Petascale WRF Simulation of Hurricane Sandy

Deployment of NCSA's Cray XE6 Blue Waters

Peter Johnsen
Meteorologist, Performance Engineering Group
Cray, Inc.
St. Paul, MN. USA
pjj@cray.com

Mark Straka
Sr. Research Programmer, NCSA
University of Illinois at Urbana-Champaign
Urbana, IL. USA
m-straka@illinois.edu

Melvyn Shapiro
Alan Norton
Thomas Galarneau
National Center for Atmospheric Research
Boulder, CO. USA
{mshapiro,alan,tomjr}@ucar.edu

***Abstract*—The National Center for Atmospheric Research (NCAR) Weather Research and Forecasting (WRF) model has been employed on the largest yet storm prediction model using real data of over 4 billion points to simulate the landfall of Hurricane Sandy. Using an unprecedented 13,680 nodes (437,760 cores) of the Cray XE6 "Blue Waters" at NCSA at the University of Illinois, researchers achieved a sustained rate of 285 Tflops while simulating an 18-hour forecast. A grid of size 9120x9216x48 (1.4Tbytes of input) was used, with horizontal resolution of 500 meters and a 2-second time step. 86 Gbytes of forecast data was written every 6 forecast hours at a rate of up to 2 Gbytes/second and collaboratively post-processed and displayed using the Vapor suite at NCAR. Opportunities to enhance scalability in the source code, run-time, and operating system realms were exploited. The output of this numerical model is now under study for model validation.**

Categories—scalability; time-to-solution

Keywords—Cray; NCSA; WRF; NCAR; hurricane; weather; forecast; simulation; storm; prediction; High Performance Computing

1. INTRODUCTION

The devastation incurred by the landfall of Hurricane Sandy on the northeast coast of the United States during the last days of October 2012 exemplifies the need for further advances in accuracy and reliability in numerical weather prediction. As costly as this storm was, there are numerous examples of communities and authorities who heeded the forecasts of the storm's probable path and took appropriate measures for food, shelter, or even evacuation. However, there clearly remains significant potential for greater accuracy when predicting exact landfall time and place, as well as expected wind and water damage [2]. High resolution numerical weather simulations carried out on hundreds of thousands of processors on the largest supercomputers can provide these insights, as they allow for rapid time to solution previously unimagined. Thus, progressively more accurate parameters can be incorporated into current storm models.

The Weather Research and Forecasting (WRF) Model is a mature, multi-component application suite for mesoscale numerical weather prediction. Among its uses are operational forecasting and atmospheric research [1]. It features multiple dynamical cores and a 3-dimensional variational data assimilation system contained within a software structure allowing for computational parallelism and system extensibility. WRF has been applied to solution domains ranging from meters to thousands of kilometers. The WRF project has been developed collaboratively through a partnership among the National Center for Atmospheric Research (NCAR), the National Oceanic and Atmospheric Administration (including the National Centers for Environmental Prediction (NCEP), the Forecast Systems Laboratory (FSL), the Air Force Weather Agency (AFWA), the Naval Research Laboratory (NRL), the University of Oklahoma, and the Federal Aviation Administration (FAA) [4].

Previous extreme-scale WRF experiments involved idealized simulations on a dry atmosphere using 2 billion grid cells [6]. Performance at that time peaked at 7.1 Tflops/second sustained on 12,500 cores of a Cray XT4 system. Subsequent tests on 148,000 cores of a Cray XT5 system reached 50 TFlops/second using the idealized case. In contrast, the simulation reported here uses two times the number of grid points and is a full non-hydrostatic forecast with full WRF moist physics.

2. SYSTEM ARCHITECTURE

The Blue Waters supercomputer provides sustained performance of 1 petaflop on a range of real-world science and engineering applications. Blue Waters is composed of 237 Cray XE6 cabinets plus 32 cabinets of Cray XK7 with NVIDIA® Kepler™ GPU computing capability [12].

The Cray XE6 processor is a 16-core 64-bit AMD Opteron 6276 series (Interlagos). It features 8x64 KB of L1 instruction cache, 16x16 KB of L1 data cache, 8x2 MB of L2 cache per processor core, and 2x8 MB shared L3 cache. Up to 192 processors can populate a cabinet.

SC '13, November 17 - 21 2013, , USA

http://dx.doi.org/10.1145/2503210.2503231

The memory system can be either 32 GB or 64 Gbytes (Blue Waters has 64) registered ECC DDR3 SDRAM per compute node, with a memory bandwidth of up to 102.4 Gbytes/s per node.

The interconnect is a 3-D torus, with 2 compute nodes connected to a Cray Gemini ASIC router. There are 48 switch ports per Gemini chip providing a 160 Gbytes/s switching capacity per chip.

Disk storage is comprised of a Sonexion 1600 with the Lustre parallel file system. Total available storage is 26.4 Pbytes and can achieve an aggregate I/O bandwidth of greater than 1 Tbyte/second.

