A Framework for Online Visualization and Simulation of Critical Weather Applications

Preeti Malakar Vijay Natarajan Sathish S. Vadhiyar

IISc-CSA-TR-2011-1
http://www.csa.iisc.ernet.in/TR/2011/1/

Computer Science and Automation
Indian Institute of Science, India

January 2011

A Framework for Online Visualization and Simulation of Critical Weather Applications

Preeti Malakar Vijay Natarajan Sathish S. Vadhiyar

Abstract

Critical weather applications like cyclone tracking and earthquake modeling require high-performance simulations and online visualization simultaneously performed with the simulations for timely collaborative analysis by geographically distributed climate science community. A computational steering framework for controlling the high-performance simulations of critical weather events needs to take into account both the steering inputs of the scientists and the criticality needs of the application including minimum progress rate of simulations and continuous visualization of significant events. In this work, we have developed an integrated user-driven and automated steering framework for simulations, online remote visualization, and analysis for critical weather applications. Our framework provides the user control over various application parameters including region of interest, resolution of simulations, and frequency of data for visualization. However, the framework considers the criticality of the application, namely, the minimum progress rate needed for the application, and various resource constraints including storage space, network bandwidth, and number of processors, to decide on the final parameter values for simulations and visualization. Thus our framework tries to find the best possible parameter values based on both the user input and resource constraints. We have demonstrated our framework using a cross-continent steering of a cyclone tracking application involving a maximum of 128 processors. Experimental results show that our framework provides high rate of simulations, continuous visualizations, and also performs reconciliation between algorithm and user-driven computational steering.

1 Introduction

Scientific applications such as weather modeling require high-fidelity simulations with complex numerical models that involve large-scale computations

generating large amount of data. Visualization is vital for subsequent data analysis and to help scientists comprehend the large volume of data output. Large-scale simulations for critical weather applications like cyclone tracking and earthquake modeling require online/"on-the-fly" visualization simultaneously performed with the simulations. Online visualization enables the scientists to provide real-time feedback in order to steer the simulations for better and more appropriate output suited to the scientific needs. Remote visualization, where the visualization is performed at a location different from the site of simulations, can enable geographically distributed climate scientists to share vital information, perform collaborative analysis, and provide joint guidance on critical weather events. Remote visualization and feedback control using computational steering can thus assist a large climate science community in analyzing large-scale scientific simulations.

High-performance simulations and simultaneous remote visualization involve the use of large stable storage for storing the weather data and networks for shipment of the data from the stable storage to the remote visualization site. However, constraints on the size and capacity of the stable storage and the network can limit the effectiveness of such online and simultaneous remote visualization of critical weather events. Contemporary climate simulations have demonstrated very high scalability on large number of modern-day processors [1]. Simulations running on thousands of cores take less than a second of execution time per time step [1]. Parallel I/O can enable very high I/O bandwidth of the range of 5 – 20 GBps on large number of cores [2,3]. A combination of high simulation rate and high I/O bandwidth leads to high rate of generation of gigabytes of climate data as output and hence rapid accumulation of data in the stable storage. This gives rise to the critical problem of storage limitation for long-running climate applications which can eventually lead to stalling of simulations due to unavailability of storage. We have developed an adaptive framework that automatically tunes various parameters including the frequency of visualization output and number of processors for simulations to enable simultaneous simulations and continuous online remote visualization of critical weather applications in environments with such storage and network constraints.

We also consider computational steering and feedback control of critical weather applications by remote scientists in addition to the automatic tuning of the parameters. *Computational steering* is a well-studied approach that allows the user to interactively explore a simulation in time or space by giving feedback to the simulation, based on visualization output. By allowing user input to instantaneously impact the simulation, interactive steering "closes the loop" between simulation and visualization [4, 5]. Unlike existing efforts on computational steering, a steering framework for controlling the

high performance simulations of critical weather events needs to take into account both the steering inputs of the scientists and the criticality needs of the application. For our work, we use the minimum progress rate (MPR) of the simulations as a parameter to represent the criticality of the application. This is a parameter input by the climate scientist to express the desired quality-of-service of the weather simulations. It denotes the minimum number of climate days that has to be simulated and output in a given wall-clock time by the application. A steering framework for critical weather applications, while allowing the scientist or user to remotely steer various application parameters including the resolution of simulations and the frequency of output for visualization, should also analyze the impact of the user-specified parameters and given resource constraints on the MPR, guide the user on possible alternative options, and possibly override the user-specified values with automatically determined values in case of infeasibility. For example, the steering framework should override a very high output frequency specified by the user if it determines that the high output frequency can lead to unavailability of storage and severely compromise the MPR or criticality needs of the application.

We have developed an integrated user-driven and automated steering framework, InSt (**In**tegrated **St**eering), for simulations, online remote visualization, and analysis for critical weather applications. InSt allows steering of application parameters including resolution of simulation, rate of simulation and frequency of data for visualization. Further, it allows scientists to specify region of interest and perform finer resolution simulation for that region of interest. However InSt also considers the criticality of the application, namely, the minimum progress rate needed for the application, and various resource constraints including storage space, network bandwidth, and the number of processors, to decide on the final parameter values for simulation and visualization. Thus our framework tries to find the best possible parameter values based on both user input and resource constraints. Our framework InSt, effectively combines computational steering by the user/scientist with the algorithmic steering performed by the runtime system of the framework. Thus our framework is unique since it considers the reconciliation of both user-driven computational steering and algorithmic steering, unlike existing work that considers only user-driven computational steering [4, 6–8]. We present results that show how the framework provides quality-of-service with respect to high rate of simulation, continuous visualizations, and also performs reconciliation between algorithm and user-driven computational steering.

Section 2 describes related work in computational steering of large-scale simulations and visualizations of scientific data. Section 3 presents our inte-

grated steering framework including the components and interactions. Section 4 explains the reconciliation between the algorithmic steering and user-driven steering. Section 5 presents our experiments involving different network bandwidths and results including simulation rates. Section 6 gives conclusions and enumerates our plans for future work.

2 Related Work

The analysis and study of time-varying output data, obtained from numerical simulations, is integral to the scientific process. Currently climate scientists have been analyzing the output of climate simulation in an offline "post-processing" step after the simulation is completed. There have been strategies on offline visualization for earthquake simulations [9]. However, these strategies cannot be applied for online visualization, which is very important for critical climate applications. Tu et al. [10] and Ma et. al. [11, 12] proposed tightly-coupled execution of the simulation and visualization components where simulation is followed by visualization on the same set of processors. They have considered the simulation of earthquake ground motion. The simulation and visualization cycles alternate executions on the same set of processors using the same shared data, minimizing the cost of communication from the simulation to the visualization component. Due to alternate executions, the simulation component is stalled while the visualization is performed. The simulation component is generally more compute-intensive than the visualization component. Hence, stalling simulation while the visualization component runs would cause the subsequent output of simulation to be produced after a considerable delay. The above efforts consider critical climate applications in tightly-coupled environments. Our work adaptively performs simultaneous simulations and online remote visualization for high-performance applications.

Computational steering has been extensively studied over the past several years [5,8,13–18]. A variety of steering systems have emerged like SCIRun [4], CUMULVS [18], Discover [13] etc. A taxonomy of steering systems and tools can be found in [5, 19]. Different kinds of steering have been used. *Exploratory steering* allows the scientists to control the execution of long-running, resource-intensive applications for application exploration. For example, in the work by Shenfield et al. [17], they propose a steering system that allows user to monitor or alter execution parameters of multi-objective evolutionary algorithm for engineering design. *Performance steering* allows scientists to change application parameters to improve application performance. *Algorithmic steering* uses an algorithm to decide application param-

eters to improve system and application performance [20]. For example, the work by Ribler et al. [15] proposes using fuzzy logic to adapt to changing application resource demands and system resource availability.

