Evaluation and Enhancement of Weather Application Performance on Blue Gene/Q

Gurbinder Singh Gill, Vaibhav Saxena, Rashmi Mittal, Thomas George and Yogish Sabharwal
IBM Research - India
Vasant Kunj, New Delhi - 110070, India
{gurbgill, vaibhavsaxena, rashmitt, thomasgeorge, ysabharwal}@in.ibm.com

Lalit Dagar
Universiti Brunei Darussalam
Brunei
lalit.kumar@ubd.edu.bn

***Abstract*—Numerical weather prediction (NWP) models use mathematical models of the atmosphere to predict the weather. Ongoing efforts in the weather and climate community continuously try to improve the fidelity of weather models by employing higher order numerical methods suitable for solving model equations at high resolutions. In realistic weather forecasting scenario, simulating and tracking multiple regions of interest (nests) at fine resolutions is important in understanding the interplay between multiple weather phenomena and for comprehensive predictions. These multiple regions of interest in a simulation can be significantly different in resolution and other modeling parameters. Currently, the weather simulations involving these nested regions process them one after the other in a sequential fashion. There exist a lot of prior work in performance evaluation and optimization of weather models, however most of this work is either limited to simulations involving a single domain or multiple nests with same resolution and model parameters such as model physics options.**

In this paper, we evaluate and enhance the performance of a popular WRF model on IBM Blue Gene/Q system. We consider nested simulations with multiple child domains and study how parameters such as physics options and simulation time steps for child domains affect the computational requirements. We also analyze how such configurations can benefit from parallel execution of the children domains rather than processing them sequentially. We demonstrate that it is important to allocate processors to nested child domains in proportion to the work load associated with them when executing them in parallel. This ensures that the time spent in the different nested simulations is nearly equal, and the nested domains reach the synchronization step with the parent simulation together. Our experimental evaluation using a simple heuristic for allocation of nodes shows that the performance of WRF simulations can be improved by up to 14% by parallel execution of sibling domains with different configuration of domain sizes, temporal resolutions and physics options.

I. Introduction

Weather forecasting is essential for accurate and timely prediction of catastrophic events such as hurricanes, heat waves, and thunderstorms. Timely prediction enables policy makers to take quick preventive actions. Such predictions require high fidelity weather simulations and simultaneous online visualization to comprehend the simulation in real-time. Numerical weather prediction uses mathematical models of the atmosphere and oceans to predict the weather based on current weather conditions. This involves solving non-linear partial differential equations numerically. Ongoing efforts in the climate science and weather community continuously improve the fidelity of weather models by employing higher order numerical methods suitable for solving model equations at high resolution discrete elements.

Simulating and tracking multiple regions of interest at fine resolutions is important to understand the interplay between multiple weather phenomena and for comprehensive predictions. For example, Figure 1 illustrates the phenomena of two storms occurring simultaneously in the Pacific Ocean. Here, it is necessary to track both depressions to forecast the possibility of a typhoon or heavy rainfall. In such scenarios, multiple simulations need to be spawned within the main parent simulation to track these phenomena. These high resolution simulations are generally executed as subtasks within the coarser-level parent simulation.

Weather simulations involving multiple regions of interest with subtask simulations are called nested domain simulations. The domain of the parent simulation is referred to as the parent domain and the domain of each subtask simulation is referred to as the child domain. Figure 2 shows the configuration of a nested domain simulation involving two child domains nested within a parent domain. The nested domains are typically simulated at a higher spatio-temporal resolution, i.e., grid points are closer together and more integration steps (simulations) are performed for the nested domains in comparison to the parent domain. This is achieved as follows. For each nested domain, there is a parameter, r, that defines the ratio of the temporal resolution of the nested simulation to the parent simulation. Thus, the nested domain is solved r number of times for each parent integration step. At the beginning of each nested simulation, data for each finer resolution child domain is interpolated from the overlapping parent domain.

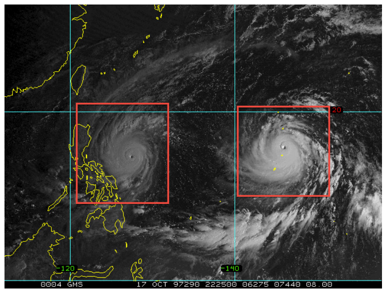

Fig. 1. Visualization of multiple storms in October 1997 on Pacific Ocean

At the end of the integration steps of the child domains, data from the child domains is communicated back to the parent domain. The nested simulations demand a large amount of computation due to their fine resolution. Hence, optimizing the executions of nested simulations can lead to a significant overall performance gain.

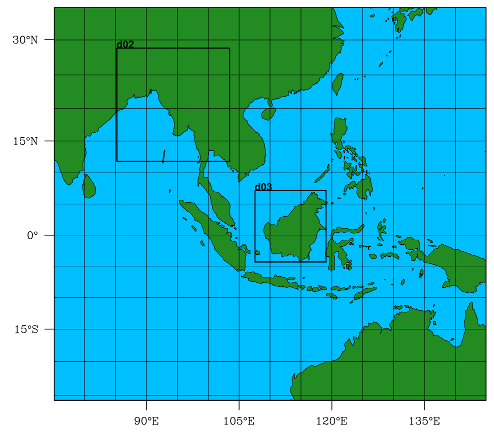

Fig. 2. Sample domain with two sibling nests at 1.5 km resolution.

