Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 116–125

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

InSt: An Integrated Steering Framework for Critical Weather Applications

Preeti Malakar[a,1], Vijay Natarajan[a,b], Sathish S. Vadhiyar[b]

[a] *Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India*
[b] *Supercomputer Education and Research Center, Indian Institute of Science, Bangalore, India*

Abstract

Online remote visualization and steering of critical weather applications like cyclone tracking are essential for effective and timely analysis by geographically distributed climate science community. 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 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 InSt for simulations, online remote visualization, and analysis for critical weather applications. InSt provides the user control over various application parameters including region of interest, resolution of simulation, and frequency of data for visualization. Unlike existing efforts, our framework considers both the steering inputs and the criticality of the application, namely, the minimum progress rate needed for the application, and various resource constraints including storage space and network bandwidth to decide the best possible parameter values for simulations and visualization.

Keywords: computational steering, critical weather application, remote visualization, adaptation, parallel simulation

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 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 and feedback control using computational steering can enable geographically distributed climate scientists to share vital information, perform collaborative analysis, and provide joint guidance on critical weather events.

High-performance simulations and simultaneous remote visualization involve the use of large stable storage for storing the weather data and networks for transferring data from the stable storage to the remote visualization site.

Email addresses: preeti@csa.iisc.ernet.in (Preeti Malakar), vijayn@csa.iisc.ernet.in (Vijay Natarajan), vss@serc.iisc.ernet.in (Sathish S. Vadhiyar)

[1] Corresponding author

Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka

doi:10.1016/j.procs.2011.04.013

However, constraints on the capacity of the stable storage and the network can limit the effectiveness of online remote visualization of critical weather events. Higher simulation and I/O rates, lower network bandwidth and lower storage capacities will lead to faster accumulation of data at the simulation site, and can eventually lead to stalling of simulations due to unavailability of storage.

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[1]. An important requirement for a steering framework that controls high-performance simulations of critical weather events is that it needs to consider 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 have 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 simulation 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. For example, the steering framework should override a very high output frequency specified by the user if it determines that the high value can lead to unavailability of storage and can severely compromise the MPR or criticality needs of the application.

In this work, 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, simulation rate and frequency of data for visualization and allows scientists to specify region of interest. However InSt also considers the criticality of the application, namely, the MPR needed for the application, and various resource constraints like storage space and network bandwidth 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. InSt effectively combines computational steering by the user/scientist with the algorithmic steering performed by the runtime system of the framework. Thus InSt 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[1, 2, 3, 4]. Using our framework, we performed cross-continent steering with the steering and visualization performed in India and simulations conducted on a TeraGrid cluster in NCSA, USA. We show that InSt performs reconciliation between algorithmic and user-driven computational steering for a 3-day weather simulation while providing quality of service with respect to simulation rate and continuous visualization.

§2 describes related work in computational steering. §3 presents our integrated steering framework. §4 explains the reconciliation between the algorithmic steering and user-driven steering. §5 presents our experiments and results involving varying network bandwidths. §6 gives conclusions and enumerates our plans for future work.

2. Related Work

Computational steering has been extensively studied over the past several years and a variety of steering systems have emerged[1, 5, 6, 2]. A taxonomy of different kinds of steering and steering systems and tools can be found in [1, 7]. *Exploratory steering* allows the scientists to control the execution of long-running, resource-intensive applications for application exploration. *Performance steering* allows scientists to change application parameters to improve application performance. *Algorithmic steering* uses an algorithm to decide application parameters to improve system and application performance[8].

Computational steering has been applied to different kinds of applications including molecular dynamics simulation, biological applications, astrophysics, computational fluid dynamics and atmospheric simulations[3, 9, 4, 10, 11, 12]. These frameworks were mainly developed for exploratory steering in order to change simulation parameters interactively and visualizing the simulation output with the new parameters.