Figure 1 illustrates the XE6 architecture with 2 nodes, each comprised of 32 AMD Opteron cores (2 sockets) connected to a Cray Gemini ASIC router.

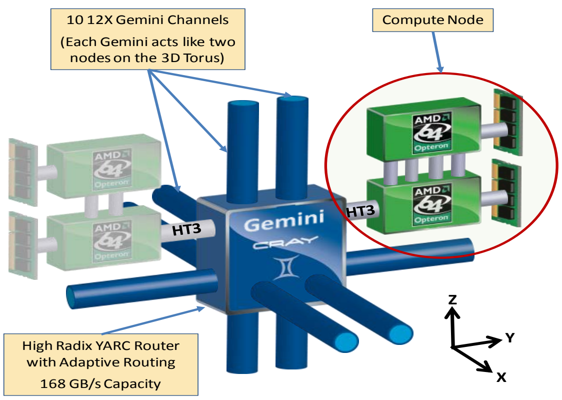

Figure 1. Illustration showing XE6 architecture, 2 nodes connected to a Cray Gemini ASIC.

3. WRF FORECAST DEFINITION

A WRF forecast problem was designed to use as much of the Blue Waters system as possible and, at the same time, simulate a meteorologically significant event. Centered on the location of Hurricane Sandy's eye on the 29th of October, a horizontal grid of 9120 by 9216 points and 48 vertical levels was defined using WRF pre-processing utilities WPS [4]. NOAA/NCEP GFS 0.5 degree model output was used to create required initial and boundary conditions for WRF.

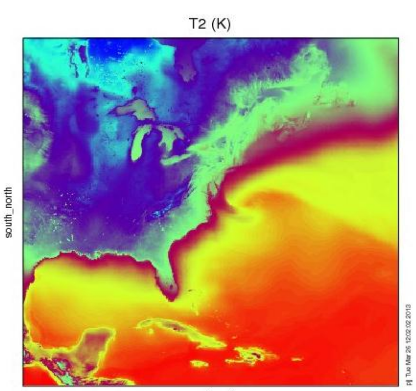

Figure 2. WRF forecast domain showing temperature at 2 meters above ground at 13Z 29Oct2012

Key forecast parameters include:

- 500 meter grid resolution
- 2 second integration time step
- Single-moment, 5-class microphysics (WSM5)
- RRTM/Dudhia radiation schemes
- Cold start at 12Z 29Oct2012 (we note that spin-up of the model was already evident at 6 minutes, or 180 time steps, into the forecast)
- WRF auxiliary history file dumped every 6 forecast hours

Note that additional experiments using 150 vertical levels instead of 48 are now underway.

4. PROFILING, ANALYSIS, AND TUNING

4.1 General Observations

Our motivation for efficient petascaling is to reduce time to solution and not necessarily to decompose the domain to fit in processor main memory. However, empirical studies of WRF have consistently shown the code to be memory bandwidth bound [3], so our strategy was to always be mindful of cache effects and loop ordering, as much as was under our control without major source code restructuring.

4.2 Methodology

Conceptually, there are 3 realms available to WRF users for tuning performance: the ***source code, run-time,*** and ***system*** level. Besides the obvious, the source code category contains build options, architecture-specific configuration flags, and compiler directives and options. The run-time realm, for our purposes, consists largely of parameters in WRF's *namelist.input* file which can be adjusted without recompiling. The system layer refers to environment variables, topology layout, *aprun* options, or any other options which are **independent** of the specific application being run. In this section we introduce our methodology and empirical studies as they fall under each of these categories.

4.2.1 Source Code Layer

The WRF version 3.3.1 code was modified from the public distribution chiefly with concerns for I/O burden per MPI task. This involved mainly limiting the creation of Runtime System Library (RSL) [13] output to only rank 0, instead of each process, as that leads to large system time overhead at scale and significantly impacts performance. Similarly, a number of WRF informational messages written by each MPI rank were limited to the root rank only.

4.2.2 Runtime Layer

WRF is a hybrid MPI/OpenMP code, and as such decomposes the global grid and distributes memory via rectangular subdomains called *patches* to the MPI ranks.

WRF allows for an internal layout of tasks using the namelist input variables *nproc_x* and *nproc_y*. It has been empirically determined to give better performance if nproc_x << nproc_y, because this leads to longer vectors on the inner compute loops (indexed over "I" in the source code). If not specified, WRF will, by default, choose as close to a square decomposition as possible; but this is seldom optimal for speed.

Each distributed-memory patch will have some number of shared-memory *tiles*. Tiles are subdivisions of the patches, and are typically bracketed by OpenMP parallel directives (although tiling can function independently of whether OpenMP has been activated into the compilation). Tiling thus provides a 2nd layer of hybrid parallelism. The effect of tiling is to allow more chunks of work to fit into cache.

