Performance Evaluation of Emerging High Performance Computing Technologies using WRF

Gregory B. Newby, Don Morton
Arctic Region Supercomputing Center
University of Alaska Fairbanks
Fairbanks, AK 99775
[newby|morton]@arsc.edu

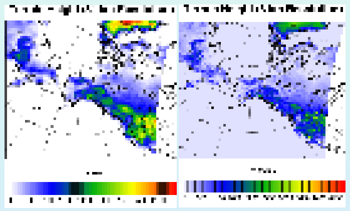
Introduction

Benchmarking of the Weather Research and Forecasting (WRF) model is performed on three different multi-core architectures available on the high end systems at the Arctic Region Supercomputing Center (ARSC). ARSC has been performing high-resolution quasi-operational runs on multi-core systems for several years, and has recently initiated a study in the performance of codes like WRF.

This poster presents a suite of benchmark cases developed to support a broad array of systems, describes a set of architectures to be evaluated, then presents a sampling of performance evaluations.

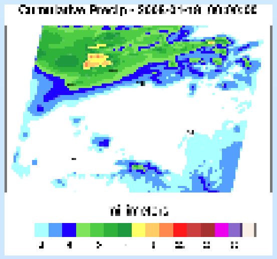
Benchmark Cases

A suite of WRF benchmark cases has been constructed to facilitate performance evaluation on architectures ranging from small-memory single-core to tens of thousands of cores. The domain for all cases is a 6,025 x 6,025 km region mapped on to a polar stereographic projection, centered (as close as resolution allows) at the Arctic Region Supercomputing Center in Fairbanks, Alaska. The benchmark cases all utilize this domain, but at grid resolutions ranging from 81km down to 3km – a 1km-resolution domain is currently being developed.



Resolution	Grid Points
81km	75x75x28 = 157,500
27km	225x225x28 = 1,417,500
09km	675x675x28 = 12,757,500
03km	2025x2025x28 = 114,817,500
01km	6025x6025x28 = 1,033,357,500

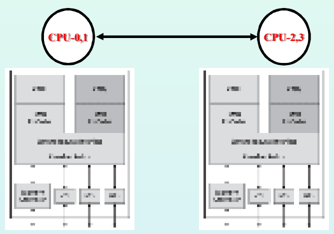
The specific benchmark case comes from a weather event in January 2008 which left Fairbanks with an unexpected, highly-localized, heavy snow event, which WRF captured well. To create the case-study, we ran a simulation for 72 forecast hours starting on 15 January 2008 at 00Z. The benchmark case consists of a WRF restart file (and a file with lateral boundary conditions, wrfbdy_d01) from Forecast Hour 48 (17 January 2008 at 00Z), and all cases are run for three forecast hours from the restart file.

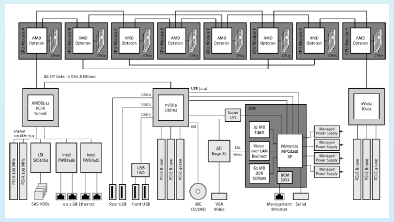


Benchmarking Systems

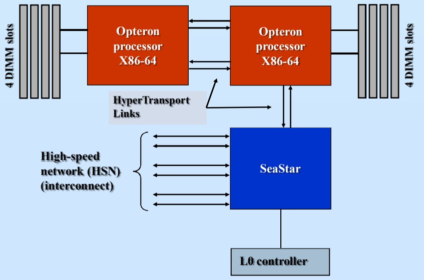
Two major systems at the Arctic Region Supercomputing Center were used for the benchmark tests – *midnight*, a Sun cluster based on Sun Fire X2200 and X4600 compute nodes, and *pingo*, a Cray XT5 based on 8-core nodes.

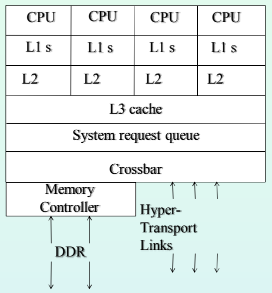
On *midnight*, the X2200 node – referred to as a 4-way node - is comprised of two dual-core 2.6 GHz AMD Opteron processors with 16 GB of shared memory per node, and a 4X Infiniband network card on PCI-Express Bus. The X4600 node – referred to as a 16-way node – is comprised of eight of the dual-core 2.6 GHz Opteron's, with 64 GB of shared memory per node.





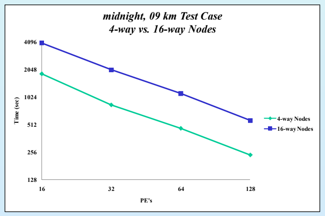
On *pingo*, all compute nodes have two quad-core 2.3 GHz AMD Opteron processors with 32 GB of shared memory per node, connected via the Cray SeaStar 2+ Interconnect.



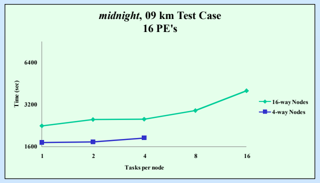


Some Performance Results

midnight, with its 4-way and 16-way nodes of dual-core Opterons was first observed. Given the common knowledge that 4-way nodes "performed better" than 16-way nodes, we considered general scalability of the 09 km test case (12.8 million nodes). The nodes were "fully-loaded" – all cores of each node were used. For both sets of nodes, scalability was excellent, but it is evident that use of the 16-way node resulted in a factor of two performance degradation.