There are some steering systems which enable performance steering[5, 6, 13]. CUMULVS[5] provides the user with a viewer and steering interface to modify the application's computational parameters and improving application

performance. Autopilot[6] dynamically adapts to changing application resource demands and system resource availability by using fuzzy logic to select and change policy parameters. In [13], the authors have developed runtime tuning algorithms to intelligently set application parameters like read-ahead parameter to tune application performance. Our previous work[14] dealt with dynamic adaptation of application and system parameters based on resource constraints for critical weather applications. The work considered various resource constraints including available storage space, network bandwidth and I/O bandwidth to perform continuous visualization with maximum simulation rate.

Brooke et al.[15] consider simultaneous online visualization and steering of application by geographically distributed teams. Jean et al.[9] have developed an integrated approach for online monitoring, steering and visualization of atmospheric simulations using Falcon[16]. To our knowledge, this is the only work on steering of weather applications prior to ours. Their work simulates physical and chemical interactions in the ocean and atmosphere. Their steering interface lets the user modify and evaluate different application parameters.

Our work differs from the above efforts since 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. To the best of our knowledge, ours is the first work that considers remote computational steering of critical applications with performance requirements.

3. InSt Steering Framework

We have developed an integrated steering framework, InSt, that performs automatic tuning as well as user-driven steering. As shown in Figure 1, it consists of the following components: *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 simulation output/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. The following subsections describe the primary components in detail.

3.1. User Interface, SimDaemon and VisDaemon

The user gives input through the user interface at the visualization site as shown in Figure 1. In particular, the 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.

3.2. Application Manager

The application manager is the primary component of InSt and acts as the bridge between algorithmic steering and user-driven steering. The manager periodically monitors the free disk space and available network bandwidth. For automatic/algorithmic steering, it periodically invokes the decision algorithm, explained in § 3.4 for obtaining the number of processors for simulation and the frequency of output to be generated by the simulation, given the resource constraints and simulation parameters. For user-driven steering, the application manager asynchronously receives the user inputs, including the upper bound for frequency of output and resolution of simulation. The manager checks the feasibility of running the simulation 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 output of the decision algorithm to the application configuration file, and starts or stops-and-restarts the simulation with the parameters. More details on the reconciliation of the algorithmic and user-driven steering are given in § 4.

3.3. Simulation Process

The simulation process is the weather application that simulates the weather events. It periodically reads the simulation parameters from the application configuration file. If the parameters in the configuration file are different from the parameters used for current execution, the simulation process stops and restarts execution with the new parameters. It 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 is available again.