This hybrid paradigm implies four logical configurations on the Cray XE node: 32, 16, 8, or 4 MPI tasks, with 1, 2, 4, or 8 OpenMP threads, respectively, for each. Since the size of a NUMA region is 8 integer cores, it would not be in the best interest of performance to exceed that number of threads. Tile sizes can similarly be defined in the input file, or WRF will choose a simple decomposition by default. Via these parameters, WRF conveniently allows the user to influence cache behavior and tune for various domain sizes, machine topology, and processor counts.

WRF allows for several types of parallel I/O, including use of the parallel netCDF library, quilting via servers, and multi-file (one file per MPI task). The latter scheme is not transportable between jobs if the number of processors changes, since each task is expecting its own dedicated file for input and/or output. This method is very fast, but the tradeoff is that the number of files will obviously be huge, at scale. Quilting allows for a certain subset of MPI tasks to be dedicated as I/O servers.

Finally, use of WRF's auxiliary history output options to select only the output fields of greatest interest, thus reducing the volume of output considerably, was of great interest to us.

4.2.3 System Layer

This regime contains the wide selection of application-independent MPICH environment variables, as well as Cray-specific topology-aware task placement tools which we found to benefit WRF and other applications on Blue Waters to varying degrees. It also includes any options to the *aprun* command or the batch system, which have no knowledge of the specific executable being run. The MPICH tunable parameters span the gamut from communication protocol and message sizes to rank reordering; the latter being the one of greatest impact to us for this project, as we will describe further below.

4.3 Preliminary Experiments

4.3.1 Load Imbalance

Although weather simulations typically exhibit some load imbalance - usually in the complex microphysics which calculates the formation of various precipitation types (e.g. rain, graupel, ice) in the atmospheric layers - with this hurricane simulation we observed little load imbalance because of the large rain and extensive cloud shield. The Cray profiling library provides interfaces to the PAPI hardware counters, produces min/max values for each, as well as the corresponding process locations. This is useful for estimating load imbalance and identifying topology refinement challenges.

4.3.2 Jitter Analysis

At larger scale (>10,000 cores), we did see periodic increases of up to 50% wall clock time in regular, periodic groups of integration steps. We knew implicitly that this was not due to the code or input model, so we began to suspect some kind of external interference. We attempted to alleviate this apparent jitter by some of the most common methods: first, the use of *core specialization*: essentially allocating non-user resources by either deliberately idling a core module or explicitly using the –r1 option to the aprun command while still utilizing all cores on a node; second, assuring that we were running on dedicated partitions of the torus; and third, considering *balanced injection*: attempts were made to tune the injection bandwidth of the compute nodes with the network for certain communication patterns. [10]

We observed that core specialization actually degraded performance somewhat, balanced injection had no perceivable effect, and running in dedicated mode only improved performance by about 1.2%. Thus, we concluded that the regular (every 75s of wall clock time) spikes in step times were most likely due to Lustre *ping* effect: essentially, clients ping all metadata and object targets in the file system. The pings serve three purposes, but for simplicity here, we shall generalize that they are all related to detecting server health. Lustre pings become exacerbated with scale. Because each client pings each target, this creates O(n*m) complexity. For Blue Water's sized systems on the order of 25,000 nodes (n) and hundreds of OSTs (m) there can be tens of millions of pings per ping interval [9]. Currently, we do not have a solution to the Lustre ping effect, and we must average in these larger times with our total integration steps.

4.3.3 Topology Effects for Communication

Overall, we noted only a little over 1% performance increase between batch and dedicated runs. This indicates that sharing links of the torus with other running jobs had minimal impact on performance. Along these same lines, we also invested a significant amount of time investigating a "best fit" node placement scheme. This involves essentially reserving the entire machine via batch, which then allows the full connectivity of the torus and all available compute nodes to be optimally picked from for a job running on some subsection of the machine. For example, in theory we could exploit the fact that the Y-dimension of the torus has ½ the bandwidth of the X and Z dimensions; thus, by assigning the "shorter" side of the WRF global rectangular domain to the Y dimension, we hoped that fewer communications in that dimension would thus be better balanced. Unfortunately, although this optimal node mapping technique proved quite successful on other applications on Blue Waters, it did not produce more than a small improvement for WRF.

Domain configuration and process layout using MPI rank ordering features of the XE6 job scheduler (ALPS) are a cornerstone in efficiently using the XE6 3D torus interconnect and allowing WRF to scale this successfully. We used the Cray *grid_order* perl script to generate improved placement of the ranks for the primary communication pattern in the WRF solver, which is nearest neighbor halo exchanges. Reducing the number of neighbors communicating off-node is the primary goal.

We found the most effective way to run WRF on the AMD Bulldozer core-modules was to use MPI/OpenMP hybrid mode with 2 OpenMP threads per MPI rank. This puts 16 MPI ranks on each XE6 node.

