Performance Analysis and Optimization of the Weather Research and Forecasting Model (WRF) on Intel® Multicore and Manycore Architectures

Samuel Elliott
University of Colorado at Boulder: Department of Applied Mathematics
National Center for Atmospheric Research: Computational and Informational Science Laboratory
3090 Center Green Drive, Boulder, CO 80301
Samuel.Elliott@Colorado.Edu

ABSTRACT

The Weather Research and Forecasting (WRF) Model is a mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting needs. The WRF benchmarks used in this study were run on the Texas Advanced Computing Center (TACC) Stampede cluster, which utilizes Intel Xeon E5-2660 CPU's and Xeon Phi SE10P coprocessors. Various aspects of WRF optimization were analyzed, many of which contributed significantly to optimization on Xeon Phi. These optimizations show that symmetric execution on Xeon and Xeon Phi can be used for highly efficient WRF simulations that significantly speed up execution relative to running on either homogeneous architecture. Performance analysis on Xeon Phi also exposes hybrid parallelization issues with the WRF model that are overlooked when running on architectures with fewer available threads per processing unit.

1. INTRODUCTION

The WRF model is a well-established and widely used code in the weather, climate and HPC communities. General performance portability is essential for any such community codes to take advantage of the ever-changing HPC architectures as we move into the exascale era of computing. Efficient shared memory models and workload distributions are critical as core counts on shared memory systems will continue to increasing. Intel's many-core architecture (MIC) is very representative of such systems and therefore performance optimization of WRF on MIC architectures will greatly contribute to the overall performance portability of the model on current and future HPC architectures. In addition to performance portability for future systems, it is necessary to understand how the model behaves on current systems to best utilize our current HPC resources.

Throughout this study we have analyzed the performance of various components of the WRF model to create a set of guidelines to inform WRF users on how to efficiently use their allocations on systems with Intel multicore and manycore architectures. In addition we used Xeon Phi to exploit hybrid MPI + OpenMP parallelization issues with the model.

2. Host Node Analysis and Optimization

2.1 Hybrid MPI + OpenMP Parallelization

2.1.1 Task/Thread Binding

The majority of WRF users initially write off using hybrid parallelization due to poor initial performance results. This is typically due to the lack of task/thread binding. Threads spawned across sockets do not share L3 cache, causing issues such as false sharing to become more prevalent.

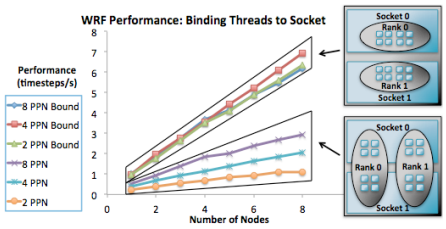

Figure 1.

2.1.2 Hybrid vs. Pure MPI Strong Scaling

When we have less MPI tasks, there are less MPI patches and therefore less ghost layer communications. By using a hybrid implementation, we cut out a significant portion of the MPI communication while still utilizing the same number of cores. This results in better scaling for hybrid runs in MPI bound workloads, which is demonstrated in figure 2.

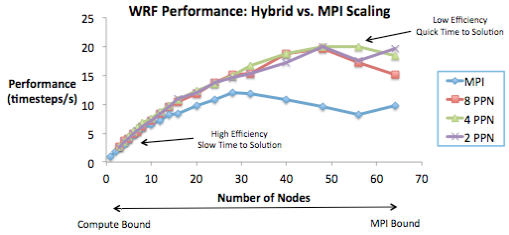

Figure 2.

2.2 I/O Considerations

WRF I/O, considering time spent in pure I/O as well as MPI communication and data formatting that results directly from I/O reads and writes, can quickly take over a simulations runtime. This is extremely significant for larger problem sizes and when running on larger numbers of nodes. The following plot shows history write times using serial I/O, PnetCDF parallel I/O, and namelist option io_form_history=102, which writes separate output files for each MPI rank. Using PnetCDF significantly reduces I/O times and although the third option requires post processing (very cheap relative to I/O related processes during the simulation), writing to separate output files made history writes negligible during the simulation. These results are demonstrated in Figure 3.

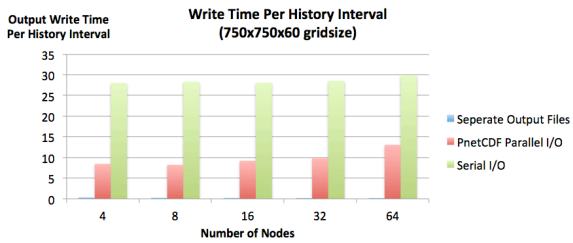

Figure 3.