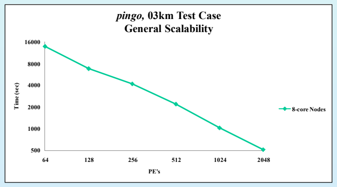
Given the uncertainty as to whether the performance degradation was a result of increased contention in the 16-way nodes, we performed another test that evaluated the performance of a 16 PE run using different numbers of cores from 1 core per node to 16 (or 4, in the 4-way nodes). In this case, it is interesting to note that even with one task per node, the performance of the 16-way nodes is worse. Neither of the nodes exhibited serious deterioration in performance as up to four tasks were added. The 16-way node exhibited increased deterioration once we used 8 and 16 cores.



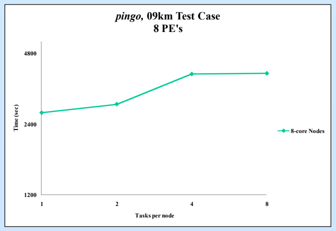
Given that each of the nodes was comprised of a number of dual-core processors (2 in the 4-way nodes, 8 in the 16-way nodes), we were interested in determining how placement of tasks across processors would influence performance. For the 81 km, 27 km and 09 km test cases we ran 2-PE evaluations on a 4-way node using the *taskset* option to map the two tasks to a single processor (cores 0 and 1) and to both processors (cores 0 and 2). The following table displays the timings (in seconds) of the cases. Note that in the 09 km case, we didn't perform a full 3-hour simulation, so the times provided are for three timesteps of the simulation (the same three timesteps in both cases). In the 81km case no significant difference in performance was noted, but as the workload increased – as in the 27km and 09 km cases – distributing the work across two processors, rather than loading down a single processor, seemed to provide a reasonable improvement.

	81km	27km	09km*
Cores 0 and 1	25.4	489	100.4
Cores 0 and 2	25.0	432	92.5

pingo is comprised solely of 8-core nodes, each consisting of two quad-core Opterons. The first test performed was a general scalability evaluation of the 03 km case, consisting of 114.8 million grid points. Scalability was excellent, with an approximate halving of simulation time with a doubling of nodes.

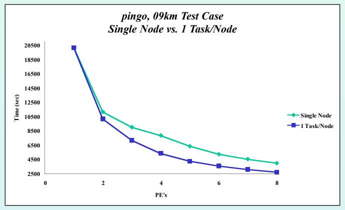


After the general scalability test we were interested in learning how the "loading" of a node would affect performance, so we ran tests on the 09 km test case, all using 8 PE's, but in different node configurations ranging from 1 task per node to a fully-loaded 8 tasks per node. As expected, performance was best when we distributed the load across 8 cores, using only one core per node. Interestingly, performance didn't degrade when increasing the loading from 4 to 8 tasks per node.

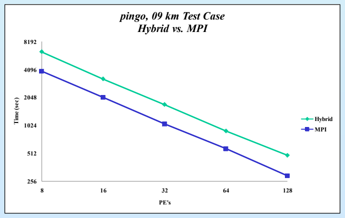


The next test performed was intended to show us how scalability varied between one-task-per-node and fully-loaded nodes. Using the 09 km case we increased the number of tasks from 1 to 8 in order to gradually increase the loading on a single node. This experiment revealed reasonable scaling in both the single-node and one-task-per-node cases, but clearly, utilization of one core per node gives us better performance than loading the tasks on to a single node.

It's interesting to note that using two cores of a node is almost as good as using two nodes, each with one core. We have theorized that in the case of two tasks on a node, the tasks are assigned to different processors and experience less contention. At this time Cray, Inc. has not implemented task-affinity directives, so we are unable to test this further.



Finally, the WRF model supports multi-core architectures through an MPI task distributed memory paradigm and an OpenMP shared memory thread paradigm. Additionally, a hybrid approach exists whereby MPI tasks are allocated in a coarse-grained fashion (presumably one task per node), and each MPI task spawns a number of threads (presumably one thread per core of a node) to utilize the shared memory in a node. We compared the MPI and the hybrid MPI/OpenMP approaches (below) and noted that the MPI-only case outperformed the hybrid case by almost a factor of two. Other experimenters have claimed less significant differences between the two, so further investigation is warranted. However, it is interesting to note that in the 8-PE case, only one 8-core node is being used, and the MPI distributed memory paradigm between the eight cores is significantly outperforming the shared memory OpenMP paradigm on the single node.



Summary

This work represents a first step in evaluating the performance of WRF on a variety of emerging HPC technologies. A flexible suite of benchmark test cases has been developed and has been utilized for performance evaluation on small and large cases.

Although no real surprise, evaluations have shown that "loading" a node with tasks degrades performance relative to spreading the tasks amongst lightly-utilized nodes. However, scalability of WRF does not seem to suffer significantly when utilizing all of the cores in a node. There is some evidence to suggest that a balanced distribution of tasks amongst processors in a node will yield better performance than an unequal distribution.

The number of compilation and run-time parameters in a multi-core environment is large, and each one of the tests presented here could be analyzed in more detail.