2011

Quality of service based data-aware scheduling

Archit Kulshrestha
Louisiana State University and Agricultural and Mechanical College, akulsh1@lsu.edu

QUALITY OF SERVICE BASED DATA-AWARE SCHEDULING

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Archit Kulshrestha
B.Tech., Jawaharalal Nehru Technological University, 2002
M.S., Louisiana State University, 2004
May 2011

Acknowledgments

I would like to start by thanking my advisory committee, Dr. Gabrielle Allen, Dr. Tevfik Kosar, Dr. Sitharama Iyengar, and Dr. Supratik Mukhopadhyay for their guidance and Dr. Shahab Mehraeen for agreeing to serve as the Dean's representative. Dr. Allen has been one of the biggest influences on my career and life and this work would be impossible without her continued support and encouragement. Also this work would not be possible without the help and support of my colleagues at the Center for Computation & Technology (Louisiana State University).

I would like to thank all my collaborators from SCOOP, CERA, LONI, TeraGrid and SURAGrid, in particular the following (in no particular order): Philip Bogden, Jon MacLaren, Gayathri Namala, Sirish Tummala, Edward Seidel, Matt Smith, Tom Gale, Gary Almes, Gerry Creager, Joanne Bintz, Hans Graber, Sara Graves, Helen Conover, Rick Luettich, Will Perrie, Bash Toulany, Justin Davis, Harry Wang, Dave Forrest, Luis Burmudez, Andrei Hutanu, Chongjie Zhang, Ian Kelley, Hartmut Kaiser, Carola Kaiser, Chirag Dekate and Srikanth Balasubramanian

I would like to acknowledge Krzysztof Kurowski and Ariel Oleksiak from the Poznan Supercomputing Center, for their help with GSSIM and Steve Brandt from CCT for providing the Queenbee workload data for the experiments.

This work was supported by the SURA Coastal Ocean Observing and Prediction Program, funded in part by the National Oceanic and Atmospheric Administration (NOAA), the Office of Naval Research (ONR) and by the Center for Computation & Technology at LSU.

Portions of this research were conducted with high performance computational resources provided by the Louisiana Optical Network Initiative (http://www.loni.org/).

Finally, I would like to thank my family and friends for their support throughout this journey and in particular my wife Prathyusha, who has been a source of strength through out this work.

Table of Contents

List of Tables

List of Figures

Abstract

Distributed supercomputers have been widely used for solving complex computational problems and modeling complex phenomena such as black holes, the environment, supply-chain economics, etc. In this work we analyze the use of these distributed supercomputers for time sensitive data-driven applications. We present the scheduling challenges involved in running deadline sensitive applications on shared distributed supercomputers running large parallel jobs and introduce a "data-aware" scheduling paradigm that overcomes these challenges by making use of Quality of Service classes for running applications on shared resources. We evaluate the new data-aware scheduling paradigm using an event-driven hurricane simulation framework which attempts to run various simulations modeling storm surge, wave height, etc. in a timely fashion to be used by first responders and emergency officials. We further generalize the work and demonstrate with examples how data-aware computing can be used in other applications with similar requirements.

Chapter 1
Introduction

Computational science is typically performed on large, parallel systems which are shared by many different users with different application requirements. In this environment simulations may need to wait hours or days before they can execute. However, there are urgent situations (for example predicting the effects of hurricanes, tornado or earthquakes) where this model is not appropriate and a new execution methodology is needed where on-demand simulations can be completed very quickly. Research in this area is important, both for the potential impact on society from rapidly modeling natural disasters for emergency response, but also because of the interesting computer science questions raised whose solutions will be applicable to a wide range of applications. This work investigated how urgent simulations can specify the Quality of Service they need, defined and evaluated scheduling algorithms to provide this Quality of Service, and investigated how scientists can best interact with a set of distributed supercomputers to execute their urgent simulations.

This work addresses challenges in taking advantage of the capabilities provided by modern, shared computational resources for the modeling of time sensitive phenomena using a special scheduling approach that uses the content or attributes of the input data to prioritize the modeling of the time sensitive phenomena. These scenarios arise in many different areas of science and engineering, driven by the availability of supercomputing resources provided by national facilities such as NSF TeraGrid and Blue Waters, and new computer science paradigms such as dynamic data-driven application systems (DDDAS, [1]).

In disaster management, exemplary scenarios include the use of time sensitive simulations to forecast the height and location of storm surge from tropical storms based on predicted hurricane tracks; the effects of earthquakes based on sensor information; or the likely path of oil spills following a rig blow out such as the one seen in the Deepwater Horizon disaster in the Gulf of Mexico [3]. In these scenarios large scale simulations must be deployed and completed to provide timely and reliable information to guide policy makers and first responders.

In scientific investigations, time sensitive simulations are envisaged that are driven by the availability of experimental or observational data, for example modeling supernovae collapse based on detected gravitational wave data in time to point optical telescopes at the appropriate event . Other applications include state estimation and prediction for dynamical systems; or complex ecosystem modeling [2].

Time sensitive simulations also occur in industrial settings such as oil recovery, where sensors provide real time input about oil well properties, such as oil pressure or temperature, which are fed into systems that can react to these changes and make necessary control adjustments to maximize oil output from the well. Iyengar et.al in their work on oil spills [3] talk about the need for complex event and data-driven processing for detecting and preventing potentially hazardous conditions that could lead to oil spills. Other scenarios include oil cleanup where sensors provide up to date information about oil density, water current velocities, wind velocities, etc., and these inputs are fed into simulations that predict the projected path of the oil. DDDAS scenarios are also common in supply chain control, advanced driving assistance systems, electrical smart grids, etc. all of which need to react to real time input from various sources. The turnaround times for these applications varies widely from a few milliseconds to a few hours depending on the scenario. Real time applications can be classified based on their real time response requirements and their run time. The characteristics of real time systems and the conditions that they impose on the schedule are discussed in Section 2.1.3.

More broadly, time sensitive simulations occur in dynamic data-driven application systems(DDDAS), a new paradigm focused on symbiotic feedback between applications and measurements where applications are provided with the ability to incorporate real-time data and further to guide and control the measurement process itself. The DDDAS paradigm of computing is the future of all interactive and real time applications and will play a key role in the applications of future and will be seen in the everyday life of people, in many cases saving lives and also improving the quality of life of people for example by having safe cars that prevent accidents by reacting to thousands of realtime events such as obstacles or even the drivers heart rate.

All these scenarios involve a critical requirement for scheduling of simulations. Generally, the need for scheduling arises when the amount of resources available to accomplish tasks is limited, as is

the case when using shared and distributed supercomputers, and associated infrastructure such as storage, networks or experimental devices, for the scenarios outlined above. There are many challenges associated with scheduling of distributed resources, some of which have begun to be addressed through work in the field of *grid computing* (see e.g. [4]), which take into account issues related to the varying capabilities of resources owned by different organizational units and operating based on different policies. Resources in a *Grid* are often considered to be part of a *Virtual Organization* and are governed by a set of sharing rules [4] which allows users from various administrative domains to submit jobs to these shared resources using credentials obtained from their respective administrative domains. Most of the scheduling policies are however controlled by the local resource providers, hence a low level scheduling algorithm that works on global information across the Grid is not a feasible solution to the scheduling problem. Meta-schedulers such as Gridway [5] are required to coordinate job submission across the resources controlled by the local resource managers. Meta-schedulers also often act as brokers in determining the right resource for scheduling the job, however just like local schedulers these meta-schedulers are also data agnostic, which means they treat all jobs equally and attach no importance to the attributes, content or type of data.

The work described here addresses fundamental issues for scheduling dynamic time-sensitive data-driven tasks in a distributed grid-type environment provided by local and national resources, such as the NSF TeraGrid [6], Louisiana's LONI environment [7] or SURAGrid [8], by introducing the Quality of Service based data-aware scheduling.

In particular we are focused on answering the following questions:

- How well do current scheduling algorithms perform when applied to time sensitive and data-driven simulations deployed on a distributed set of supercomputer resources?
- Is a new scheduling paradigm required for handling time sensitive and data-driven simulations?
- What middleware is needed to support such scheduling and application scenarios?
- How can application needs be formally described to provide a well posed computational problem, provide scientists with easily configurable interfaces, and enable automated workflows?

The answers to these questions make up the rest of the thesis. The key contributions of this work are:

- Development of a scheduling paradigm called Quality of Service based data-aware scheduling.
- Development of middleware to support data-aware scheduling. The middleware is part of pipeline that is a prototype for operational hurricane forecast runs.
- Development of semantics to convey priority of jobs based on scientific importance to the scheduler.
- Development of scheduling plugins for GSSIM

Each of these contributions is discussed in detail in this thesis. Chapter 2 describes the compute infrastructure and the motivating scenario for this work, followed by the description and theory behind data-aware scheduling in Chapter 3. Chapter 4 describes the coastal modeling pipeline and its components and carries on to describe the additional components needed to make it data-aware. Chapter 5 addresses the other work in the areas of scheduling, pipelines, workflows etc and aims at putting into perspective this work and its merits when compared to other related work in relevant fields, followed by results in Chapter 6.