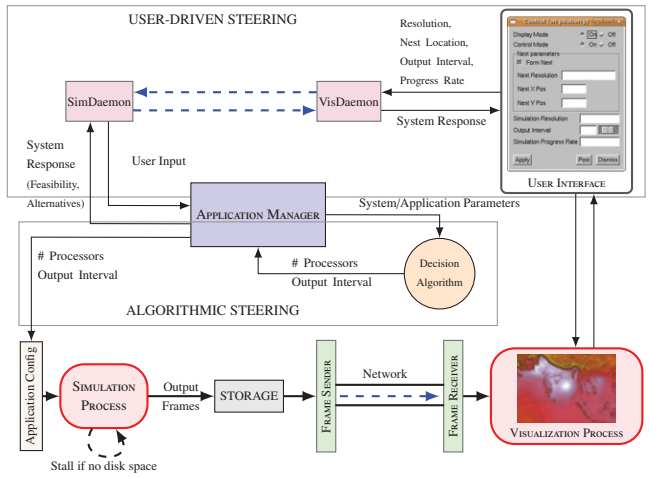


Figure 1: InSt: Integrated Steering Framework

3.4. Decision Algorithm

The decision algorithm determines the number of processors and the frequency of output of weather data for a given resolution of simulation, the network bandwidth between simulation and visualization sites, the available free disk space at the simulation site and the user-specified minimum progress rate of simulation. The algorithm also takes as input the execution times for different number of processors and simulation resolutions. It also considers the 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 wants 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. However, these objectives are contradictory. Increasing the frequency of output can decrease the rate of simulation due to increase in I/O and can also lead to rapid consumption of storage. Unlike traditional scheduling algorithms that minimize execution times, our algorithm may sometimes have to "slow down" the simulations, since faster simulations can lead to faster consumption of storage if the network bandwidth is low. Thus we can think of our problem as an optimization problem that tries to maximize the simulation rate within the constraints related to continuous visualization, acceptable frequency of output, MPR of simulations, I/O bandwidth, disk space and network speed. We have formulated this problem as a linear programming model with the objective of obtaining the number of processors and the frequency of output within the constraints related to minimum stalling at the visualization end and availability of disk space for storing the simulation output. We have added an important constraint, the *rate constraint* involving the MPR of simulations, for considering the criticality of the application and ensuring quality of service to the climate scientist for online visualization. The objective function and the constraints are described below.

The objective function is to maximize simulation throughput or *minimize* t, where t is the execution time to solve a time step. The parameters used in the formulation are listed below. $\mathcal{S}$, $\mathcal{F}$, $\mathcal{T}$ and t are the decision variables.

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

Rate Constraint: Critical weather applications require a steady rate of progress in simulations. This rate is represented by the ratio of the time simulated by the application (simulation time) to the wall-clock time taken by the application for the simulation (wall-clock time). This ratio has to be greater than 1 for critical applications like cyclone tracking where scientists require advance information to provide timely guidance to decision makers. The simulation rate depends on the computation speed and the output frequency. The higher the output frequency, the higher will be the number of I/O writes to the disk and hence lesser will be the simulation rate. Also, the lower the I/O bandwidth,

more significant will be the impact of high output frequency on the simulation rate. In most cases, scientists may want to specify a minimum acceptable limit for this ratio, denoted by MPR. Equation (1) specifies this constraint. Let ts denote the integration time step associated with a simulation resolution. It is the amount of time simulated in one time step. In an interval I, if S frames are solved and F frames are produced, then simulation time is $ts \cdot S$ and wall-clock time is the sum of solve time and I/O time i.e. $t \cdot S + T_{IO} \cdot F$ as shown in Equation (2). This can be rewritten as Equation (3) which forms the rate constraint equation.

$$\frac{Simulation\ time}{Wall-clock\ time} \geq MPR \quad (1) \qquad \Longrightarrow \frac{(ts \cdot S)}{(t \cdot S + T_{IO} \cdot F)} \geq MPR \quad (2) \qquad \Longrightarrow \frac{ts}{(t + T_{IO} \cdot F/S)} \geq MPR \quad (3)$$

Other Constraints: For continuous visualization, the time to solve $\mathcal{S}$ frames and produce $\mathcal{F}$ frames in an interval $\mathcal{I}$ should be less than the time to transfer $\mathcal{T}$ frames in that interval since the next set of $\mathcal{F}$ frames should be ready for transfer by the time the current $\mathcal{T}$ frames are transferred. This forms the time constraint. The I/O bandwidth, the computation speed and the output size affect the rate of input to the disk and the network bandwidth affects the rate of output from the disk. Using these, we form the disk constraint. These constraints are detailed in [17, 14].

Bounds: Depending on the total number of processors, t has a lower bound $\mathcal{T}_{LB}$, as specified in Equation (4). We can specify upper bound OI_{UB} for the output interval based on the minimum frequency of visualization of weather events specified by climate scientists. Output interval also has a lower bound OI_{LB} based on the limitations of the simulation application. These bounds are specified in Equation (5). In InSt, OI_{UB} can be specified or steered by the user.

$$t \geq \mathcal{T}_{LB} \quad (4) \qquad\qquad OI_{LB} \leq OI \leq OI_{UB} \quad (5)$$

We linearize the non-linear constraint equations by change of variables. We solved our optimization problem using GLPK (GNU Linear Programming Kit)[18]. We then obtain the number of processors from t and output interval from F/S and ts. This decision algorithm is invoked periodically 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 which is used to reschedule the simulations with the new configuration.

Although we have used a weather forecast model as the simulation process in this work, with very minimal changes, we can use InSt to steer other applications as well.

4. Reconciling User-driven and Algorithmic Steering

Initially, the application manager specifies default values for simulation resolution and MPR. The manager then determines the frequency of output using the decision algorithm based on resource constraints for the given resolution of simulation. The simulation is 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, the simulation process restarts and continues with a new configuration involving the nest and the new resolution.

