An Adaptive Framework for Simulation and Online Remote Visualization of Critical Climate Applications in Resource-constrained Environments

Preeti Malakar*, Vijay Natarajan*†, Sathish S. Vadhiyar†
*Department of Computer Science and Automation
†Supercomputer Education and Research Centre
Indian Institute of Science, Bangalore, India
preeti@csa.iisc.ernet.in, vijayn@csa.iisc.ernet.in, vss@serc.iisc.ernet.in

Abstract—Critical climate applications like cyclone tracking and earthquake modeling require high-performance simulations and online visualization simultaneously performed with the simulations for timely analysis. Remote visualization of critical climate events enables joint analysis by geographically distributed climate science community. However, resource constraints including limited storage and slow networks can limit the effectiveness of such online visualization. In this work, we have developed an adaptive framework that simultaneously performs numerical simulations and online remote visualization of critical climate applications in resource-constrained environments. Our framework considers both application and resource dynamics to adapt various application and resource parameters including simulation resolutions, resource configurations and amount of data for visualization. We have developed two algorithms for processor allocation for simulations and the frequency of data for visualization. We show that our optimization method is able to provide about 30% higher simulation rate and consumes about 25-50% lesser storage space than the greedy approach.

Index Terms—climate simulation; remote visualization; processor allocation; adaptation;

I. INTRODUCTION

Critical climate applications like cyclone or hurricane tracking, earthquake modeling and tsunami predictions require high-performance and high-fidelity simulations to obtain real-time forecasts and high-resolution visualization by the climate scientists for subsequent analysis and scientific discovery. For timely analysis and rapid response, these applications require simultaneous and online/"on-the-fly" visualization simultaneously performed with the simulations. This will enable the scientists to provide real-time guidance to policy and decision makers, and feedback control for refining the simulations. 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 climate events, and hence can help harness the expertise of a large climate science community.

Such high performance simulations and simultaneous visualization involve the use of large stable storage or disk for storing the climate data and networks for shipment of the
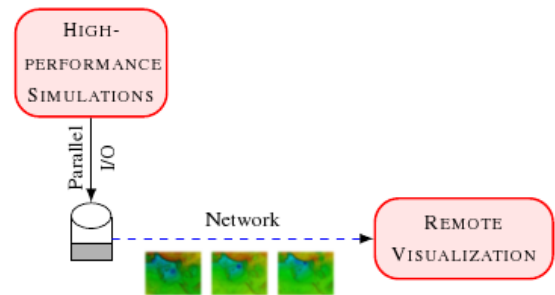

Fig. 1: Illustration of simultaneous simulations and remote visualization using stable storage

data from the stable storage to the remote visualization site as shown in Figure 1. However, constraints on the size and capacity of the stable storage and the network can limit the effectiveness of online and simultaneous remote visualization of critical climate events. In this work, we assume that the data that is transferred to the visualization site is removed from the simulation site thereby increasing the available free disk space at the simulation site.

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. The network bandwidth between the simulation and visualization site impacts the rate at which data is moved out from the simulation site and hence determines the amount of remaining disk space available for simulation output. The continuous development of high resolution simulation models for fine-grained analysis also increases the climate data volume and hence exacerbates the

TABLE I: Illustration of Disk Space Limitation. Climate simulation of grid size 4486x4486 points, 10 KM resolution, execution on 16,384 cores with 1.2 seconds of execution time per time step, and I/O bandwidth of about 5 GBps

Disk Space	Network Bandwidth	Time when storage becomes full
5 TB	1 Gbps	25 minutes
	10 Gbps	36 minutes
100 TB	1 Gbps	8 hours
	10 Gbps	12 hours
300 TB	1 Gbps	24.5 hours
	10 Gbps	36 hours
500 TB	1 Gbps	41 hours
	10 Gbps	60 hours

problem of storage space limitation for climate simulations. Eventual unavailability of storage for simulations, due to the disk becoming completely full, can result in either the stalling of the simulations or loss of visualization of critical climate events.

The problem is illustrated in Table I that shows the estimated time when the stable storage becomes unavailable for climate simulations (last column) for a climate simulation of grid size 4486 x 4486 points, 10 KM resolution resulting in about 31 GB of output per frame, execution time of 1.2 seconds for a simulation time step on 16,384 processor cores, I/O bandwidth of about 5 GBps, and for various total disk spaces and network bandwidths. The values for simulation grid sizes and resolutions, parallel I/O bandwidths and execution time on about 16,000 cores are reported and projected in recent research efforts [1]–[3]. We find that even the presence of large disk spaces, and fast networks can result in the storage becoming unavailable within few minutes to hours of high-resolution climate simulations that are envisaged to execute for few days to weeks on large-scale machines. This in turn hampers effective remote visualization of critical climate events. Hence it is highly essential to adaptively use the processor space and adjust the frequency of output based on the application and resource dynamics. Such a dynamic solution will ensure that the disk space is always available to store the output of the simulation and the climate scientists are able to constantly monitor progress of the simulation.