Computational steering has been applied to different kinds of applications like molecular dynamics simulation, biological applications, astrophysics, atmospheric simulations, computational fluid dynamics etc [6, 7, 21, 22]. These frameworks were mainly developed for exploratory steering in order to change simulation parameters interactively and thereafter, visualizing the simulation output with the new parameters. In the work in [17], the authors show that steering of multi-objective evolutionary algorithm improves quality of the solutions. The work in [6] describes an exploratory simulation environment for Smoothed Particle Hydrodynamics simulation of astrophysical phenomena in areas such as star formation and evolution. They allow the user to alter input parameters to influence simulation behaviour.

There are some steering systems which do performance steering [14, 15, 18, 23–25]. CUMULVS [18, 23] provides the user with a viewer and steering interface for modifying the application's computational parameters and improving application performance. It allows user-directed checkpointing for fault-tolerance. Autopilot [14, 15] is about dynamically adapting to changing application resource demands and system resource availability. They use sensors to capture system performance and actuators to configure application behaviour. They have used fuzzy logic in their decision mechanism to balance conflicting performance goals. The fuzzy logic decides where and what existing policy parameters are needed to be changed. Active Harmony [25] deals with automated performance tuning. It allows runtime tuning of application parameters like read-ahead parameter, switching of algorithms etc. They have developed runtime tuning algorithms to intelligently set the parameters at runtime to tune the application performance. The most common performance metrics considered are CPU time or memory space used.

There have been some efforts on remote computational steering [26, 27]. Wu et al. [27] focus on computational steering in distributed environments. They formulate visualization pipeline configuration problems with the objective of maximizing the frame rate. They show that these problems are NP-complete and propose heuristics based on a dynamic programming approach. In the work by Brooke et al. [26], the authors show how geographically distributed teams can view simultaneously the visualization of a running simulation and can steer the application. They have presented application steering of simulations in condensed matter physics, plasma physics, and fluid dynamics in a collaborative environment.

Jean et al. [28] have developed an integrated approach for online monitoring, steering and visualization of atmospheric simulations using Falcon [29].

In this work, they have done simulation of physical and chemical interactions in the ocean and atmosphere. In order to evaluate different parameter settings by the user, they have built a steering interface to let the user dynamically modify the application execution.

Our work differs from the above efforts on computational steering because we not only let the user interactively steer the application, but we also let the system override the user decision in order to meet resource constraints and application performance of critical weather applications.

3 Adaptive Integrated Steering Framework

Simultaneous and continuous visualization for user-guided simulation of critical weather applications require robust middleware for better application performance and efficient resource management. We have developed an adaptive integrated steering framework, InSt, that performs automatic tuning as well as user-driven steering. Our framework, shown in Figure 1, consists of the following components to perform coordinated simulations, online remote visualizations, and user-driven steering: *an application manager* that determines the application configuration for weather simulations based on resource characteristics and user input, *a simulation process* that performs weather simulations with different application configurations, *a visualization process* for visualization of the frames, *frame sender and receiver daemons* that deal with transfer of frames from simulation to visualization sites, *simdaemon and visdaemon* for communication of user-specified simulation parameters and system response and *user interface* for accepting user input. In our work, we remove the frames from the simulation site once they are transferred to the visualization site. The following subsections describe in detail the primary components.

3.1 User Interface, SimDaemon and VisDaemon

The user gives input through the user interface as shown in Figure 1. In particular, user can specify nest location, simulation resolution, and bounds for output interval and simulation progress rate through the user interface. The input values from the user are sent to the application manager through the *VisDaemon* and *SimDaemon*. The VisDaemon receives user input from the user interface through the visualization process, and communicates the same to the SimDaemon. The SimDaemon specifies this user input to the application manager. The response from the manager is conveyed back by the daemons to the user through the user interface.

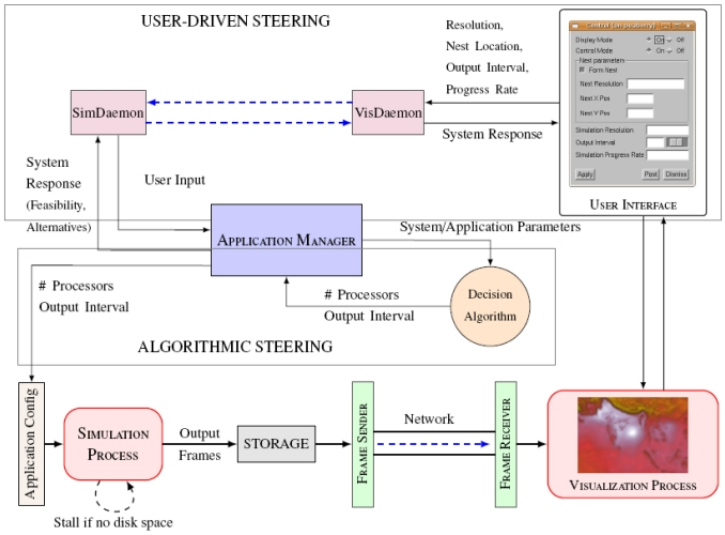


Figure 1: InSt: Integrated Steering Framework

3.2 Application Manager

The application manager is the primary component of InSt and acts as the bridge between automatic steering and user-driven steering in our framework. The application manager periodically monitors the resource parameters, namely the free disk space and available network bandwidth. For automatic/algorithmic steering, the application manager periodically invokes the decision algorithm, explained in Section 3.4, for obtaining the number of processors for simulations and the frequency of weather data output to be generated by the simulations for continuous visualization. For user-driven steering, the application manager asynchronously receives the user inputs, including the upper bound for frequency of output and the simulation resolution, from the visualization site. The manager checks the feasibility of running the simulations with the user inputs, advises the users of alternate options if not feasible, and invokes the decision algorithm with the user inputs and resource parameters if feasible.

The manager writes the simulation parameters output by the decision algorithm to the application configuration file, and starts or stops-and-restarts the simulations with the parameters. More details on the reconciliation of the automatic/algorithmic and user-driven steering are given in Section 4.

3.3 Simulation Process

The simulation process is the weather application that simulates the weather events for a desired number of days. The simulation process periodically reads the simulation parameters from the application configuration file written by the application manager and stops for restarting when the parameters in the configuration file are different from the parameters used for current execution. The simulation process also stalls execution (using *sleep*) if the available free disk space becomes less than a threshold. It periodically checks the disk space and continues execution only when the disk space becomes available again.

3.4 Decision Algorithm

The decision algorithm invoked by the application manager determines

1. the number of processors, and
2. the frequency of output of weather data

for execution of weather simulations for a given

1. resolution of simulation,
2. the bandwidth of the network connecting the simulation and visualization sites,
3. the available free disk space at the simulation site, and
4. the minimum progress rate (MPR) of simulations desired by the user.

The algorithm also takes as input the execution times for different number of processors and simulation resolutions. The decision algorithm also considers lower bound for frequency of output or upper bound for interval between outputs, *upper_output_interval*. This upper bound corresponds to the minimum frequency with which the climate scientist would want to visualize the weather events.

The objective of the decision algorithm is to maximize the rate of simulations and to enable continuous visualization with maximum temporal resolution. We define temporal resolution as the frequency at which successive frames are visualized. However, these objectives are contradictory. Increasing the frequency of output of weather data by simulations can decrease the rate of simulations due to increase in number of writes to the disk and can also lead to rapid consumption of storage, eventually stalling the simulations. Unlike traditional scheduling algorithms that minimize execution

times or maximize simulation rates, our decision algorithm may have to sometimes "slow down" the simulations, since faster simulations can lead to faster consumption of storage if the network to the visualization site is slow.

