Performance Evaluation and Optimization of Nested High Resolution Weather Simulations

Preeti Malakar[1], Vaibhav Saxena[2], Thomas George[2], Rashmi Mittal[2], Sameer Kumar[3], Abdul Ghani Naim[4], and Saiful Azmi bin Hj Husain[4]

[1] Indian Institute of Science
preeti@csa.iisc.ernet.in
[2] IBM Research - India
{vaibhavsaxena,thomasgeorge,rasmitta}@in.ibm.com
[3] IBM T.J. Watson Research Center
sameerk@us.ibm.com
[4] Universiti Brunei Darussalam, Brunei
{ghani.naim,saiful.husain}@ubd.edu.bn

Abstract. Weather models with high spatial and temporal resolutions are required for accurate prediction of meso-micro scale weather phenomena. Using these models for operational purposes requires forecasts with sufficient lead time, which in turn calls for large computational power. There exists a lot of prior studies on the performance of weather models on single domain simulations with a uniform horizontal resolution. However, there has not been much work on high resolution nested domains that are essential for high-fidelity weather forecasts.

In this paper, we focus on improving and analyzing the performance of nested domain simulations using WRF on IBM Blue Gene/P. We demonstrate a significant reduction (up to 29%) in runtime via a combination of compiler optimizations, mapping of process topology to the physical torus topology, overlapping communication with computation, and parallel communications along torus dimensions. We also conduct a detailed performance evaluation using four nested domain configurations to assess the benefits of the different optimizations as well as the scalability of different WRF operations. Our analysis indicates that the choice of nesting configuration is critical for good performance. To aid WRF practitioners in making this choice, we describe a performance modeling approach that can predict the total simulation time in terms of the domain and processor configurations with a very high accuracy ($< 8\%$) using a regression-based model learned from empirical timing data.

1 Introduction

Operational weather forecasting is critical for planning operations in weather sensitive sectors such as energy, transportation, urban planning, and public safety. Such weather forecasting is performed using fine resolution regional and global atmospheric models that discretize the nonlinear partial differential equations representing evolution of atmospheric flows in time, which entails a huge computational effort. It is also imperative that the forecasts are provided with sufficient lead time (24-48 hrs) in order to allow actions that mitigate the socio-economic impact. Hence, it is critical to have a highly efficient and scalable execution of the weather models on a high-performance computing platform.

Weather and Research Forecasting model (WRF) is a state-of-the-art regional to global-scale numerical weather prediction model that is used by weather agencies all over the world. WRF has been designed to perform well on massively parallel computers. It can be built in serial, parallel (MPI) and mixed-mode (OpenMP and MPI) forms and is available on various HPC machines. Motivated by earlier WRF performance studies on the IBM Blue Gene series [1,8], we explore optimizing WRF on the Blue Gene/P machine. Past studies on WRF are mainly focused on simple single domain benchmarks. These are not representative of real world short-term high-fidelity weather simulations[1,2] that often require nested domain configurations with one or more small high resolution domains (nests) embedded into a coarse resolution parent domain such as those in Figure 1. Fine resolution runs can effectively model weather at meso-micro scale because of higher granularity, but also require a proportionally smaller time step for numerical stability resulting in a quadratic increase in the net computational effort. Domain nesting is essential to achieve good prediction accuracy over small regions of interest (child domains) while avoiding expensive computation across the whole parent domain.

Nested domain simulations differ from single domain simulations in one key aspect. Modeling (i.e., the solve operation) needs to be performed at multiple (parent and child) spatial resolutions and the results need to be communicated and aligned at the points of overlap. The data for the finer resolution child domains are interpolated from the coarser domain by a process called *forcing* in WRF. In a two-way nest integration, the finer grid solution also overwrites the coarser grid solution for the coarse grid points that lie inside the finer grid by a process called *feedback* [11].