In this paper, we have developed an adaptive framework that simultaneously performs numerical simulations and online continuous remote visualization of critical climate applications in resource-constrained environments. The objective of our framework is to enable continuous progress in simulation and maximize temporal resolution in visualization considering the limitations in storage and network capacities. We define temporal resolution as the frequency at which successive frames are visualized. High temporal resolution would mean that more number of successively produced frames are visualized. Our framework considers both application and resource dynamics

including the intensity of climate events, available disk space and the network bandwidth to adapt various application and resource parameters including simulation resolutions, selecting the number of processors for simulations, and the frequency of data output for visualization. We have developed two algorithms for processor allocation and the frequency of data output for visualization. The first algorithm is a greedy strategy that attempts to maximize the simulation rate and frequency of output while the second algorithm is based on linear optimization that attempts to provide steady-state simulation and visualization rate. We have applied our framework for large-scale and long-range tracking of cyclones. We conducted experiments with our framework using our algorithms for three experiment settings corresponding to inter-department, intra-country and cross-continent visualizations. We show that our optimization method is able to provide about 30% higher simulation rate completing the entire simulations for all network configurations, consumes about 25-50% lesser storage space completely avoiding the disk overflow problem and the resulting stalling of simulations, and provides higher and more consistent rate of visualization than the greedy approach.

Section II describes related work in large-scale simulations and visualizations of scientific data. Section III presents our adaptive framework including the components and interactions. Section IV explains our adaptive algorithms for deciding processor allocation and output frequency. Section V presents our experiments involving different network bandwidths and results including simulation rates. Section VI gives conclusions and enumerates our future efforts.

II. 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 [4]. However, these strategies cannot be applied for online visualization, which is very important for critical climate applications.

Tu et al. [5] and Ma et. al. [6], [7] 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.

Another effort from NASA [8] uses shared memory as a medium of communication between simulation and visu-

alization. They have conducted weather simulation of the 2005 Hurricane season using a global climate model. Though this approach decouples the set of processors on which each component runs, it requires large amount of shared memory. In their work, 1TB of shared memory was used. This approach is clearly not suitable for long running high-resolution climate simulation dealing with large amount of data due to limitation on the sizes of shared memory.

All the above efforts consider critical climate applications in tightly-coupled environments. Ours is the first work that adaptively performs simultaneous simulations and online remote visualization for these applications.

III. ADAPTIVE INTEGRATED FRAMEWORK

Critical climate application simulations require immediate attention on the occurrence of critical events. Hence executions of these applications require efficient processor allocation and a robust disk-space management because of the sheer amount of data produced by the simulations. The absence of such a middleware can lead to problems including disk overflow, stalling of simulation, and low temporal resolution.

We have developed an adaptive framework that performs efficient processor allocation and robust disk-space management to handle the large amount of data produced by the simulations to enable continuous online visualization at the remote visualization engine. Our framework, shown in Figure 2, consists of the following components to perform coordinated simulations and online remote visualizations: *an application manager* that determines the application configuration for climate simulations based on resource characteristics, *a job handler* that coordinates the execution of climate simulations, *a simulation process* that performs climate simulations with different application configurations, *frame sender and receiver daemons* that deal with transport of frames from simulation to visualization sites, and *a visualization process* for visualization of the frames. For our work, we use a mesoscale numerical weather forecast model, WRF (Weather Research and Forecasting Model) [9], [10] in the simulation process for simulating climate events. The following subsections describe in detail the components and their interactions.

A. Application Manager

The application manager is the primary component that makes our framework adaptive to resource configuration changes. It invokes a decision algorithm periodically or at specified times. The decision algorithm considers as input the bandwidth of the network between the climate simulation and visualization sites, the available free disk space, and the resolutions of climate simulations. 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 decision algorithm then determines the number of processors for execution of climate simulations

and the frequency of output of climate data for continuous visualization. The application manager stores these parameters to an application configuration file. The application manager also notifies the other components in our framework if the available free disk space becomes significantly low by setting a *CRITICAL* flag in the application configuration file.

The efficiency of the decision algorithm used in the application manager impacts the rate of simulations and online visualization in our framework. We have developed two decision algorithms for the application manager. These algorithms are described in Section IV.

B. Job Handler and Simulation Process

The job handler component is responsible for scheduling the WRF climate simulation application with the application configuration determined by the application manager. The WRF climate simulation process is executed on a certain number of processors with a simulation resolution and frequency of output of the climate data specified in the application configuration. The job handler starts, stops and restarts the simulation process whenever the application configuration changes. The simulation process simulates the climate over a specified period of time and produces output for visualization.

The simulation process continuously simulates the climate events across time steps and outputs climate data to disks as long as the available disk space is sufficient for accommodating the output. The WRF simulation process also periodically reads the application configuration file written by the application manager. If the available free disk space is significantly low, the application manager sets the *CRITICAL* flag. In this case, the simulation process *stalls* execution, and periodically checks the application configuration file. When the free disk space becomes sufficient again, the application manager resets the *CRITICAL* flag, and WRF continues execution. When the application configuration specified in the configuration file changes from the *current configuration* used for the WRF execution, the WRF process stops. The job handler then restarts WRF using WRF checkpointed data with the new application configuration and continues execution.