By default the XE6 job scheduler places MPI ranks in serial order on the machine, packing processes in accordance with the defined system topology. This is illustrated in Figure 3, where the first 16 MPI ranks are all placed on the first node, the next 16 on the next neighboring node, and so on. But halo exchange partners are not mapped this way in WRF. For instance, MPI rank 17 exchanges halo regions with MPI ranks 1, 16, 18, and 33. Only neighbors 16 and 18 are on the same node. But, using an alternate placement allows us to get 3 communications partners for most MPI ranks on the same node. At very high scales, this strategy improves overall WRF performance by 18% or more.

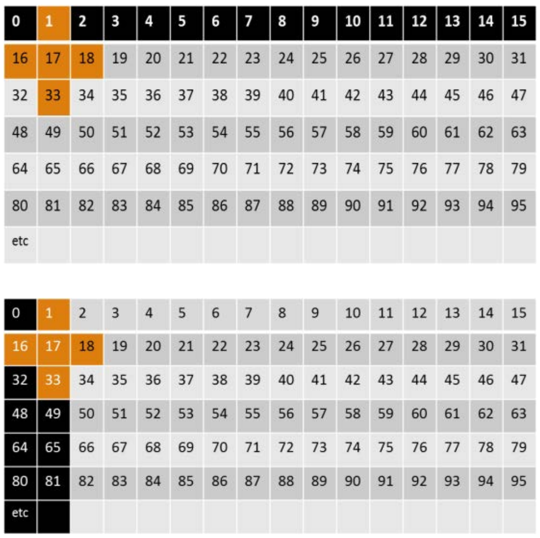

Figure 3. The top figure represents the default MPI rank placement (black squares) for a WRF 16xNPROC_Y decomposition. Note that, at most, each MPI rank has 2 communication partners on the same node (orange squares). The bottom figure shows the alternate rank placement strategy used to improve WRF scaling. In this scheme, most MPI ranks now have 3 partners on the same node.

As mentioned previously, WRF allows the flexibility of decomposing the horizontal domain at runtime, as long as the number of processors in the X and Y directions are integer factors of the total number of processors being launched. In keeping with our observations for optimal layouts, we decompose the WRF grid into rectangles with latitudes longer than longitudes for each subdomain. The optimized placement we've employed also has the benefit of sending smaller east-west direction exchanges off-node and keeping as many larger north-south messages on-node as possible – 75% fewer bytes are sent over the network.

Table 1 provides message statistics for the 13,680 node run assuming 42 halo exchanges occurring for each WRF integration step. The Cray XE6 interconnect is easily handling over 12 million off-node halo exchange messages totaling 280 Gbytes every WRF time step.

Table 1. Halo exchange messaging statistics for a single WRF time step on 13,680 XE6 nodes (218,880 MPI ranks). *Total Bytes Exchanged* assumes each message contains two packed 3D single precision variables and a halo region of 5 slices

Placement Method	*Total Messages*	*Total Bytes Exchanged*	*On-node Messages*	*Off-node Messages*	*Off-node Bytes Exchanged*
Default Placement	3.6E07	1.5E12	1.8E07	1.8E07	1.1E12
Optimized MPI rank Ordering	3.6E07	1.5E12	2.4E07	1.2E07	2.8E11

5. PERFORMANCE RESULTS

Through simulation of a compelling real-world problem, we have demonstrated that a complex scientific application code can be run at heretofore unmatched scale while achieving impressive levels of performance.

5.1 Strong Scaling

Sustained performance was calculated for forecast integration only and obtained on a lightly loaded Blue Waters system before it was available to the general community. All weather calculations and nearest neighbor halo exchanges are included, but not I/O overhead. This methodology is used by NCAR when computing sustained WRF performance [7]. FLOP count, per integration time step, was obtained using CrayPat performance analysis tool and PAPI library from AMD Opteron hardware counters. Specifically, PAPI counter RETIRED_SSE_OPS:ALL was used to collect full FLOP count. Average FLOP count per time step was then obtained by dividing by the total number of time steps in the forecast run. This yielded an average Tflop count of 32.454 Tflops/second.

Sustained performance is shown in Figure 4 along with the parallel efficiency compared to a base run on 8,192 cores. Parallel efficiency is still above 60% even on 13,680 XE6 nodes. At the top of the scaling curve, each subdomain, or MPI rank, is working on a subset of only 18,342 grid points. Thus, total halo exchange overhead is still relatively low, even at extreme scale.