We can think of our decision algorithm problem as an optimization problem that primarily attempts to maximize the simulation rate within the constraints related to continuous visualization, acceptable frequency of output, minimum quality-of-service expressed by the user in the form of minimum progress rate of simulations (MPR), I/O bandwidth, disk space and network speed. We consider an important constraint involving the minimum progress rate (MPR) of simulations, for considering the criticality needs of the application. This constraint is also useful for ensuring quality of service to the climate scientist for continuous and fast visualization. We formulate our problem as a linear programming problem with constraints to obtain the number of processors and the frequency of output for simulations. Since we want the best possible throughput of the simulation in spite of the resource constraints, we express the objective of our optimization problem as

$$minimize\ t$$

where t is the execution time to solve a time step. The parameters used in the formulation are listed in Table 1. Among these parameters, the decision variables involved in the formulation are $\mathcal{S}$, $\mathcal{F}$, $\mathcal{T}$ and t. In the table, a frame is the simulation output of one time step of simulation and corresponds to the smallest unit of simulation output that can be visualized. Interval corresponds to some fixed execution time for the simulations. The following sub-sections describe the formulation of the constraints.

Table 1: Problem Parameters

t	Time to solve one simulation time step
$\mathcal{S}$	Number of frames solved in an interval
$\mathcal{F}$	Number of frames output in an interval
$\mathcal{T}$	Number of frames transferred in an interval
$\mathcal{O}$	Size of one frame output in one time step
$\mathcal{D}$	Total remaining free disk space
$\mathcal{T}_{IO}$	Time to output one time step
b	Network bandwidth

Time Constraint: For minimum stalling at the visualization end, it is desirable to transfer frames continuously. Consider an interval $\mathcal{I}$ when $\mathcal{T}$

frames are transferred, $\mathcal{S}$ frames are solved and $\mathcal{F}$ frames are output. For continuous visualization, the time to produce $\mathcal{F}$ frames should be less than the time to transfer $\mathcal{T}$ frames since the next set of frames should be ready for transfer by the time the transfer of current frames is over. If the next set of frames are not available, the continuity of the visualization will be affected and the visualization process will incur idling. The time to produce a frame corresponding to a time step includes the time to solve the time step and the time to write the frame onto the disk. Thus, the time to produce $\mathcal{F}$ frames includes the time to solve $\mathcal{S}$ frames and to write $\mathcal{F}$ frames onto the disk. This gives equation (1) where *tts* is the time to solve, *tto* is the time to output and *ttt* is the time to transfer. Expanding equation (1), we obtain the constraint specified in equation (2) which can be rearranged as equation (3). The relation between $\mathcal{S}$ and $\mathcal{F}$ is determined by the output frequency for the simulation. For example, if the output frequency is 1 then $\mathcal{S} = \mathcal{F}$, i.e. every frame that is solved is written to the disk.

$$tts + tto \leq ttt \tag{1}$$

$$\mathcal{S} \cdot t + \mathcal{F} \cdot \mathcal{T}_{IO} \leq \frac{\mathcal{O}}{b} \cdot \mathcal{T} \tag{2}$$

$$t + (\mathcal{F}/\mathcal{S}) \cdot \mathcal{T}_{IO} \leq \frac{\mathcal{O}}{b} \cdot (\mathcal{T}/\mathcal{S}) \tag{3}$$

Disk Constraint: Assuming that the rate of input to the disk from the simulation is greater than the rate at which the simulation output data is transferred to the visualization site, then the time n in which the disk will overflow is given by equation (4) where $\mathcal{R}_{in}$ and $\mathcal{R}_{out}$ are the rate of input to the disk and rate of output from the disk respectively. $\mathcal{R}_{in}$ is calculated using the solve time t, the output data size $\mathcal{O}$ and the interval of output (inverse of frequency expressed in simulated time units) OI, and $\mathcal{R}_{out}$ is calculated using network bandwidth b. From this we derive equation (5) which can be rearranged as equation (6).

$$n \leq \frac{\mathcal{D}}{(\mathcal{R}_{in}) - (\mathcal{R}_{out})} \tag{4}$$

$$\frac{\mathcal{O} \cdot \mathcal{F}}{t \cdot \mathcal{S} + \mathcal{T}_{IO} \cdot \mathcal{F}} - b \leq \frac{\mathcal{D}}{n} \tag{5}$$

$$t \geq \left[\frac{\mathcal{O}}{(\frac{\mathcal{D}}{n} + b)} - \mathcal{T}_{IO} \right] \cdot (\mathcal{F}/\mathcal{S}) \tag{6}$$

Rate Constraint: A steady rate of progress in simulation is required for critical weather applications in order for scientists to provide advance information based on visualization output. For weather applications, the rate of progress is represented by the ratio of the time simulated by the application (simulation time) and the wall-clock time taken by the application

for the simulation (wall-clock time). For critical applications like cyclone tracking where advance information is needed by the scientists to provide timely guidance to decision makers, this ratio has to be greater than 1, i.e., the simulation time has to be greater than the wall-clock time taken for the simulation. The rate of simulation is dependent on the computation speed as well as on the output frequency. Higher the frequency of output, higher will be the number of I/O writes to the stable storage and hence lesser will be the simulation rate. Also, lower the I/O bandwidth, more significant will be the impact of output frequency on the simulation rate. In most cases, the scientists may want to specify a minimum acceptable limit for this ratio. We denote this minimum ratio as MPR (Minimum Progress Rate), that is provided by the user/scientist in our steering framework. Equation (7) specifies this constraint related to the rate of simulations. Let ts denote the integration time step associated with the resolution of the simulation. This is the amount of time simulated or solved per time step and depends on the simulation resolution. For example, in our weather application, if the simulation resolution is 18 km, then the integration time-step was taken as $3 \cdot 18 = 54$ seconds. If S frames are solved in an interval I and F frames are produced in that interval, then simulation time will be $ts \cdot S$ and wall-clock time will be the summation of solve time and time to output F frames i.e. $t \cdot S \ + \ T_{IO} \cdot F$ as given by equation (8). This can be rewritten as equation (9).

$$\frac{Simulation\ time}{Wall-clock\ time} \ \geq \ MPR \tag{7}$$

$$\implies \frac{(ts \cdot S)}{(t \cdot S \ + \ T_{IO} \cdot F)} \ \geq \ MPR \tag{8}$$

$$\implies \frac{ts}{(t \ + \ T_{IO} \ \cdot \ F/S)} \ \geq \ MPR \tag{9}$$

Bounds: Depending on the total number of processors, t has a lower bound $\mathcal{T}_{LB}$, as specified in equation (10). We can specify upper bound OI_{UB} for the output interval based on the minimum frequency of visualization of weather events desired by the users/climate scientists. In our framework, OI_{UB} can be specified or steered by the user. Output interval also has a lower bound OI_{LB} based on the limitations of the simulation application. For our weather application, the output interval has a lower bound of 1 simulated minute[1]. The bounds for the output interval are specified in equation (11).

$$t \ \geq \ \mathcal{T}_{LB} \tag{10}$$

$$OI_{LB} \ \leq \ OI \ \leq \ OI_{UB} \tag{11}$$

Linearizing the Constraints: To linearize the non-linear constraints in equations (3), (6) and (9), we substitute $\frac{\mathcal{F}}{\mathcal{S}}$ by z and $\frac{\mathcal{T}}{\mathcal{S}}$ by y respectively to

[1]Simulated time units denote the time that is simulated and does not represent the wall-clock time.

obtain the constraints (12), (13) and (14) for our optimization problem.

$$t \ + \ z \cdot \mathcal{T}_{IO} \ \leq \ \frac{\mathcal{O}}{b} \cdot y \tag{12}$$

$$t \ \geq \ \frac{\mathcal{O}}{(\frac{\mathcal{D}}{n} \ + \ b)} \ - \ \mathcal{T}_{IO} \cdot z \tag{13}$$

$$\frac{ts}{(t \ + \ T_{IO} \ \cdot \ z)} \ \geq \ MPR \tag{14}$$

It is clear that OI depends on the ratio between the number of frames solved by the simulations and the number of frames output to the disk as explained above. Let ts denote the integration time step associated with the resolution of a climate simulation. This is the amount of time simulated or solved per time step and is constant for a given simulation. OI is a multiple of ts. A frame is solved after every ts simulated time and a frame is output to disk after every OI simulated time. Thus the total time simulated in an interval of execution time, where $\mathcal{S}$ frames are solved and $\mathcal{F}$ frames are output to the disk, is given by equation (15). Using equation (15), the bound constraint of equation (11) can be rewritten as equation (16).

$$OI \cdot \mathcal{F} \ = \ ts \cdot \mathcal{S} \tag{15}$$

$$OI_{LB} \ \leq \ \frac{ts}{z} \ \leq \ OI_{UB} \tag{16}$$

We used GLPK (GNU Linear Programming Kit) [30] to solve the above linear programming problem and obtain the values for t, z and y. From the value for t, we determine the corresponding number of processors using the benchmark profiling runs with the WRF simulations. We obtain the output interval, OI, by substituting for $z = \frac{\mathcal{F}}{\mathcal{S}}$ and ts in equation (15).