In our framework, the WRF simulation process is made to stall if the available free disk space is very low. This is a reasonable strategy since our framework is primarily intended for online and continuous remote visualization. Hence continuing the simulation without generating the output will result in large time "gaps" in the visualization of climate events.

The modifications to the WRF climate application for our application are minimal. Whenever WRF finds the values of its certain variables drop below a certain threshold, it stops and the job handler reschedules it using a new configuration input from the application manager. WRF also stops and is rescheduled on different number of processors when job handler signals change in the number of processors. WRF also changes the frequency of output whenever it receives a signal from the job handler.
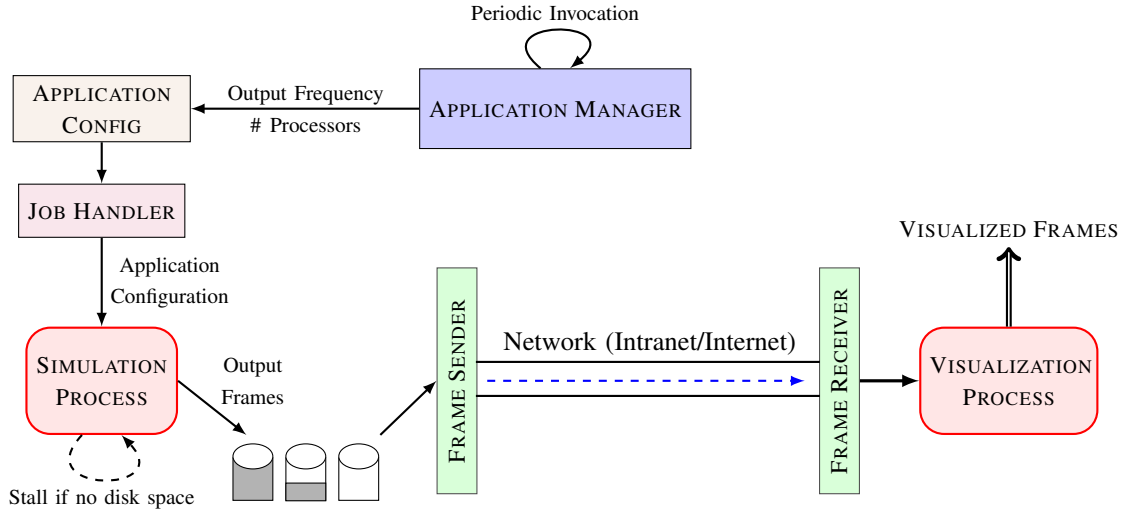
Fig. 2: Adaptive integration framework

C. Frame Sender and Receiver, and Visualization Process

Critical climate applications require continuous visualization of the simulated output. The frame sender daemon continuously checks for the availability of climate data output frames and sends the available frames over the network to the remote visualization site. The frame receiver daemon at the remote visualization site receives the frames and invokes the visualization process for visualization of the frames.

IV. DECISION ALGORITHMS FOR APPLICATION MANAGER

The decision algorithm invoked by the application manager determines
  1) the number of processors, and
  2) the frequency of output of climate data
for execution of climate simulations for a given
  1) resolution of simulation,
  2) the bandwidth of the network connecting the simulation and visualization sites, and
  3) the available free disk space at the simulation site.
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. Visualization of maximum number of output frames can be achieved by increasing the frequency of output of climate data by simulations but this can increase the time for I/O and hence decrease the rate of simulations. Increasing the frequency of output can also lead to rapid accumulation of output data and hence rapid decrease in the available free disk space, eventually stalling the simulations. Decreasing the frequency of output can increase the simulation rate, but will result in visualization of fewer frames.

Faster networks with high bandwidths result in faster transfer of output frames from the simulation to visualization sites and hence results in large-scale freeing of disk space at the simulation site. Also, unlike traditional scheduling, executing the simulations on large number of processors corresponding to the maximum simulation rate may not be the optimum strategy. Sometimes, the simulations may have to be "slowed down" if a large number of output frames corresponding to previous time steps has not been transferred to the visualization site, resulting in lesser available free disk space.

Thus a decision algorithm has to carefully consider these various dependent impacting factors to achieve a good balance between its contradictory objectives of maximizing simulation rate and the temporal resolution for visualization. The decision algorithm considers both application and resource parameters as inputs. The simulation resolution is an application-specific parameter and impacts the visualization quality. For example, in the case of cyclone tracking, a climate scientist may want to visualize with coarser resolutions during the initial stages of cyclone formation and with finer resolutions when the cyclone intensifies. The bandwidth and the free disk space are resource parameters. The algorithm also takes as input the execution times or simulation rates of WRF simulations for different number of processors and simulation resolutions. The execution times of a subset of configurations have been experimentally found by running sample WRF runs for simulation time of 1 hour for different discrete number of processors, spanning the available processor space and using performance modeling or curve fitting tools [11] to interpolate for other number of processors. This has been done for each of the experimental set-ups. 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 climate events.