Chapter 2
Background

In this section we describe some of the different components and issues that are important for understanding this work, and its application and importance for real world systems and production high performance computing(HPC) resources. First, in Section 2.1 we describe the computing environment, focusing in Section 2.1.1 on the policy issues and technical capabilities of the underlying resources, in Section 2.1.2 on scheduling of compute jobs, in Section 2.1.3 on real time systems, which establishes the background and introduces common solutions for scheduling time sensitive jobs, in Section 2.1.4 on the use of batch queues on supercomputers, and in Section 2.1.4 on the role of metaschedulers. Section 2.2 describes the motivating scenario behind this work, which is running time sensitive hurricane forecast simulation on shared supercomptuers. Section 2.2 describes the impact of hurricanes and the importance of timely simulation results to save lives and property. Section 2.2.1 describes the ensemble forecasting process followed by a description of the various components that make up a coastal modeling pipeline in Section 2.2.2.

2.1 Modeling Infrastructure

2.1.1 Compute Resources

In this section we describe the various resources that were used for developing and deploying the coastal modeling scenario and discuss the various scheduling policies on these resources and how they affect the choices that we made when designing our solution.

TeraGrid

TeraGrid is a national Grid of supercomputers funded by the National Science Foundation to support scientific research. TeraGrid [6] integrates high-performance computers, data resources and tools, and high-end experimental facilities around the country and forms the largest unified resource in the world. TeraGrid is used by national and international researchers to carry out scientific simulations ranging from few minutes to few days or weeks and from using one core to running across thousands of processors. Resources in TeraGrid are owned and operated by local Resource Providers and resource

policies are set by these local resource owners, for instance some sites prioritize large parallel jobs while others are targeted at throughput computing, while some other resources also provide on-demand access to jobs. This presents a challenge when trying to develop a single system that must ensure timely completion of time sensitive applications submitted to these resources. Also these novel scheduling scenarios like on-demand computing and advance reservations compete with traditionally established scientific use and are not welcome by dominant users of TeraGrid. Compute resources are made available to users by a process of allocation request and approval. Users that have an allocation are allowed to run on resources. The allocation process currently oversubscribes various resources because of heavy demand for compute resources. This obviously means that resources are usually busy and instant access to compute resources required special mechanisms to be put in place. The TeraGrid provides resources with next to run access and also resources with preemptive access. Access to these resources is also controlled by allocations and once it has been approved, local resource providers enable special use of these systems for approved users. We make use of these special access mechanisms in our solution.

LONI and SURAGrid

The Louisiana Optical Network Initiative, or LONI, is a fiber optics network that runs throughout Louisiana, and connects Louisiana and Mississippi research universities to one another as well as National LambdaRail and Internet2. LONI connects supercomputers at the participating universities and other computing resources throughout Louisiana, and centers around a 50-teraflops supercomputer called Queen Bee, supplemented by six five teraflop Intel Linux Clusters and five IBM Power5 575. The Queenbee workload data was used to generate the results for this work as Queenbee is a large LONI and TeraGrid resource which is representative of all large and medium sized HPC resources available today. In contrast to TeraGrid, LONI is centrally managed and the scheduling policies at the various sites are strictly controlled and are often based on the size of the resource and its perceived use, for example large jobs are favored on the resources with higher number of processors while single processor jobs are favored on smaller machines distributed across the state. Even in such a controlled environment the challenges remain almost the same in terms of scheduling time sensitive simulations. The South Eastern Universities Regional Association (SURA) Grid is a

grid infrastructure that combines the compute power at various universities in the Southeast US to form a large collaborative resource. However unlike LONI, SURAGrid resources are locally governed and controlled, but abide by a set of standard policies specifically designed to support on-demand computing. LONI like TeraGrid has an allocations policy and requires users to formally apply for compute time before using the resource. On demand access to resources is also provided upon request and on a case by case basis. For this work we had an allocation on LONI which allowed preemptive and high priority access to resources. SURAGrid did not enforce allocations on users, but used authorization mechanisms to control access to resources in any manner.

Due to the complexity of developing efficient schedules across such a large number of resources for thousands of jobs, modern day schedulers are focused on developing efficient schedules based on the resource requirements (compute time, number and type of resources) alone. Special middleware and tools developed as part of this work are needed to translate the specific job requirements of time sensitive and deadline bound jobs to requirements that the local schedulers can identify and develop a schedule that meets these requirements. Such middleware takes the form of a set of services that interact with job submission interfaces, and use information provided by the job owners that can be used to determine the right scheduling strategy for these jobs. The key here is that not all jobs are equal and not all jobs have the same scheduling requirements, hence it seems obvious that a scheduler that treats all jobs the same is not suitable in such scenarios. In particular time sensitive jobs need special treatment from the schedulers so that they can be executed and finished before the deadline or not be executed at all.

2.1.2 Compute Job Scheduling

The primary focus of this work is developing scheduling paradigms for use on modern day supercomputers that better serve time sensitive simulations. This chapter will introduce the concept of scheduling, discuss various commonly used scheduling algorithms used for executing jobs on supercomputing resources and discuss the effectiveness of these algorithms in scheduling deadline sensitive simulations. We will also present the commonly used scheduling policies on supercomputing resources and the challenges that these present when dealing with time sensitive applications and finally discuss the role of meta-schedulers when dealing with distributed super computers for submitting jobs.

Resource sharing is required when enough resources are not available to execute all the requested tasks simultaneously without incurring a penalty on performance. Two modes of resource sharing are popular when scheduling jobs on supercomputing resources namely time-sharing and space-sharing. Space sharing partitions the resource into pools of processors that are assigned to tasks, while time sharing allocates all resources to a single task for a quantum of time and schedules the next task once the time expires. Space sharing has become the most popular mechanism of using distributed supercomputers for running compute jobs mainly because some parallel jobs are not able to tolerate interruptions in execution and do not recover elegantly. Checkpointing of parallel jobs is however common practice and is used to restart the simulation using state information that is written to checkpoint files. This ability of jobs to checkpoint and restart enables us to introduce time sharing along with space sharing to improve utilization of resources, specially when scheduling dynamic time sensitive data-driven applications. The specific mechanisms involved in using the time and space sharing hybrid model will be discussed in Chapter 3. In this chapter we will also describe the scheduling problem and the working of traditional batch scheduler used on these supercomputers, to motivate the need for additional services and software to assist in building a suitable schedule for the time sensitive simulations.

The scheduling problem dates back to the time when industries were faced with the problem of managing various activities occurring in a workshop. In the late 1960s computer scientists encountered scheduling problems in the development of operating systems. The primary reason for the study of scheduling is provided by the economic implications of providing abundant resources that do not need to be scheduled anymore. As problems began to increase in complexity scheduling became a difficult task. In the 1970s, many scheduling problems were shown to be NP-hard [11]. Most effort was then focused on developing approximation algorithms that provided near-optimal schedules that improved utilization [11]. Early efforts were focused on developing scheduling algorithms for operating systems that needed to schedule tasks on one or more processors. Scheduling algorithms are evaluated using various metrics such as the total execution time known as makespan, average turnaround time and resource utilization. The average turnaround time is the metric that we will focus on in this work and try to reduce that for important jobs. Most basic scheduling algorithms are based in some way

on arrival order, task length and ability to preempt other tasks. The most notable of these are are First In First Out (FIFO), Shortest Job First (SJF) and Round Robin. These algorithms are simple yet effective in terms of providing good resource utilization. As mentioned above the most widely used scheduling algorithms, because of their simplicity and effectiveness are decribed below:

- *First in First Out:* The most basic algorithm a scheduler can implement is known as First In, First Out (FIFO) scheduling (sometimes also called First Come, First Served (FCFS)). As the name suggests FIFO dispatches jobs in the order in which they were submitted. FIFO has a number of positive properties: it is clearly very simple and thus easy to implement. The algorithm works at 100% utilization when there is a single resource to be scheduled, but is not optimal when multiple resources are present and tasks need one or more resources to execute. FIFO also performs poorly with respect to average turnaround time when task lengths vary significantly when compared to other scheduling strategies that allow reordering of jobs in the queue.

- *Shortest Job First:* The problem of bad average turnaround time is solved by reordering jobs to move shorter jobs to the top of the queue. In this strategy the scheduler arranges processes with the least estimated processing time remaining to be next in the queue. This strategy does not do well for scheduling supercomputing jobs, specially where the goals are to implement fairness or to favor large jobs.

- *Round-robin Scheduling:* With round-robin scheduling, the scheduler assigns a fixed time unit per process, and cycles through them. This ensures a guaranteed response time for all jobs and the response time is a factor of the number of jobs in queue rather than the length of jobs. This strategy assumes all tasks can be preempted, which is not the case in HPC jobs.

Scheduling evolved as compute resources grew in size and solutions were developed that could efficiently schedule thousands of jobs across thousands of processing units, however as mentioned earlier these schedulers focused only on number of resources and the amount of time requested.

2.1.3 Real-time Systems

In computer science, real-time computing (RTC), or reactive computing, is the study of hardware and software systems that are subject to a "real-time constraint" - i.e., operational deadlines from event to system response. By contrast, a non-real-time system is one for which there is no deadline, even if fast response or high performance is desired or preferred [40]. Hard real time systems require that a response to an event must be generated within a strict deadline failing which will cause the system to fail. Soft real time systems are able to tolerate delays in response even though a deadline is specified. Firm real time systems make up a third category of real time systems in which the only jobs that finish within the deadline are useful and the system is tolerant to some jobs missing their deadlines or not being executed when it is perceived that they would miss the deadline. Our work addresses simulations that belong to the class of firm real time systems and the specific deadline requirements are discussed in greater detail in Section 2.2.