This decision algorithm is invoked every 1.5 hours during the simulation run period. Given the inputs $\mathcal{D}$, $\mathcal{T}_{IO}$, b and $\mathcal{O}$, this algorithm outputs t and OI to the application configuration file. The job handler reads the file to look for changes with the current configuration and accordingly reschedules WRF with the new configuration. Due to changing disk space, it might give a different set of outputs, namely the number of processors and the output interval OI, at different points of time during the simulation run-period.

4 Reconciling User-driven and Algorithmic Steering

Initially, before starting the simulations, the application manager specifies default values for simulation resolution and MPR. The application manager

then determines the frequency of output using the decision algorithm based on resource constraints for the given resolution of simulation. The simulations are then started with these values. The user at the visualization site can change these simulation parameters during execution through the user interface. The interface also allows the user to specify a location for formation of nest or sub-region in the domain for finer resolutions. When the user requests nest placement or a finer simulation resolution, the simulation process restarts and continues with a new configuration involving the nest and the new resolution.

When a user specifies an output interval (inverse of frequency) and/or MPR value, InSt considers the criticality of the application, proactively checks the feasibility of executing the simulations with these inputs, and guides the user with possible alternative values for ensuring continuous and reasonable progress of simulations and visualization. When a user specifies an output interval, the framework checks if the specified interval can be used without violating the rate constraint of equation (9), i.e., if the simulations can generate output with the specified interval such that the simulation rate will continue to be greater than the current MPR used by the application manager. This relationship between output interval, OI, and MPR is given by equation (17) which is derived using equations (9) and (15).

$$MPR \leq \frac{ts}{[t + T_{IO} \cdot (ts/OI)]} \tag{17}$$

It can be clearly seen that there is a direct relation between OI and MPR. For example, let the resolution be 15 km, the integration time-step, ts, be 45 seconds, T_{IO} be 43 seconds and t be 3.95 seconds. Now if the user requests OI of 180 seconds (i.e. solved time-steps are output onto the storage every 3 simulated minutes), then from equation (17), MPR will be less than or equal to 3.06. But if the current MPR is 5, then the system clearly cannot satisfy his request for an OI of 180 seconds. Therefore in such cases, InSt has to override the user-requested value for OI.

Also, it can be seen from equation (17) that if OI is too low, it will decrease the simulation rate as well. Hence to continuously simulate and visualize at a steady rate, InSt determines the lower bound of OI from equation (17) using the current value of MPR used by the application manager. If the value of OI requested by the user is less than this lower bound, then InSt does not incorporate the user-specified OI. In this case, the InSt framework informs the user of the lowest feasible OI. If the user-specified value of OI is greater than the lower bound, this value forms the upper bound for OI in the decision algorithm, i.e. the decision algorithm in InSt tries to find the best possible OI within the user-specified bound.

If the user specifies a MPR value, then the application manager attempts to change the current MPR value it uses in the decision algorithm with the user-specified value. However, it first checks if the user-specified MPR, $uMPR$ is feasible for the current resolution of simulations by calculating an upper bound, MPR_{max}, feasible for the resolution and comparing the user-specified $uMPR$ with MPR_{max}. For calculating MPR_{max}, the feasible upper bound, we substitute OI with ∞ and t with its lower bound T_{LB} in equation (17) and obtain MPR_{max} as the ratio of ts and T_{LB}. If the user-specified MPR, $uMPR$, is greater than this feasible upper bound, MPR_{max}, for the current resolution, InSt proactively tries to find a coarser resolution and checks the feasibility of $uMPR$ for the coarser resolution by calculating MPR_{max} for the coarser resolution. The application manager then provides the user with the options of coarser resolution at which the $uMPR$ is feasible. Thus, InSt proactively tries to find a balance between the algorithmic steering of the decision algorithm and the user-driven steering values considering the criticality, represented by MPR, of the application.

The flowchart in Figure 2 depicts the reconciliation or handshaking dialogue between the algorithmic steering and user-driven steering. In the flowchart, MPR is the value currently used by the application manager for simulations, Res represents the current resolution of simulation, $uMPR$ and uOI are the user-provided MPR and OI (output interval) values. The flowchart shows the various feasibility analysis done by the application manager before accepting or overriding user request. If the user-specified values are approved by the application manager, the simulation process is updated with the new values. This is denoted in the flowchart using connectors marked 'U'. When the application manager is not able to satisfy user-request due to infeasibility with respect to the current minimum progress rate used in the simulations, then the user is recommended an optimal set of values. This is denoted by connectors marked 'I' in the flowchart.

Therefore the user drives the simulation process and the automatic tuning framework steers the weather simulation to the favorable state of continuous visualization with minimum stalling and maximum progress rate.

5 Experiments and Results

5.1 Resource Configuration

For all our experiments, visualization was performed on a graphics workstation in Indian Institute of Science with a dual quad-core Intel® Xeon® E5405 and an NVIDIA graphics card GeForce 8800 GTX. We used hardware accel-

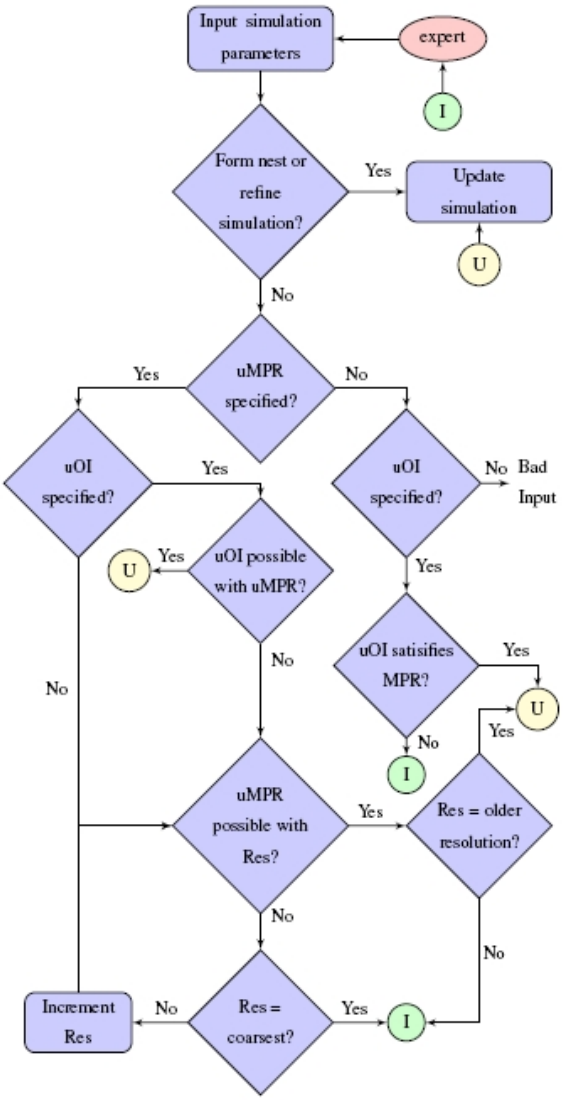


Figure 2: Flowchart depicting reconciliation

eration feature of VisIt [31] for faster visualization. We executed the simulations on two different sites resulting in two different remote visualization and computational steering settings, namely, *intra-country* and *inter-country* steering. In the *intra-country* configuration, the simulations were performed on a quad-core Intel® Xeon® X5460 cluster, *gg-blr*, in the Centre for Development of Advanced Computing (C-DAC), Bangalore, India. The transfer between simulation and visualization site for this *intra-country* configuration was carried out on the National Knowledge Network (NKN) [32] with the maximum bandwidth of 1 Gbps. In the *inter-country* configuration, the WRF