We have devised two decision algorithms: a greedy algorithm that uses thresholds for modifying parameters, and an optimization-based approach. The following subsections describe these algorithms.

A. Greedy-Threshold Algorithm

This algorithm attempts to employ the maximum number of processors for maximum simulation rate and output every simulated time step for maximum temporal resolution. However, since this greedy strategy can result in rapid decrease in available free disk space, the algorithm also uses thresholds for free disk space to dynamically adjust the frequency of output and number of processors for execution. The algorithm considers two sets of thresholds, *lowdiskspace-thresholdset* when the remaining disk space is low and *highdiskspace-thresholdset* when the remaining disk space is high. For our current work, we set *lowdiskspace-thresholdset* = $\{50, 25\}$, and *highdiskspace-thresholdset* = $\{60\}$ When the remaining disk space is less than an upper bound of *lowdiskspace-thresholdset*, the algorithm first decreases the frequency of output i.e. increases the interval of output, *output_interval*. If the *output_interval* is already equal to its maximum value, *upper_output_interval*, and if the free disk space is still less than the thresholds in *lowdiskspace-thresholdset*, the algorithm "slows down" the simulation or increases the execution time by decreasing the number of processors used for simulation. If the free disk space is less than the lowest threshold in *lowdiskspace-thresholdset*, the algorithm sets the *CRITICAL* flag in the application configuration file, thereby leading to stalling of the simulations.

The observation is that decreasing the rate of simulation and the frequency of output may eventually lead to freeing up of disk space. At some point when the remaining free disk space increases sufficiently, the algorithm follows a reverse process using the thresholds in *highdiskspace-thresholdset*, whereby it increases the simulation rate by increasing the number of processors for execution first. If the maximum simulation rate is achieved and the remaining free disk space is sufficient, then the algorithm decreases the *output_interval*. Thus this algorithm gives more preference to maximizing the simulation rate than to maximizing the output frequency.

The psuedocode for this is shown in Algorithm 1. This algorithm is invoked periodically every 1.5 hours. In the pseudocode, OI refers to the *output_interval*. $oldOI$ and $newOI$ refer to the old and new values of *output_interval*. $minOI$ and $maxOI$ refer to the minimum and maximum values of *output_interval*. $mintime$ and $maxtime$ refer to the minimum and maximum values of execution time per time step of simulation, and correspond to execution with maximum number of processors and minimum simulation rate respectively. The algorithm calculates the new execution time $newtime$ for simulation from the previous value $oldtime$ in lines 7 and 11 and determines the corresponding number of processors using the benchmark profiling runs with WRF.

B. Optimization Method

The primary objective of a traditional scheduling problem for parallel simulations is to maximize the rate of simulations. The rate of simulations is typically high for large number of processors and large I/O bandwidth. However faster execution time and larger I/O bandwidth can lead to faster consumption

---

Input: oldOI, minOI, maxOI, oldtime, mintime, maxtime

1. $D \leftarrow$ Remaining free disk space;
2. **if** *(D ≤ 10%)* **then** set CRITICAL flag;
3. **else if** *(D ≤ 50%)* **then**
4.   **if** *(D ≥ 25%)* **then**
5. $newOI \leftarrow oldOI + \frac{(50-D)}{25} \cdot (maxOI - oldOI)$;
6.   **else if** *(oldOI = maxOI)* **then**
7. $newtime \leftarrow oldtime + \frac{(25-D)}{15} \cdot (maxtime - oldtime)$;
8.   **end**
9. **else if** *(D ≥ 60%)* **then**
10.   **if** *(oldtime > mintime)* **then**
11. $newtime \leftarrow oldtime - \frac{(D-60)}{40} \cdot (oldtime - mintime)$;
12.   **else if** *(oldOI > minOI)* **then**
13. $newOI \leftarrow oldOI - \frac{(D-60)}{40} \cdot (oldOI - minOI)$;
14.   **end**

Output: $newOI$ and corresponding number of processors for $newtime$ to application configuration file

Algorithm 1: Greedy-Threshold Algorithm

---

of storage space by the simulations. In addition, if the network bandwidth from the simulation to the visualization end is low, then the disk can overflow soon. It is also interesting to note that the fastest rate of simulation can be achieved and the disk space limitation or problem can be avoided in spite of high I/O bandwidth, slow network and small execution time, if the output frequency is 0, i.e. output is not generated by the simulations at all. But for critical climate applications, it is vital to output as frequently as possible in order to perform continuous visualization of the output. However frequent I/O can decrease the simulation rate and also leads to faster accumulation in the storage. Thus we can think of our problem as an optimization problem that primarily attempts to maximize the simulation rate within the constraints related to continuous visualization, acceptable frequency of output, I/O bandwidth, disk space and network speed.

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 II. 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.

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,

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

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).

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

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

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.