2.1.4 Supercomputers and Batch Queues

Parallel supercomputers consist of multiple nodes connected via a high speed interconnect, with each node running its own operating system image. These supercomputers are built for targeted workloads and jobs perform best when isolated from one another and provided dedicated compute units, which may be processor cores or threading units on a single die. Due to this nature of parallel supercomputers and jobs running on them, special scheduling software is needed to maximize utilization. The most common setup divides the resource into classes attached to queues that receive jobs and the resource execute the jobs in a First In First Out manner using batch queue systems such as Torque, Loadleveler, etc. These batch queue systems typically consist of two components, the resource manager and the scheduler. The resource manager is responsible for interacting with the resources it manages and determining the current state and availability. It also provides the matchmaking capability to assign resources to jobs based on the job requirements specified by the user. Jobs are submitted to resource managers using a job description language that the resource manager understands. Resources are described using keys and values that can be matched against the requirements. The scheduler polls the resource manager for jobs and resource status and uses these to build a schedule for execution. The jobs are then dispatched to the execution nodes and managed by the resource manager. Modern

day schedulers add techniques such as backfill that allow jobs to execute out of order while the next in order job waits for sufficient resources as shown in Figure 2.1. Policies are used to control the behavior of a scheduler for example, fair-share is used to uniformly distribute resource usage in a multi-user environment. Since most modern day schedulers are priority-queue based schedulers, scheduling policies are implemented by manipulating the relative priority of jobs, causing jobs to execute out of order. These schedulers are optimized to maximize resource utilization in a multi user environment and use the number of resources and time requested by a task to build the schedule. The schedule does not take into account any information about the type of job itself and all jobs from a user are treated equally when building a schedule. The data-aware scheduling system will in contrast build a schedule based on the scientific significance of the simulation in relation to other similar simulations.

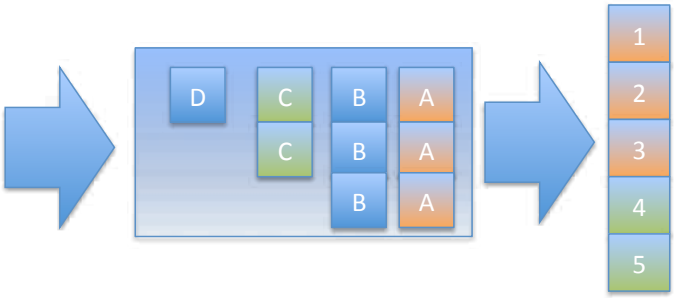

FIGURE 2.1: Backfill in FIFO queue allowing job C to execute out of order. The letters A, B, C and D denote jobs and the blocks denote the number of processing units requested (A needs 3, B needs 3, C needs 2 and D needs 1 processing unit).

Compute jobs that must be completed within a certain deadline need special scheduling mechanisms when being executed with other jobs on compute resources that rely on queues to order and execute jobs. Also local schedulers on resources are not aware of job level priority and deadline information and most schedulers schedule jobs using some Backfill and Fairshare scheme. Special queueing techniques as discussed in this work can be used to prioritize jobs that need to meet deadlines.

2.1.5 Meta-schedulers

A meta-scheduler is responsible for execution of jobs submitted to it. The meta-scheduler hands the job over to the local Distributed Resource Manager (DRM) systems that then schedule the job onto the nodes they control. Figure 2.2 shows a layered diagram depicting the interaction of the meta-

scheduler with the various components involved in execution management on the Grid [19]. These

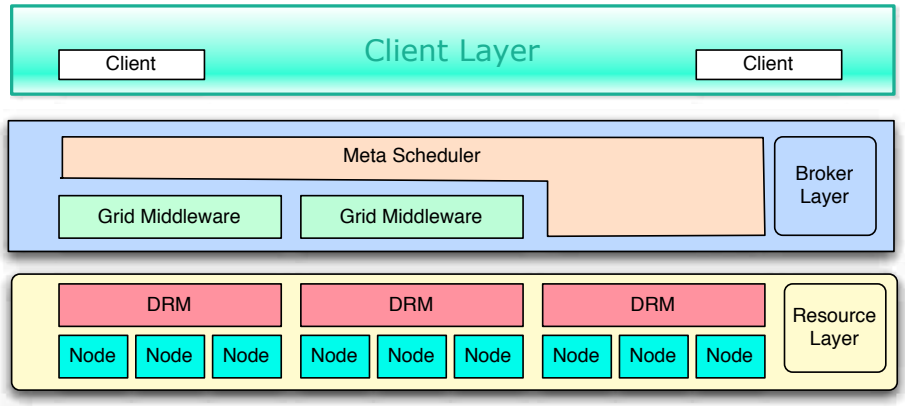

FIGURE 2.2: The Meta-scheduler interacts with Grid Middleware and Distributed Resource Managers (DRMs) to submit jobs to local DRM systems which then schedule the job onto the nodes they control.

meta-schedulers attempt to obtain information from local job schedulers and then use that to dispatch jobs to appropriate resources, thus acting also a broker of resources. The meta-scheduler hence has better global knowledge of the capabilities that resources provide, which as we will describe later is key to be able to build a suitable schedule for time sensitive applications. Since the local schedulers accept jobs directly from clients as well, the meta-scheduler may not always have the most up to date information about the resources controlled by the DRMs as is often the case when dealing with large Grid infrastructures such as TeraGrid where thousands of users submit jobs directly to the local DRM. The meta-scheduler is a key component in our architecture which is responsible for properly scheduling deadline sensitive simulations by determining the Quality of Service offered by various resources and scheduling the jobs on these resources based on certain metrics as will be discussed in Chapter 3.

2.2 Motivating Scenario

Hurricanes cause of massive destruction and loss of life and property when they intersect with populated areas. Even low intensity hurricanes pose a significant threat mandating an evacuation of the areas in and around the path of the hurricane. Hurricanes in the past have been responsible for large scale destruction both due to winds knocking down trees, power lines, buildings and also from flooding of coastal areas.

On the 29th August 2005, Hurricane Katrina (Fig. 2.5) hit New Orleans in Louisiana, causing massive storm surge and flooding, which resulted in a tragic loss of life and destruction of property and infrastructure. Soon after, Hurricane Rita caused similar devastation in the much less populated area of southwest Louisiana, and once again parts of New Orleans were under water. In both cases mandatory evacuations were enforced only 19 hours before the hurricanes made landfall. Speedier and more accurate analysis from prediction models could allow decision makers to evacuate earlier and with more preparation. Such hurricane prediction infrastructure is one goal of the SURA SCOOP Project. Figure 2.5 shows the various forecast tracks as released by NHC 2 days before it made landfall. As is evident from the predicted tracks there is a great deal of uncertainty in the predicted tracks and a reliable prediction can only be made when the storm center is closer to landfall.

The conditions during Katrina favored an accurate prediction when the storm center was a couple of days out but in most cases even 2 days is a stretch when trying to predict hurricane tracks. However with hurricanes the real interest lies in predicting the effects of the storm at the coastline and further inland, hence techniques like ensemble forecasting described in section 2.2.1.1 can be used to provide probabilistic bounds on the predicted values of storm surge and inundation. During an active hurricane the National Hurricane Center(NHC) produces an updated storm track every 6 hours and new input data is available. Modelers use this data to run various models to predict water levels, storm surge, inundation, etc. The models need to start running as soon as new data is available and visualized output needs to be made available as soon as possible to allow as much time as possible for emergency response activities. The absolute deadline for the runs is six hours since the data was available and the runs are useless for prediction if the deadline is missed as more current data is now available. The storm tracks vary substantially when the storm is further away from land and hence using new storm track data is important. The storm track data is received in six hour cycles starting at 00:00:00 UTC. The cycles will be referred to as 00Z, 06Z, 12Z and 18Z and the system starts receiving data as soon as a tropical storm forms from a tropical depression and continue until the storm weakens to a tropical depression or dissipates.