3. Xeon Phi Optimization

3.1 Initial Scaling Performance

Although initial scaling of WRF on Xeon Phi shows poor results (see figure 4.), we can see that the performance approaches that of the host cpu's in extreme high workload per core simulations. To better understand this we scaled up the gridsize while running on a constant number of nodes. We should expect that for either architecture, due to insufficient workloads per core, the performance per grid point should be low initially and level off to a maximum for sufficient workloads per core. What we find is that this maximum is initially reached for much larger workloads per core on Xeon than on Xeon Phi and for greater than approximately 60,000 horizontal gridpoints/processor, it actually becomes more efficient to use Xeon Phi over the host Xeon CPUs (see figure 5.).

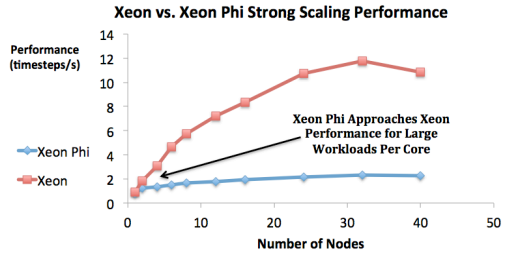

Figure 4.

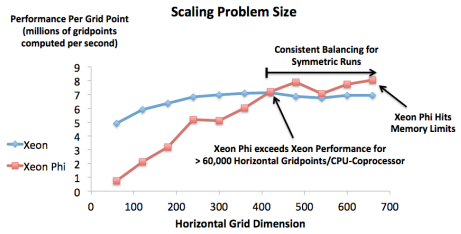

Figure 5.

3.2 Symmetric Xeon + Xeon Phi Performance

For large workloads per core, due to low MPI overhead and constant efficiency it is possible to have well balanced symmetric CPU-Coprocessor WRF runs that are more efficient than running on either homogeneous architecture. Figure 5 demonstrates a case where over a 1.5X speedup can be achieved running on the same number of nodes symmetrically.

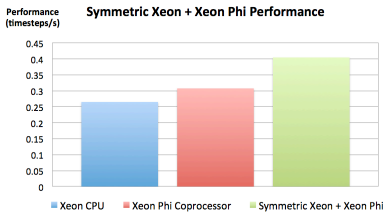

Figure 6.

4. OpenMP Tile Decomposition

WRF parallelization is done by breaking down the grid into two-dimensional horizontal patches, which are distributed between the MPI tasks to execute. Each OpenMP thread then solves and updates a subset of this patch, which we call an OpenMP tile. We found that WRF's patching and tiling strategies often cause imbalancing and categorized these as follows:

1. Number of Patch Rows > Number of OpenMP Threads
2. Number of Patch Rows < Number of OpenMP Threads
3. Number of OpenMP threads is any multiple of the number of patch rows

These issues are much more prevalent on Xeon Phi as there are many more threads available (up to 244) opposed to the Xeon CPU's (up to 8 on Stampede).

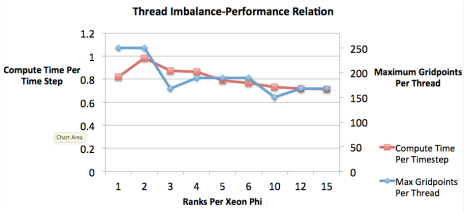

Figure 6. Plotting the maximum number of gridpoints assigned to each thread overlaid by the total compute time per iteration shows the general relation between the two.

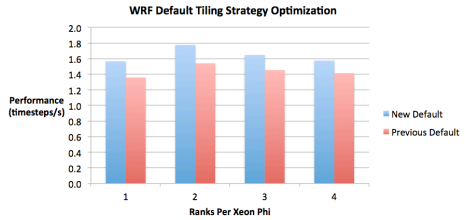

Figure 7. Demonstrating performance benefit for new default tiling strategy.

5. Future Work

We would like to continue developing better hybrid parallelization strategies. This includes better patching and tiling strategies as well as the consideration of alternative non-uniform work distribution such as OpenMP task constructs. We would also like to compare our results on a variety of systems to ensure the portability of these results. The upcoming Knights Landing architecture is of significant interest in particular.

6. ACKNOWLEDGMENTS

I would like to first thank my advisor Davide Del Vento (NCAR) as well as Dave Gill (NCAR), John Michalakes (NCAR), Indraneil Gokhale (Intel), and Negin Sobhani (NCAR) for their guidance and contributions. I would also like to thank NCAR, the SIParCS internship program, and XSEDE for providing me the opportunity, community and resources to work on this project.

7. REFERENCES

[1] Michalakes, J., J.Dudhia, D. Gill, J. Klemp, and W. Skamarock. Design of a next-generation weather research and forecast model. In proceedings of the Eighth Workshop on the Use of Parallel Processors in Meteorology, European Center for Medium Range Weather Forecasting. World Scientific, Singapore, 1999

[2] Skamarock, William C., Joseph B. Klemp, Jimy Dudhia, David O. Gill, Dale M. Barker, Wei Wang, and Jordan G. Powers. A description of the advanced research WRF version 2. No. NCAR/TN-468+ STR. National Center For Atmospheric Research Boulder Co Mesoscale and Microscale Meteorology Div, 2005.