Disk Constraint: Assuming that the rate of input to the disk from the simulation is greater than the rate at which the data is transferred to the visualization, then the time n in which the disk will overflow is given by Equation (3) 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 (4).

$$n \leq \frac{\mathcal{D}}{(\mathcal{R}_{in}) - (\mathcal{R}_{out})} \tag{3}$$

$$\frac{\mathcal{O} \cdot \mathcal{F}}{t \cdot \mathcal{S} + \mathcal{T}_{IO} \cdot \mathcal{F}} - b \leq \frac{\mathcal{D}}{n} \tag{4}$$

Linearizing the Constraints: To linearize the non-linear constraints (2) and (4), we divide both sides of these equations by $\mathcal{S}$ to obtain the constraints specified in Equations (5) and (6). Here, z and y are substituted for $\frac{\mathcal{F}}{\mathcal{S}}$ and $\frac{\mathcal{T}}{\mathcal{S}}$, respectively. Depending on the total number of processors, t also has a lower bound $\mathcal{T}_{LB}$, as specified in constraint (7). We have specified the upper bound for output frequency to be

25 simulated minutes[1] i.e. the interval between visualization of climate events is maximum of 25 simulated minutes. This gives a lower bound for z. Since the output frequency OI can at the minimum be 1 simulated minute, this gives an upper bound for z. These constraints are specified in (8).

$$t + z \cdot \mathcal{T}_{IO} \leq \frac{\mathcal{O}}{b} \cdot y \tag{5}$$

$$t \geq \frac{\mathcal{O}}{(\frac{\mathcal{D}}{n} + b)} - \mathcal{T}_{IO} \cdot z \tag{6}$$

$$t \geq \mathcal{T}_{LB} \tag{7}$$

$$LB \leq z \leq UB \tag{8}$$

We used GLPK (GNU Linear Programming Kit) [12] 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. From the value for z, we determine the frequency of output, OI, as follows. 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 as

$$OI \cdot \mathcal{F} = ts \cdot \mathcal{S} \tag{9}$$

Substituting for $z = \frac{\mathcal{F}}{\mathcal{S}}$ and ts in the above equation, the interval of output, OI, and the corresponding frequency can be obtained.

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.

Although the threshold values used in the algorithms are specific to our experiment settings and WRF simulations, the general principles of our threshold-based greedy heuristic and optimization-based strategy are generic and applicable to other applications.

V. EXPERIMENTS AND RESULTS

In this section we present our experimental setup including the details of the climate application used for simulation and visualization, the resource configuration used for simultaneous

---

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

simulations and online remote visualization involving networks of different bandwidths, and the results.

A. Climate Application: Tracking Cyclone Aila

We have applied our framework for large-scale and long-range tracking of cyclones that involves high amount of parallel computations with large data sets, subsequent data analysis and high-resolution visualization for scientific discovery. 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 [13]. 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 WRF (Weather Research and Forecasting Model) [9], [10].

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 of cyclonic regions of interest. To track the lowest pressure region or eye of the cyclone Aila, we employ a finer resolution nest on the region of our interest inside the parent domain as shown in Figure 3. We have performed the simulations for an area of approximately 32×10^6 sq. km. from $60°E$ - $120°E$ and $10°S$ - $40°N$, comprising of the region of formation and dissipation of Aila over a period of 2.5 days. The nesting ratio i.e. the ratio of the resolution of the nest to that of the parent domain, was set to 1:3. The 6-hourly 1-degree FNL analysis GRIB meteorological input data for our model domain was obtained from CISL Research Data Archive [14].
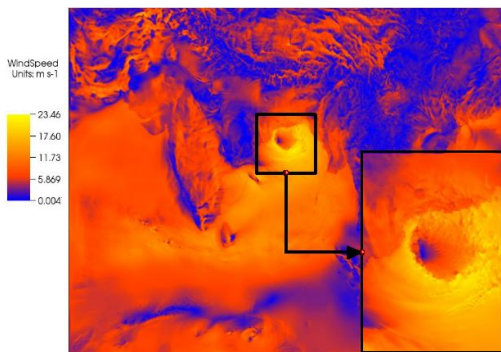

Fig. 3: Windspeed visualization in finer resolution nest inside parent domain

As WRF is a regional model, with each level of refinement, it needs input data at a finer resolution. Before executing

TABLE III: Resolutions for different Pressure Values

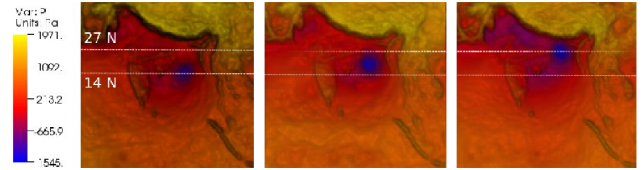| Pressure (hPa)	995	994	992	990	988	986
Resolution (km)	24	21	18	15	12	10

Fig. 4: Visualization of Perturbation Pressure at 18:00 hours on 23rd, 24th and 25th May, 2009

WRF, the WRF Preprocessing System (WPS) is executed to interpolate the meteorological data onto the domain of interest. 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. 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. Table III shows the pressure values used for different resolutions.