There are three key challenges faced by WRF users while performing nested domain simulations. First, nesting entails significant communication between the parent and child domains in the form of forcing and feedback operations, which results in an increased run time and poor scalability that in turn affect the forecast lead time. Second, there does not exist much work on scalability analysis of nested domain simulations that can provide guidance to WRF users on the potential benefits and trade-offs associated with extra computing resources. Lastly, there is risk of over-decomposition on a small-sized domain when the number of processors is large. Therefore, it is critical to choose the nesting configuration to ensure that the nest domain sizes are appropriate for the available processor configuration. Due to the relative opaqueness of the WRF code, most users typically employ a tedious and time-consuming trial and error approach that involves running the code multiple times in order to make this decision.
We make the following contributions:

(a) We significantly reduce the runtime of WRF nested domain simulations (up to 29%) via compiler optimizations on the IBM Blue Gene/P machine, efficient mapping of 2D process topology of WRF on to the 3D torus topology of BG/P to reduce the communication time in WRF and optimization of some

[1] `http://www.emc.ncep.noaa.gov/mmb/mmbpll/nestpage/overlay_hiresw4km.jpg`
[2] `http://www.metoffice.gov.uk/research/modelling-systems/unified-model/weather-forecasting`

critical portions of WRF source code. Unrolling the Z dimension loops and parallelizing the X and Y dimension communications leads to further optimization.

(b) We conduct a detailed performance evaluation of the original as well as the optimized WRF code by performing simulations with varying number of processors over four nested domain configurations with 2-level nesting and varying sibling domains at the innermost level. Our results indicate that forcing and feedback operations do not scale well. Further, performance comparison of similar nesting configurations indicates that optimal nesting choice varies with the number of processors.

(c) We propose a performance modeling approach for estimating the total integration time for any multi-level nested domain WRF simulation, which is based on learning regression models for each of the key WRF operations (solve, forcing, feedback) using empirical timing data. Practitioners can use such a model to determine the best (lowest runtime) nesting configuration among multiple choices for a given number of processors.

2 Related Work

WRF has been extensively studied in the HPC community since weather modeling is a key HPC application. Most of this work can be broadly grouped into three categories: (a) optimization for specific HPC architectures, (b) performance analysis, and (c) performance modeling.

Architecture-Specific Optimization. In the past, WRF code has been optimized for a number of HPC architectures such as BG/L [8], Cray XT4 [9]. For the BG/P architecture, Bhatele et al.[1], present a framework for automatic mapping of WRF onto the BG/P torus topology using the WRF communication graph. On a single domain configuration, they demonstrate that their mapping reduces the average number of hops per process by up to 60%, but increases the total communication time by 40% in certain cases. This approach is yet to be validated on nested domain configurations. Currently, the only existing work that deals with optimization for nested domain configurations is that of Porter et al. [9]. They studied compiler optimizations for improving WRF performance using the PathScale and PGI compilers on Cray XT4/5, which do not apply to BG/P. They also report improvement in WRF performance upon changing the default X-Y processor decomposition, which does not hold true in case of BG/P.

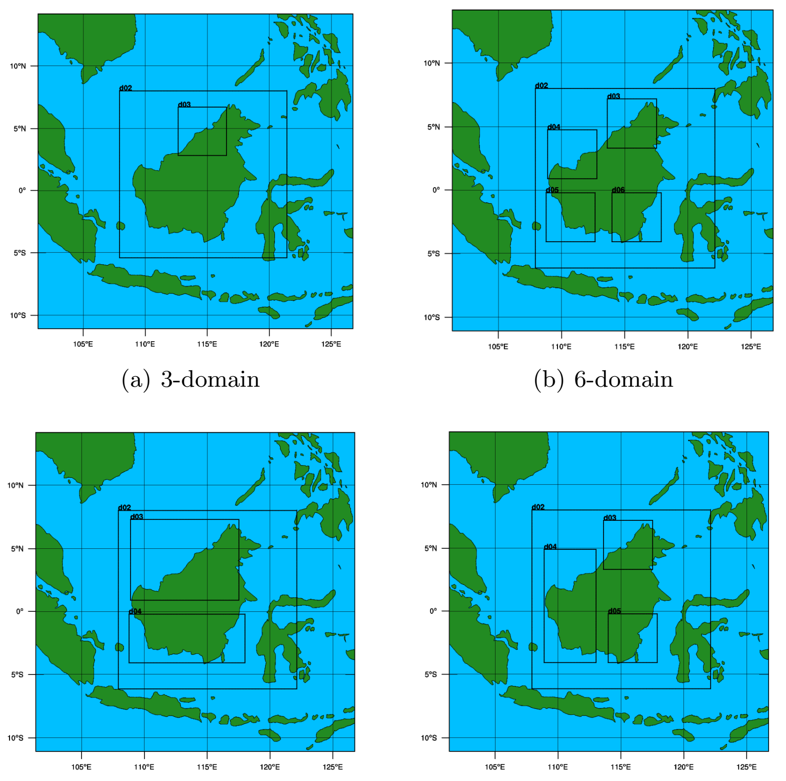
Performance Analysis. Wright et al. [12] examine the scalability of WRF (version 2.1.2) across different architectures using IPM to analyze the performance. They show that for most of the architectures, WRF exhibits a sublinear speedup of both computation and communication times with increasing number of cores and also identify bottlenecks such as MPI_Wait. There exists a number of other performance and profiling studies [10] focusing on other aspects of WRF. However, these studies only focus on a single domain configuration and do not consider forcing/feedback operations that are prominent in nested configurations.