Table 2: Simulation and Visualization Configurations

Configuration	*Simulation Configuration*	*Maximum Cores for Simulation*	*Maximum Disk Space Used*	*Average Sim-Vis Bandwidth*
intra-country	*gg-blr*: HP Intel Xeon Quad Core Processor X5460, 40 nodes, 320 3.16 GHz cores, RHEL 5.1 on Rocks 5.0 operating system, each with 16 GB RAM and 500 GB SATA based storage, and connected by Infiniband primary interconnect and Gigabit Ethernet secondary interconnect. For our work, we used the Gigabit network	96	150 GB	40 Mbps
inter-country	*abe*: Dell PowerEdge 1955 dual-socket quad-core compute blades, 1200 blades, 9600 2.33 GHz cores, RHEL 4 operating system, 1 GB RAM per core, 100 TB Lustre filesystem, and connected by Infiniband	128	700 GB	8 Mbps

simulations were conducted on the dual-socket quad-core Intel 64 (Clovertown) PowerEdge 1955 cluster, *Abe*, in National Center for Supercomputing Applications (NCSA), Illinois, USA. Table 2 gives the detailed specifications of the two resource configurations including the maximum cores used for simulations, the maximum disk space used by our adaptive framework for the experiments, and the average available bandwidth between the simulation and the visualization sites for each of the configurations.

5.2 Weather Model and Cyclone Tracking

We have applied our framework for an important critical weather application, namely, large-scale and long-range tracking of cyclones. Visualization of cyclones is vital for subsequent data analysis and to help scientists comprehend the huge volume of data output. Visualization can be helpful in identifying important aspects of the modeled region. For example, the development of low pressure or the appearance of high vorticity can be easily detected.

In our experiments, we used our framework for tracking a tropical cyclone, *Aila*, in the Indian region. Aila was the second tropical cyclone to form in the Northern Indian Ocean during 2009 [33]. The cyclone was formed on May 23, 2009 about 400 kms south of Kolkata, India and dissipated on May 26, 2009 in the Darjeeling hills. There were 330 fatalities, 8,208 reported missing and about $40.7 million estimated damage. We simulated Aila upto a finest resolution of 3.33 km using a mesoscale numerical weather forecast model, WRF (Weather Research and Forecasting Model) [34, 35].

The modeled region of forecast is called a *domain* in WRF. The WRF simulations involve one parent domain which can have child domains, called *nests*. WRF supports nesting to perform finer level simulations in specific

regions of interest. We used the *nesting* feature supported by WRF to track the lowest pressure region or eye of the cyclone and perform finer level simulations in the region of interest inside the parent domain as shown in Figure 3. The nesting ratio i.e. the ratio of the resolution of the nest to that of the parent domain, was set to 1:3.

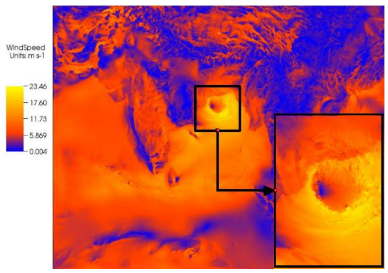


Figure 3: Windspeed visualization in finer resolution nest inside parent domain

As WRF is a regional model, with each level of refinement, it needs input data at a finer resolution. Before executing WRF, the WRF Preprocessing System (WPS) is executed to interpolate the meteorological data onto the domain of interest. The 6-hourly 1-degree FNL analysis GRIB meteorological input data for our model domain was obtained from CISL Research Data Archive [36]. WRF allows for writing the weather data at specified output frequencies. WRF outputs data in the form of NetCDF [37] files. Each NetCDF file contains output of a number of simulation time steps. WRF also supports *restart files* for checkpointing the data during execution, and continuing the application from the checkpointed data.

For the intra-country experiments, we performed simulations for an area of approximately 32×10^6 sq. km. from 60°E - 120°E and 10°S - 40°N over a period of 2 days. For the inter-country experiments, we performed simulations over a larger area or domain and for a longer period of time, since the *Abe* cluster supports faster rate of simulations (approximately 1.5 time steps per second) and has faster Infiniband interconnect. For these inter-country experiments, the domain was approximately from 30°E - 150°E and 10°S - 40°N and the simulation was done over a period of 3 days and 18 hours. These domains correspond to the areas of formation and dissipation of Aila.

5.3 Framework Implementation

The modifications to the WRF weather application for our work are minimal. For tracking cyclones, our framework contains mechanisms for identifying the formation of cyclones in addition to the functionalities described in the

earlier sections. Our framework forms the nest dynamically based on the lowest pressure value in the domain and monitors the nest movement in the parent domain along the eye of the cyclone. The main modification is to make the WRF stop for rescheduling on different number of processors when application configuration file specifies the number of processors and output interval that are different from the current configuration. For our WRF executions, the default upper bound for output frequency was specified as 30 simulated minutes and the lower bound was specified as 3 simulated minutes. However, the user can change these values during steering.

To obtain the simulation rates for different number of processors, that are used by our decision algorithm, sample WRF runs, each with a simulation period of 1 hour, were executed for different discrete number of processors spanning the available processor space and using performance modeling or curve fitting tools [38] to interpolate for other number of processors. These WRF profiling runs were executed on 16, 24, 32, 48, 56, 64, 80 and 96 processors in *gg-blr* cluster and on 32, 48, 64, 80, 96, 112 and 128 processors in *abe* cluster.

For faster I/O we used WRF's split NetCDF approach, where each processor outputs its own data. This approach is beneficial, especially for low bandwidth, low latency networks for faster data transfer from the simulation site. It also significantly reduces the I/O time per time step. We have developed a utility to merge these split NetCDF files at the visualization site. We have also developed a plug-in for VisIt to *directly read* the WRF NetCDF output files, eliminating the cost of post-processing before data analysis. We have customized VisIt to automatically render as and when these WRF NetCDF files are merged after arriving at the visualization site. We have also visualized the output using volume rendering, vector plots employing oriented glyphs, pseudocolor and contour plots of the VisIt visualization tool. For the steering interface, we have developed a GUI inside VisIt using Qt. A snapshot of the GUI can be seen in Figure 1.

The application manager periodically (in our work, every 1.5 hours) monitors the available disk space using the UNIX command *df*. The application manager also uses the average observed bandwidth between the simulation and visualization sites, obtained by using the time taken for sending about 1 GB message across the network. The application manager also periodically invokes our decision algorithm every 1.5 hours. This frequency was sufficient for our experiment settings where the storage space and network bandwidth did not exhibit high fluctuations. For highly dynamic environments, the decision algorithm will have to be invoked more frequently. Although the threshold values used in our InSt are specific to our experiment settings and WRF simulations, the general principles of our steering framework are

generic and applicable to other applications.

5.4 Automatic Tuning Results

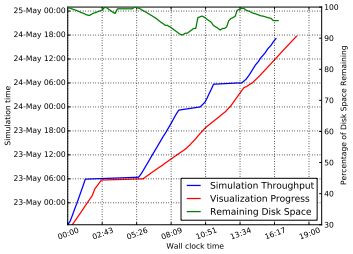
We first demonstrate the efficiency of the decision algorithm and the automatic tuning of parameters in the InSt framework in the absence of user inputs. For experiments in this section, our framework automatically determines the WRF nest location and changes the resolution of simulations based on pressure values in the weather data. Our framework spawns a nest when the pressure drops below 995 hPa. The nest is centered at the location of lowest pressure in the parent domain. We also use a configuration file that specifies the different resolutions for simulations and visualization for different pressure gradients or intensity of the cyclone. This can be specified by the climate scientists who typically use coarser resolutions for the initial stages of cyclone formation and finer resolutions when the cyclone intensifies. As and when the cyclone intensifies i.e. the pressure decreases further, our framework changes the resolution of the nest multiple times to obtain a better simulation result from the model.