WRF outputs data in form of NetCDF [15] files. Each NetCDF file contains output of a number of simulation time steps. We have visualized the output using volume rendering, vector plots employing oriented glyphs, pseudocolor and contour plots of the VisIt [16] visualization tool. We have developed a plug-in for VisIt to *directly read* NetCDF output files, eliminating the cost of post-processing before data analysis.

The track of Aila as produced by simulation can be seen in Figure 4. It can be observed from the figure that the depression was formed in the central Bay of Bengal region (around $14°N$) and traversed north-east upto Darjeeling ($27°N$).

B. Resource Configuration

For all our experiments, visualization was performed on a graphics workstation in Indian Institute of Science (IISc) with a Intel(R) Pentium(R) 4 CPU 3.40 GHz and an NVIDIA graphics card GeForce 7800 GTX. We used hardware acceleration feature of VisIt for faster visualization. We executed the simulations on three different sites resulting in three different remote visualization settings, namely, *inter-department*, *intra-country* and *cross-continent* visualizations. In the *inter-department* configuration, the WRF simulations

were executed on a dual-core AMD Opteron 2218 cluster, *fire*, in Indian Institute of Science (IISc). In the *intra-country* configuration, the simulations were performed on quad core Intel Xeon X5460 cluster, *gg-blr*, in Centre for Development of Advance 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) [17] with the maximum bandwidth of 1Gbps. In the *cross-continent* configuration, the WRF simulations were conducted on the dual-core AMD Opteron 265 cluster, *moria*, in Innovative Computing Lab of University of Tennessee, Knoxville, USA. Table IV gives the detailed specifications of the three 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. WRF simulations have limitations in the number of cores that can be used depending on the grid size. Specifically, each MPI process should have at least 6x6 parent domain grid points and 9x9 nest domain grid points to process. For our simulations, we used a minimum nest grid size of 100x127 that is appropriate for the region of interest of cyclone Aila. This imposed the limitation on the number of cores for simulations.

C. Results

Figure 5(a) shows the simulation rate in the *inter-department* configuration. The graph plots the times simulated with the progress in executions. We find that the optimization method provides steady and faster rate of simulations. The optimization approach resulted in about five hours of less execution time than the greedy-threshold algorithm for the same total simulated time. The optimization approach, due to its solution for maximization of simulation rate within the resource constraints, is able to provide steady-state solution for the simulation rate. The greedy-threshold algorithm, due to its reactive behavior, shows inconsistent rates of simulations throughout the simulation period.

Similar results for the progress of simulation for the *intra-country* and *cross-continent* are shown in Figures 5(b) and 5(c) respectively. In the *cross-continent* case, since the bandwidth was only 60 Kbps, the disk got filled at a faster rate and hence in the greedy approach, WRF simulation had to be stalled even before the completion of the simulation time-period. This is shown by means of dotted lines in the graph after the wall clock time value of 24 hours. However, using the optimization approach, WRF simulation was able to complete without stalling. It is also clear from this result that a non-adaptive solution would result in stalling of the simulation much earlier than in the greedy algorithm.

Figure 6(a) shows the rate of disk consumption for the *inter-department* configuration. The graph plots the percentage of available free disk space with progress in simulation. The greedy-threshold algorithm, due to its emphasis on minimizing the execution time and maximizing the frequency of disk output in initial stages, results in high rate of storage space

consumption in the initial stages. The available free disk space becomes less than 40% within 4 hours of execution time. The algorithm then tries to take corrective action and consumes less storage by decreasing the simulation rate and the frequency of output to the disk. Nevertheless, the greedy algorithm consumes about 90% of the disk space by the end of the simulations. The optimization method, due to its steady state behavior, is able to determine the appropriate simulation rate and the frequency of disk output considering the different constraints and arrive at a *global* solution over a longer period of time. The efficiency of the method results in about 25% less consumption of storage space than the greedy algorithm.

The rate of disk consumption for *intra-country* configuration is shown in Figure 6(b). In the greedy-threshold heuristic, the disk space is quickly consumed and the free disk space falls below 20% because of faster solve time and slower network. In the optimization approach, the free disk space never drops below 50% during the entire run of the simulation because of the consideration of the global solution by this method.

The storage space decreases rapidly for the *cross-continent configuration* for the greedy-threshold heuristic because it tries to maximize the number of frames output without considering available disk space in the initial stages. Since the network bandwidth is very low, this approach suffers from disk over-flow (less than 5%) before completion of simulation as shown in Figure 6(c). The optimization method is able to complete simulation with more than 20% of the disk space remaining. This is because though the disk is filled faster due to slower network, the optimization method tries to adjust the output frequency and number of processors from the beginning of the simulation.