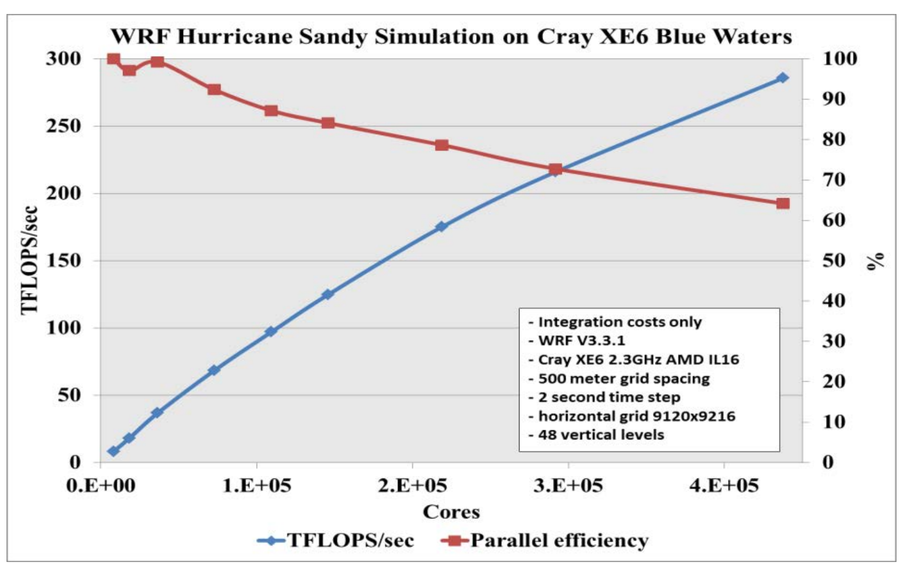

Figure 4. Strong scaling of Hurricane Sandy run. Sustained performance in Tflops/second (y-axis, left) and parallel efficiency over base run on 8,192 cores (y-axis, right) are shown.

Table 2 lists, for each scaling run, the number of nodes used, WRF decomposition strategy employed, the average integration time step in seconds along with sustained performance.

Table 2. Strong scaling details. Each MPI rank uses 2 OpenMP threads.

Core Count	*XE6 Nodes*	*Horizontal Decomposition (MPI ranks)*	*Average Time Step (seconds)*	*Sustained Performance (Tflops/sec)*
8192	256	32x128	3.895	8.3
18240	570	38x240	1.802	18.0
36480	1140	76x240	0.882	36.8
72960	2280	95x384	0.474	68.5
109440	3420	120x456	0.334	97.1
145920	4560	190x384	0.260	124.8
218880	6840	228x480	0.185	175.1
291840	9120	285x512	0.150	216.0
437760	13680	285x768	0.114	285.7

5.2 Weak Scaling

As the Blue Waters system was being built and stages came online, it was convenient to have a suite of weak-scaled datasets to track performance. We found the WRF benchmark suite from the Arctic Region Supercomputing Center (ARSC) conveniently suited for this purpose. It consists of five datasets, all based on an overall horizontal domain of 6075x6075, but with a factor of 3x in grid resolution (in each dimension) between consecutive datasets, leading to a factor of 9 in work between each case. The smallest case has a resolution of 81km; the largest, 1km. The smallest case can be run on from 1 to 128 nodes (2048 MPI ranks); any further decomposition is not possible, as the patch size is already down to just a couple of cells. Similarly, the 27km and 9km datasets can scale out to 256 and 512 nodes (8192 MPI ranks), respectively, while the 3km and 1km were run up to 3072 nodes and 6561 nodes before reaching their respective limits of work to effectively decompose. Exploiting the 9x factor of work between these input sets, we ran the problems as detailed in the following table:

Table 3. Weak-scaling details. Parallel efficiency normalized to smallest dataset on single node. Nodes populated with 16 MPI ranks and 2 OMP threads.

km	*Core Count*	*XE6 Nodes*	*Horizontal Decomp. (MPI ranks)*	*Patch size*	*Patch cells*	*Ave. Time Step (secs)*	*Parallel eff. (%)*
1	209952	6561	144x729	43x9	387	0.053005	124
3	23328	729	81x144	25x15	375	0.050091	131
9	2592	81	16x81	43x9	387	0.049692	132
27	288	9	8x18	28x13	364	0.052637	125
81	32	1	2x8	37x10	370	0.065783	100

Increasing the work by 9x while also increasing the number of compute nodes by 9x should result in a flat scaling line, ideally. The graph below shows our results, with the counter-intuitive observation that the smallest node partition actually produces the largest relative step times, despite each MPI patch having essentially the same number of cells to compute over.

As a point of comparison, the smallest (81km) case, when run on a single core with a single OMP thread, produced average step times of 0.92s; with 2 OMP threads this time improved by about 12.5% to 0.82s. Within a node, if this time scaled perfectly, we would expect the 16-MPI rank, 2-tile step time to be 0.051s. While this is very close to our measured times for the larger weak-scaled datasets, we could not reconcile why the measured single-node performance was 28% larger. Because of this anomaly, if we normalize our parallel scaling efficiency against the smallest case (as one would normally do), we actually achieve "super" efficiency on the order of 125%, as noted in the above table. Our runs chose the rectangular decomposition for each size in order to maximize WRF's inner loops' stride counts. We plan to conduct more localized hardware counter analysis of the various epochs comprising an integration step in order to see what trends may correlate to the observed timings. For example, D1 cache utilization/refills and TLB utilization may reveal some key variance.