When a user specifies an output interval and/or MPR value, the InSt framework considers the criticality of the application, proactively checks the feasibility of simulating with these inputs, and guides the user with possible alternative values for ensuring continuous progress of simulation 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 (3), i.e., if the simulation 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 OI and MPR is given by Equation (6) which is derived using Equation (3) and the relation $\mathcal{F}/\mathcal{S} = ts/OI$ described in [17, 14].

$$MPR \leq \frac{ts}{[t + T_{IO} \cdot (ts/OI)]} \quad (6)$$

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, then from Equation (6), 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 users' requested value for OI. Also, it can be seen from Equation (6) 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 (6) using the current value of *MPR* used by the application manager. If the user-specified *OI* is less than this lower bound, then InSt does not incorporate the request. In this case, InSt informs the user of the lowest feasible *OI*. If the user-specified *OI* is greater than the lower bound, then this value forms the upper bound for *OI* in the decision algorithm.

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 simulation by calculating an upper bound, MPR_{max}, feasible for the resolution and comparing *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 (6) and obtain MPR_{max} as the ratio of ts and T_{LB}. If *uMPR*, is greater than 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.

If the user wants to place a nest in the parent domain, the WRF simulation is restarted with the nest. If the user specifies both *uMPR* and *uOI* then InSt checks for the feasibility of the combination. If its feasible, then InSt incorporates the *uMPR* otherwise the framework checks for feasibility of *uMPR* at a coarser resolution, since the rate of simulation increases with coarser resolution. If the user specifies only *uOI*, then InSt checks if *uOI* satisfies the current *MPR*. If feasible, simulation is updated otherwise the user is informed. Thus, InSt 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.

5. Experiments and Results

5.1. Resource Configuration

For all our experiments, visualization was performed on a graphics workstation in Indian Institute of Science (IISc) with a dual quad-core Intel® Xeon® E5405 and an NVIDIA graphics card GeForce 8800 GTX. We used hardware acceleration feature of VisIt[19] 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 the 3.16 GHz quad-core Intel® Xeon® X5460 320-core cluster, *gg-blr*, in the Centre for Development of Advanced Computing (C-DAC), Bangalore, India. We used the Gigabit interconnect for this configuration. The transfer between simulation and visualization site was carried out on the National Knowledge Network[20] with the maximum bandwidth of 1 Gbps and average available bandwidth of 40 Mbps. We used a maximum of 96 cores and a maximum of 150 GB disk space for this configuration. In the *inter-country* configuration, the WRF simulations were conducted on a quad-core Intel 64 (Clovertown) PowerEdge 1955 9600-core cluster, *Abe*, in National Center for Supercomputing Applications (NCSA), Illinois, USA. We used the Infiniband interconnect for this simulation. We used a maximum of 128 cores and a maximum disk space of 700 GB. The average available bandwidth was 8 Mbps. Our framework can be used for steering applications running on more than 128 cores and is not dependant on the maximum number of cores used.

5.2. Weather Model and Cyclone Tracking

For our work, we used a mesoscale numerical weather forecast model, WRF (Weather Research and Forecasting Model)[21, 22] for simulating weather events. 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. WRF outputs data in the form of NetCDF[23] files. Each NetCDF file contains output of a number of simulation time steps. We have applied InSt for an important critical weather application, namely, large-scale and long-range tracking of cyclones. In our experiments, we used InSt for tracking a tropical cyclone, *Aila* which was formed in the Northern Indian Ocean during May 2009[24]. 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. The nesting ratio, i.e., the ratio of the resolution of the nest to that of the parent domain, was set to 1:3. 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 from 30°E - 150°E and 10°S - 40°N and for a longer period

of time of 3 days and 18 hours. The 6-hourly 1-degree FNL analysis GRIB meteorological input data for our model domain was obtained from CISL Research Data Archive[25].

5.3. Framework Implementation

The modifications to the WRF application for our work are minimal. The main modification is to stop WRF 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 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 curve fitting tools[26] were used 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.