Existing weather applications employ a default strategy of executing the nested simulations corresponding to a single parent domain sequentially one after the other using the full set of processors. However, these applications typically exhibit sub-linear scalability resulting in diminishing returns as the problem size becomes smaller relative to the number of available cores. For example, we observed the popular Weather Research and Forecasting model (WRF) [1], [2] is scalable up to large number of cores [3] when executed without a subdomain, but exhibits poor scalability when executed with subdomains. Figure 3 shows the scalability of WRF up to 512 nodes of IBM Blue Gene/Q with each node containing 16 processor cores. The simulation corresponds to the region around Brunei Darussalam and includes the parent domain containing 860×780 grid points as well as a child subdomain of size 625×625 grid points (specifically parent domain d01 and nest d02 in Figure 2). Note that the performance of WRF involving a subdomain saturates at about 4096 processor cores (256 BGQ nodes). Hence, in a WRF simulation with two subdomains executed on a total of 8192 processor cores, the performance of a subdomain executed on 4096 cores will not improve when using all the 8192 cores. Thus, partitioning the 8192 cores equally among the subdomains for simultaneous execution will give better performance than serial execution on all the 8192 cores.

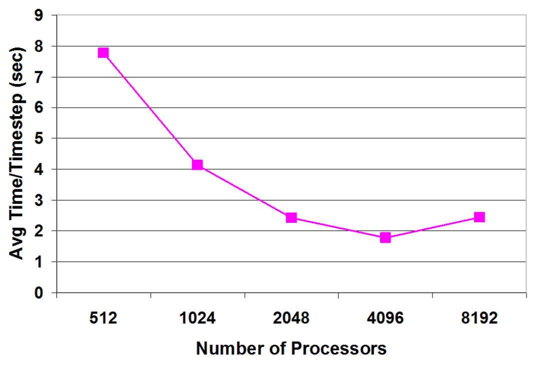

Fig. 3. Execution time of a weather simulation with two domains on Blue Gene/Q

In recent work, Malakar et al. [4] described how nested domain simulations can be optimized by executing the sibling domains in parallel - they carefully partitioned the node space into smaller node sets and after each time step of the parent domain processing that runs on the entire set of nodes, the processing of each child domain is performed on one of the partitioned node sets. As the scalability of weather modeling software is limited for practical configurations[1], the sibling domains benefit by parallel execution on smaller set of nodes in comparison to executing sequentially one after the other on the entire set of nodes. Malakar et al. studied how the performance varies with different subdomain sizes, and how to efficiently partition the entire processor space into multiple disjoint rectangular processor grids that can be assigned to the different nested simulations such that the parallel execution time is minimized. However, this work was limited to studying the impact of different domain sizes on the computational requirements of the domains and optimizing these computations by parallelizing the execution of the subdomains assuming homogeneous modeling configurations for all the subdomains. In practice, the appropriate weather simulation for a region is influenced by many factors. For instance, the geographical location of the domain, terrain, vegetation and soil-types along with the purpose for which the land is being used and, dominant atmospheric characteristics, the geophysical conditions of the neighboring areas, and many other parameters. For instance, in the tropical precipitation dominant regions it may be more desirable to use the physics option for convection based on spectral formulation [5] as it better simulates the mesoscale convective clusters that are more prevalent in these region. Thus, the suitable configurations for different subdomains can vary considerably even if they are in spatial proximity.

Similarly, for a region where higher prediction accuracy is required (e.g., urban areas with high population density), it is desirable to have a higher spatio-temporal resolution whereas in other areas, a lower prediction accuracy might suffice. Weather models such as WRF are very flexible in allowing different configurations to be set up for different child domains. For instance, different sibling domains within a parent domain can have different grid sizes and resolution, different time step ratios with the parent domain, and they can run with different physics options. While highly desirable, these features also lead to load imbalance amongst the child domains as different configurations result in different computational requirements. In this paper, we consider nested simulations with multiple child domains and study how parameters such as physics options and time step ratios for child domains affect the computational requirements and how such configurations can benefit from parallel execution of the children domains. A critical requirement is that the processors are allocated in proportion to the work load associated with each nested simulation. This ensures that the time spent in the different nested simulations is nearly equal, and the nested domains reach the synchronization step with the parent simulation together.

[1]Weather modeling software is known to show very good scalability on artificially constructed benchmarks.

We make the following main contributions:

- We present the first ever performance evaluation of WRF for nested domain configurations on the Blue Gene/Q supercomputer. Specifically, we target the case of multiple child domains at the same level and study scalability on BG/Q for multiple MPI tasks and thread combinations.
- We extend the partitioning scheme by Malakar et. al. [4] to handle child domains with varying physics options and timesteps. Empirical evaluation of our approach demonstrates a reduction in run times by up to 14 % when the child domains at the same level are executed in parallel.

The rest of the paper is organized as follows: Section II described related work on nested WRF simulations. Section III provides background on the WRF model and various physics parameters. Section IV discusses optimizations related to partitioning the processors into subsets to which the nested sibling domains are assigned. Section V describes the experimental setup while Section VI presents the results of the performance evaluation. Section VII provides the concluding remarks.