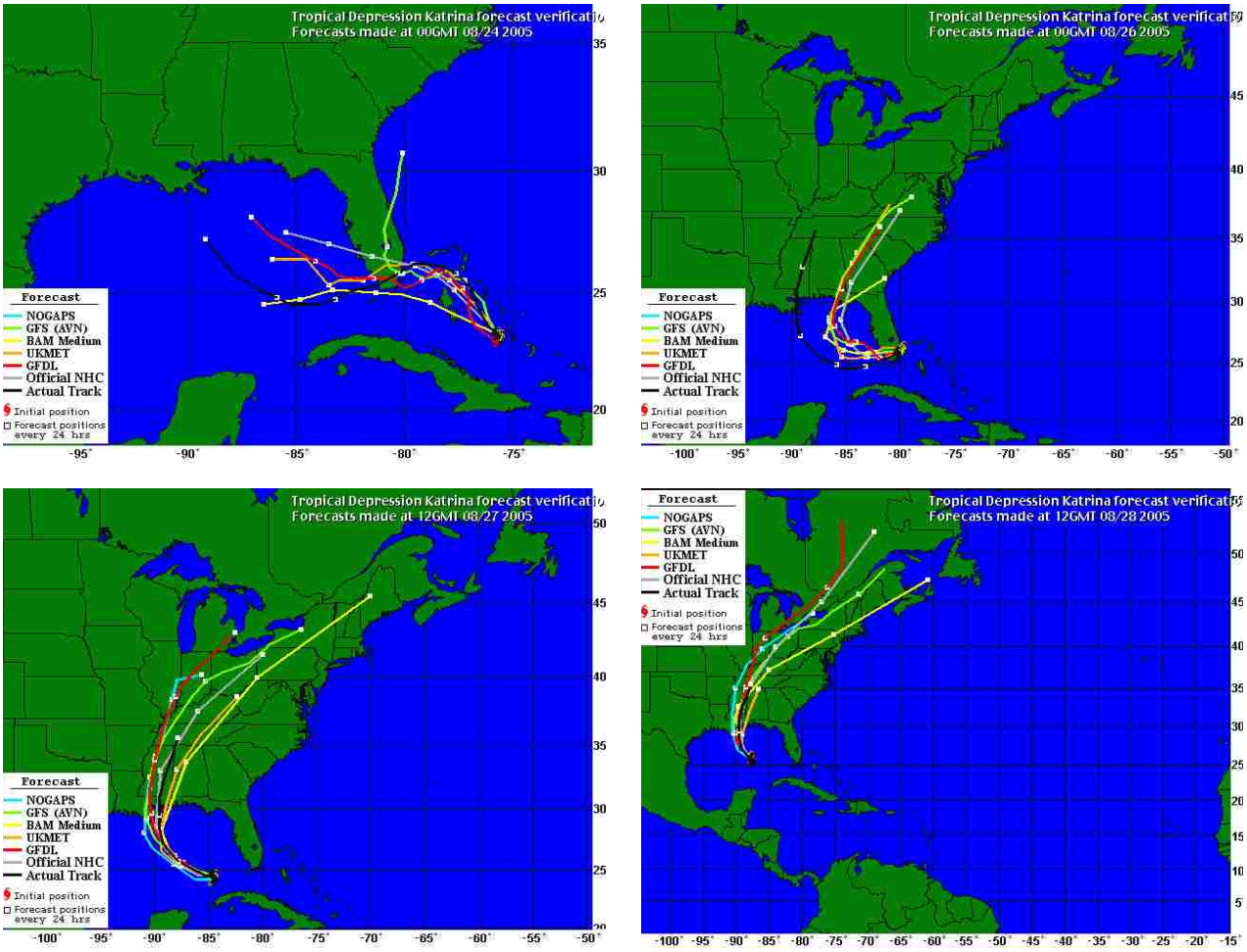

FIGURE 2.3: The forecast tracks as predicted by various models. Reading clockwise from top the first image shows the forecast tracks when the storm center was 5 days out, the next shows forecast tracks when the center was 3 days out followed by the tracks when the storm center was 2 days out and finally when it was 1 day away from making landfall.

Coordinated Universal Time (UTC) is a time standard based on International Atomic Time (TAI) with leap seconds added at irregular intervals to compensate for the Earth's slowing rotation. Leap seconds are used to allow UTC to closely track UT1, which is mean solar time at the Royal Observatory, Greenwich also popularly known as GMT. UTC provides a convenient way of describing timings of tropical events as they scale multiple time zones.

FIGURE 2.4: A hurricane day is divided into 4 sections with new advisories and storm track data received every six hours.

2.2.1 Ensemble Forecasting for Coastal Hazard Prediction

The scheduling problem that is the focus of this work is motivated by the problem of running hurricane forecasts in a timely manner. This chapter will introduce the coastal modeling scenario by describing the impact of hurricanes and the timeline requirements that the scenario places on the scheduling of hurricane simulations. We also introduce various efforts that have been focused at developing solutions for hurricane forecasting.

The Gulf of Mexico and the East coast of the America is prone to severe weather due to Tropical cyclones that develop over warm waters in the Atlantic ocean and travel westwards making landfall over continental USA and neighboring countries. The threat is immense: around half the U.S. population live in coastal areas, at risk from a range of coastal hazards including hurricane winds, storm surge, floods, tornados, tsunamis and rising sea-level. While changes in sea-level occur over time scales measured in decades or more, other hazards such as hurricanes or tornados occur on timescales of days or hours, and early accurate predictions of their effects are crucial for planning and emergency response.

Tropical cyclones are regions of low pressure that develop due to the latent heat of condensation that is released when water vapor rises from the sea surface and then condenses. Tropical cyclones are classified into various categories depending upon their maximum sustained wind speed. Table 2.1 shows the categorization of storms in the North Atlantic region.

TABLE 2.1: Saffir-Simpson Scale: Divides storms formed in the Atlantic Ocean and the Pacific Ocean east of the International Date Line, into categories distinguished by the intensities of their sustained winds.

Category	Wind Speed (mph)
Tropical Depression	0-38
Tropical Storm	39-73
Hurricane Cat 1	74-95
Hurricane Cat 2	96-110
Hurricane Cat 3	111-130
Hurricane Cat 4	131-155
Hurricane Cat 5	$\geq$ 155

The National Hurricane Center watches any activity in the Atlantic Ocean and the Gulf of Mexico for possible storms and generates storm tracks, indicating the storm's projected route and intensity at various points on the route, by running various oceanographic models such as the Geophysical Fluid Dynamics Laboratory (GFDL) model, United Kingdom Met Office model, automated tracker (UKMET), NWS/Hurricane Weather Research and Forecasting Model (HWRF) etc. Also released is the Official NHC forecast (OFCL) which is based on the various model outputs and input from coastal scientists [44].

FIGURE 2.5: Satellite image of hurricane Katrina which resulted in the loss of nearly 2000 lives and caused some $120 billion of property damage. The storm size at landfall was 460 miles, with 145mph winds (Category 3), and storm surges of up to 22 feet. [Image credits: MODIS Rapid Response Gallery].

2.2.1.1 Ensemble Modeling

Ensemble forecasting is a numerical prediction method that is used to generate a representative sample of the possible future states of a dynamic system. Ensemble forecasting is a form of Monte Carlo analysis: multiple numerical predictions are conducted using slightly different initial conditions that are all plausible given the past and current set of observations, or measurements. Sometimes the ensemble of forecasts may use different forecast models for different members, or different formulations of a forecast model [33]. The multiple simulations are conducted to account for the two sources of

uncertainty in weather forecast models: (1) the errors introduced by chaos or sensitive dependence on the initial conditions; and (2) errors introduced because of imperfections in the model, such as the finite grid spacings [34]. Ensemble modeling allows scientists to place a confidence interval on the results obtained from the simulations. Each ensemble consists of sets of runs which when run provide a larger confidence interval.

2.2.2 Data driven Coastal Modeling Pipeline

Data driven pipelines are a common architecture for components that rely on production and consumption of data products. The Coastal modeling pipeline is an example of one such data driven pipeline. The work described here has been carried out as part of efforts to develop and deploy two different hurricane forecasting systems. The first system, Coastal Emergency Risk Assessment(CERA) at LSU, is an operational hurricane forecast system that runs the ADCIRC model to generate storm surge data. The second system, SCOOP, is a project to develop a prototype of a distributed coastal modeling laboratory across the south-east USA.

CERA

The Louisiana Coastal Emergency Risks Assessment (CERA) [28] group is a coastal modeling research & development effort providing operational advisory services related to impending hurricane events and other coastal hazards. CERA works closely with various local, state and federal emergency response teams, including the Louisiana Governor's Office of Homeland Security & Emergency Preparedness (GOHSEP).

SCOOP

SURA Coastal Ocean Observing and Prediction Program (SCOOP) [20] is a multi-institutional effort to build a prototype of a distributed coastal laboratory to model and predict coastal events. The SCOOP project aimed at creating an open access, distributed laboratory for oceanographic scientific research and coastal operations by:

- Supporting the development and implementation of data standards that comprise the technical language of interoperability,

- Demonstrating the potential for integration and added value that occurs when disparate and diverse communities employ a common, standardized framework for information exchange; and

- Deploying the technical infrastructure to create an environmental prediction system that can be used as a research tool and handed off to the responsible entity that will use it to support the decision-making activities that benefit society.

The SCOOP program was a partnership between different institutions namely, The Souther Universities Regional Association (SURA), Gulf of Maine Ocean Observing System (GoMOOS), Louisiana State University (LSU), Bedford Institute of Oceanaography (BIO), Texas A&M University (TAMU), University of Alabama in Huntsville (UAH), University of North Carolina (UNC), University if Florida (UFL), Virginia Institute of Marine Sciences (VIMS) and Renaissance Computing Institute (RENCI) working together to define standards, share data and models across geographic and political borders, and develop a common infrastructure for automated modeling based on a service oriented architecture. One of the motivating scenarios for SCOOP is predicting the impact of tropical storms and increase the warning time that first responders and emergency officials have before an impending hurricane strike. The data products generated by SCOOP are available to researchers and officials via numerous avenues such as visualized output on websites such as at http://www.openioos.org as shown in Figure 2.6, raw data from the archives via file downloads and via LDM feeds to subscribers for various data products generated in SCOOP.

The SCOOP infrastructure including the various components and services and their interaction is described in Chapter 4. In order to understand why special treatment is needed for jobs that need to be completed within a deadline, we must study how jobs are executed on the resources. Most resource providers setup queues on the resource and jobs are submitted to these queues which are scheduled for execution based on the policies placed on the queue. The most common setups use prioritized batch queues that allow backfill to improve resource utilization. The primary execution order is First In First Out (FIFO) and priorities are used to implement fair-share. Resources also use multiple queues to separate different classes of jobs and prioritize them based on these classes.