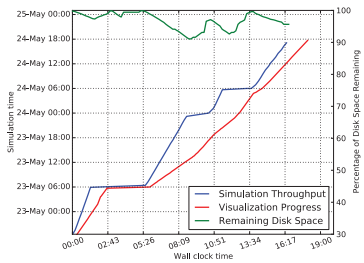
For faster I/O, we used WRF's split NetCDF approach, where each processor outputs its own data. We have developed a utility to merge these split NetCDF files at the visualization site. We have also developed a plug-in for VisIt to *directly read* the WRF NetCDF output files, eliminating the cost of post-processing 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 visualized the output using volume rendering, vector plots employing oriented glyphs, pseudocolor and contour plots. 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 monitors the available disk space using the UNIX command *df*. The average observed bandwidth between the simulation and visualization sites is obtained from 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 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 InSt are specific to our experiment settings and WRF simulations, the general principles of our steering framework are generic and applicable to other applications.

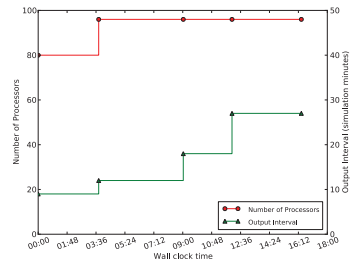
5.4. Automatic Tuning Results

We first demonstrate the efficiency of the decision algorithm and the automatic tuning of parameters in the absence of user inputs. For experiments in this section, InSt automatically determines the WRF nest location and changes the resolution of simulations based on pressure values in the weather data. When the pressure drops below 995 hPa, InSt spawns a nest, centered at the location of lowest pressure in the parent domain. We also use a configuration file that specifies different simulation resolutions 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, InSt changes the resolution multiple times to obtain a better simulation result from the model.

For the *intra-country* configuration, 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. Figure 2(a) shows the results for the *intra-country* experimental setup. 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 available storage space. For the simulation and visualization curves, the left y-axis shows the corresponding simulation times of the frames. For the disk availability curve, the right y-axis shows the percentage of remaining disk space. Figure 2(a) shows that the lag between visualization and simulation is minimal resulting in a truly online visualization. This is primarily because 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 needs input data at that 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 gg-blr 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. The remaining disk space is always above 90% because disk was freed at a high rate due to high network bandwidth.

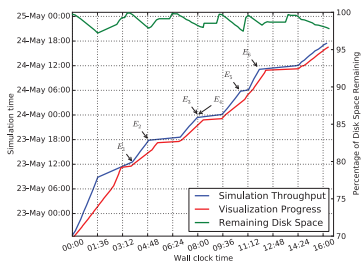


(a) Simulation (blue) and Visualization (red) progress, and Disk Consumption (green). Frames are visualized with a minimal time lag due to the high network bandwidth.

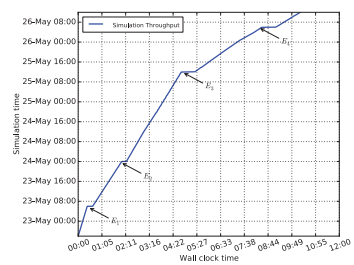


(b) Adaptivity of the framework showing variation in the number of processors (red, left y-axis) and output interval (green, right y-axis).

Figure 2: Simulation and Visualization progress, Disk usage and Adaptivity for algorithmic steering in *intra-country* configuration.



(a) Simulation progress (blue), Visualization progress (red) and Disk Consumption (green) for *intra-country* configuration with computational steering.



(b) Simulation progress for *inter-country* configuration with computational steering. Both algorithmic and user-driven steering events ($E_1 - E_4$) affect the simulation throughput.

Figure 3: Effects of algorithmic and user-driven steering events in *intra-country* and *inter-country* configurations.