II. Related Work

Weather/Climate modeling being a key application in the HPC community, their performance has been studied widely on different HPC architecture. These studies include, performance analysis of these models on specific architecture and the net gain in performance with architecture specific optimizations. As the lead time is of most interest in all the forecasting models, there also exists some studies on the performance modeling to predict the execution time. A lot of prior work in this area is 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, a state-of-the-art weather model on IBM Blue Gene/Q. To our knowledge this is the first work on WRF performance analysis for nested domains on BG/Q. Our current work is primarily related to two search areas: (1) performance analysis, and (2) static load balancing for parallel applications.

A. *Performance Analysis*

Wright et al. [6] 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 [7] focusing on other aspects of WRF. These studies only focus on a single domain configuration. WRF has also been optimized for a number of HPC architectures such as BG/L [3], Cray XT4 [8]. For the BG/P architecture, Bhatele et al. [9], present a framework for automatic mapping of WRF onto the BG/P torus topology using the WRF communication graph. On the nested domain, the only existing work that deals with optimization for nested domain configurations is that of Porter et al. [8] and Malakar et al. [10] . Porter et al. studied compiler optimizations for improving WRF performance using the PathScale and PGI compilers on Cray XT4/5. A detailed analysis on BGP has been presented in Malakar et al. They demonstrated a significant reduction (up to 29%) in runtime on the BGP 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. They also described a performance modeling approach that can predict the total simulation time in terms of the domain and processor configurations with a very high accuracy. Kerbyson et al. [11] and Delgado et al. [12] has also describe performance model with different parameters but they have focused on single domain simulations which doest not take into account the effects of force and feedback in the nested configurations. On the other hand Malakar et. al learn separate models for solve/forcing/feedback operations, which can be used for predicting performance for even multi-level nested configurations.

B. *Static Load Balancing for Parallel Applications.*

Most of the work on the parallelization of weather model is based on domain decomposition where the domain of interest is divided into smaller subdomains that are assigned to individual processors such that the load is balanced across the processors while minimizing the communication costs. There currently exist a number of approaches such as recursive bisection [13] and graph partitioning [14] that cater to both regular and irregular domains and yield very good performance. In recent years, there has also been work [15] addressing scenarios involving processors with heterogeneous computational capacity and communication bandwidth that requires partitioning the domain of interest into multiple subdomains in proportion to the computational capacity of the processors while taking into account other constraints. In a recent past, Malakar et al. [4] addressed the WRF specific constraint on the nested domains. In WRF model single nested domain is assigned to a rectangular processor grid to achieve optimal performance. Instead of decomposing the multiple nested domains into even smaller ones to be assigned to individual processors, they partitioned the processor space into multiple disjoint rectangular grids that are assigned to the individual nested domains so that the computational capacity is proportional to the domain workload.

In current work, the scalability of the WRF (version 3.3.1) is examined on on BGQ and the its performance is evaluated in various different load imbalance conditions using the the approach suggested by Malakar et al.

III. WRF Model

Weather Research and Forecasting (WRF) model is a numerical weather prediction model useful for both research and forecasting purposes. It serves a wide range of meteorological applications at different scales ranging from a few meters to hundreds of kilometers. The model provides a lot of flexibility in its configuration for producing accurate forecasts at desired spatial and temporal resolution, for different geographies, different weather and climate systems etc. The WRF configuration consists of setting up various parameters for model physics and dynamics that are appropriate for the target meteorological application and domain. The physics options correspond to the parameterization of various sub-grid scale processes like radiation budget, cloud micro-physics, convection, planetary boundary layer, land surface etc. In the WRF model there are multiple options available for the parameterization of every sub-grid scale process. This multi-physics option provided in WRF has proven to be very useful in operational forecasting and has led to the widespread use of WRF – almost every operational short term forecasting center uses a WRF model. Another very useful feature of WRF is that one can configure the model with multiple nested domains embedded within the each other – commonly termed as "nesting". Nesting leads to a huge reduction in the computation cost as the target area can be solved at a finer resolution whereas the parent domain that contains the large-scale atmospheric information required by atmospheric modeling can be resolved at a relatively coarser resolution. Operational forecasting for large areas can sometimes become very complicated as the model may show a very good skill in one part of the target domain and a bad skill in another part of the domain. This generally happens because the same physics configuration may not be suitable for the whole target domain. Thus, the use of multi-physics option along with nesting becomes very useful as one can divide the areas of the target domain in to different nests and use appropriate physics in each sibling domain. This approach is adopted in many operational centers, for example, in the operational forecasting of India, the Himalayan region, Western desert region, coastal southern region and eastern region are divided in to different sibling nests and different physics options are used in each one of them that are suitable for cold mountainous, hot desert plains, convectively active costal regions respectively. An important aspect of nested domain simulations is that the 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*[2].

Amongst the various sub-grid scale processes the micro-physics and long-wave radiation parameterization are the most time consuming ones. The micro-physics includes explicitly resolved water vapor, cloud, and precipitation processes. Four-dimensional arrays with three spatial indices and one species index are used to carry such scalars. The advection of all the species required by the micro-physics option is carried out. In general advection or transport of species is computationally expensive as higher order numerical schemes are used for this. The three options, namely Kessler, Thompson and Morrison double moment scheme, representative of different computational complexity are used in this study. Kessler is the simplest of them with only three species: water vapour, cloud water, and rain. The Thompson scheme includes seven variables and the Morrison schemes is based on six species of water: vapor, cloud droplets, cloud ice, rain, snow, and graupel/hail. Also the number concentrations and mixing ratios of cloud ice, rain, snow, and graupel/hail, and mixing ratios of cloud droplets and water vapor are included, making it a total of 10 species. The radiation schemes provide atmospheric heating due to radiative flux divergence and surface downward long-wave and shortwave radiation for the ground heat budget. Long-wave radiation includes infrared or thermal radiation absorbed and emitted by gases and surfaces. Upward long-wave radiative flux from the ground is determined by the surface emissivity that in turn depends upon land-use type, as well as the ground (skin) temperature. The radiation package is computationally very expensive as calculations are performed for a number of different wavelengths emitted from the earth's surface and atmospheric gases. Unlike micro-physics, the radiation package is not generally called at every time step because of its computational cost. The radiation scheme and the frequency of calls to radiation physics can be selected based on the nature of the simulation and the underlying geography.