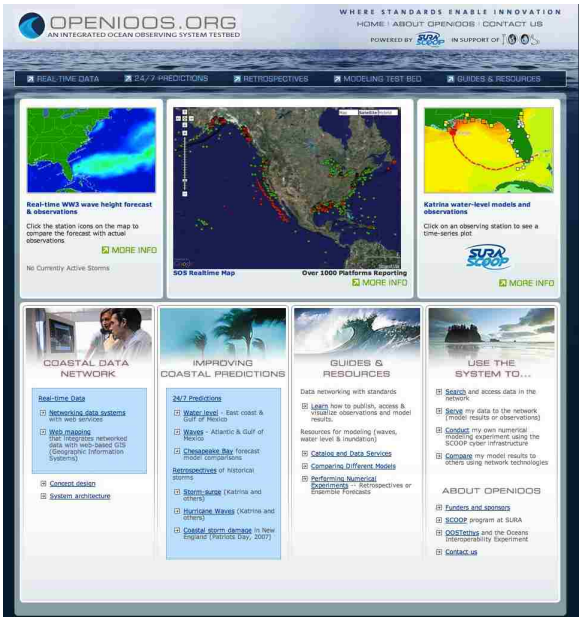

FIGURE 2.6: The OpenIOOS website powered by SCOOP data products

In this chapter we introduced the various scheduling algorithms and policies found in traditional supercomputers, the special requirements of real-time systems, the role of meta-schedulers in distributed supercomputer scheduling and introduced the motivating scenario for this work. These concepts are key to building a data-aware scheduling system, that needs to schedule deadline-sensitive jobs on distributed supercomputers.

Chapter 3
Data-aware Scheduling

Traditional scheduling mechanisms used by HPC resources and meta-schedulers are static with respect to attributes of the data, such as file names, data sizes, data types and the data itself, being used by the job and also the Quality of Service classes offered by resources. The Quality of Service classes need to be provided along with the job, for HPC schedulers to use as part of scheduling the jobs. However to support the scenario described in Chapter 2, we need a scheduling mechanism that takes into account the attributes of the data and also adjusts the Quality of Service classes for jobs automatically and dynamically based on certain preset metrics. We introduce "Quality of Service based data-aware scheduling" as the means to achieve the dynamic behavior desired by time sensitive scenarios described earlier.

In this chapter we provide the theoretical and analytical background that shows that data-aware scheduling is indeed a solution to the problem of providing measurable guarantees for running time sensitive simulations on shared resources and also explain how Quality of Service levels are used to provide these guarantees. We introduce the specific algorithm used and describe how the metrics for data-aware scheduling are computed for the ensemble-based hurricane modeling scenario. The second important aspect to running deadline sensitive jobs is to be able to provide guarantees on jobs meeting their deadline. This in turn means that there needs to be a way of determining what are the various Quality of Service (QoS) levels offered by the resources and how these can be mapped to the priority of jobs.

Formally, in the context of deadline sensitive scheduling, we define Quality of Service as the confidence that can be placed in a job completing within its specified deadline. Quality of Service levels provide different confidence values from low to high. Resources may provide various QoS classes (levels) and we build an ordered set of these QoS classes to map jobs with various priorities to these QoS classes.

If P is the set of priorities assigned to jobs and Q is the set of QoS levels, the QoS function is defined as,

$$QoS = (P, Q, \{(p, q) : p \in P, q \in Q\}) \tag{3.1}$$

where p is the priority assigned to a job and q is a QoS level. In order to be able to generate the mapping of jobs to QoS levels using a priority value p, we also need a way to generate these priority values based on attributes of the job. We call such a function as the priority function PF and will define it as,

$$PF = (J, R, \{(x, i) : x \in J, i \in \mathbb{R}\}) \tag{3.2}$$

where J is the set of jobs and $\mathbb{R}$ is the set of Real numbers.

$PF(m)$ returns a real number that represents the numeric representation of the priority of job m. It should be noted that for the purpose of definition we specify no bounds on the value returned by PF, however specific scenarios may formulate the function so that it is bounded as will be seen in the case of the SCOOP priority function described in Section 3.1 where the priority is a function of the set membership and the deadline information. It is assumed that the priority function will be defined by application owners who have the knowledge of how to classify jobs and assign importance to them. The priority function can be as simple as a mapping between two ordered sets or may take into account multiple variables.

3.1 Priority and QoS levels for SCOOP Scenario

In this section we will discuss the specific priority values and the QoS levels that were used for the coastal modeling jobs in SCOOP. As described in Section 2.2.1.1, ensemble modeling is used to provide guarantees on results generated from the simulations. These ensembles are divided into different sets based on the level of confidence they provide on the results of the simulations. For the SCOOP scenario there are six ensemble sets each with a different number of ensemble members. Each set is a superset of the previous one, so that if we name these sets E_1, E_2, E_3, E_4, E_5 and E_6 then

$$E_1 \subset E_2 \subset E_3 \subset E_4 \subset E_5 \subset E_6 \tag{3.3}$$

The first set E_1 has no perturbed tracks and the runs are based on actual forecast tracks from different models. Table 3.1 shows the number of members in each set:

TABLE 3.1: SCOOP ensemble sizes

Set	E_1	E_2	E_3	E_4	E_5	E_6
Ensemble Size	2	5	10	16	34	88

To generate initial results it is sufficient to run just the tasks in the E_1 set, but to provide increasing stringent guarantees on the correctness of the results the new members of other sets must be run. Since E_1 contains the tasks deemed most important by the coastal scientists, these tasks in E_1 are assigned the highest priority and the members of E_2 not in E_1 get the next highest priority and so on. It can be seen that the priority of the runs depends on which set E_i with the lowest index i they belong to. We will henceforth refer to an ensemble member as m and the set of ensemble members as E.

Equation 3.4 shows the relation between set members and their priorities. The sets E_1 to E_5 enforce an oder on the priority of members based on which sets the members belong to.

$$\text{If } M_i \in E_x \text{ and } M_j \notin E_x \text{ and } M_j \in E_y \text{ then } PF(M_i) > PF(M_j) \text{ iff } x < y \tag{3.4}$$

If D_m is the deadline for member $m \in P_j, 1 \leq j \leq n$, where n is the number of sets of ensemble members then,

$$PF(m) = \begin{cases} (n-j+1)/n & \text{if Deadline } D_m \text{ has not passed,} \\ 0 & \text{if Deadline } D_m \text{ has passed.} \end{cases} \tag{3.5}$$

$PF(m)$ is the priority function for member m. As shown in Equation 3.5 for the SCOOP scenario, $PF(m)$ is chosen to be between 0 and 1, 0 indicating a do not run status and 1 indicating the highest priority which translates to an on-demand job which must be run within the deadline. It would suffice to use a simple mapping between priorities and QoS classes but we chose the linear relation of equation 3.5 to easily accommodate more sets than 6 and also place upper and lower bounds of 0 and 1 respectively on the priority values.

Now that we have defined the Priority function for SCOOP ensembles, we will discuss the various QoS levels available for running SCOOP jobs and how these QoS levels were implemented on the resources.

The LONI and SURAGrid resources provided three levels of Quality of Service. The highest level is preemptive access to resources, followed by a next to run status in the queue and finally a best effort level, which offers no priority. For jobs to finish at the earliest on an already busy resource we make use of preemptive scheduling that kills running jobs to free the required number of execution units. We will call jobs that used the highest QoS level of preemptive access, "urgent" jobs. In the SCOOP scenario since the jobs have different priorities not all jobs needed to have preemptive access. Sometimes it is sufficient to place the jobs at the top of queue and wait for the existing job to complete. Such jobs are called high priority jobs and receive "next to run" status. Jobs that are submitted to queues and wait for their turn to begin execution are termed best effort jobs. Hence for SCOOP our set of ordered QoS classes is

$$Q = \{\text{Preemptive, Next to Run, Best Effort}\} \tag{3.6}$$

The Quality of Service that a job receives is a function of various parameters. These include the user set or computed priority of the job, the time remaining to the deadline and the status of other jobs in the queue, for example there is no point providing next to run status to a job if current long running jobs will make the next-to-run job miss its deadline. We do not take into account such conditions in our algorithms, however as will be seen from the results, on large resources (relative to the size of the job) this is usually not a big issue and can be ignored. Better resource brokers that can provide such information to the scheduler will need to be developed and when this is possible the scheduler will be able to make complete use of this information when placing jobs into the right QoS classes. Our goal is to develop an optimum schedule that maximizes the number of jobs that complete within their specified deadline using the information available to the scheduler.

An important property of the jobs is that they arrive at the scheduler for execution in batches. Hence if the scheduler waits for a specified interval before starting to execute the jobs, it can make better scheduling decisions based on these jobs. It is however not guaranteed that all jobs will arrive within the specified interval, hence the scheduler must adopt a dynamic schedule and evaluate the candidacy of jobs for execution at every scheduling interval. The time period that the scheduler waits for jobs to arrive is called the Dispatch Threshold Time (Th_D). The scheduling interval is a constant

time and will be represented as SI. In order to compute the execution order of the jobs, we need to come up with a mathematical function that can be used to order the jobs within their Quality of Service domains. We will treat each of these QoS levels as a separate queue that can receive jobs. The queues will be named U_q for the urgent queue, N_q for the next to run queue and B_q for the best effort queue. The submit time of the job will be represented as S_i for job i and this is a real instance of time. The length of the deadline window (the time difference between the submit time and the deadline) will be denoted as D_i for job i.