Figure 2(b) shows the number of processors and output interval for the simulations automatically determined by InSt at different stages of execution for the *intra-country* configuration. Initially, InSt chooses an initial value of 9 and 80 for output interval and the number of processors respectively. During the execution, the number of processors and output interval change a few times when one or more parameters change in the constraint equations as explained in § 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 I/O time also change. So, the decision algorithm re-evaluates the correct number of processors and the best output interval for the current parameters. At finer resolution, the output size and hence the I/O time increases. For this experiment, MPR was chosen as 5. The decision algorithm increases the output interval to satisfy the MPR and to prevent disk overflow due to frequent I/O. Thus InSt algorithmically steers and adaptively changes the execution and application parameters based on the application and resource configurations and simulation rates.

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 algorithmic steering. For these experiments, the simulations are started at a particular resolution. However, unlike in the automatic tuning experiments, only the user can place the nest and change the simulation resolutions.

5.5.1. Intra-country Steering

Figure 3(a) 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 simulation was 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 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. At E_3, 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. This can be estimated using Equation (6). 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 vice versa. At event E_4, 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 3(a) 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 space 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 whereas 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 USA. Figure 3(b) 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. 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 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. The user requests for change in output interval to 60 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 and timely analysis of critical weather events. 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. InSt 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 InSt with experiments involving intra and inter country steering. Results of these experiments demonstrate that the framework guarantees a minimum rate of simulation, continuous visualization and reconciliation between algorithmic 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 plan to expand InSt to tackle the challenges of simultaneous steering of multiple simulations and multi-user steering in grid environments. Furthermore, we intend to apply our techniques for other critical applications and form a more generic framework.

References

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

[2] H. Wright, R. H. Crompton, S. Kharche, P. Wenisch, Steering and visualization: Enabling technologies for computational science, in: Future Generation Computer Systems, Vol. 26, 2010, pp. 506–513.

[3] J. A. Insley, M. E. Papka, S. Dong, G. Karniadakis, N. T. Karonis, Runtime Visualization of the Human Arterial Tree, in: IEEE Transactions on Visualization and Computer Graphics, 2007.

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

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

[6] R. L. Ribler, et al., The Autopilot performance-directed adaptive control system, in: Future Generation Computer Systems, 2001.

[7] W. Gu, J. Vetter, K. Schwan, An annotated bibliography of interactive program steering, in: ACM SIGPLAN Notices, 1994.

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

[9] Y. Jean, T. Kindler, W. Ribarsky, W. Gu, G. Eisenhauer, K. Schwan, F. Alyea, Case study: an integrated approach for steering, visualization, and analysis of atmospheric simulations, in: Proceedings of IEEE Visualization, 1995.

[10] C. R. Johnson, S. G. Parker, A computational steering model applied to problems in medicine, in: SC '94: Proceedings of the 1994 ACM/IEEE conference on Supercomputing, 1994.

[11] B. Huang1, D. Xiong, H. Li, An Integrated Approach to Real-time Environmental Simulation and Visualization, in: Journal of Environmental Informatics, Vol. 3, 2004, pp. 42–50.

[12] A. Modi, L. N. Long, 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, 2002.

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

[14] P. Malakar, V. Natarajan, S. Vadhiyar, An Adaptive Framework for Simulation and Online Remote Visualization of Critical Climate Applications in Resource-constrained Environments, in: SC '10: Proceedings of the 2010 ACM/IEEE conference on Supercomputing, 2010.

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

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

[17] P. Malakar, V. Natarajan, S. Vadhiyar, A framework for online visualization and simulation of critical weather applications, Tech. rep., Department of Computer Science and Automation, Indian Institute of Science, `http://www.csa.iisc.ernet.in/TR/2011/1` (2011).

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

[19] VisIt Visualization Tool, `http://www.llnl.gov/visit`.

[20] National Knowledge Network, Department of IT, Government of India, `http://www.mit.gov.in/content/national-knowledge-netwo`

[21] J. Michalakes, et al., 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, 2004.

[22] J. Michalakes, et al., WRF Nature Run, in: SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, 2007.

[23] R. Rew, 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.

[24] Cyclone Aila, `http://en.wikipedia.org/wiki/Cyclone_Aila`.

[25] UCAR CISL Research Data Archive, `http://dss.ucar.edu`.

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