Automatic Parallelization and Locality Optimization of Beamforming Algorithms[1]

Albert Hartono[†], Nicolas Vasilache[†], Cédric Bastoul[†], Allen Leung[†], Benoît Meister[†], Richard Lethin[†], Peter Vouras[‡]
{hartonoa, vasilache, bastoul, leunga, meister, lethin}@reservoir.com, peter.vouras@nrl.navy.mil
[†]Reservoir Labs, Inc., New York, NY
[‡]Naval Research Laboratory, Radar Division, Washington, DC

Abstract

This paper demonstrates the benefits of a global optimization strategy using a new automatic parallelization and locality optimization methodology for high performance embedded computing algorithms that occur in adaptive radar systems, for modern multi-core computing chips. As a baseline, the resulting performance was compared against the performance that could be obtained using highly optimized math libraries.

Adaptive Beamforming Algorithms

Adaptive beamforming algorithms eliminate interference and clutter in a phased array antenna. Typically, for a small number N of array elements, the weight vector application to the incoming sensor stream represents the majority of the computation. However, with the introduction of solid state transceiver elements and the transition to conformal arrays, the number of antenna elements may go into the tens of thousands. This means that the computational challenge of weight computation algorithms with $O(N^2)$ and $O(N^3)$ complexities (versus weight applications with O(N) complexity) will dominate and require high performance computation.

Many different beamforming algorithms exist to perform the task of signal detection [1]. Traditional methods, such as direct inversion to determine the covariance, have more trivial parallelism, but in terms of operation count, they are not scalable to larger number of sensors. Iterative algorithms are more efficient on an operation count basis, but present a more difficult optimization challenge. The focus of this paper is on improving parallel performance of three different iterative beamforming algorithms: Minimum Variance Distortionless Response using Sequential Regression method (MVDR-SER), Coherent Sidelobe Cancellation using Least Mean Square algorithm (CSLC-LMS), and Coherent Sidelobe Cancellation using Robust Least Square algorithm (CSLC-RLS). Sequential implementations of these algorithms are available in the radar simulation framework developed by Reservoir Labs for the purpose of high performance benchmarking. The framework is constructed in such a way that the most computationally intensive parts (i.e., the kernels) of the beamforming algorithms can be optimized separately using external optimization tools. Kernel implementations based on the leading industry math library (the Intel MKL) are also included in the radar simulation framework.

[1] This work was funded in part by US Army Space and Missile Defense Command, Code: W9113M, P.O. 1500, Huntsville, AL 35807.

Integrated Optimization Framework for Parallelism and Locality

Achieving high performance and efficient operation on emerging processors requires finding significant parallelism. However, this is not the only requirement. Locality of reference – through the memory hierarchy, feeding bandwidth starved functional units, is also critical [2]. Parallelization can come at the expense of locality. We have developed optimization modules in the R-Stream compiler [3] that allow for fine tradeoffs between parallelism and locality. We illustrate the importance of this optimization for a classic beamforming algorithm below.

R-Stream is a source-to-source compiler that takes as input a sequential C code and automatically generates optimized parallel code for several different architectures including SMPs (using OpenMP), FPGAs, Tilera, ClearSpeed, STI Cell, and GPUs. The core component of the R-Stream compiler is the *polyhedral mapper* that uses the *polyhedral model* to precisely represent loop programs and to compute complex sequences of loop transformations (interchange, fusion, fission, skewing, tiling, etc.) while preserving the original program semantics. The polyhedral mapper exposes parallelism via affine scheduling transformation. It also simultaneously augments data locality using loop fusion. Its analytical model employs a unified formulation of parallelism and fusion/distribution profitability to obtain an optimal trade-off between parallelism and locality. The model computes cost function parameters to estimate the performance benefits coming from applying fusion and parallel execution.

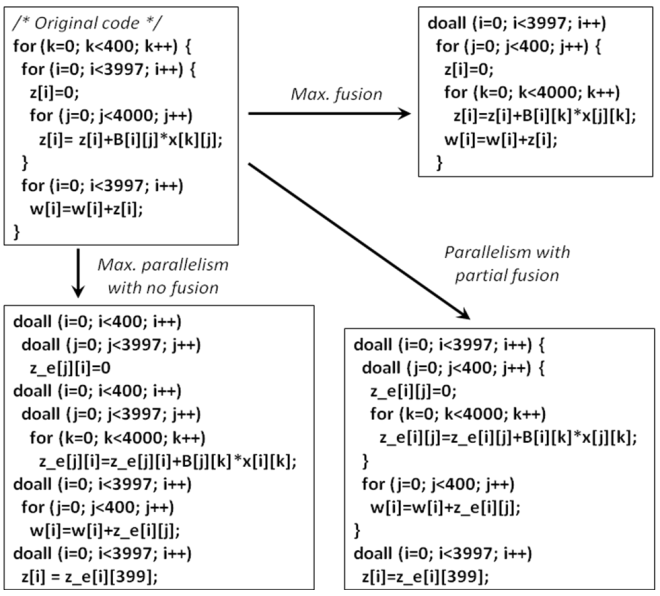
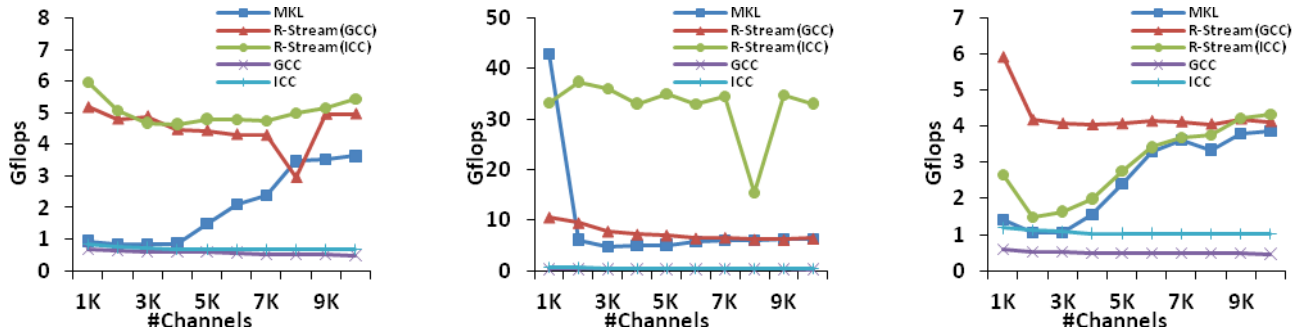


