore and multiprocessor systems, programmers often use explicit parallel programming models such as OpenMP or Intel Thread Building Blocks (TBB) to overcome the semantics enforced by inherently sequential programming languages such as C and C++. However, these programming models are plagued by the difficulty of determining where and how to parallelize the code most efficiently, further complicated by the fact that these choices may need to be reconsidered based on the target architecture and subsequent code changes. Moreover, they frequently exhibit less-than-optimal speedups, stemming from the inability to transform code due to strict conventions of the programming model and large overheads in the runtime. On the other hand, implicit parallelism with automatic parallelization shifts this burden from the programmer to the compiler, which can take advantage of state-of-the-art static analysis, speculation techniques, and parallelization schemes to transform the code in ways not conceivable at the source level. This work bridges the dichotomy between these two approaches by repurposing OpenMP annotations as additional memory dependence information in an automatic parallelization framework, outperforming its use as directives for explicit parallel programming. We evaluate the number of dependences and sequential strongly connected components (SCC) removed on 17 C/C++ benchmarks parallelized with OpenMP from the Rodinia benchmark suite compared to each unannotated counterpart to understand if programmer annotations can indeed aid in better automatic parallelization.

Reading list:

Textbook:

[1] A. W. Appel, Modern Compiler Implementation in ML. New York, NY, USA: Cambridge University Press, 2004.

Papers:

[1] D. Koes, M. Budiu, and G. Venkataramani, “Programmer specified pointer independence,” In Proceedings of the 2004 workshop on Memory system performance (MSP ’04), 2004.

[2] Milind Kulkarni, Keshav Pingali, Bruce Walter, Ganesh Ramanarayanan, Kavita Bala, and L. Paul Chew. “Optimistic parallelism requires abstractions,” In Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '07), 2007. p. 211–222.

[3] M. J. Bridges, N. Vachharajani, Y. Zhang, T. Jablin, and D. I. August, "Revisiting the sequential programming model for multi-core," in 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO ‘07), 2007, p. 13.

[4] H. Vandierendonck, S. Rul, and K. De Bosschere, “The Paralax infrastructure: automatic parallelization with a helping hand,” in 2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT), 2010.

[5] P. Prabhu, S. Ghosh, Y. Zhang, N. P. Johnson, and D. I. August, "Commutative Set: A Language Extension for Implicit Parallel Programming," in Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2011.

[6] A. Udupa, K. Rajan, W. Thies, “ALTER: exploiting breakable dependences for parallelization,” in Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2011.

[7] S. Campanoni, T. Jones, G. Holloway, V. J. Reddi, G.-Y. Wei, and D. Brooks, "HELIX: automatic parallelization of irregular programs for chip multiprocessing," in Proceedings of the Tenth International Symposium on Code Generation and Optimization - CHO '12, San Jose, California, 2012, p. 84.

[8] N. P. Johnson, H. Kim, P. Prabhu, A. Zaks, and D. I. August, "Speculative separation for privatization and reductions," in Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation, 2012, pp. 359–370.

[9] N. P. Johnson, J. Fix, S. R. Beard, T. Oh, T. B. Jablin, and D. I. August, "A collaborative dependence analysis framework," in 2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2017, pp. 148–159.

[10] N. B. Jensen, S. Karlsson, “Improving loop dependence analysis,” in 2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2017.

[11] S. Apostolakis, Z. Xu, G. Chan, S. Campanoni, and D. I. August, "Perspective: a sensible approach to speculative automatic parallelization," in Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2020.

[12] S. Apostolakis, Z. Xu, Z. Tan, G. Chan, S. Campanoni, and D. I. August, "SCAF: a speculation-aware collaborative dependence analysis framework," in Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2020.