First let us consider jobs whose priority is set as urgent. The scheduler reads the priority parameter from the submission script and finds that it is an urgent job. It then checks with the resource manager to see if there are sufficient on-demand resources available for execution of the job. If so it places this job in U_q. The availability of the resources means that on-demand resources will be available at some point in time such that the execution of the job will start and execution will complete before the deadline. The resources need not be available immediately. On-demand resources can be provided using a variety of means such as using advance reservations, dedicated resources, preemptive queues and compute clouds. The scheduler chooses between these based on the availability and cost of the resource. The economics of finding resources for computation leads to the requirement of a cost model that can be used to determine the appropriate resource for the execution of a job. In this work we will not address the details of developing such a cost model, but instead focus on developing the system that can make scheduling decisions based on the relative significance of jobs and the QoS levels provided by resources.

One of the techniques that is often used to guarantee resources for a job at a particular time is the use of advance reservations. Advance reservations are used to block off a subset of the system and only allow the the owner of the reservation to run jobs in the reserved time slot. Advance reservations sometimes cause the scheduler to generate non-optimal schedules that wastes CPU cycles. Also using advance reservations requires precise knowledge of when and how many resources will be needed well before the time of the reservation. On resources with long queue times this can mean requiring a long lead time. Most supercomputing centers have queue limits set to an average of two days of wall time. This means that to guarantee a time the reservations must be made at least two days in advance on

a resource that is close to 100% utilization. Two days is a long time on the hurricane time scale and forecasts greater than two days are often inaccurate. This results in scientists reserving large chunks of resources for a long duration (multiple cycles) when they are only needed for much shorter time spans.

Jobs that have a QoS parameter of high priority are treated as important jobs that are not allowed to preempt other running jobs. Hence these jobs must be submitted to the resource that has the best chance of running it to completion before the deadline based on the current running jobs. The third QoS parameter is the best effort level of completion guarantee. These jobs are considered less important than the other two previous categories. That being said these must still be scheduled so that they have a good chance of completion before the deadline, while being limited by the local schedulers and the policies they enforce on all jobs. Although this might be the most challenging and interesting aspect of scheduling it is not the most fruitful to solve, which is why our proposed three level QoS scheduler focuses on splitting up the jobs into categories that identify the perceived importance of the job.

Chapter 4
Coastal Modeling Pipeline

We described in Section 2.2 the coastal modeling scenario and the need for ensemble modeling, thus establishing grounds for the need of a coastal modeling pipeline. The data-aware coastal pipeline draws most of its components from the data-driven pipeline. In this chapter we will go into greater detail in describing all of the components and services that make up the coastal modeling pipeline as implemented for SCOOP and CERA. The need for an automated pipeline is obvious in time-sensitive systems as human input into the system unless necessary can cause unwanted delays and shrink the window of the advance warning time that is so crucial when dealing with hurricane events. SCOOP provides multiple services such as the Coastal Data Archive, SCOOP catalog, SCOOP Portal etc. that support the distributed coastal laboratory. It was envisioned that a coastal scientist would use the SCOOP portal to interact with the SCOOP services and configure them to his liking. This included selecting the number and frequency of runs, the input data used for example, analytical or observed wind forcing data, type of output to generate such as storm surge graphs or inundation maps, etc.

The SCOOP coastal modeling pipeline is composed of various components and services that interact with each other to orchestrate a complex workflow that involves receiving input data from various sources such as NHC, submitting the models, collecting the output and generating the necessary data visualization. We will call the SCOOP coastal modeling pipeline the data driven coastal modeling pipeline and in our analysis of the data-aware scheduling use it to compare various metrics related to performance of such pipelines.

4.1 Pipeline Components

4.1.1 Data Transport

SCOOP uses both existing legacy data transport mechanisms such as the Logical Data Manager (LDM) [31] and state of the art Grid based data transfer mechanisms in different parts of the pipeline. LDM is a widely used data transport tool based on a push model where receivers "subscribe" to data feeds and senders push out data on channels or feeds to subscribed receivers. LDM is designed for

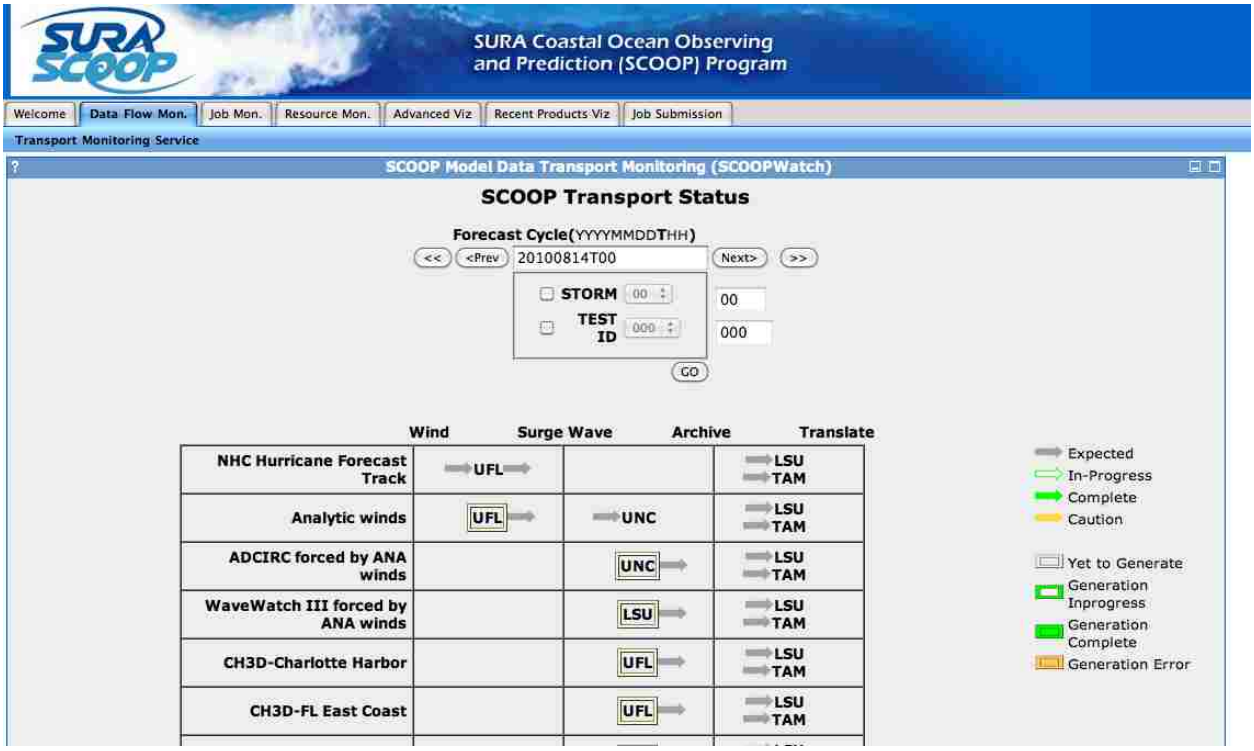

FIGURE 4.1: The scoop portal showing a portlet that visualizes the flow of data between various services hosted at different locations.

event-driven data distribution developed and maintained by Unidata and includes network client and server programs and their shared protocols. LDM is used in SCOOP to manage transfer of data products from various sources to the archives. LDM has the facility to trigger arbitrary actions based on the file names of received data products. The data-driven coastal pipeline used this triggering mechanism to initiate archiving process and also trigger model workflows for ADCIRC. This design was later changed in favor of a centrally orchestrated workflow trigger mechanism that had a better global picture of the data products received and generated. The new mechanism and its role in the data-aware pipeline will be described in detail in Section 4.2. GridFTP [32] is used as the data transport mechanism of choice when moving data products from the archives to High Performance Computing resources where models are executed and to transfer the output back to the archives. GridFTP is a high-speed transport service which extends the popular FTP protocol with new features required for Data Grid applications, such as striping, parallel transfers and partial file access.

The data products used and generated in SCOOP were required to follow a file naming convention [35], which was used to identify various meta-data attributes of the data, such as the Grid identifier, forecast dates, data source, etc.

For example, WANAFe01-UFL_20050825T0000_20050825T0000_20050830T0000_12gsr1_Z.nc.gz is the netCDF analytical wind product generated by UFL with a model initialization time of 20050825T0000 and the forecast start and end dates of 20050825T0000 and 20050830T0000 respectively for storm number 12.

4.1.2 Data Archive

All of the input data received by the pipeline and the output data generated by the system is stored in the data archives. The SCOOP data-driven system used a system of two redundant archives housed at Louisiana State University and Texas A&M University. The archives also act as sinks and sources for the LDM transport, receiving data products generated at various sites and maintaining the data in the two archives in sync. The architecture 4.2 and working of the LSU SCOOP Archive as described in the paper by Maclaren et. al [29] is described below:

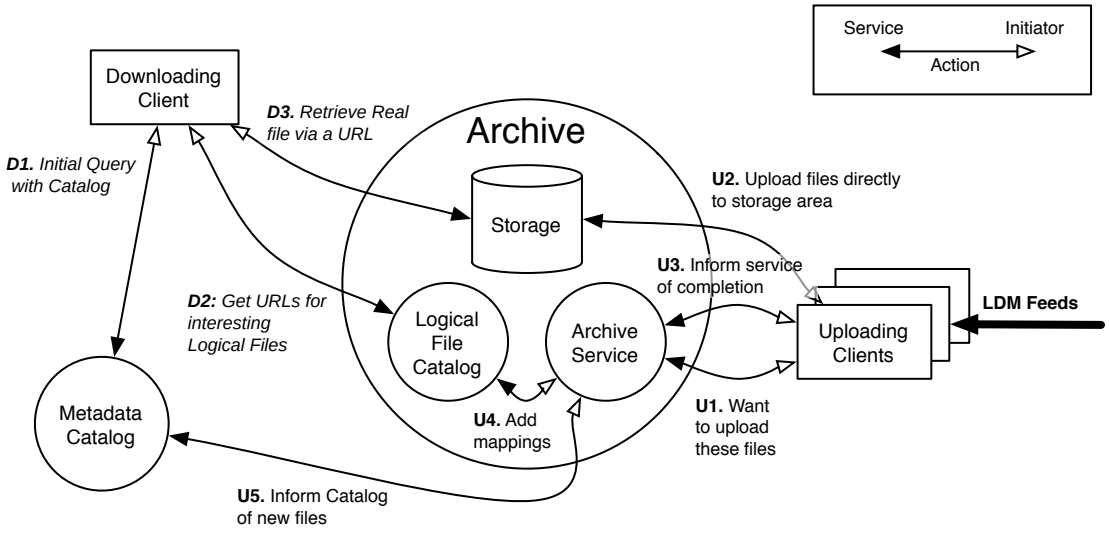

FIGURE 4.2: The LSU Scoop Archive [29] architecture showing the interactions between the internal components upon a file ingestion triggered by a client.