5.4.1 Intra-country Results

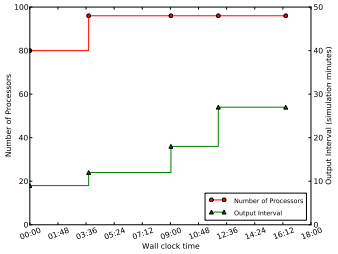
Figure 4(a) shows the results for the *intra-country* experimental setup using the *gg-blr* cluster. The graph shows three curves: a simulation curve (blue) that plots the simulated time versus wall-clock time, a visualization curve (red) that plots the times at which the corresponding output were visualized, and a disk availability curve (green) that shows the consumption of disk space during the execution. The x-axis shows the wall-clock time progression of execution for the simulation and disk availability curves. For the visualization curve, the x-axis shows the wall-clock time when the different frames are visualized. For the simulation and visualization curves, the left y-axis shows the corresponding simulation times represented by the frames. For the disk availability curve, the right y-axis shows the percentage of remaining disk space.

For this experiment, the initial resolution of the simulation was chosen as 24 km. The resolution was changed to 21 km when the lowest pressure drops below 994 hPa, 18 km for pressure less than 991 hPa, 15 km for pressure less than 989 hPa and 12 km for pressure less than 987 hPa. The minimum progress rate (MPR) of simulations for the rate constraint of our algorithm was chosen as 5.

Figure 4(a) shows that the lag between visualization and simulation is minimal resulting in a truly online visualization. This is primarily because



(a) Simulation (blue) and Visualization (red) progress, and Disk Consumption (green) for *intra-country* configuration. Frames are visualized with a minimal time lag due to the high network bandwidth

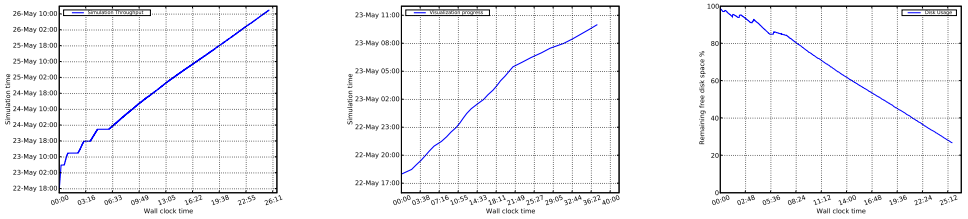


(b) Adaptivity of the framework showing variation in the number of processors (red, left y-axis) and output interval (green, right y-axis) for *intra-country* configuration. Decision algorithm computes the optimal number of processors and output interval.

Figure 4: Simulation and Visualization progress, Disk Consumption and Adaptivity for *intra-country* configuration. Initial WRF resolution = 24 km, MPR = 5.

of the high network bandwidth. However, the lag is not constant because of variation in the network bandwidth and difference in simulation rates at different points of execution. When WRF restarts at a finer resolution, it also needs input data at the finer resolution. These regions can be seen as the flat regions in the curve. The long flat regions are due to the unusually low I/O bandwidth in the ggblr cluster. These regions also correspond to the increase in available disk space because of the continuous transfer of frames from the simulation site to the visualization site even during these restart events in simulation. It can be observed that the remaining disk space is always above 90%. This is because the network bandwidth for this experiment is quite high, which implies that the rate at which the disk is freed is quite high.

Figure 4(b) shows the values for the number of processors and output interval by the simulations automatically determined by InSt at different stages of execution for the *intra-country* configuration. During the initial stages, our framework chooses an initial value of 9 and 80 for output interval and the number of processors respectively. During the course of execution, the number of processors and output interval changes a few times. This happens when one or more parameters change in the constraint equations as explained in Section 3.4. For example, when resolution changes from 18 km to 15 km, the time to solve a time step, the output data size per time step and the time to output a time step also change. So, the decision algorithm re-evaluates the correct number of processors and the best output interval for the current parameters. As the resolution becomes finer, the output size



(a) Simulation Progress (b) Visualization Progress (c) Disk Space Consumption

Figure 5: Simulation and Visualization progress, and Disk Consumption for *inter-country* configuration. Initial WRF resolution = 18 km, MPR = 3. Steady progress in simulation and visualization without stalling is observed.

and hence the time to output increases. Our InSt framework also provides guarantees regarding the rate of simulations (simulation time / wall-clock time). Specifically, the framework attempts to maintain higher simulation rates than the minimum progress rate (MPR), using the rate constraint in equation 9 of the decision algorithm. For the *intra-country* experiment, the MPR was specified as 5. The decision algorithm increases the output interval to satisfy the MPR and to prevent disk overflow due to frequent I/O. Thus, our framework InSt algorithmically steers and adaptively changes the execution and application parameters based on the application and resource configurations and application simulation rates.

5.4.2 Inter-country Results

Figure 5 shows the simulation, visualization and disk usage graphs obtained in the *inter-country* configuration with NCSA's Abe cluster in USA for simulations, and the visualization engine in IISc, India. For this experiment, the initial resolution of the simulation was chosen as 18 km. The resolution was changed to 15 km when the lowest pressure drops below 993 hPa, 12 km for pressure less than 991 hPa and 10 km for pressure less than 989 hPa. The minimum progress rate (MPR) of simulations for the rate constraint of our algorithm was chosen as 3.

From Figure 5(a), we can see that the simulation for 3 days and 18 hours was completed in about a day because of the high simulation rate. The simulation curve shows a rapid change in slope for the initial few hours when the cyclone is being formed and the simulation resolution is being refined. After it reaches the finest resolution, it continues at a steady rate and there is not much change in slope. At a coarser resolution, the time steps are solved at a much faster rate because the number of grid points are lesser than in the finer resolution. Hence the slope is steeper in the beginning of the simulation.

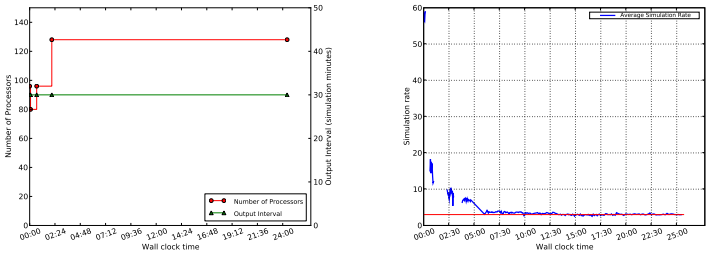
Figure 5(b) shows that we are able to continuously visualize despite the slow network bandwidth. It can be seen that there is a change in slope of the curve at a point corresponding to 23^{rd} May, 05:00 hours simulation time step. This is because a nest is formed at this time step, and hence the total amount of output data for a single time step increases and hence the time to transfer one time step increases.

Figure 5(c) shows that the simulation completed without overflowing available disk space. This is because of adjusting the number of processors to the correct value so that the disk space is not a problem even though the simulation rate is high and the network bandwidth is low.

We find a considerable time lag between the time when a time step was simulated and when the corresponding frame was visualized. This is because of the high simulation rate in Abe cluster, and the slow Internet-based transfer of the frames to the visualization site in India. The time taken for simulating a time step in the coarsest resolution in Abe is approximately 0.6 seconds. Our framework chose the initial number of processors for WRF executions in Abe as 96, and the output interval for the simulations as 30. The maximum number of processors, i.e 128 processors were not chosen by the optimization algorithm because of the disk space constraint due to the slow network. If simulation had started on 128 processors, it would have progressed at a much faster rate leading to storage problem later. At the same time, it was also not drastically reduced so that a minimum progress rate of simulation is maintained without overflowing the disk. The output interval was chosen to be 30 because more output frames would mean a higher disk space consumption rate due to slow rate of data transfer from the disk.

