Performance Comparison of the Multi-zone Scalar Pentadiagonal (SP-MZ) NAS Parallel Benchmark on Many-core Parallel Platforms

Christopher P. Stone
Computational Science and Engineering, LLC
1830 N Winchester Ave. #305
Chicago, IL 60622
1+(678)-938-7292
chris.stone@computational-science.com

Bracy Elton
Engility Corporation
U.S. Air Force Research Laboratory
Wright-Patterson Air Force Base, OH 45433
1+(937)-904-5139
bracy.elton@engilitycorp.com

Categories and Subject Descriptors
D.1.3 Concurrent Programming.

General Terms
Performance, Algorithms.

Keywords
Accelerators, Compiler Directives, Performance Optimization.

1. INTRODUCTION
The NAS multi-zone scalar pentadiagonal (SP-MZ) [1] parallel benchmark (NPB3.3.1-MZ) is widely used to model the performance of a class of computational fluid dynamics algorithms based on the alternating direction implicit (ADI) scheme on massively parallel computing systems. SP-MZ uses the Pulliam-Chaussee ADI scheme to solve the three-dimensional, compressible Navier-Stokes equations on structured meshes. The dominant cost of the SP-MZ algorithm is the factorization of the five SP matrices along each spatial direction. The global mesh is partitioned into multiple zones which can be computed concurrently within in each iteration. The SP-MZ benchmark mimics many multi-zone CFD codes, e.g., OVERFLOW-2, used throughput the aerospace industry.

We have ported the SP-MZ benchmark to run on NVIDIA GPU and Intel Xeon Phi many-core accelerator devices, using OpenACC and OpenMP compiler directives, respectively. Xu et al.[2] explored OpenACC and CUDA parallelization for the single-zone SP benchmark and found run-time acceleration between five and ten times the single-core run-time with an NVIDIA (K20) GPU. The multi-zone design of the SP-MZ benchmark allows us to measure the performance on clusters comprised of accelerators and when using the host CPU cores and accelerator devices concurrently in a heterogeneous fashion.

The implementations proceeded in several stages for both platforms: (i) vectorization, (ii) instrumentation, (iii) single-zone optimization, and (iv) multi-zone optimization. Vectorization involved reformatting the array syntax throughout the SP-MZ code. SP-MZ was optimized for cache-based CPUs by using an array-of-structures-like Fortran format: `U(N,I,J,K)` where `N` is the index of the independent variables and `IJK` are the mesh coordinates. We reformatted this to `U(I,J,K,N)` throughout the code to promote unit-stride access since vector parallelism is used across the inner mesh dimension. Instrumentation involved adding OpenACC directives or augmenting the original OpenMP directives. All loops within the ADI routine were instrumented with parallel loop and data region directives. Data regions define the size and lifetime of device data and control host-device transfers. Note that the current Xeon Phi benchmarks operate in *native* mode; therefore, data resides exclusively on the device and host-device transfers are not necessary. The host-device transfer overhead was assessed using both whole array transfers and RIND-only transfers. RIND transfers copy only the surrounding halo of each zone as opposed to the zone's entire volume.

During single-zone optimization, all routines within the ADI algorithm were optimized to increase the thread concurrency and vector parallelism. Significant restructuring of loops was required to improve array access patterns and loop vectorization. A major hurdle of the SP-MZ algorithm was how to efficiently factorize the SP. These are solved directly using the Thomas Algorithm (TMA) which forces iterating sequentially in one spatial direction. However, loops along the other two directions can be efficiently computed in parallel if structured properly. The five matrix systems are linearly independent and were factorized concurrently as well to further increase the thread parallelism. Finally, the ability to process multiple zones in parallel on one or more devices was enabled during multi-zone optimization.

2. RESULTS
We present on the poster the run-time for the Class-A mesh (128x128x16 with sixteen zones) for several development stages on an NVIDIA Kepler GPU and an Intel E5-2670 CPU. These were run on the Navajo cluster at the Air Force Research Laboratory using the PGI v14.10 compiler. We also show the same benchmark run on an Intel Xeon Phi with one MPI process per zone. On the GPU, we observe a significant improvement (75%) with full optimization within a single zone. The performance more than doubles when multiple zones are processed in parallel asynchronously. On the Xeon Phi, array vectorization increased the baseline throughput by 30% and optimization added an additional 13% performance improvement.

Other performance measurements presented on the poster include the impact of improved host-device communication with OpenACC and by bypassing the host altogether with GPUDirect. These benchmarks were run on a large GPU-based supercomputer and show promising multi-device scaling for the larger Class-C mesh. Hybrid OpenMP and MPI computations are presented on the Intel Xeon Phi which show the impact of increasing thread parallelism within the ADI zone calculations.

Planned additions and revisions on the final poster include the use of data offload directives for the Intel Xeon Phi. This will be

analogous to the OpenACC implementation and will permit profiling across any number of Xeon Phi devices. We also shall include an assessment of the optimization effort required for the two accelerator devices.

3. REFERENCES

[1] Van der Wijngaart, R.F., Haoqiang, J., "NASA Parallel Benchmarks, Multi-Zone Versions," *NAS Technical Report* NAS-03-010, July, 2003.

[2] Xu, R., Tian, X., Chandrasekaran, S., Yan, Y., and Chapman, B., "OpenACC Parallelization and optimization of NAS parallel benchmarks," *GPU Technology Conference* 2014.