U1 The client contacts the archive, providing a list of files which need to be uploaded to the archive. The archive decides where each file will be located within the archive. The archive's response

groups the original files into one or more **transactions**, each of which is associated with: a location in the storage area (specified by a set of URLs for various scheme names); a subset of the list of files; and an identifier which the client uses in later interactions.

The following steps are then carried out for each of the transactions.

U2 The client uploads the files to the archive storage with some third-party software, e.g. GridFTP [32], and a URL given to it by the archive.

U3: request After the client has uploaded the files to the location, it informs the archive that the upload is complete (or aborted), using the identifier.

U4 The archive constructs Logical File Names (LFNs) for the files which have been uploaded, and adds mappings to Logical File Catalog that link the LFNs to various URLs that can be used to access the physical files.

U5 The archive informs the catalog that there are new files available in the archive, providing the LFNs.

U3: response The archive returns the LFNs to the client.

The steps for downloading a file from the archive are as follows:

D1 A client queries the **Metadata Catalog** to discover interesting data, e.g. ADCIRC Model output for the Gulf of Mexico region, during June 2005. Some Logical File Names (LFNs) are returned as part of the output, together with a pointer to the Archive's Logical File Catalog.

D2 The client chooses some LFNs that they are interested in, and contacts the Logical File Catalog service to obtain the files' URLs.

D3 The client picks URLs (based on scheme name, e.g. `gsiftp` for GridFTP) and downloads the files directly from the Archive Storage.

4.1.3 Data Catalog

This Scientific Catalog for Open Resource Exchange (SCORE) [30] encompasses a comprehensive information repository for SCOOP, and a suite of web services using WSDL, SOAP, and XML messages to provide low level access to the catalog. SCORE's catalog services provide for automated interactions by other SCOOP partners such as the archives, and by SCOOP applications such as workflow tools. In addition, they provide the basis for search applications designed for human interaction. The Catalog is a key component because it contains essential data and meta-data based on which the data-aware system will make its scheduling decisions.

4.1.4 Workflow Trigger

The LSU Archive [29] provides additional features such as a trigger mechanism that allows workflows to be initiated once the data products have been received and stored in the archive. The archive uses the file naming convention to trigger appropriate workflows for example, the analytical Wind product files trigger the ADCIRC and WWIII workflows and the storm surge output data file triggers the visualization workflow.

4.1.5 Workflow Manager

Once the archive triggers the workflow, the workflow manager orchestrates the workflow and manages the execution of the jobs on the distributed supercomputers. We choose DAGMan [17] as the workflow manager for orchestrating SCOOP workflows. DAGMan was chosen because the SCOOP workflows were simple consisting of preprocessing, model execution and post-processing and did not need complex workflow tools such as Pegasus [18], Triana [26], Kepler [25] etc. Workflow managers and their capabilities will be described in more detail in the related work chapter.

4.1.6 Grid Middleware

The data-driven pipeline's primary platform is TeraGrid and LONI, which are both Globus [14] based grids. Hence we needed to have a way to submit to Globus based resources. Since we chose DAGMan as our workflow manager, which uses Condor [15] to manage jobs, we use Condor-G [16] to submit to the TeraGrid and LONI Globus based resources.

4.1.7 Monitoring

Monitoring is a key component of any real-time system as the system needs to keep track of successes and failures and be able to recover from errors in a timely and elegant fashion. The SCOOP system provides both data and job monitoring features. Data monitoring is provided using the SCOOP Data transport monitor, developed by the University of Alabaman at Huntsville, which monitors data as it moves from one point to another and keeps track of the products generated by various sites and the products received at various sites. The data transport monitor uses information from the Catalog to determine the expected number and type of data products and is able to detect missing files at different points in the system. Job Monitoring is performed using UNC's notification service that is used to report the status of jobs at different stages in the workflow execution, such as pre-proessing, model execution and post-processing.

4.1.8 Simulation Models

Coastal modeling is performed using legacy and usually closed source or restricted code, and models are generally not versatile or standards compliant, for example, data formats such as netCDF are only now being established and being implemented by models. These features have restricted current models to be run only in constrained environments, with significant effort involved in porting these models to various HPC resources. The SCOOP project aimed at developing a production level end-to-end dynamic data-driven solution for coastal modeling using these legacy coastal models and making them part of the automated system that uses standard data formats and defines standards for file naming conventions. This system receives input data such as storm track, central pressure, wind speed etc from various sources such as the National Hurricane Center (NHC) and uses it to run various coastal models such as ADCIRC, Wave Watch III and CH3D to predict storm surge, wave height and inundation levels.

- **ADCIRC** (Advanced Circulation) is a system of computer programs for solving time dependent, free surface circulation and transport problems in two and three dimensions. These programs utilize the finite element method in space allowing the use of highly flexible, unstructured grids. Typical ADCIRC applications have included modeling tides and wind driven

circulation, analysis of hurricane storm surge and flooding, dredging feasibility and material disposal studies, larval transport studies, near shore marine operations. In SCOOP ADCIRC was used primarily for storm surge forecasting.

- **CH3D** is a Curvilinear-grid Hydrodynamics 3D model developed originally by Dr. Y. Peter Sheng at the Aeronautical Research Associates of Princeton, Inc. (ARAP, now part of Titan Corporation) during 1983-1986. The CH3D model uses a horizontally boundary-fitted curvilinear grid and a vertically sigma grid, and hence is suitable for application to coastal and nearshore waters with complex shoreline and bathymetry.

- **WAVEWATCH III** [42] is a third generation wave model developed at NOAA/NCEP in the spirit of the WAM model [43]. It is a further development of the model WAVEWATCH, as developed at Delft University of Technology and WAVEWATCH II, developed at NASA, Goddard Space Flight Center. WAVEWATCH III (WWIII), however, differs from its predecessors in many important points such as the governing equations, the model structure, the numerical methods and the physical parameterizations. The WWIII model generates several gridded output wave fields including mean wave water depth, wave frequency, wave direction and spectra. Table 4.1 describes the run times for 1 ensemble track with a 0.2 resolution grid and includes times for pre and post processing.

TABLE 4.1: Wave Watch III benchmark results for the SCOOP scenario on LONI and LSU resources

Procs(↓)/Time(hh:mm:ss)(→)	Eric	Pelican	Ducky	Zeke	Bluedawg
2			02:56:19	02:54:37	02:59:16
4	00:56:22	01:31:22	01:26:48	01:25:22	01:28:37
8	00:30:45	00:49:35	00:43:00	00:44:35	00:44:13
16	00:15:28	00:32:11	00:23:00	00:21:29	00:23:14
32	00:09:08	00:17:18	00:12:55	00:11:04	00:13:30
64	00:05:35	00:10:12	00:08:03		
104	-	-	00:07:25	-	-
128	00:04:20	00:06:41			
256		00:05:11			

The system that was developed was capable of archiving and cataloging input data, pre-processing the input data, triggering the appropriate models and then post-processing the output data. The existing system makes use of tools such as the Local Data Manager (LDM) developed at University Corporation for Atmospheric Research (UCAR), to push data to sites that can consume this data such as data archives. The archives then store and catalog the data using the Archive [29] and Catalog [41] Services. SCOOP uses a set of geographically distributed redundant archives for storing input and output data. Figure 4.3 shows the production SCOOP system based around LDM. Input data that was pushed out to LDM is received by the archives and the ADCIRC workflow system. The ADCIRC workflow is triggered as soon as data is downloaded from the LDM pipe. The Wave Watch III model workflow is triggered by the archive once the data has been received from LDM, archived and cataloged.

