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

Samuel Elliott – Advisor: Davide Del Vento

National Center for Atmospheric Research

University of Colorado, Boulder

Introduction

The Weather Research and Forecasting Model (WRF) is a widely used mesoscale numerical weather prediction system used for both research and operational forecasting needs. General performance portability is essential for any such community code to take advantage of ever-changing HPC architectures as we move into the exascale era of computing. Efficient shared memory models are critical as core counts on shared memory systems will continue to increase. Intel's many integrated core (MIC) architecture 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. 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.

Goals

1. **Generalize optimizations and best practices to educate current WRF users on how to best utilize HPC resources.**
2. **Identify performance bottlenecks and issues with WRF hybrid parallelization on both Xeon CPUs and Xeon Phi Coprocessors.**

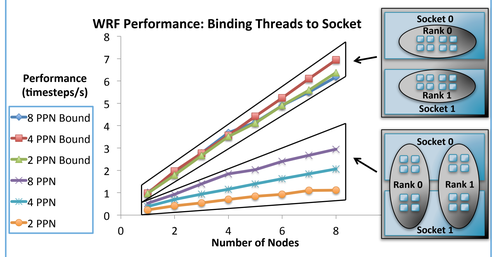
Benchmarking

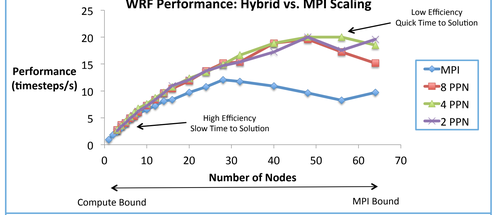
All benchmarks in this study were run on the Texas Advanced Computing Center's Stampede supercomputer. Each node is comprised of two Intel Xeon 2680 CPU's and utilizes Intel Xeon Phi SE10P coprocessors.

Xeon E5 2680 CPU	Xeon Phi SE10P Coprocessor
8 Cores	61 Cores
2-Way Hyperthreading	4-Way Hyperthreading
256 Bit Vector Registers	512 Bit Vector Registers
2.7 GHz	1.1 GHz
32 KB L1	32 KB L1
256 KB L2	512 KB L2
20 MB Shared L3	No L3 Cache
32 GB Main Memory	8 GB Main Memory

Hybrid WRF Scaling and Task/Thread Binding

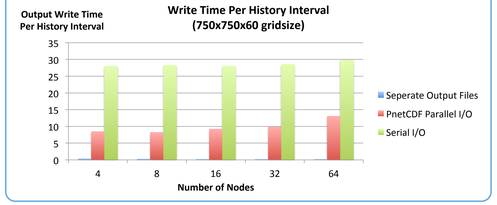
The majority of WRF users initially write off using hybrid parallelization due to poor initial performance results. This is typically due to the lack of task/thread binding. Threads spawned across sockets do not share L3 cache, causing issues such as false sharing to become more prevalent. The following performance results show how critical using thread binding within MPI ranks is (First plot below). By using a hybrid implementation, we cut out a significant portion of the MPI communication while still utilizing the same number of cores. The following shows how using a hybrid implementation allows WRF to scale more efficiently to a larger number of nodes.





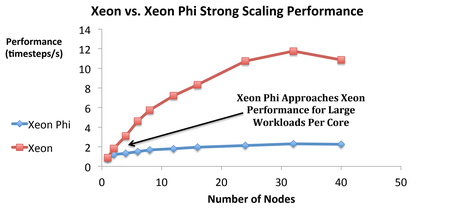
I/O Considerations

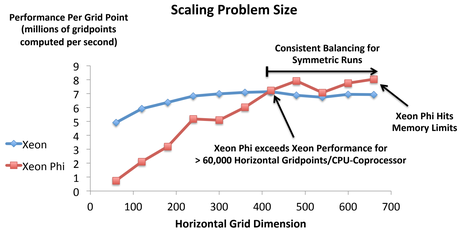
WRF I/O, considering time spent in pure I/O as well as MPI communication and data formatting that results directly from I/O reads and writes, can quickly take over a simulations runtime. This is extremely significant for larger problem sizes and when running on larger numbers of nodes. The following 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.



