A Study on the Impact of Memory Hierarchy on Stencil Operations for Modern Architectures
ABSTRACT

Federico Bassetti, Kei Davis, Olaf Lubeck, Fabrizio Petrini
Scientific Computing Group, CIC-19,
Los Alamos NM, 87545, USA
Tel: (505) 667-7492, Fax: (505) 667-1126
{fede,kei,oml,fabrizio}@lanl.gov

In the context of large-scale parallel scientific applications, research and experience have shown that the greatest performance gains are realized by directing the most effort toward efficient utilization of the uppermost levels of the memory hierarchy and interprocess communication. Benchmarking has shown that even more than is the case for more mainstream application areas, for numerical codes the bulk of the computation is represented by a very small fraction of the code.

For several families of numerical algorithms the dominant computational kernels are stencil and stencil-like operations, made conspicuous and amenable to parallelization and optimizing transformation by their repetitive and regular access of (typically) very large arrays.

Numerical algorithms used for solving partial differential equations–multigrid methods and those entailed by linear solvers–are rich with stencil-like operations to effect smoothing and relaxation. The distinguishing characteristic of stencil-like operations is the evaluation of a computationally inexpensive expression, for each element of a (possibly multi-dimensional) array, in which the arguments are all of the elements within a given radius of that element.

Elsewhere we have shown that naively implemented, stencil operations are severely memory-bound, hence very inefficient, even in the presence of a cache. In essence the problem is that the arrays, being many times the size of the cache, cycle through the cache with minimal spatial or temporal reuse. Optimizing compilers attempt to improve performance by tiling Again, in this context improvement is measurable but minimal. Processor time can be traded for communication time for net performance gain, and shown simple sequential optimizations (also applicable in the parallel case), to give significant improvement in performance. The parallel aspect of this technique is a subject of ongoing investigation.

The focus of this presentation will be a memory-centric performance analysis. The main object of detailing problems and address solutions is achieved with the development of a simple analytical model for stencil computation which has a sequential component and a parallel one as well. We validate the model with a cache simulator and performance counters.

Taking advantage of the flexibility offered by a simulator, that enables us to vary the memory hierachy, we evaluate the sequential and parallel performance. Of particular interest is the analysis of how changes in the memory hierarchy affect the performance gap existing between traditional (state of the art compilers) and stencil-aware implementations.