Figures 7(a), 7(b) and 7(c) show the progress of visualization at the visualization site for the three configurations. On the x-axis, the wall-clock time progression is shown when different frames are visualized and on the y-axis, the corresponding simulation time steps represented by the frames are shown. Any given point in the graphs corresponds to the time when a simulation frame was visualized. For example, in Figure 7(a), at time 12:00 hours corresponding to the x-axis, the frame corresponding to 23rd May, 9:00 hours simulation time step is visualized in the optimization-based approach.[2] It can be seen in the figures that the visualization progress is much faster for the optimization method whereas the greedy heuristic approach lags behind in visualizing frames because it tries to send every time step from the simulation to the visualization site in the intial stages. It can be seen that even after 21 hours, the greedy heuristic is able to visualize less than 10 hours of simulation frames for *intra-country* and *cross-continent* simulations. Since the optimization method outputs frames depending on the resource constraints, in all three cases reasonable visualization progress were made. The optimization method is able to visualize about a day of simulation progress in *inter-department* and *cross-continent*

---

[2]The visualization was stopped after simulation time of May 23, and not upto May 25, to facilitate completion of experiments involving slow networks.
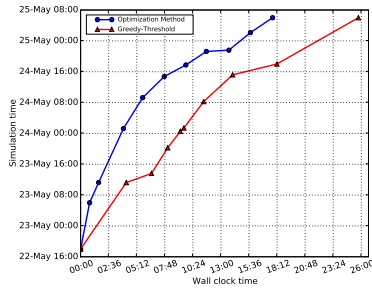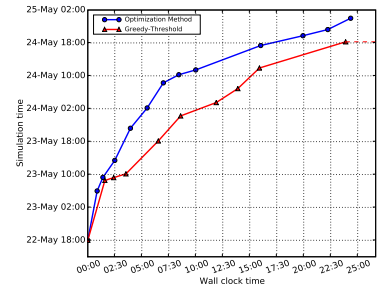
Configuration	Simulation Configuration	Maximum Cores for Simulation	Maximum Disk Space Used	Average Sim-Vis Bandwidth
inter-department	*fire*: 12x2 dual-core AMD Opteron 2218 based 2.64 GHz Sun Fire servers, CentOS release 4.3, each with 4 GB RAM, 250 GB Hard Drive, and connected by Gigabit Ethernet	48	182GB	56 Mbps
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	90	150GB	40 Mbps
cross-continent	*moria*: dual AMD Opteron 265 (Dual Core) 1.8GHz cores, RHEL 5, each with 4GB RAM and 1 x 80GB SATA Hard Drive, and connected by Gigabit Ethernet	56	100GB	60 Kbps
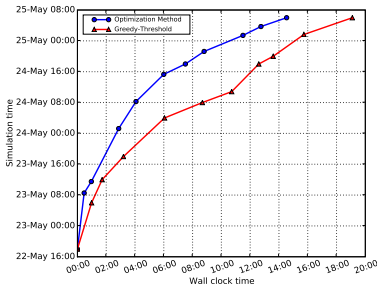
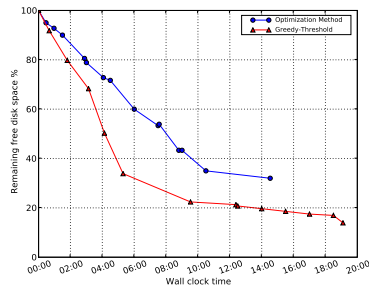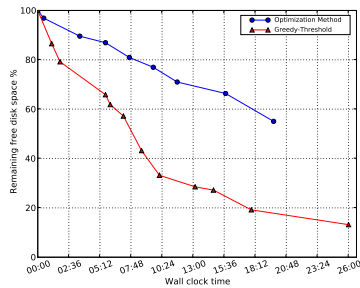(a) *inter-department* configuration (b) *intra-country* configuration (c) *cross-continent* configuration
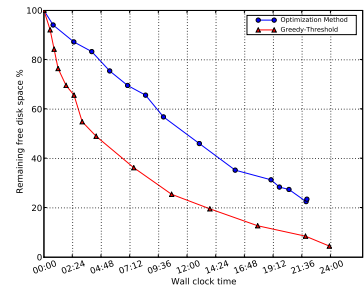
Fig. 5: Simulation times with progress in executions for different configurations. The graphs show faster rate of simulation for Optimization-based approach in all the configurations. Greedy-Threshold (red) and Optimization-based Approach (blue)

(a) *inter-department* configuration (b) *intra-country* configuration (c) *cross-continent* configuration

Fig. 6: Free disk space with progress in executions for different configurations. The graphs show the decrease in available disk space as simulation progresses in time. Greedy-Threshold (red) and Optimization-based Approach (blue)

configurations. Among the three configurations, the progress is fastest in the *inter-department* case because of the availability of highest bandwidth.