Figure 6(a) shows the change in the number of processors during the course of execution. It can be seen that the change is not frequent, this is because the decision algorithm selects an optimal value considering the parameters like simulation resolution, I/O bandwidth, network bandwidth, time to solve and free disk space available. When the simulation runs at a given resolution, only the disk space varies. However since the total available disk space is already considered by the algorithm while selecting the optimal value, there is not much variation later during the run. When the resolution changes, some of the parameters change and hence we can see some variation in the number of processors. Since the simulation starts with a coarser resolution and then changes to a finer resolution at different points of the experiment, the number of processors is more for the finer resolution. This is because at a finer resolution, the time to solve a time step increases.

Figure 6(b) shows the actual rates of simulations for different stages of WRF execution for the experiment with the *inter-country* configuration. We set an MPR value of 3 for this experiment and we find that the rates are al-



(a) (b)

Figure 6: Actual rates of simulations and change in number of processors for *inter-country* configuration . The rate constraint of our decision algorithm ensures higher simulation values than the MPR.

ways maintained higher than the MPR value, thus ensuring minimum guarantees in the speed of simulations, which is very essential for continuous tracking of critical weather events. Even though the simulation is slowed down in the beginning of the experiment when the simulation rate is quite high due to coarser resolution, yet the framework ensures that the MPR is satisfied.

5.5 Computational Steering Results

In this section, we demonstrate the user-driven steering supported by InSt, namely, the various steering capabilities provided to the user, the feedback mechanisms in the framework, and the reconciliation of the user-driven steering and automatic tuning by the framework. For these experiments, the simulations are started at a particular resolution but unlike in the automatic tuning experiments of the previous section, the resolutions of the ongoing simulations can be changed only by the user. Similarly, the placement of the nest for finer simulations can also be performed only by the user.

5.5.1 Intra-country Steering

Figure 7 shows the steering results for the *intra-country* experimental setup using the *gg-blr* cluster. The graph also shows the various steering events provided by the user.

Initially, the simulations were started with a resolution of 24 km, and MPR of 5. User input at various stages of the simulation and visualization resulted in steering events. Below, we list the steering events together with the system response and the effect of these events.

- E_1**:** This event occurs after 3.8 hours of execution. In this event the user decides to form a nest based on the visualization output and also

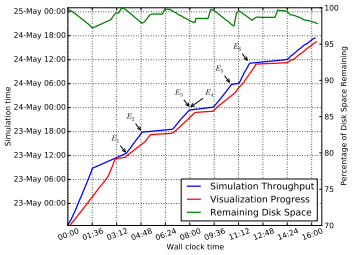


Figure 7: Simulation (blue) and Visualization (red) progress, and Disk Consumption (green) for *intra-country* configuration with computational steering. Initial WRF resolution = 24 km, MPR = 5. Events $E_1 - E_6$ affect the simulation throughput and the visualization as reflected in the graph.

provides the nest location. This causes the decision algorithm to reconsider the various system and application parameter values and compute the optimal number of processors and output interval. The formation of nest decreases the simulation rate and hence the slope decreases after the event.

- E_2**:** This event occurs after about 4.8 hours of execution. In this event the user decides to refine the simulation to a finer resolution of 18 km. To start at a finer resolution, WRF has to preprocess input data at that resolution. Hence we observe the flat region after E_2 corresponding to the time required for preprocessing. This time depends on the I/O bandwidth of the system. The framework then starts with the user-provided resolution of 18 km.

- E_3**,** E_4**:** This event occurs after about 8 hours of execution. E_3 denotes the event where the user requests for a simulation rate of 8 but the system denies owing to infeasibility. The maximum rate possible with an output interval of 30 at resolution of 18 km is 6 considering the time to solve and time to output in this experimental setup. This value can be estimated using equation (17). The system then tries to find a coarser resolution at which the user's desired rate is feasible as explained in Section 4. In this case, the system provides an alternate option of making the resolution 21 km. In this way, the system tries to satisfy one requirement of the user, while compromising another based on feasibility analysis. The user can then prioritize resolution over rate or the other way round. E_4 denotes the event where the user asks for a resolution of 21 km, as suggested by the system. As these events demonstrate, InSt takes a proactive approach towards user-driven computational steering. While it attempts to steer the simulations based on user inputs, it also analyzes the impact of the user inputs on the criticality

of the application, namely, the MPR desired for the simulations, and "advises" the user about possible violations of the "quality-of-service" due to his inputs, and provides him with suitable alternate options. Thus, InSt follows an effective reconciliation approach towards steering executions. Unlike existing work that mainly focuses on user-driven steering, this reconciliation of the user inputs and the criticality needs of the application is very essential for critical weather applications like cyclone tracking.

- E_5**:** This event occurs after about 10.6 hours of execution. In this event the user decides to increase the output interval to 21. As can be seen in the simulation curve, the slope slightly increases signifying an increase in the simulation rate. The simulation rate increases because the increased output interval causes the simulation process to spend lesser amount of time writing files to disk.

- E_6**:** This event occurs after about 11.8 hours of execution. This is where the user decides to refine the resolution for better visualization. Since finer resolution implies more time to solve a time step, it can be seen from the graph that the slope of the simulation curve after E_6 is lower compared to before.

Figure 7 shows that the available disk space is always above 95%. This is because of adjusting the number of processors by the decision algorithm to the correct value so that disk space is not a problem even when inputs are given by the user. Fluctuations in the disk curve can be seen at times corresponding to the events E_1 to E_6. Increase in disk space can be seen after the events and before restarting WRF because during this period, the transfer rate remains the same as there is almost no input to the disk.

5.5.2 Inter-country Steering Results

We also performed *inter-country* steering from the visualization site in India to the simulation site in NCSA, USA. Figure 8 shows the results obtained in the *inter-country* configuration with NCSA's Abe cluster in USA for simulations, and the visualization engine in IISc, India. The graph shows the various algorithmic steering events and user-driven steering events.

Initially, the simulations were started with a resolution of 18 km, and MPR of 3 on 96 processors with an output interval of 30 simulated minutes. Below, we list the algorithmic events (E_1 and E_4) and user-driven steering events (E_2 and E_3) that occurred during the 11 hours of execution. The

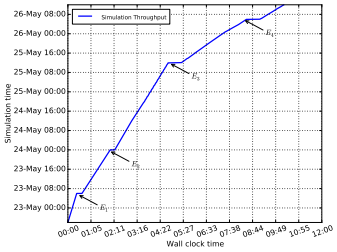


Figure 8: Simulation progress for *inter-country* configuration with computational steering. Initial WRF resolution = 18 km, MPR = 3. Both algorithmic and user-driven steering events ($E_1 - E_4$) affect the simulation throughput as reflected in the graph.

response to these events in terms of the simulation throughput and visualization progress is similar to the intra-country experiment.

- E_1**:** This event occurs after 30 minutes of execution. In this event, the decision algorithm computes the number of processors as 80. This change from 96 to 80 is because of the rapid disk space consumption due to the high simulation rate at coarser resolution.

- E_2**:** This event occurs after 2 hours of execution. In this event, the user requests for change in output interval to 60 simulated minutes.

- E_3**:** This event occurs after 5 hours of execution. In this event, the user requests for change in resolution from 18 km to 12 km for better simulation output.

- E_4**:** This event occurs after 8 hours of execution when finer resolution causes the simulation rate to decrease. The decision algorithm increases the number of processors from 80 to 112 in response to this event and maintains the minimum simulation rate.

