Advanced tiling techniques for memory-starved streaming numerical kernels

Tareq Malas
Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
tareq.malas@kaust.edu.sa

Georg Hager
Erlangen Regional Computing Center (RRZE)
Friedrich-Alexander University of Erlangen-Nuremberg
Erlangen, Germany
georg.hager@fau.de

Hatem Ltaief
Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
hatem.ltaief@kaust.edu.sa

David Keyes
Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
david.keyes@kaust.edu.sa

I. Introduction

Several challenges prevent stencil computations from utilizing the compute power of contemporary and future processors. When the codes do not leverage the shared cache feature of modern processors, the required cache size is multiplied by the increasing number of cores, leading to low or no performance benefit from traditional temporal blocking techniques.

We build on previous work about cache block sharing [1], [2] and introduce a generalized multi-dimensional intra-tile parallelization scheme using different tiling techniques in three dimensions. We also introduce a fixed-execution-to-data multi-core wavefront approach that allows each thread to update local grid data to its private caches, reducing inter-core data traffic.

II. Approach: Multi-dimensional intra-tile parallelization algorithm

We introduce a $(d+1)$-dimensional intra-tile parallelization algorithm for tiled d-dimensional grids. Each Cartesian dimension, i, of a tile is divided equally into T_i chunks that can be updated concurrently. Additionally, the components of each grid cell may be divided among T_c chunks, when at least c equations per grid cell can be updated concurrently. The "Thread Group" (TG) size is $T_c \cdot \prod_{n=1}^{d} T_n$. Multiple TGs may update tiles concurrently.

In our implementation (Fig. 3) diamond tiling is performed along the y-axis by maximum of 2 threads. Wavefront blocking is performed along the z-axis. Any number of threads can share the data in the x- and z-axes. We propose a new wavefront scheme (Fig. 3e) resolving the data pipelining issue by "fixing the execution" (FE) of each each grid point to a single thread. This is particularly useful for high-order stencils, where more data is transferred between threads. A software sub-group barrier is used to synchronize threads along y- and x-axes to ensure results correctness.

Our system implementation (Fig. 4) consists of (1) parametrized tiling codes that take the stencil kernels and their specifications, for example, the stencil radius, (2) MPI wrappers to handle the distributed memory communication at domain boundaries, (3) a runtime system for dynamic scheduling of diamond tiles, and (4) an auto-tuner searching for the best diamond width, wavefront tile width, and threads along the x-, y-, and z- axes. The auto-tuner uses a cache block size model and a user-specified cache size range to reduce the search space.

III. Results

We compare PLUTO, Pochoir, our implementation of the CATS2 algorithm introduced by Strzodka *et al.* [3] (we call it 1-thread wavefront diamond blocking, 1WD), our approach (Multi-thread wavefront diamond blocking, MWD), and an optimal spatial blocking implementation. We use auto-tuning for optimal parameter selection in MWD and PLUTO. PLUTO is set up with flags "`--tile --parallel --pet --partlbtile`".

We show performance in billion lattice site updates per second (GLUP/s) vs. grid size (cubic domain) on an 18-core Intel Haswell EP with fixed 2.3 GHz base frequency (AVX slowdown was not observed) for four stencil schemes: 7-point with constant coefficients, 7-point with variable coefficients, 25-point with constant coefficients, and 25-point with variable coefficients. We also show memory traffic per LUP (code balance) and memory bandwidth. We include data for optimal spatial blocking as a sensible baseline.

MWD is consistently fastest for all stencils at almost all problem sizes. It exhibits the lowest code balance since it can make better use of the cache memory than the other schemes due to its block sharing concept. This is also why 1WD, PLUTO, and Pochoir can still achieve some significant speedup versus spatial blocking for the 7-point stencils, but fail to do so on the high-order stencils. Although in general a low code balance also means low memory bandwidth, this is not always the case: Pochoir can achieve very low bandwidth in some cases, notably for the 25-point variable-coefficient stencil, but it is still not competitive performance-wise.

Figs. 10–13 show a comparison of our model with measurements for the code balance of the four studied stencils. The model predicts the code balance for a given diamond width, i.e., the possible reduction in memory traffic per LUP, and how much cache is required for it. For example, the 25-point variable-coefficient stencil (Fig. 13) can barely fit a single cache block in the cache memory when using temporal blocking, which shows the importance of our intra-tile parallelization scheme.

IV. Conclusion

Our multi-dimensional cache block sharing approach is clearly well suited for stencil computations on modern CPUs with shared caches, allowing more in-cache data reuse than competing schemes that only use a single thread per cache block. Especially for long-range stencils it is the only method that can significantly outperform perfect spatial blocking across the full problem size range.

Our implementation achieves near-optimal code balance as long as there is sufficient cache space available to support the intra- and inter-tile parallelization. For the future we plan to integrate our framework into PLUTO and to add support for complex memory hierarchies to be found in future architectures.

References

[1] G. Wellein, G. Hager, T. Zeiser, M. Wittmann, and H. Fehske, "Efficient temporal blocking for stencil computations by multicore-aware wavefront parallelization," in *Computer Software and Applications Conference. 33rd Annual IEEE International*, vol. 1, July 2009, pp. 579–586.

[2] T. Malas, G. Hager, H. Ltaief, H. Stengel, G. Wellein, and D. Keyes, "Multicore-optimized wavefront diamond blocking for optimizing stencil updates," *SIAM Journal on Scientific Computing*, vol. 37, no. 4, pp. C439–C464, 2015. [Online]. Available: http://dx.doi.org/10.1137/140991133

[3] R. Strzodka, M. Shaheen, D. Pajak, and H.-P. Seidel, "Cache accurate time skewing in iterative stencil computations," in *Proceedings of the International Conference on Parallel Processing*. IEEE Computer Society, September 2011, pp. 571–581.