Performance Modeling. Given the criticality of lead time, there is a strong interest in predicting the execution time of WRF runs. Kerbyson et al. [3] describe an analytical performance model with parameters such as the grid size and processor configuration. This model was developed via a careful manual inspection of the dynamic execution behaviour of the WRF application and was subsequently validated using performance measurements on real systems. Unlike [3], our model makes use of regression analysis to directly learn the coefficients of potential influencing factors using empirical timing data. Delgado et al. [2] also describe a regression-based approach for modeling WRF performance on systems with < 256 processors, but their primary focus is on capturing the system related factors such as clock speed, network bandwidth, which they do via a multiplicative effects model. Our modeling focuses on the interaction of domain and processor configurations assuming fixed architecture and simulation parameters allowing us to obtain much better accuracy even up to 1024 processors. Further, [2] and [3] only focus on single domain configurations whereas we learn separate models for solve/forcing/feedback operations, which can be used for predicting performance for even multi-level nested configurations.

3 Experimental Setup

Blue Gene/P Overview. The IBM Blue Gene/P (BG/P) is the second generation in the line of Blue Gene machines after Blue Gene/L. Each BG/P node has four 850 MHz embedded PowerPC 450 cores on a single ASIC and can achieve a peak floating point throughput of 13.6 GF/node. The software stack supports three modes: symmetric multi-processing mode (or SMP mode) with one process and up to four threads, dual mode with two processes, each with up to two threads and quad mode (also known as virtual node mode, or VN mode) with four processes. Systems software provides optimized MPI libraries [4] and an OpenMP runtime via the IBM XL compiler to take advantage of the 4-way SMP node. MPI point-to-point messages are sent on the 3D torus network, while global collective communication operations such as barrier, broadcast and allreduce on MPI_COMM_WORLD are executed on the collective network [5].

Nested Domain Configurations. For our experiments, we simulated a heavy rainfall event that occurred on 20 January 2009, between 00z to 04z UTC in the Borneo island. In order to capture this event, we started a 48-hour forecast run from 19 Jan 00z UTC to 21 Jan 2009 00z UTC and generated a restart file at 20 Jan 00z UTC. This choice of restart file at the end of a 24-hour simulation, ensured that the model is well spun off. Figure 1 shows the nested domain configurations used for our study. Figure 1(a) shows the 3-domain nested configuration with the innermost nest focused on the country of Brunei. Figure 1(b) shows the case which has 4 sibling nests of same size at the innermost level. The innermost nests were chosen such that some of the highly populated regions in the Borneo island are well represented. In order to study the effects of over-decomposition of innermost nests, we also created two sibling domain configurations that are formed by merging a subset of the domains in 1(b). More specifically, Figure 1(c) has 50% more grid points after combining, whereas, Figure 1(d) has 10% more grid points. The same spatial resolution is used for all sibling domains.



(a) 3-domain (b) 6-domain

(c) 4-domain (d) 5-domain

Fig. 1. Nested domain configurations used for the experiments

WRF Runtime Setup. WRF-ARW version 3.2.1, compiled in hybrid mode (dm+sm), was used for all the experiments. In all the simulations, Kain-Fritsch convection parameterization, Thompson microphysics scheme, RRTM long wave radiation, Yonsei University (YSU) boundary layer scheme, and Noah land surface model were used. File I/O was restricted to just the beginning and end of the run, however, for high resolution operational simulations that require forecast output very frequently, it is possible that I/O could become a bottleneck for scaling to large number of processors. This is especially true in production runs that typically involve two or more levels of nesting and a forecast output every ten minutes. We also explored the use of I/O quilting feature in WRF, however, we excluded quilting from the current study since parallel I/O gave the best performance. For this study we primarily use the total integration time since the I/O time is still a small fraction of the total simulation time when parallel I/O is used with moderate number of processors (up to 1024).