To summarize, the different micro-physics options have varying levels of complexity and computational requirements. In general, a more complex option resolves the atmospheric processes better but is more compute intensive. Also, increasing the frequency of calls to the radiation package results in computational overhead. The appropriate choices of these two physics options will depend on the nature of simulation and the underlying geography. Different sibling domains often require configuration with different set of options that lead to load imbalance in their processing time.

The model simulation proceeds in small units of time called 'time step'. For a fixed simulation period (such as 2-days), a simulation with a smaller time step frequency will require more number of time step simulations and therefore an increased run time for the simulation.

IV. Optimizations with Partitioned Node Sets

In this section we discuss details of the optimizations related to partitioning the node space and assigning sibling domains to different partitions. The strategy for allocation of nodes to the sibling domains is similar in spirit to that of Malakar et al.[4]. We briefly summarize this strategy for the sake of completeness. We note that their strategy is primarily designed to handle sibling domains with different node sizes. We propose a simple heuristic for handling different physics options and time step ratios as well. We later show that this heuristic does well in practice.

The motivation for this approach comes from the fact that weather codes do not scale perfectly and therefore executing multiple sibling domains in parallel on partitions of the processors is more beneficial than executing them sequentially one after the other on the entire node set. However, it is also important to note that the communication does not stand to benefit from this approach. The data exchange in the force and feedback operations remains the same.

We next describe how the processing for different domains is handled on the set of nodes. The parent simulation domain is solved on the full set of available processors. The processor space P can be considered as a virtual processor grid of size $P_x \cdot P_y$ $(= P)$. Consider the parent simulation domain of size $D_x \times D_y$. Initially, this domain is distributed over the processors by assigning rectangular regions of size $D_x/P_x \times D_y/P_y$ to each processor. The sibling domains are assigned processors as follows. The virtual processor grid is partitioned into multiple rectangular subgrids. The number of partitions is equal to the number of nested simulations. The area of a region allocated for a nested simulation is proportional to the predicted execution time of the nested simulation. This is illustrated in Figure 4 – it shows the sub-grids of the processor space allocated to 2 nested simulations whose predicted execution times (as obtained from our performance prediction model) are in the ratio of 2 : 1.

The subdivision of the virtual process topology into n rectangular regions is a variant of the rectangular partitioning problem, which is known to be NP-hard[16]. When there are two nests, then a simple strategy is to divide the virtual processor topology into 2 rectangular regions in the desired proportion. As the desired proportion may not yield exact rectangular regions, we suffice with the rectangular regions approximately having the desired proportion. The partitioning is always done along the longer dimension to ensure that the rectangles are as square-like as possible – this minimizes the difference in the communication volume along the two dimensions. When there are more sibling domains, we can use the algorithm of [4] based on Huffman trees that divides the node set into regions having roughly the desired ratios.

We use a simple heuristic to determine the proportion into which the node set should be partitioned for execution of the sibling domains. We obtain the execution times of the individual sibling domains when they execute on the full node set (note that this is the way the WRF model executes). These execution times of the sibling domains represent a good first order estimate of the computation load of the sibling domains. We therefore divide the node set based on the ratio of the above obtained execution time of the sibling domains. We remark here that determining the optimal allocation of nodes is not an online problem – typically a configuration is setup once and then used for daily operational runs – it is therefore acceptable to use execution times from runs of the model itself to determine the strategy for the allocation of nodes.

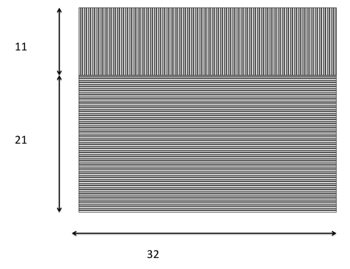

Fig. 4. Partition for 1024 processors when the execution times for the two domains are in the ratio 2:1

V. Experimental Setup

A. IBM Blue Gene/Q

Blue Gene/Q (BGQ) [17], [18], [19] is the third generation of highly scalable, power efficient supercomputers in the IBM Blue Gene product line, following Blue Gene/L and Blue Gene/P. Each BGQ compute node contains sixteen 1.6 GHz PPC A2 core processors with 16 GB of physical memory. Each processor is 4-way multi-threaded (SMT) and therefore supporting simultaneous execution of up to 64 threads per node. An application can run 1 to 64 processes per node using 64 to 1 threads per process. Each node has 32MB of shared L2 cache supporting transactional memory and speculative execution. BGQ nodes are interconnected with a 5D Torus based communication network. Each bidirectional torus link can send/receive data at 2 GB/s. The earlier generations of Blue Genes used a 3D Torus based network with comparatively lower link bandwidth than BGQ. The BGQ system software provides optimized MPI libraries for inter-process communication and an OpenMP runtime via IBM XL Compilers to take advantage of multi-threaded BGQ node.

