An S_N Algorithm for Modern Architectures

Randal S. Baker, rsb@lanl.gov

LANL discrete ordinates transport packages are required to perform large, computationally intensive time-dependent calculations on massively parallel architectures, where even a single such calculation may need many months to complete. While KBA methods scale out well to very large numbers of compute nodes, we are limited by practical constraints on the number of such nodes we can actually apply to any given calculation. Instead, we describe a modified KBA algorithm that allows realization of the reductions in solution time offered by both the current, and future, architectural changes within a compute node.

Background and Motivation

The KBA method, essentially a wave front method that results in a 2-D spatial domain decomposition for 3-D geometries, has successfully provided parallelization for SN transport sweeps as computing platforms have evolved from the SIMD (Single Instruction, Multiple Data) architectures of the early 1990's to today's large clusters of RISC-based (Reduced Instruction Set Computing) commercial compute nodes. Indeed, with additional algorithmic modifications and extensions, it forms the basis for "hybrid KBA" domain decomposition methods that allow transport sweeps to operate with reasonable parallel efficiency on computing platforms with core counts approaching one million. However, while these KBA-like methods are necessary, they are not sufficient for LANL radiation transport applications for reasons discussed below.

At LANL, we are now transitioning into a computational realm where large, 3-D, time-dependent radiation transport calculations will be performed on a regular basis. A single such calculation may require in excess of fifty million CPU-hours over the course of several months of calendar time, with memory requirements exceeding fifty TBytes. In our experience, fabric scaling alone (i.e., increasing node count until job turnaround time is acceptable) will not be sufficient for these calculations since, as node count increases, power requirements (and therefore costs) increase linearly, while mean time to job interrupt decreases super-linearly. The combination of these two effects limits our practical machine allocations for such calculations to no more than $O(10^3)$ nodes. It is therefore essential in order to meet job turnaround time requirements for our applications that per-node performance be maximized.

Description

In order to maximize the number of independent work units, we have completely decoupled all phase-space dependencies for angles and energy groups during a transport sweep. That is, during the KBA sweep itself, in Cartesian geometries, the solution for any given angle or energy group no longer depends in any fashion upon any other angle or energy group (in R-Z geometries angles are replaced with polar levels due to the angular derivative in the streaming term). Then, in order to minimize data movement for this algorithm within our Fortran implementation, we have reordered many arrays, such as those used for storage of angular fluxes or flux moments, so that their fastest algorithmically-varying index is now left-most. In other cases we duplicated storage, such as that used for convergence controls or temporary sweep working arrays, so that they may be accessed simultaneously and independently across energy groups.

These changes have been incorporated into the instantiation of our modified KBA algorithm (hereafter referred to as the KBA1 algorithm) targeted at many- and multi-core architectures. The KBA1 implementation requires only ~3,000 lines of additional source code within PARTISN (110,000+ lines of source code). The overall strategy of our implementation for these many-

and multi-core architectures may be summarized as follows:

1. Spatial domain decomposition using MPI and the KBA1 algorithm.
2. Vectorization over all angles within an octant (polar levels in R-Z geometries).
3. Threading via OpenMP over (at least) energy groups, with a Point Jacobi treatment for the down-scatter term.

Anticipated Impact

Disabling vectorization actually increases performance with the KBA0 algorithm since it exposes insufficient vectorization opportunities to offset the associated increases in loop overhead costs. The KBA1 algorithm, on the other hand, shows substantial benefits from vectorization on all architectures, although always less than the theoretical maximum. Our analysis on Knights Corner has shown that, although our algorithm is indeed well-vectorized, i.e., average vector length of eight, we are limited by memory bandwidth during the cell balance solution. This limitation is further aggravated by the use here of all threads as independent energy groups, which induces cache thrashing. We are currently exploring alternative approaches for the cell balance solution which will minimize these effects. Finally, even with all vectorization disabled, we see solution times for the KBA1 algorithm are still less than those for the KBA0 algorithm. This is a result of the KBA1 algorithm's design goal of minimizing data movement.

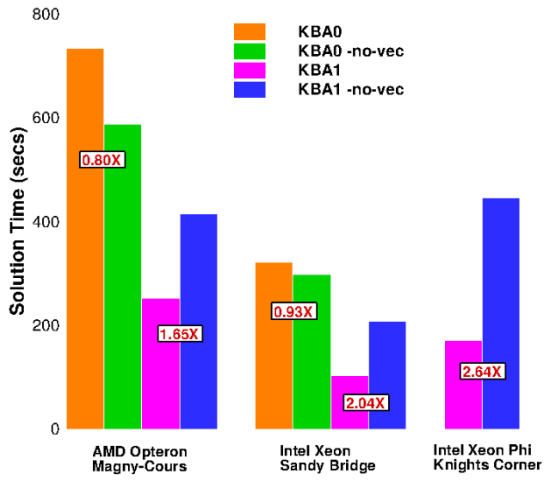

Figure 1- Vectorization Performance

Path Forward

Effective use of current and future computer architectures depends upon maximizing independent works units while minimizing data movement. The modified KBA algorithm discussed here does so, while still preserving the intrinsic domain decomposition that defines KBA sweeps. Building upon the lessons learned from Roadrunner, we have implemented a vectorized, "MPI+OpenMP" version of our algorithm that already offers substantial reductions in run time on current multi-core platforms, and shows promise for sustaining these performance improvements on the many-core platforms of the near future. We believe use of this algorithm will enable us to meet our requirements for job turnaround times while staying within realistic power and mean time to job interrupt constraints.

We also believe use of our modified KBA algorithm will allow similar performance gains to be obtained on heterogeneous architectures with an accelerator, such as GPUs, both because such an architecture resembles that which our algorithm was originally designed for (i.e., Roadrunner) and, more importantly, because the fundamental principles of maximizing independent works units while minimizing data movement are independent of architectural type.

Acknowledgements

Los Alamos Report LA-UR-16-26593. Funded by the Department of Energy at Los Alamos National Laboratory under contract DE-AC52-06NA25396.