4 Optimizations

We explored several optimizations for the WRF application on BG/P that include compiler options, code modifications and tuning the MPI libraries.

4.1 Compiler Options

We modified the default WRF configure from UCAR to include parallel NetCDF for parallel I/O on BG/P. This serves as the *base* WRF configure to make *base* WRF run for our experiments. We enhanced the *base* WRF configure with a few new options. As the WRF moisture physics routines have several calls to exponents, square-root, trigonometric functions and divisions, we enabled the mathematical acceleration libraries mass and the vector variant library massv. We also modified WRF source to call the massv library for the vspow call. In addition, we also used the -qhot=vector compiler option that converts loops with exponents, square-root, division and trigonometric functions to massv library calls. The mass and massv library calls are optimized via SIMD instructions that have higher throughput. Most of WRF is compiled with the -O2 option, with select performance-critical routines compiled with -O3. To further optimize compiler performance, we added the -qmaxmem=128000 to enable the compiler to aggressively optimize WRF source. We also enabled OpenMP via the IBM XL compiler option -qsmp=omp. The WRF source can tile the X or the Y loops to enable the tile computation to be executed on different threads.

4.2 Communication Libraries

Communication overhead is a significant fraction of the WRF timestep. We explored mapfiles to efficiently map the 2D decomposition on to the 3D torus. Our mapfiles map the processors to diagonals on the 3D torus planes when the 3D torus dimensions cannot be folded to the WRF 2D stencil communication dimensions. In addition, we explored increasing the MPI eager limit and enabling the FIFO mode RZVANY in the DCMF library [4] that drives MPI communication on BG/P. This mode improves performance of messages to diagonal neighboring nodes by allocating more network resources to those messages. In addition, to minimize synchronization between nodes, we explored the Async Rectangle Broadcast that implements broadcast without forcing synchronization between the nodes. By default, the collective network is used that forces all nodes to synchronize before the broadcast data is transmitted on the network. We also replaced a call to MPI_Allgather in the forcing and feedback computation with an MPI_Alltoall call as MPI_Alltoall is efficient on BG/P.

4.3 Source Code Optimizations

Loop Unrolling. The XL compiler on BG/P typically only unrolls the innermost loops. WRF is a strong scaling application where the application domain is decomposed along the X and Y dimensions to MPI ranks. By default, the WRF compute loops execute in an XZY order that results in short X loops on

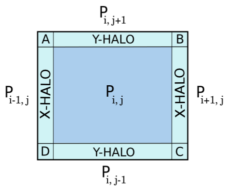


Fig. 2. Halo regions in X-Y dimensions

Table 1. Steps in WRF halo exchange

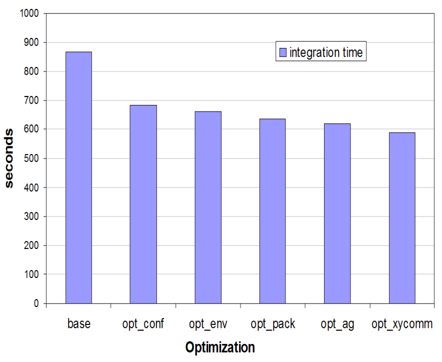
Pack data into buffer to send to the Y-neighbours.
Receive message from and send message to the Y-neighbours.
Wait for completion of communication with the Y-neighbours.
Unpack data received from Y-neighbours.
Pack data into buffer to send to the X-neighbours.
Receive message from and send message to the X-neighbours.
Wait for completion of communication with the X-neighbours.
Unpack data received from X-neighbours.

a large number of nodes. Hence, we explored unrolling the Z dimension loops in the pack unpack functions that serialize application buffers into MPI messages functions and WRF dynamics computation.