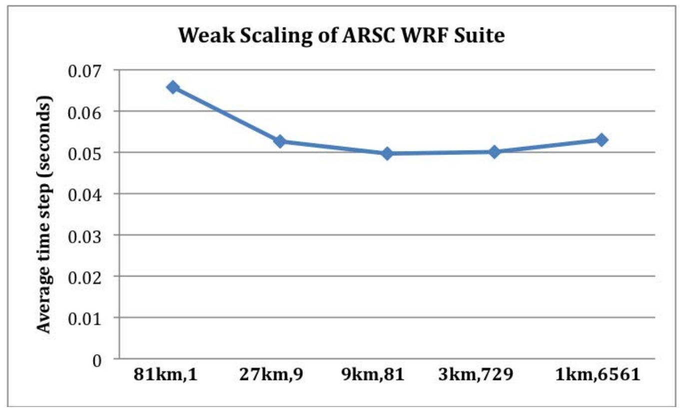

Figure 5. Step times maintain flat profile at large scale even as work and node count are increased incrementally by factor of 9x.

5.3 I/O Considerations

While I/O is not factored into the sustained performance discussed above, reading initial and boundary conditions and writing forecast output are crucial aspects to any scientific simulation. On the Blue Waters system, the Lustre file system was used for all file activity.

Two techniques were used to handle the large I/O requirements for the Sandy simulation -

1. Parallel NetCDF (pnetcdf) from Argonne National Lab was used where practical [8]. The MPICH library from Cray has a tuned MPI-IO implementation that aligns parallel I/O with the Lustre file system. This format is required when post-processing tools are used.

2. WRF has a multi-file option where each subdomain, or MPI rank, reads and writes unique files. This was used for very large restart files and some of the pre-processing steps. The Blue Waters Lustre file system was able to open and read 145,920 restart files in 18 seconds for the 4560 node case.

Table 4 gives an example of effective I/O rates achieved for the hurricane Sandy simulation. Effective I/O rate includes data gather/scatter operations across Blue Waters interconnect as well as Lustre I/O, which also uses the same interconnect, along with data formatting.

Table 4. Effective I/O rates for a 4560 node run. Includes data gather/scatter operations, formatting, and Lustre I/O.

Core Count	*XE6 Nodes*	*Operation*	*Total Gbytes Read or Written*	*Effective I/O Rate (Gbytes/sec)*
145920	4560	Read initial conditions, pNetCDF	1,400	77.6
145920	4560	Read restart multi-files	281	1.4
145920	4560	Write forecast output, pNetCDF	86	2.3

6. FORECAST ANALYSIS AND VALIDATION

The landfall of Hurricane Sandy along the New Jersey shoreline at 2330 UTC 30 October 2012 produced a catastrophic storm surge extending from New Jersey to Rhode Island. This case demonstrates the capability of the NCSA/Cray Blue Waters Petaflop computer to conduct a cloud-resolving WRF-ARW simulation of an intense cyclone over a relatively large domain at a very-high spatial resolution. The results discussed here are from the ARW simulation generated at 500-m horizontal grid spacing and 150 vertical levels spanning the surface to 26 km. The initialization time was chosen to be relatively late in the life-cycle of Hurricane Sandy in order to examine the intensification of the low-level wind field in the 12-h period prior to landfall.

Figure 6 shows a comparison of the maximum radar reflectivity (a surrogate for precipitation) verifying at 1500 UTC 29 October 2012 from the simulations at 3-km and 500-m horizontal resolution. In both simulations, a broad region of heavy precipitation is located on the west and southwest side of Sandy, and is organized in a region where warm moist northeasterly flow intersects a northwesterly surge of cold continental air (not shown). The increase in the temperature gradient in this region over time, termed frontogenesis, helped to focus *convection* along northwest-to-southeast-oriented bands by enhancing the vertical circulation. There is also a region of convection located closer to the center of Sandy on its north and east flank associated with the warm core embryo (not shown).

The 500-m simulation is superior to that at 3-km because it shows the fine-scale linear structure of the convective precipitation bands, consistent with the available observations (not shown). Figure 7 on the next page shows a zoomed-in view of maximum radar reflectivity and 300-m wind speed within the inner-core of Sandy at 1800 UTC 29 October 2012. This zoomed perspective allows for examination of the full detail of the simulation, noting that the resolution of the simulation exceeds the resolution of standard computer monitors by a factor of seven. Here we note the utility of ultra-advanced computational capability to represent the full range of scales spanning the storm-scale circulations down to fine-scale turbulent motions and individual cloud and precipitation systems.

Given recent advances in accessing and displaying large volume geophysical datasets as exemplified by the NCAR VAPOR visualization software, it is now possible to view the full temporal evolution of numerical simulations and predictions of atmospheric and other geophysical systems. Examples of the advanced visualizations of Hurricane Sandy with VAPOR can be found on the NCAR website [14].