The 96 rack, Blue Gene/Q system called Sequoia installed at Lawrence Livermore National Laboratory (LLNL) delivers more than 20 Petaflop of computing power and is currently the #2 Supercomputer in the Top500 Supercomputer listing[20]. There are four Blue Gene/Q supercomputers amongst the top 10 supercomputers in the Top500 list.

B. WRF Runtime Setup

We used WRF-ARW version 3.3.1, compiled in hybrid mode (dm+sm) for conducting our experiments. The model equations represent fully compressible, nonhydrostatic atmosphere with terrain-following hydrostatic pressure vertical coordinate. The higher-order numerics such as 3rd-order time integration schemes, 5th and 3rd order scalar advection schemes in the horizontal and vertical directions are used in this model. It uses a time-split small step for acoustic and gravity-wave modes. The base configuration uses Kain-Fritsch convection parameterization, Yonsei University planetary boundary layer scheme, RRTM long wave radiation, Thompson microphysics scheme and Noah Land surface model.

Our experiments utilize a three domain configuration with one parent domain and two sibling domains. The model performance on BGQ has been evaluated under various conditions that cause load imbalance in the sibling domains. The sibling domains used in our experiments are designed as follows:

- We setup sibling domains of different sizes keeping all the other configuration parameters same. In this case, simulations are performed with and without allocating the resources in the proportion of their size. The sizes of the two sibling domains were 1351x1351 and 850x850 grid points in the horizontal mesh.
- We setup sibling domains utilizing different time steps. Again the resources are allocated to domains in different proportions. In this case the sizes of the two siblings are kept the same.
- We setup sibling domains with different physics options. In one sibling domain relatively simpler Kessler microphysics scheme is used with invocations to radiation physics after every nine time steps whereas in the other sibling domain we use the more computationally expensive Thompson microphysics with three times higher frequency if invocation to the radiation physics.

In all our experiments, the WRF output files are produced only at the beginning of the simulation with no other output file generated during the simulation, to avoid any additional I/O contributions to runtime.

VI. Performance Results and Analysis

In this section, we present and analyze the results of our performance evaluation of WRF with sibling domains on BGQ. We discuss WRF scaling with varying number of nodes (processors), threads per process and different process-thread combinations. We also discuss the performance when the two sibling domains have load-imbalanced configurations, i.e., they have different sizes (number of grid points), different simulation time steps and physics options as discussed in the previous section.

We consider two different domain configurations for our experiments – one consisting of small domain sizes and the the other large domain sizes. The small domain configuration (config-small) has 3 sub-domains (nests) with the outer (parent) domain containing 860x780 grid points. This parent domain has two child domains (siblings) of the same size containing 625x625 grid points each at same resolution. The large domain configuration (config-large) also has 3 sub-domains but of larger size compared to the domains in small domain configuration (config-small). The outer parent domain contains two child domains (siblings) with different number of grid points but same grid resolution. The parent domain, second sub-domain (first sibling domain) and third sub-domain (second sibling domain) have 1720x1560, 1351x1351 and 850x850 grid points respectively.

A. Scaling

We start by discussing the scaling of WRF on BGQ for some realistic configurations.

1) Scaling with Nodes: Figure 5 shows the WRF performance with varying number of BGQ nodes using a total of 16 threads per node in four different rank-thread configurations. We plot the average time taken per time step of the outer (parent) domain using (i) 16 MPI tasks per node and 1 thread per task, (ii) 8 MPI tasks per node and 2 threads per task, (iii) 4 MPI tasks per node and 4 threads per task and (iv) 2 MPI tasks per node and 8 threads per task. We did not use more than 16 threads per node in order to make each thread run on a different A2 processor core in a non-SMT fashion. We remark here that our experiments are conducted with more realistic configurations as opposed to artificial benchmarks that are known to scale well. There is close to 5x performance speed up when increasing the nodes from 32 to 256 (i.e. 8 times) for most of these four setups which is good for the realistic configurations under consideration. The scaling slowly drops as we increase the number of nodes. This is as expected in strong scaling of weather codes as the computational work per node reduces with increasing number of nodes and the communication overheads increase. We see that the performance drops when we reach 256 nodes with 16 MPI tasks per node – this is attributed to over-decomposition of the domain when distributing on 4K MPI tasks - it is dependent on the specific problem size (domain configuration) and number of BGQ nodes used.

2) Scaling with Threads: Figure 6 shows the WRF performance with varying number of threads per process on 128 nodes using the small domain configuration (config-small). The plot shows scaling with increasing number of threads when the number of MPI tasks per node is fixed. We obtain these scaling