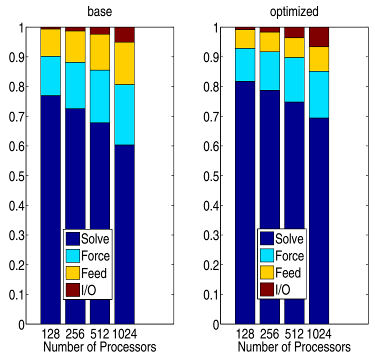
Parallel X and Y communication. The WRF solver sweeps over a 2D-XY-spatial grid performing nearest neighbour computations called *stencils*[7]. Due to the overlapping spatial decomposition, each process $P_{i,j}$ communicates its boundary regions, called *halo* regions, with its two X-neighbours $P_{i-1,j}$ and $P_{i+1,j}$ and two Y-neighbours $P_{i,j-1}$ and $P_{i,j+1}$ as shown in Figure 2. Each integration time-step in WRF involves a large number of halo exchanges, which are fairly expensive and comprise of the steps shown in Table 1. Using IBM's HPCT profiling tools, we found that the packing/unpacking of the halo regions is a computation-intensive step. Further, most of the time is spent in MPI_Wait since every process waits for the completion of the communications without doing any computation. To improve the performance, we modified the halo exchange sequence of steps, specifically delaying the MPI_Wait calls and performing the X and Y communications in parallel, which is favoured by the BG/P torus topology. In the new sequence, the Y-communications are overlapped with the packing of data for the X-neighbours and the X-communications are overlapped with the unpacking of data from the Y-neighbours. This affects only the corners of the halo regions (e.g. regions A,B,C,D in Figure 2) and does not make a substantial difference since these only comprise a tiny part of the subdomain. Note parallelizing X and Y communication does not affect bitwise reproducability in the application as we still maintain a deterministic order.

5 Performance Analysis

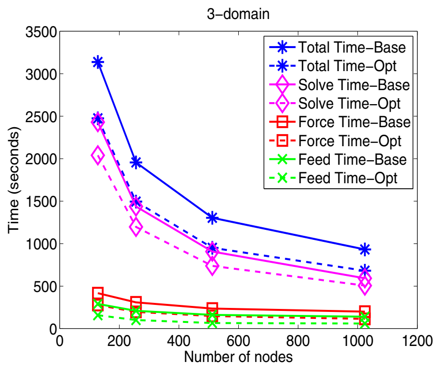
In this section, we present the results of our performance evaluation on multiple nested domain configurations. Specifically, we discuss the various BG/P execution modes, the benefits of various WRF optimizations, the scalability of the original and optimized WRF code, and also the importance of choosing a nesting configuration to match the processor configuration. We used 128 to 1024 BG/P nodes for performance evaluation to keep the simulation runtimes to reasonable limits as well as to prevent over-decomposition of the domains beyond 1024 nodes.



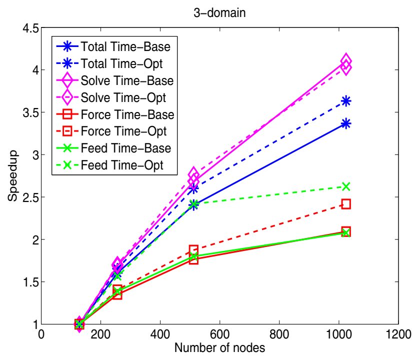
(a) Performance variation with different optimizations on 1 rack



(b) Variation of fractional times of solve, forcing, feedback, and I/O operations vs. number of processors



(c) Computation time of WRF operations



(d) Speedup of WRF operations

Fig. 3. WRF performance on nested domain configurations

BG/P Execution Modes. We compared the base WRF performance with different BG/P execution modes (SMP, DUAL and VN). The performance was best with SMP mode with 4 OpenMP threads compared to other modes. On 1024 nodes (1 rack) of BG/P, integration time for DUAL and VN modes increased by 4% and 48% over SMP mode respectively. The total simulation runtime with DUAL and VN modes increased by 12% and 65% over SMP mode respectively. The increase in total runtime was partly due to higher I/O overheads in DUAL and VN modes because of the increase in the number of MPI ranks. The I/O times increased 1.5x and 2.8x in DUAL and VN modes respectively.