The NCSA/Cray Blue Waters computer simulations of Hurricane Sandy demonstrate the capability of present and next-generation computers to address high-impact-weather-related scientific and societal issues, such as i) the cost effective value to implementing Petaflop computational capability for operational, high-impact weather forecasts and warning spanning global to regional scales. In particular, regarding hurricane storm surges and sea-state; ii) the value-added information of very-high spatial resolution probability (ensemble prediction systems) to assist in risk-management decision making; iii) the application of visualization of advanced prediction to facilitate interpretation, preparedness, and public/government/private-sector awareness/response/use of high-impact weather information.

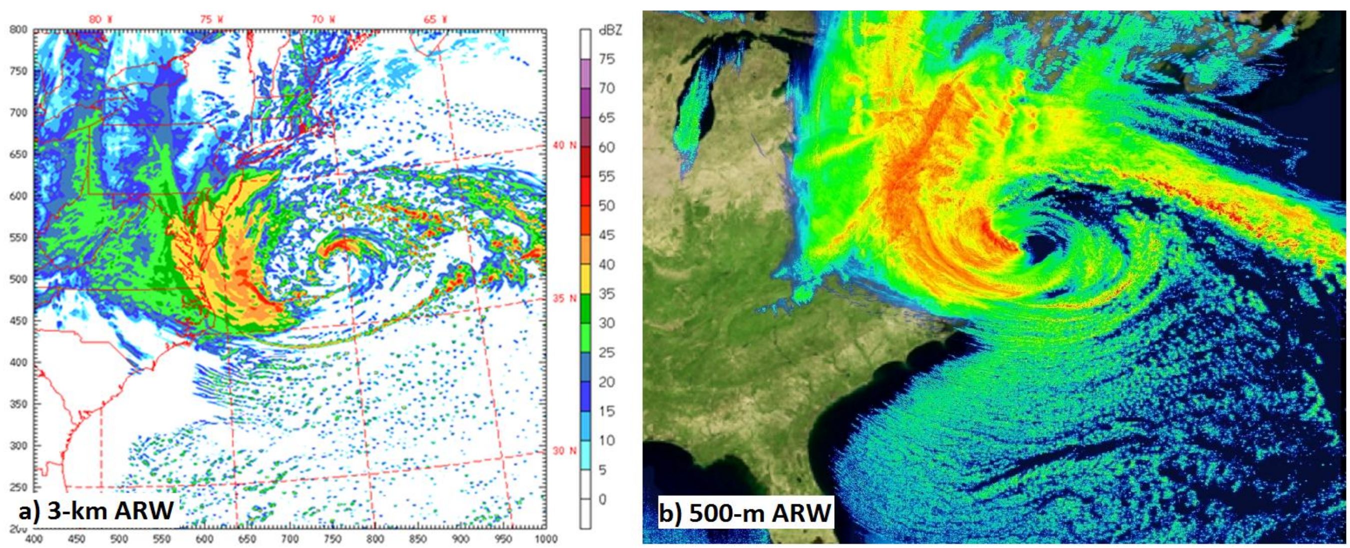

Figure 6. Comparison of (a) 3-km and (b) 500-m horizontal resolution ARW simulations of maximum radar reflectivity (shaded according to the color bar in dBZ) verifying at 1500 UTC 20 October 2012.