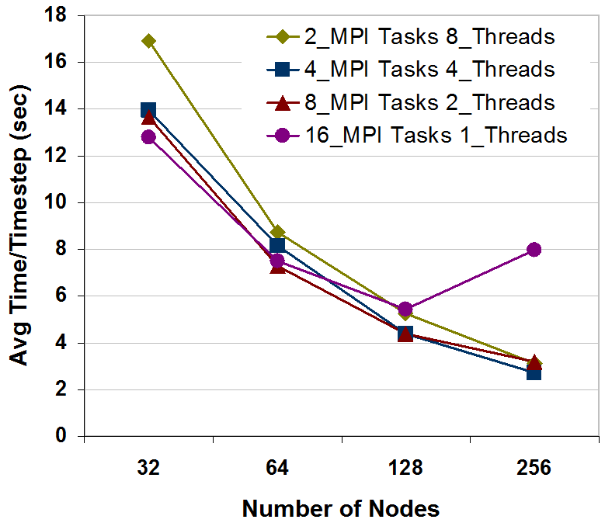

Fig. 5. Performance with varying number of nodes

results for 2,4 and 8 MPI tasks per nodes. The maximum number of threads for 2, 4 and 8 MPI tasks per node are 32, 16 and 8 respectively so that the total number of threads per node doesn't exceed 64. For 2 MPI tasks per node, there is a speed up of about 5x when going from 1 thread to 32 threads per MPI task. We obtain good scaling when increasing the threads per task from 1 to 8. Beyond this the scaling drops. This is expected because beyond 8 threads per task (i.e. total of 16 threads per node for both the tasks), the threads are running in the SMT mode with more than one thread running on each A2 processor core. A similar behavior is seen with 4 and 8 MPI tasks per node. For 4 MPI tasks per node, the speed up is about 3.5x as we increase the number of threads per MPI task from 1 to 16 and for 8 MPI tasks per node, the speed up is about 2.3x as we increase from 1 to 8 threads per MPI task.

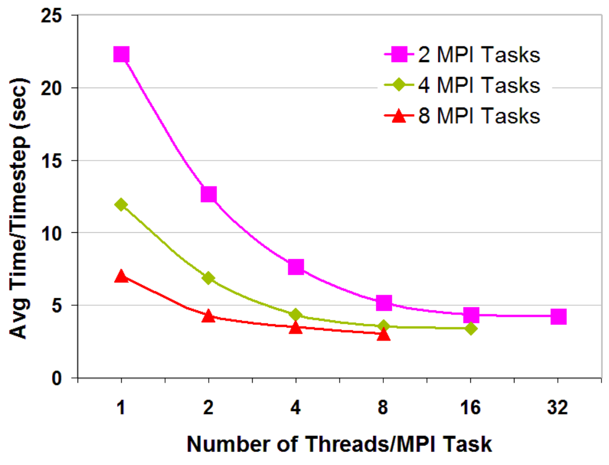

Fig. 6. Performance with varying number of threads

3) Varying process-thread Combination: Hybrid architectures such as the BGQ offer applications to run in many different combinations of MPI task and threads when running on a fixed size system. For example, if we wish to utilize all the 64 threads on every BGQ node, this can be run in many different combinations such as 1 MPI task per node with 64 threads per task or 2 MPI tasks per node with 32 threads per task, etc. Depending on the domain decomposition characteristics of the application that determine the amount of communication between MPI tasks and its suitability of the computations to shared memory parallelism, the application performance may vary significantly for these different combination of MPI tasks per node and threads per task for the same system size.

Here, we examine the performance of WRF on BGQ as we vary the combination of MPI tasks per node and threads per task. We fix the the total threads per node to 64, i.e., tasks per node $\times$ threads per task = 64, and examine how WRF performs on systems of 64, 128 and 256 nodes for MPI tasks per node varying from 1 to 16. The results are shown in Figure 7. The plot shows that for WRF, it is generally beneficial to go with more MPI tasks with fewer threads per task than to go with fewer MPI tasks and more threads per task. The performance steadily increases as we increase the number of MPI tasks and dedicate fewer threads for shared memory parallelism. However, we see a sudden drop in performance when the number of MPI tasks per node reaches 16 on the 128 node system and when it reaches 8 on the 256 node system. This is attributed to over-decomposition of the domain and is dependent on the specific problem size (domain configuration) and number of BGQ nodes used. We observe that the performance can drastically drop when the domain is over-decomposed. Therefore, it is recommended to increase the MPI ranks ensuring the domain is not over-decomposed; once the domain is optimally decomposed, then the number of threads per task should be increased to take advantage of shared memory parallelism.

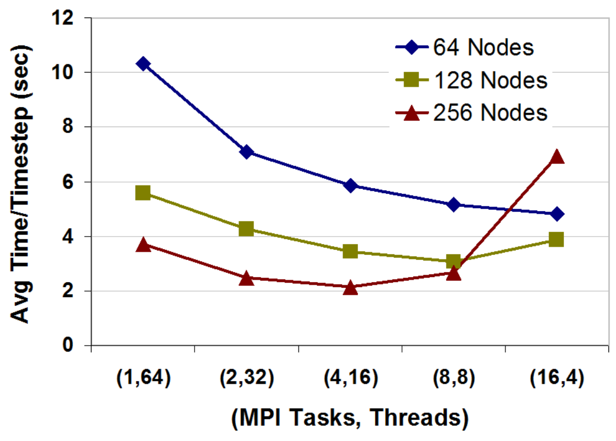

Fig. 7. Performance with varying process-thread combination

B. *Parallel Execution of Subdomains*

In this section, we discuss the performance of WRF modified with the optimizations related to partitioning the node set and executing the sibling domains.