6 Conclusions and Future Work

High-performance simulations, effective "on-the-fly" remote visualizations, and user-driven computational steering of simulations based on feedback to focus on important scientific phenomena are essential for efficient monitoring of critical weather events, and providing timely analysis. In this paper, we have described our integrated steering framework, InSt, that combines user-driven steering with automatic tuning of application parameters based on resource constraints and the criticality needs of the application to determine the final parameters for simulations. Our steering framework proactively analyzes the impact of user inputs on the criticality of the application, advises

the user on violations, guides with alternate options, and arrives at the final agreeable parameters. We have demonstrated the algorithmic and steering aspects of our framework with experiments involving intra and inter country steering. Results of these experiments demonstrate how the framework guarantees a minimum rate of simulation, continuous visualizations, and reconciliation between algorithm and user-driven steering.

In future, we plan to investigate research challenges related to steering across very-slow networks similar to our *inter-country* configuration. We also plan to expand our framework and research to include simultaneous steering of multiple simulations and multi-user steering in grid environments. We plan to apply our techniques for other critical applications and form a more generic framework.

References

[1] J. Michalakes, J. Hacker, R. Loft, M. O. McCracken, A. Snavely, N. J. Wright, T. E. Spelce, B. C. Gorda, and R. Walkup, "WRF Nature Run," in *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*.

[2] H. Yu, R. K. Sahoo, C. Howson, G. Almsi, J. G. Castaos, M. Gupta, J. E. Moreira, and J. J. Parker, "High Performance File I/O for The Blue Gene/L Supercomputer," in *Proceedings of the 12th International Symposium on High-Performance Computer Architecture*, 2006.

[3] S. Lang, P. H. Carns, R. Latham, R. B. Ross, K. Harms, and W. E. Allcock, "I/O performance challenges at leadership scale." in *SC '09: Proceedings of the 2009 ACM/IEEE conference on Supercomputing*, 2009.

[4] S. Parker and C. Johnson, "SCIRun: A Scientific Programming Environment for Computational Steering," in *SC '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, 1995.

[5] S. Parker, M. Miller, C. Hansen, and C. Johnson, "An integrated problem solving environment: the SCIRun computational steering system," *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, vol. 7, pp. 147–156, January 1998.

[6] R. Walker, P. Kenny, and J. Miao, "Exploratory simulation for astrophysics," *Proceedings of the SPIE, Volume 6495, Conference on Visualization and Data Analysis*, 2007.

[7] A. Modi, L. N. Long, and P. E. Plassmann, "Real-time visualization of wake-vortex simulations using computational steering and Beowulf clusters," in *VECPAR'02: Proceedings of the 5th International conference on High Performance Computing for Computational Science*, 2003.

[8] H. Wright, R. H. Crompton, S. Kharche, and P. Wenisch, "Steering and visualization: Enabling technologies for computational science," *Future Generation Computer Systems*, vol. 26, no. 3, pp. 506–513, 2010.

[9] H. Yu, K.-L. Ma, and J. Welling, "A Parallel Visualization Pipeline for Terascale Earthquake Simulations," in *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, 2004.

[10] T. Tu, H. Yu, L. Ramirez-Guzman, J. Bielak, O. Ghattas, K.-L. Ma, and D. O'Hallaron, "From Mesh Generation to Scientific Visualization: an End-to-End Approach to Parallel Supercomputing," in *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006.

[11] K.-L. Ma, C. Wang, H. Yu, and A. Tikhonova, "In Situ Processing and Visualization for Ultrascale Simulations," *Journal of Physics (Proceedings of SciDAC 2007 Conference)*, vol. 78, 2007.

[12] K.-L. Ma, "In Situ Visualization at Extreme Scale: Challenges and Opportunities," *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 14–19, 2009.

[13] H. Liu, L. Jiang, M. Parashar, and D. Silver, "Rule-based visualization in the Discover computational steering collaboratory," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 53–59, 2005.

[14] R. Ribler, J. Vetter, H. Simitci, and D. Reed, "Autopilot: adaptive control of distributed applications," in *Proceedings of the Seventh International Symposium on High Performance Distributed Computing*, July 1998.

[15] R. L. Ribler, H. Simitci, and D. A. Reed, "The Autopilot performance-directed adaptive control system," *Future Generation Computer Systems*, 2001.

[16] E. V. Zudilova, "Simulation-Visualization Complexes as Generic Exploration Environment," in *ICCS '01: Proceedings of the International Conference on Computational Science-Part II*, 2001, pp. 903–911.

[17] A. Shenfield, P. J. Fleming, and M. Alkarouri, "Computational steering of a multi-objective evolutionary algorithm for engineering design," *Engineering Applications of Artificial Intelligence*, 2007.

[18] G. I. James, G. A. Geist, I. James, A. Kohl, and P. M. Papadopoulos, "CUMULVS: Providing Fault-Tolerance, Visualization and Steering of Parallel Applications," *International Journal of High Performance Computing Applications*, vol. 11, pp. 224–236, 1996.

[19] W. Gu, J. Vetter, and K. Schwan, "An annotated bibliography of interactive program steering," *ACM SIGPLAN Notices*, 1994.

[20] J. S. Vetter and K. Schwan, "High Performance Computational Steering of Physical Simulations," in *IPPS '97: Proceedings of the 11th International Symposium on Parallel Processing*, 1997, p. 128.

[21] J. A. Insley, M. E. Papka, S. Dong, G. Karniadakis, and N. T. Karonis, "Runtime Visualization of the Human Arterial Tree," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, 2007.

[22] B. Huang1, D. Xiong, and H. Li, "An Integrated Approach to Real-time Environmental Simulation and Visualization," *Journal of Environmental Informatics*, vol. 3, no. 1, pp. 42–50, March 2004.

[23] J. A. Kohl, T. Wilde, and D. E. Bernholdt, "Cumulvs: Interacting with High-Performance Scientific Simulations, for Visualization, Steering and Fault Tolerance," *International Journal of High Performance Computing Applications*, 2006.

[24] I. Dooley and L. V. Kale, "Control Points for Adaptive Parallel Performance Tuning," *PPL Technical Report*, 2008.

[25] C. Tapus, I.-H. Chung, and J. Hollingsworth, "Active Harmony: Towards Automated Performance Tuning," in *SC '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002.

[26] J. Brooke, T. Eickermann, and U. Woessner, "Application Steering in a Collaborative Environment," in *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, 2003.

[27] Q. Wu, M. Zhu, Y. Gu, and N. S. V. Rao, "System Design and Algorithmic Development for Computational Steering in Distributed Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, 2010.

[28] Y. Jean, T. Kindler, W. Ribarsky, W. Gu, G. Eisenhauer, K. Schwan, and F. Alyea, "Case study: an integrated approach for steering, visualization, and analysis of atmospheric simulations," in *Proceedings of IEEE Visualization '95*.

[29] W. Gu, G. Eisenhauer, E. Kraemer, K. Schwan, J. Stasko, J. Vetter, and N. Mallavarupu, "Falcon: on-line monitoring and steering of large-scale parallel programs," *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, pp. 422–429, 1995.

[30] "GNU Linear Programming Kit," http://www.gnu.org/software/glpk.

[31] "VisIt Visualization Tool," http://www.llnl.gov/visit.

[32] "National Knowledge Network, Department of Information Technology, Government of India," http://www.mit.gov.in/content/national-knowledge-network.

[33] "Cyclone Aila," http://en.wikipedia.org/wiki/Cyclone_Aila.

[34] 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 *In proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology*, October 2004.

[35] W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, D. Barker, W. Wang, and J. G. Powers, "A Description of the Advanced Research WRF version 2," *NCAR Technical Note TN-468*, 2005.

[36] "UCAR CISL Research Data Archive," http://dss.ucar.edu.

[37] R. Rew and G. Davis, "The Unidata netCDF: Software for Scientific Data Access," in *6th International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, California, American Meteorology Society*, 1990.

[38] "LABFit Curve Fitting Software," http://www.angelfire.com/rnb/labfit.