WRF Optimizations. Figure 3(a) shows the incremental performance benefits of the various optimizations described in Section 4 on a single rack of BG/P. Each column bar in the figure also incorporates the optimizations indicated by the previous column bars. The performance of the base WRF configuration is presented in column bar *base*. The *opt_conf* column bar indicates the performance after incorporating compiler options mentioned in Section 4.1, that results in a 21% improvement over the *base* configuration. The column bar *opt_env* shows the benefit of environment variables (Section 4.2) that enable an optimized processor mapping, set the eager to rendezvous cutoff and the DCMF FIFO mode. We see a total improvement of 3% over *opt_conf* (23.7% over *base*). Loop unrolling of the pack and unpack routines (*opt_pack*) described in Section 4.3 results in an improvement of 3.6% over *opt_env* (26.5% over *base*). Replacing *MPI_Allgather* with *MPI_Alltoall* (*opt_ag*) provided 2.7% further improvement over *opt_pack* (28.4% over *base*). Finally, optimization for the parallel X and Y halo communication (*opt_xycomm*) results in a performance improvement of 5% over *opt_ag*. This corresponds to an overall improvement of 32% to the integration time and an improvement of 28.9% in the total simulation runtime over the *base* run.

WRF Scaling. Figure 3(b) shows the variation in fractional time for solve, forcing and feedback operations across number of processors for base and optimized codes. Observe the I/O and forcing components increase with the number of nodes suggesting that they are performance bottlenecks. Figures 3(c) and 3(d) show the scaling of solve, forcing and feedback operations in terms of actual timings and speedup respectively. The total time is the sum of the timings for these 3 operations. Though sub-linear, the observed speedups of solve operation in both the original and optimized WRF code (3.5x - 5x going from 128 to 1024 processors) exhibit the same general behavior as the theoretical model presented by Kerbyson et al. [3]. Relative to solve, the forcing and feedback operations exhibit much poorer scaling, attaining speedups of 2x - 2.7x for the same increase in the number of processors. From these figures, we also observe that with our optimizations, the solve time improves and the forcing and feedback components have lower overheads and better scaling.

Influence of Nesting Configuration. We also compare the performance of the three nested domain configurations in Figures 1(b), 1(c), and 1(d), which are equivalent in terms of practical utility and differ only in the sibling domains at the innermost level. Figure 4(a) shows the speedup (relative to 128 processors) as the number of processors increase. We observe that increasing the number of domains results in poorer scalability. One reason for this behavior is that more domains entail additional sequential forcing and feedback operations, which do not scale well. In addition, the innermost nests with small domain sizes get over-decomposed for higher number of processors resulting in lower computation to communication cost ratio. The superior scalability of the 4-domain and 5-domain cases suggest that as we increase the number of processors, the gap in the total integration time due to the additional number of innermost grid points in comparison to that of the 6-domain case should progressively decrease. Figure 4(b) shows the total integration time for the three nested domain configurations with varying number of processors. Even though the 6-domain case has the least number of grid points, for 128 processors, the 5-domain case provides the best performance (8984s in comparison to 10701s and 9108s for 4-domain and 6-domain cases). As the number of processors increases to 1024, the 4-domain configuration exhibits superior scaling and gives the best performance (2411s in comparison to 2459s and 2794s for 5-domain and 6-domain cases). Therefore, it is sometimes beneficial to consider a smaller number of consolidated domains instead of a large number of small focused domains.

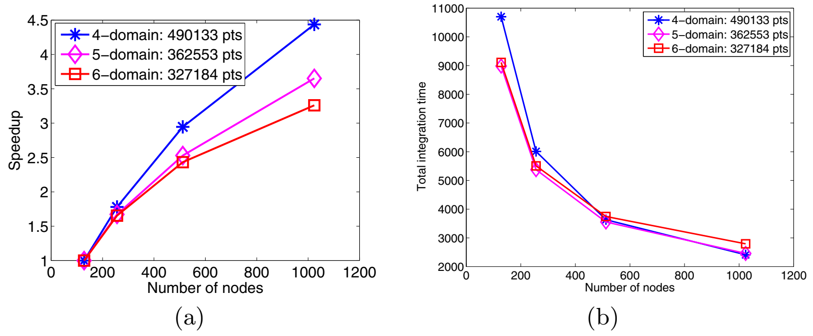


Fig. 4. Performance over the three sibling configurations across varying number of processors

6 Empirical Modeling of Computation Time

Observations from Section 5 highlight the importance of carefully choosing the nesting configuration that can extract the best performance for a given processor configuration. To aid in this effort, we use the empirical timing data from our evaluation to learn a statistical model of the computation time for the three key operations in WRF (solve, forcing and feedback). We then describe how these individual models can be combined to estimate the total modeling time for complex nesting scenarios that involve multiple levels of nesting.