1) Varying sub-domain sizes: In the large domain configuration (config-large), the sibling domains are of different sizes and contain different number of grid points but at the same horizontal resolution. Figure 8 shows the performance of base WRF (*Domain-Large_base*) and WRF modified with parallel processing of sibling domains on 256 nodes. We consider two different partitions for processing the sibling domains – *Domain-Large_1:1* corresponds to the case where two partitions of equal size (ratio 1:1) are created and assigned to each of the sibling domains and *Domain-Large_2.35:1* corresponds to the case where two partitions with sizes in the ratio 2.35:1 are created and the larger (smaller resp.) domain is assigned to the larger (smaller resp.) partition. This ratio is determined by our heuristic based on the execution time ratio of the two sibling domains on the full set of nodes in base WRF. We see that *Domain-Large_2.35:1* shows improvement of up to 14% over the base run. *Domain-Large_1:1* shows poorer performance compared to the base code. This is attributed to the load imbalance amongst the sibling domains as the larger sibling domain now uses only half the processor compared to what is used in the base code whereas it can perform better with more procesors.

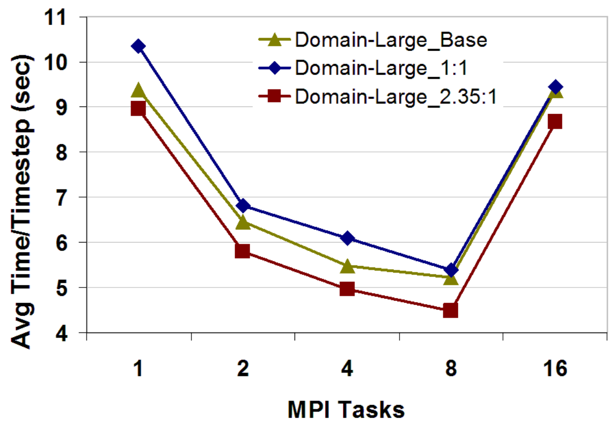

Fig. 8. Performance with different sub-domain sizes

2) Varying sub-domain time steps: We modified the small domain configuration (config-small) by changing the time step of second sibling domain to make it half of the time step of first sibling domain. This results in the processing time of the second sibling domain to become slower by a factor of 2 compared to first sibling domain. Figure 9 shows the performance for this case with the base code (*Base Code*) and different processor divisions for sibling domains on 128 nodes – we consider the division of processors in the ratios of 1:1 (*Domain-Small_1:1*), 1:2 (*Domain-Small_1:2*) and 1:3 (*Domain-Small_1:3*). We see that the processor allocation with 1: 2 ratio, where second sibling gets twice the processors as first sibling, performs the best with up to 12% improvement over the base performance. The division of processors across siblings in other ratios of 1:1 and 1:3 results in performance degradation compared to base because of load imbalance in the processing of sibling domains in both these ratios.

3) Varying sub-domain physics options: In this experiment we modified the small domain configuration (config-small) by setting the physics options of first sibling domain to the more compute intensive Thompson microphysics scheme with higher frequency of calls to the radiation module and setting the physics options of the second sibling domain to the comparatively less compute intensive Kessler microphysics scheme with lower frequency of calls to the radiation module. Figure 10 shows the performance in this case with base code (*Domain-Small_Base*) and different processor divisions for sibling domains on 128 nodes – we consider the division of

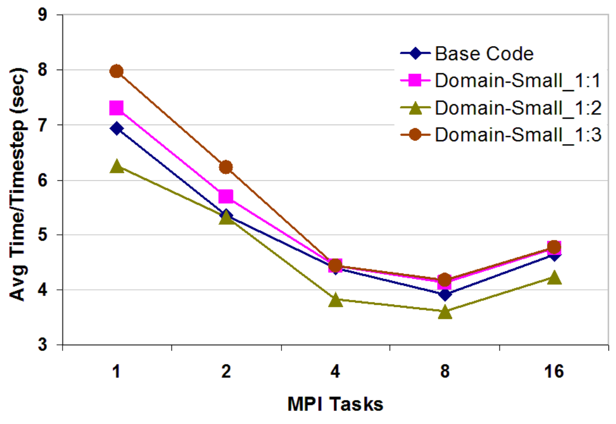

Fig. 9. Performance with different sub-domain time steps

processors in the ratios of 1:1 (*Domain-Small_1:1*) and 1.35:1 (*Domain-Small_1.35:1*). The domain involving the more (less resp.) compute intensive physics options is assigned to the larger (smaller resp.) partition. This ratio is determined by our heuristic based on the execution time ratio of the two sibling domains on the full set of nodes in base WRF. We see that *Domain-Small_1:1* and *Domain-Small_1.35:1* both perform better than *Domain-Small_Base*. Both these partitioning schemes show a performance improvement of more than 8% over the WRF base code. *Domain-Small_1.35:1* shows an improvement of up to 12% over the base code. We see that *Domain-Small_1.35:1* is consistently better than *Domain-Small_1:1* except in the case when over-decomposition occurs. We also conducted experiments with the processors divided in other ratios of 2:1 and 3:1 but the performance was worse than that of the base WRF code. As discussed earlier, this happens due to load imbalance in the processing of two sibling domains for these ratios.