Xeon Phi Scaling

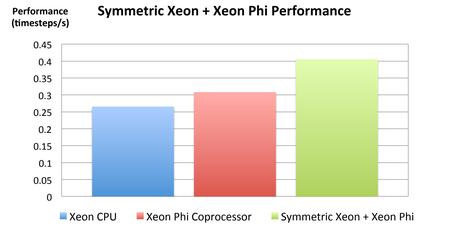
Although initial scaling of WRF on Xeon Phi shows poor results (first plot), we can see that the performance approaches that of the host cpu's in extreme high workload per core simulations. To better understand this we scaled up the gridsize while running on a constant number of nodes. We should expect 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 (second plot).





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. The following results demonstrate a case where over a 1.5X speedup can be achieved running on the same number of nodes symmetrically.



WRF OpenMP Tile Decomposition

WRF parallelization is done by breaking down the grid into two-dimensional horizontal patches, which are distributed to 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

   In this case each thread will execute a single tile but since the number of rows in a patch may not break down evenly into the number of tiles, some tiles will consist of more rows than others (up to a 2x imbalance between threads).

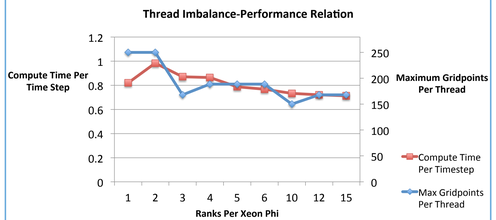
2. Number of Patch Rows < Number of OpenMP Threads

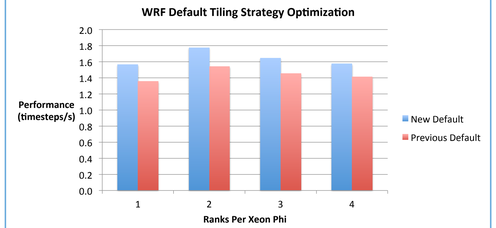
   In this case WRF will decompose the tiles 2-dimensionally, each tile with a contiguous subset of a single row. The number of tiles will be greater than the number of threads, requiring that some threads execute two tiles while others only execute one (always 2x imbalance between threads).

3. Number of OpenMP threads is any multiple of the number of patch rows

   This case should be optimal for thread balancing but found that WRF's default strategy would overdecompose the problem causing unnecessary imbalances. We have since changed the tiling algorithm to fix this issue which will be included in the next release of WRF.

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). The first plot below shows a case that whose performance is hindered by the first and second issues explained above. This case demonstrates the correlation between execution time and maximum workload per core. The second plot below shows the performance benefits in the third case after changing the default tiling algorithm.





Conclusions: WRF Best Practices

- **Hybrid WRF parallelization will give consistent performance benefits, especially for MPI bound workloads.**
  - **Ensure utilization of task/thread binding using runtime scripts or environment runtime settings (this is specific to each system and MPI implementation).**
  - **Ensure that the environment variable OMP_STACKSIZE is set correctly based on the memory limitations of your system**
- **Using parallel I/O options with PnetCDF will significantly decrease I/O time spent in I/O related processes.**
- **Using separate output files for each MPI task makes writing history negligible.**
  - **Change namelist option: io_form_history = 102**
  - **This option does require post-processing but is very worthwhile as the I/O overhead dominates runtimes on larger core counts.**
- **Understand the tradeoff between quick time to solution/low efficiency and high efficiency/slow time to solution.**
- **Only use Xeon Phi if you are extremely focused on efficient utilization of HPC allocations on systems such as stampede.**
  - **Using optimal WRF I/O options is critical for utilization of Xeon Phi**
  - **Efficient utilization of Xeon Phi is limited to a very small window of workloads.**
  - **For specific cases, symmetric execution can be used for efficient utilization of HPC resources.**

Future Work

- **Further develop methods for better patching and tiling strategies**
- **Extend studies onto various systems including the Knights Landing architecture which is more representative of future manycore HPC systems**
- **Assess whether other shared memory models such as OpenMP task constructs will further WRF's performance portability for current and future HPC architectures**
- **Better understand performance issues associated with memory allocation that would otherwise dramatically decrease memory utilization in hybrid WRF simulations**

References

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

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.

Acknowledgements

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 discussions. 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.