Figures 8(a) and 8(b) show the adaption of the number of processors and the output interval of the simulations by the framework based on the application and resource configurations with the progress in execution times for *inter-department* and *cross-country* configurations. In all cases, the greedy method starts with the maximum number of processors in order to have the best simulation rate. It also starts with

a lowest output interval of 3 minutes in order to output as many time steps as possible. However with time, as the free disk space decreases, the output interval is increased and the number of processors is decreased as shown in Figure 8(a). The optimization method adjusts output frequency as and when needed according to the remaining disk space. Thus it adapts the frequency of output to the best possible value for the given resource constraints from the beginning of the simulations. Since it is able to satisfy the upper bound of output interval without altering the number of processors, it

(a) *inter-department* configuration   (b) *intra-country* configuration   (c) *cross-continent* configuration
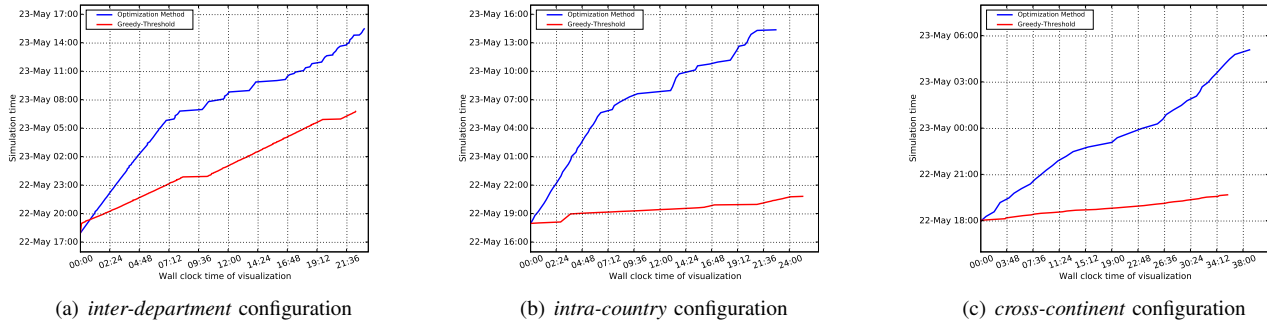
Fig. 7: Progress at the visualization end for different configurations. Greedy-Threshold (red) and Optimization-based Approach (blue)

uses the maximum number of processors since its primary objective is to minimize the solve time. In the *cross-continent* configuration, when the greedy approach cannot take either of these actions and the disk space is sufficiently low, the WRF simulations have to stall. Whereas in the optimization approach, it tries to avoid stalling from the beginning by considering the various impact factors for smooth simulation and visualization. We also find that the disk output interval is almost constant and variation is less for the optimization method when compared to the greedy-threshold heuristic. Thus, the optimization method is able to provide a consistent "quality-of-service" for visualization by the climate scientists, which is very essential for remote visualization of critical climate science applications.

VI. CONCLUSIONS AND FUTURE WORK

We have described an adaptive integrated framework for simulation and visualization of critical climate applications like cyclones. We show how we are able to simultaneously visualize the output without having to wait for the whole simulation to complete. Our framework adapts the simulation rate and the output interval based on the disk space and network speed constraints. In the process, the framework also considers the dynamics of the changing resource configurations. As shown in Section V, a simple and intuitive greedy approach may lead to low throughput, stalling of simulation and disk overflow. This shows the importance of considering network bandwidth, execution time and output frequency for a smooth online visualization. Our optimization method is able to provide about 30% higher simulation rate completing the entire simulations for all network configurations, consumes about 25-50% lesser storage space completely avoiding the disk overflow problem and the resulting stalling of simulations, and provides higher and more consistent rate of visualization than the greedy approach. Hence we claim that it is very important to consider the currently available network bandwidth, and disk space to be able to do online visualization with best throughput and best temporal resolution.

In future work we would like to extend our framework for a larger grid and on different configuration settings. We intend to parallelize the visualization process as well. We also intend

(a) *inter-department* configuration
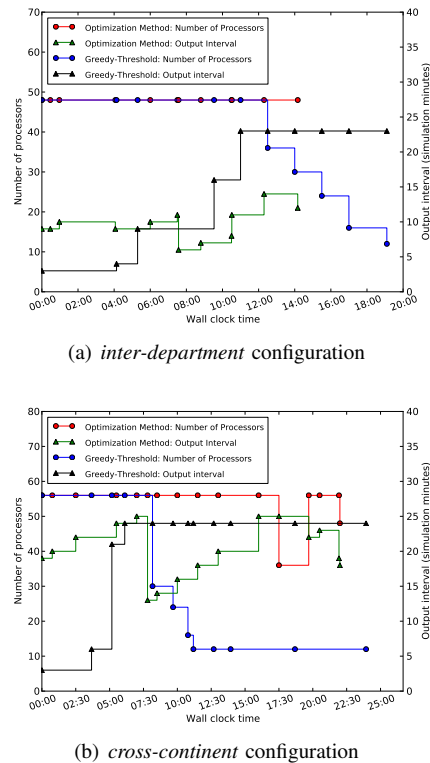

(b) *cross-continent* configuration

Fig. 8: Adaptivity of the framework showing variation in number of processors (Left y-axis) and output interval (Right y-axis)

to investigate interactive simulation/visualization, so that user input based on the visualization can steer the simulation.

VII. ACKNOWLEDGEMENTS

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*, 2007, p. 59.

[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] 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, p. 49.

[5] 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, p. 91.

[6] 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.

[7] 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.

[8] D. Ellsworth, B. Green, C. Henze, P. Moran, and T. Sandstrom, "Concurrent Visualization in a Production Supercomputing Environment," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 997–1004, 2006.

[9] 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.

[10] 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.

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

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

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

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

[15] 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.

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

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