Regression Model. For each operation and domain/processor configuration, the runtime for each iteration is computed as the maximum time across the different nodes since they are operating in parallel. Since the different iterations are executed sequentially and the number of iterations vary due to nesting or simulation configuration, the mean runtime across the different iterations is a good target. We choose the median value across the different iterations as the target variable to be modeled instead of the mean to disregard outlier cases, but the results are comparable in either case. Each domain/processor configuration is represented as a vector of features (column headings in Table 2) based on sizes of the domain(s) involved and the processor grid. Let the domain grid size be denoted by $n_x \times n_y$ and processor grid be denoted by $p_x \times p_y$. In case of forcing and feedback, we also consider the parent grid size denoted by $n_x^p \times n_y^p$. Using the median iteration time as the target response and the feature vector as independent variables, we learn a least squares linear regression model [6]. Table 2 shows the model coefficients. For instance, the computation time for solve operation is given by T_{solve} =7.69e-4$\frac{n_x n_y}{p_x p_y}$+3.37e-3$\frac{n_x}{p_x}$+2.35e-3$\frac{n_y}{p_y}$+4.047e-2. A similar model was computed for the optimized WRF code as well and for each operation, the coefficients of all the features were found to be positive, but lower than that of the base WRF timing model. Note that the model coefficients

Table 2. Coefficients of the model for base WRF version

Op	$\frac{n_x n_y}{p_x p_y}$	$\frac{n_x^p n_y^p}{p_x p_y}$	$\frac{n_x}{p_x}$	$\frac{n_y}{p_y}$	**constant**
Solve	7.693e-4	NA	3.3676e-3	2.3472e-3	4.04716e-2
Forcing	6.2493e-5	2.0758e-4	2.78171e-3	1.7226e-3	1.4445e-1
Feedback	9.8716e-5	5.2550e-5	2.4233e-3	1.2726e-3	8.5323e-2

Table 3. Actual (s), predicted (s) and relative error (%) of integration time for the base WRF model

Processors	128			256			512			1024		
Domain	Act	Pred	Err(%)	Act	Pred	Err(%)	Act	Pred	Err(%)	Act	Pred	Err(%)
3-domain	2976	2877	3.3	1809	1753	3.1	1200	1143	4.7	867	815	5.9
6-domain	9109	9064	0.5	5508	5524	0.29	3745	3603	3.8	2794	2574	7.9
4-domain	10701	10225	4.5	6011	5909	1.7	3632	3455	4.9	2412	2276	5.6
5-domain	8984	8989	0.06	5374	5266	2.01	3553	3367	5.2	2460	2283	7.2

are specific to choice of BG/P and the other fixed parameters of the simulation (e.g., mp_physics = 8), but the methodology can be adapted as needed.

Computation Time for Multilevel Nested Domains. Let $\mathcal{D} = \{D_0, \cdots, D_n\}$ be a set of nested domains with D_0 being the root domain and the domain hierarchy specified by the $p(\cdot)$ function. Let $T_{solve}(D)$ denote the computation time for a solve iteration on domain D and $T_{force}(D, D_p)$ and $T_{feedback}(D, D_p)$ denote the computation time for forcing and feedback operations between domain D and its parent domain D_p. Let $T^{SH}_{solve}(D)$ denote the solve time for the entire sub-hierarchy under the domain D. Given the constraints of the current WRF code, the forcing, feedback, and solve operations associated with the child domains have to be done sequentially. For two-way nesting, one can obtain the following recursive equation,

$$T^{SH}_{solve}(D) = T_{solve}(D) + \sum_{D_c|p(D_c)=D} (T_{force}(D_c, D) + rT_{solve}(D_c) + T_{feedback}(D_c, D))$$

where r is the parent-child time-step ratio (=3). The total integration time is then given by $T^{SH}_{solve}(D_0)$ multiplied by the number of timesteps.