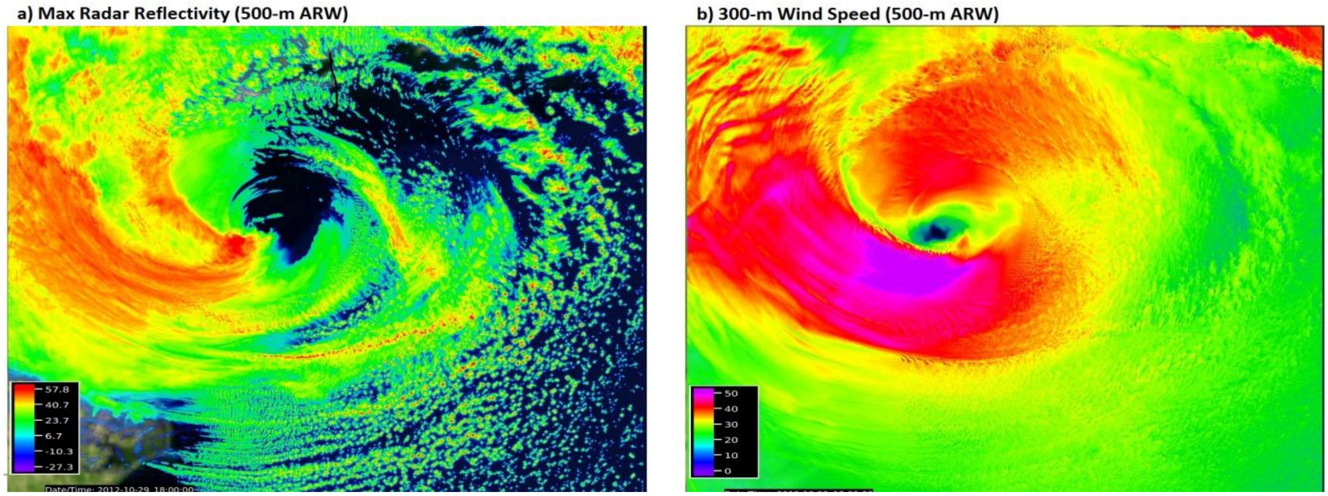

Figure 7. 500-m ARW simulation results showing (a) maximum radar reflectivity (shaded according to the color bar in dBZ) and (b) wind speed at 300 meters above sea level (shaded according to the color bar in m s^{-1}) verifying at 1800 UTC 29 October 2012.

7. CONCLUSIONS

The devastation caused by hurricane Sandy is a testament to our imperfect ability to shield ourselves from natural disasters. However, the advance warning and forecasts for the storm's approach allowed millions of people to seek shelter. This easily validates the investment in numerical weather prediction models. This paper described the validation of a weather forecast model using the WRF code as applied to real data from hurricane Sandy at a resolution and scale unprecedented in numerical weather prediction.

Performance characterizations of the WRF code on the Cray XE6, Blue Waters, at NCSA revealed several opportunities for optimization at the source code, run time, and operating system layers. Most of these discoveries only became salient at the scale of the new Blue Waters machine, and thus represent the next generation of true benchmarks by which future architectures will be judged and procured. These practices were documented for dissemination to the WRF supercomputing community at large.

The model accuracy for predicting such key output fields as rainfall, pressures, wind speeds, and storm track was graphically validated against actual atmospheric measurements from the storm using NCAR's Vapor software suite. The new scientific discoveries made by the simulations of hurricane Sandy support the need for increased resolution in these models, along with architectures such as the Cray Blue Waters system, where such codes map exceptionally well.

8. ACKNOWLEDGEMENTS

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

We thank Eric Johnsen for his expert editing of this text, and are grateful to Don Morton of the Arctic Region Supercomputing Center at the University of Alaska, Fairbanks for making his benchmarks available.

9. REFERENCES

[1] Skamarock, William C., et al, A description of the advanced research WRF V3, NCAR technical note, June, 2008, TN-475+STR

[2] Blake, Eric S; Kimberlain, Todd B; Berg, Robert J; Cangialosi, John P; Beven II, John L; National Hurricane Center (February 12, 2013) (PDF). Hurricane Sandy: October 22 – 29, 2012 (Tropical Cyclone Report). United States National Oceanic and Atmospheric Administration's National Weather Service

[3] Porter, A. R., Ashworth, M., Gadian, A., Burton, R., Connolly, P., and Bane, M., WRF code optimisation for meso-scale process studies (WOMPS) dCSE project report, June, 2010

[4] National Center for Atmospheric Research (NCAR) WRF model web site (http://wrf-model.org/index.php)

[5] Blake, Eric S; Landsea, Christopher W; Gibney, Ethan J; National Climatic Data Center; National Hurricane Center (August 10, 2011). The deadliest, costliest and most intense United States tropical cyclones from 1851 to 2010 (and other frequently requested hurricane facts) (NOAA Technical Memorandum NWS NHC-6). National Oceanic and Atmospheric Administration. p. 47. Retrieved August 10, 2011.

[6] Michalakes, J.; Hacker, J.; Loft, R.; McCracken, M.O.; Snavely, A.; Wright, N.; Spelce, T.; Gorda, B.; Walkup, B. (2007). "WRF Nature Run." *Proceedings SC'07/Gordon Bell prize finalist*; Portland, Oregon.

[7] http://www.mmm.ucar.edu/wrf/WG2/benchv3/, see

[8] http://trac.mcs.anl.gov/projects/parallel-netcdf, see

[9] Spitz, Cory; Henke, Nic; Petesch, Doug; Glenski, Joe, Cray, Inc., Minimizing Lustre ping effects at scale on Cray systems, 2012

[10] Using Balanced Injection, http://docs.cray.com/books/S-0040-A//S-0040-A.pdf

[11] http://weather.arsc.edu/BenchmarkSuite/, see

[12] NCSA Blue Waters system summary, see https://bluewaters.ncsa.illinois.edu/hardware-summary

[13] http://www.mcs.anl.gov/~michalak/rsl/, see

[14] http://www.vis.ucar.edu/~alan/shapiro/sandy500m150lev/T1000New.mov http://www.vis.ucar.edu/~alan/shapiro/sandy500m150lev/winduvNew.mov http://www.vis.ucar.edu/~alan/shapiro/sandy500m150lev/pvoNew.mov