Figure 1: Trade-off between parallelism and locality.



(a) MVDR-SER (b) CSLC-LMS (c) CSLC-RLS

Figure 2: Performance results of radar codes on a dual Intel Xeon E5405 machine (theoretical peak performance: 128 Gflops).

To illustrate the trade-off between parallelism and locality, consider the code in Fig. 1 (top left), a simplified version (for illustration purposes) of CSLC-LMS. The code fragment on the bottom left of the figure is the corresponding code restructured by R-Stream for maximum parallelism with no fusion consideration. The original loop body is fissioned into distinct loop nests, where each loop nest contains only a single statement. R-Stream performs array expansion optimization that expands the dimension of array *z* to introduce another level of parallelism. As a result, the code now has two levels of parallelism. However, increasing parallelism by solely using loop fission with no fusion may lead to inferior performance because of poor data reuse, especially for codes where many references to the same array elements exist. Hence, when transforming codes for parallelism, loop fusion must also be considered in a unified manner. The top right of Fig. 1 shows the code optimized by R-Stream with a strong bias toward fusion. Application of fusion in this example is too aggressive as it destroys an extra level of parallelism that may be important for some highly parallel architecture in which more than one degree of parallelism is needed. On the contrary, fusion that can significantly enhance locality may sometimes be favored over an additional degree of parallelism, if sufficient degrees of parallelism to fill the hardware resources are already achieved. An example of loop fusion that does not decrease the available degree of parallelism is shown in the bottom right portion of Fig. 1. The parallel code still has the maximum two degrees of parallelism, and the reuse distances of references to elements of array *z_e* are minimized by partial fusion.

Table 1: R-Stream performance with different mapping strategies for CSLC-LMS for 4K channels (in Gflops).

Low-level compiler	Max. fusion	Max. parallelism with no fusion	Parallelism with partial fusion
GCC	3.1	3.3	7.3
ICC	1.4	28.7	33.0

We experiment with the actual CSLC-LMS code to show the performance of different mapping strategies. The same experimental setup described in the next section was used here. The results are presented in Table 1. The code with maximum fusion loses a significant amount of parallelism as its most compute-intensive loop nest becomes completely sequential. The code that balances parallelism and locality has the best performance. The code that maximizes parallelism with no fusion performs worse than the version using partial fusion, caused by lack of locality.

Parallel Performance Results

The performance of the three beamforming algorithms mentioned earlier was evaluated on an eight-core Intel Xeon machine with dual quad-core E5405 Xeon processors clocked at 2.0 GHz with 9 GB memory, running Linux kernel 2.6.25 (x86-64). The codes were compiled with R-Stream 3.1.2 with the unified parallelism/locality model enabled. The used low-level compilers were GCC 4.3.0 (with "-O6 -fno-trapping-math -ftree-vectorize -msse3 -fopenmp" flags) and ICC 11.0 (with "-fast -openmp" flags). The implementations based on math libraries used Intel MKL 10.2.1. To utilize all available computing cores, the number of OpenMP threads was set to eight. Single precision floating point matrices were used. All experiments were run ten times and then averaged.

For each beamforming algorithm, we set the total number of radar iterations to 400. The CLSC-LMS and CSLC-RLS algorithms compute for 3 radar sidelobes. The performance results for the three beamforming algorithms are provided in Fig. 2. The performance of ICC and GCC is low since they are unable to parallelize the codes. R-Stream performance is better than MKL for most of problem instances. R-Stream enables us to attain very good speedup over MKL. In particular, the performance of CSLC-LMS optimized using R-Stream and ICC is seven times better than MKL. This is because R-Stream is able to parallelize the outermost loop iterating over CSLC-LMS computation, whereas in the MKL version, such an outermost loop cannot be parallelized due to loop-carried dependences. For MVDR-SER and CSLC-RLS, effective exploitation of data locality dominates the performance of R-Stream as the outermost loops of the most compute-intensive parts must be executed sequentially because of loop-carried dependences.

References

[1] P. Vouras and B. Freburger. "Application of adaptive beamforming techniques to HF radar," In IEEE Radar Conference, pages 1-6, Rome, May 2008.

[2] M. Wolf and M. Lam, "A Data Locality Optimizing Algorithm," In Proceedings of ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation, pages 30-44, Toronto, Canada, June 1991.

[3] R. Lethin, A. Leung, B. Meister, P. Szilagyi, N. Vasilache, and D. Wohlford, "Final report on the R-Stream 3.0 compiler DARPA/AFRL Contract #F03602-03-C-0033, DTIC AFRL-RI-RS-TR-2008-160," Technical report, Reservoir Labs, Inc., May 2008.