Using the above recursion and the prediction models for $T_{solve}(\cdot)$, $T_{force}(\cdot, \cdot)$ and $T_{feedback}(\cdot, \cdot)$, one can obtain a reasonable estimate of the computation time for a specified nested domain configuration and processor configuration without having to actually run the code. Table 3 shows the actual integration timings as well as the timings predicted by the model and the absolute relative error for different domain and feasible processor configurations for the base WRF code. We observe that the prediction error is fairly low $(0.06 - 7.9\%)$. Note that the 5-domain case is in fact a new configuration which was not used for training the statistical model, but the predicted timings are still fairly accurate. This timing prediction model can be very useful for practitioners since they can choose the domains and nesting configurations based on the estimated run times. We plan to do a more exhaustive evaluation over multiple nested domain cases in future.

7 Conclusions and Future Work

We performed a detailed study of nested domain weather simulations that are critical for operational weather forecasting. We described several optimizations to the base WRF code that reduces the total runtime by 29%. We also conducted a performance evaluation using four test configurations and demonstrated that high resolution nested weather simulation presents a number of challenging opportunities in terms of scaling to large number of processors especially in the case of multiple small-sized sibling domains. This is partly due to the design of the WRF code wherein, multiple nests at the same level are handled by all the processors in a sequential manner resulting in over-decomposition. The current design of the WRF code makes it both critical and challenging for practitioners to choose a good nesting configuration. Accounting for the constraints of the current WRF code, we also presented a regression-based model for predicting integration time for multi-level nested weather simulations that can be used by WRF users to determine the domain configurations best suited for a given processor grid. Future directions include improving the performance model by incorporating additional features based on the network topology, modifying WRF code to allow subsets of processors working in parallel over sibling domains and designing algorithms that can effectively balance the load under such circumstances. We also plan to explore MPI non contiguous datatypes to optimize the pack/unpack operations in WRF. In addition, weak scaling studies could potentially assist in further understanding of the effects of the proposed optimizations. Optimizing file I/O performance in WRF is another potential future work.

Acknowledgements. We would like to thank Yogish Sabharwal, Bob Walkup, Dong Chen, Sathish S. Vadhiyar, Vijay Natarajan and Lloyd A. Treinish for their help, technical support and valuable suggestions. The work presented in this paper was funded in part by the US Government contract No. B554331.

References

1. Bhatele, A., Gupta, G.R., Kale, L.V., Chung, I.H.: Automated Mapping of Regular Communication Graphs on Mesh Interconnects. In: HiPC (2010)
2. Delgado, J., et al.: Performance Prediction of Weather Forecasting Software on Multicore Systems. In: IPDPS, Workshops and PhD Forum (2010)
3. Kerbyson, D.J., Barker, K.J., Davis, K.: Analysis of the Weather Research and Forecasting (WRF) Model on Large-Scale Systems. In: PARCO, pp. 89–98 (2007)
4. Kumar, S., et al.: The Deep Computing Messaging Framework: Generalized Scalable Message passing on the Blue Gene/P Supercomputer. In: ICS 2008 (2008)
5. Kumar, S., et al.: Architecture of the Component Collective Messaging Interface. IJHPCA 24(1), 16–33 (2010)
6. Kutner, M.H., Nachtsheim, C.J., Neter, J.: Applied Linear Regression Models, Fourth International edn. McGraw-Hill (September 2004)
7. Michalakes, J.: RSL: A Parallel Runtime System Library For Regional Atmospheric Models With Nesting. Tech. Rep. ANL/MCS-TM-197, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne (1997)
8. Michalakes, J., et al.: WRF Nature Run. In: SC, p. 59 (2007)

9. Porter, A.R., et al.: WRF code Optimisation for Mesoscale Process Studies (WOMPS) dCSE Project Report (June 2010)
10. Shainer, G., et al.: Weather Research and Forecast (WRF) Model Performance and Profiling Analysis on Advanced Multi-core HPC Clusters. In: 10th LCI ICHPCC (2009)
11. Skamarock, W.C., et al.: A Description of the Advanced Research WRF version 3. NCAR Technical Note TN-475 (2008)
12. Wright, N.J., Pfeiffer, W., Snavely, A.: Characterizing Parallel Scaling of Scientific Applications using IPM. In: 10th LCI International Conference on High-Performance Clustered Computing (2009)