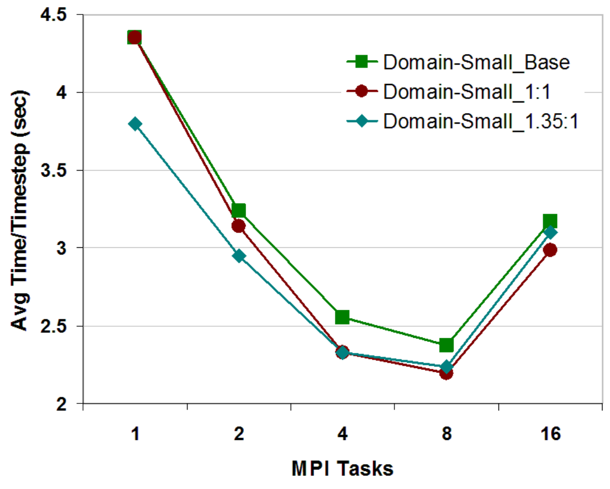

Fig. 10. Performance with different sub-domain physics options

VII. Conclusion

Weather forecasting is important for the accurate and timely prediction of catastrophic events. Weather forecasting applications based on numerical weather prediction are very compute intensive and a lot of research and effort has been put into making them scale on modern day high performance computing systems. We evaluated the performance of WRF on Blue Gene/Q and showed that the application scales quite well for real-life configurations. We also showed that with proper choice of system parameters, such as number of MPI tasks per node and threads per task, the scalability can be improved.

We also studied the impact of parallelizing sibling domains to overcome the scaling limitations of the application and showed that even with a very simple heuristic for partitioning, we can achieve decent improvement in performance particularly when the sibling domains are load imbalanced due to different domain sizes, temporal resolutions and physics parameterization. We thus conclude that the strategy of parallelizing sibling domains is quite effective. This is one of the first few studies on this idea and we believe that designing partitioning strategies for such sibling domains will be interesting future work in this area.

References

[1] J. Michalakes, J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang, "The Weather Reseach and Forecast Model: Software Architecture and Performance," in *Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology*, October 2004.

[2] W. C. Skamarock and et al., "A Description of the Advanced Research WRF version 3," *NCAR Technical Note TN-475*, 2008.

[3] J. Michalakes and et al., "WRF Nature Run," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, 2007.

[4] P. Malakar, T. George, S. Kumar, R. Mittal, V. Natarajan, Y. Sabharwal, V. Saxena, and S. S. Vadhiyar, "A divide and conquer strategy for scaling weather simulations with multiple regions of interest," in *SC'12*, 2012, pp. 37–37.

[5] A. Arakawa and W. H. Schubert, "Interaction of a Cumulus Cloud Ensemble with the Large-Scale Environment," *Journal of Atmospheric Sciences*, vol. 31, pp. 674–701, Apr. 1974.

[6] N. J. Wright, W. Pfeiffer, and A. Snavely, "Characterizing Parallel Scaling of Scientific Applications using IPM," in *10th LCI International Conference on High-Performance Clustered Computing*, 2009.

[7] G. Shainer and et al., "Weather Research and Forecast (WRF) Model Performance and Profiling Analysis on Advanced Multi-core HPC Clusters," in *10th LCI ICHPCC*, 2009.

[8] A. R. Porter and et al., "WRF code Optimisation for Mesoscale Process Studies (WOMPS) dCSE Project Report," June 2010.

[9] A. Bhatele, G. R. Gupta, L. V. Kale, and I.-H. Chung, "Automated Mapping of Regular Communication Graphs on Mesh Interconnects," in *HiPC*, 2010.

[10] P. Malakar, V. Saxena, T. George, R. Mittal, S. Kumar, A. Naim, and S. A. b. H. Husain, "Performance evaluation and optimization of nested high resolution weather simulations," in *Euro-Par 2012 Parallel Processing*, ser. Lecture Notes in Computer Science, C. Kaklamanis, T. Papatheodorou, and P. Spirakis, Eds., vol. 7484. Springer Berlin Heidelberg, 2012, pp. 805–817.

[11] D. J. Kerbyson, K. J. Barker, and K. Davis, "Analysis of the Weather Research and Forecasting (WRF) Model on Large-Scale Systems," in *PARCO*, 2007, pp. 89–98.

[12] J. Delgado and et al., "Performance Prediction of Weather Forecasting Software on Multicore Systems," in *IPDPS, Workshops and PhD Forum*, 2010.

[13] H. D. Simon and S.-H. Teng, "How good is recursive bisection?" *SIAM J. Sci. Comput.*, vol. 18, no. 5, pp. 1436–1445, Sep. 1997. [Online]. Available: http://dx.doi.org/10.1137/S1064827593255135

[14] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *J. Parallel Distrib. Comput.*, vol. 48, no. 1, pp. 96–129, 1998.

[15] I. Moulitsas and G. Karypis, "Architecture aware partitioning algorithms," in *ICA3PP*, 2008, pp. 42–53.

[16] A. Lingas, R. Pinter, R. Rivest, and A. Shamir, “Minimum Edge Length Rectilinear Decompositions of Rectilinear Figures,” in *20th Allerton Conference on Communication, Control, and Computing*, 1982, pp. 53–63.

[17] D. Chen and et al., “The IBM Blue Gene/Q Interconnection Fabric,” *IEEE Micro*, vol. 32, no. 1, pp. 32–43, 2012.

[18] R. A. Haring and et al., “The IBM Blue Gene/Q Compute Chip,” *IEEE Micro*, vol. 32, no. 2, pp. 48–60, 2012.

[19] D. Chen and et al., “The IBM Blue Gene/Q interconnection network and message unit,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’11, 2011, pp. 26:1–26:10.

[20] “Top 500 Supercomputing Sites,” http://www.top500.org.