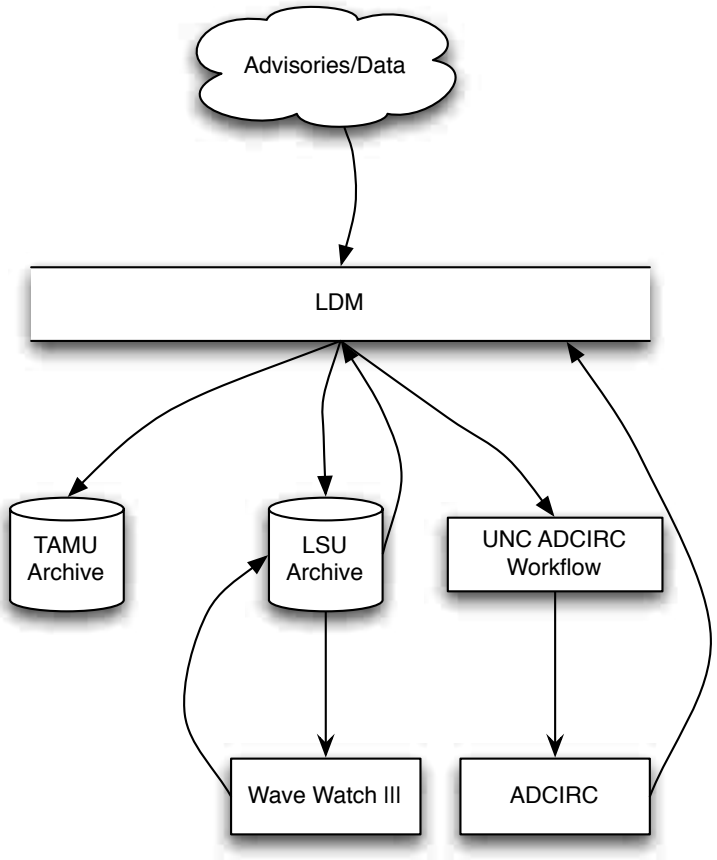

FIGURE 4.3: Flow of data and control through the legacy SCOOP production pipeline. The architecture is centered around LDM and all data in the system is pushed out to LDM and received by subscribers.

The data-driven implementation of the coastal modeling pipeline was a major improvement over the previously manual pipeline systems and it contributed several key components that will feature in the new data-aware modeling pipeline.

4.2 Data-aware Coastal Modeling Pipeline

The coastal modeling pipeline developed for SCOOP and described earlier in this chapter is an end-to-end system that was driven based on availability of data. The SCOOP system succeeded in developing a Service Oriented Architecture that fully automated the hurricane simulation process. However in a resource constrained environment the key challenge is to run all the necessary ensemble members within their deadline. There needs to be a way to make smart decisions about scheduling the right ensemble members for execution. The large size of the ensemble and the large individual model runtime requires that multiple resources are needed to finish all the runs. Also on large resources medium sized jobs have lower priority, hence using distributed resources is essential for scheduling deadline sensitive simulations. A system driven strictly by data availability loses the key capability of being able to react to the data content. The data-aware system aims at making scheduling decisions based on the information present in the data and the metadata. This translates into a system that knows about the scientific significance of the data and uses it to make scheduling decisions. The data-aware system also has another key difference aimed at improving the turnaround time of jobs that have a higher scientific significance than other jobs for which data is available. This is the use of Quality of Service levels provided by the resources that execute the models. The scheduling system is now aware of the QoS levels provided by various resources and can make smart scheduling decisions based on these parameters.

Figure 4.4 shows the architecture of the data-aware coastal pipeline, showing the interaction between various components. The Simulation Application Manager is at the core of the architecture, interacting with the Catalog to keep track of available data products, status of the storms and uses this information to generate the proper description of urgency and priority requirements and sends these to the scheduler to execute. The scheduler determines the appropriate schedule and resources and dispatches the job to the workflow manager that executes the task on the selected resource. Files

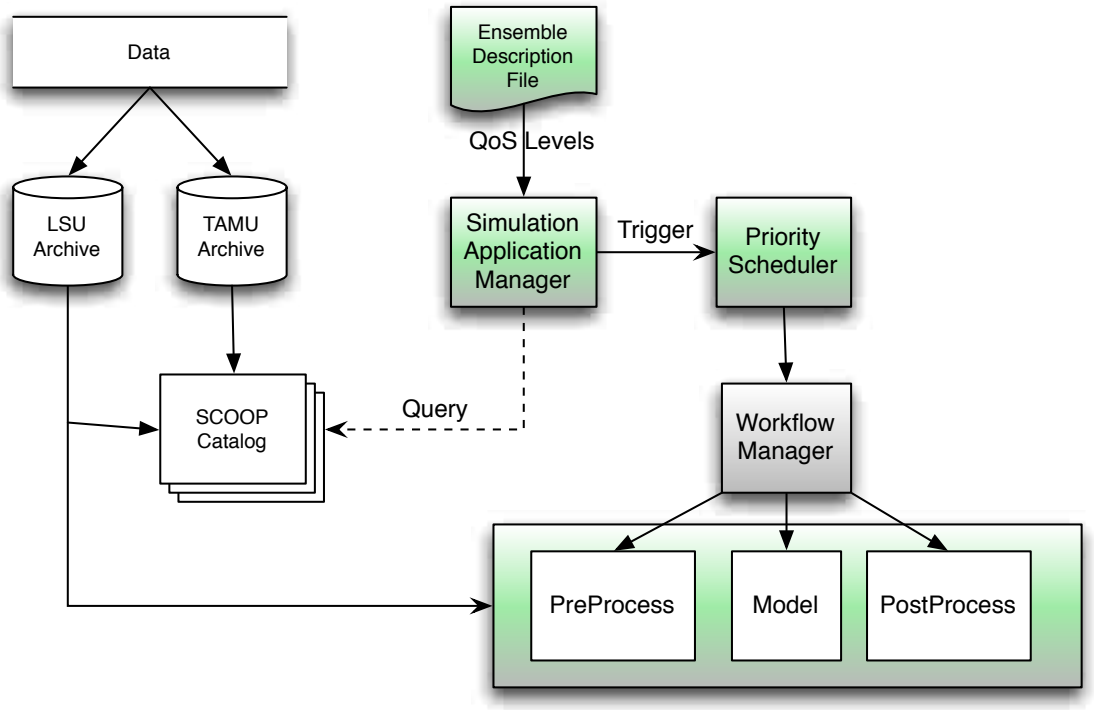

FIGURE 4.4: The data-aware coastal modeling pipeline: The Simulation Application Manager (SAM) [36] sits at the heart of the system coordinating with the Catalog and scheduler, while the archives interact with the Catalog to register data products.

are staged in and out as part of the workflow and all output data is ingested by the archives and cataloged. The availability of visualized output triggers the Visualization service and the visualized data products are generated and once again archived for consumption as part of web sites and for information dissemination.

Listing 4.1 describes algorithmically the decision making process using the Priority notation described in Chapter 3.

Listing 4.1: Data-aware Scheduling algorithm

```
Receive model data for job m.
Analyze model data and determine Quality of Service (q) needed.
If current time + run_time > deadline then
        Set PF(m) = 0 and QoS(m) = Best Effort.
else
```

```
            PF(m) = (n - j + 1)/n, where n is number of sets
             and m belongs to set j
            Set QoS(m) to Best Effort, High Priority,
             or  On-demand based on PF(m) value.
  end if
 Insert job into the right QoS level's priority queue.
START: Set queue = On Demand
    WHILE: while queue is not empty
         Dispatch the job at the top of queue
      end while
      Set queue = High Priority
      if queue is not empty GOTO WHILE
      Set queue = Best Effort
      Dispatch job at the top of queue
GOTO START
STOP
```

The algorithm makes use of the priority function and the QoS function in the decision making process. When a model is received for dispatch by the scheduler, it evaluates if it is possible to complete the execution of the job before its deadline, using the estimated run time data based on the benchmarking information of the model on the resource. If the job will miss the deadline, it is placed in a best effort queue with a priority of 0. This signifies that although this job was important if it would meet its deadline, the fact that it is impossible to run it so that it finishes before its deadline makes it no longer important. It must be noted that no job is discarded and hence the lowest Quality of Service and Priority is assigned to the job. When the scheduler finds multiple jobs with the same priority value it treats them on a first come first serve basis.

If the job will not miss its deadline, the scheduler evaluates the value of the priority using the SCOOP priority function given in equation 3.5. If a priority value is specified in the Ensemble Description File, it is used to override the value returned by the function for that ensemble member. Once the priority value is known, the job is placed in the appropriate QoS level, based on the priority to QoS mapping function. The scheduler operates on a scheduling cycle and at every cycle it dispatches all jobs in the on-demand queue, followed by jobs in the high priority queue in priority order and then dispatches jobs in the best effort queue one by one in priority order. If any jobs make it into the On-demand or high priority queue, they are dispatched before the next job in the Best-Effort queue is dispatched.

4.2.1 Simulation Application Manager

To make use of these parameters in scheduling jobs the system architecture was redesigned to improve the flow of information in the system. The Simulation Application Manager was introduced which is the key component that consolidates the information in the system and uses it to make the scheduling decisions. The architecture also introduces the archives as the data consumers and SAM is made aware of data availability after the archives receive the data and catalog it with the Catalog service. This ensures that data safely arrives into the system and is duly archived and cataloged. In the traditional system, LDM's queue subscription mechanism needed the LDM client machines to be extremely fast and responsive to avoid data loss. Although multiple data receivers existed it was not easy to sync these and recover the lost data and then re-trigger the runs. SAM with the help of the Catalog services is able to keep track of the data received by the archives and uses this information to point the execution resources to the right location for the input data. This adds the much needed reliability and fault recovery components into the system in addition to the smart application aware scheduling. Figure 4.5 shows the flow of control and data through the SCOOP pipeline. The system can be triggered into operation by the arrival of a National Hurricane Center(NHC) advisory or manually using the portal interface. The system is still data-driven and the analytical Wind Generation model, Windgen, is triggered by availability of new track data at the FTP sites hosted by NHC. If a new forecast track is found, analytical winds are generated by the Wind Generating Service hosted at University of Florida, Gainesville. The wind data is then pushed onto LDM for the archives to

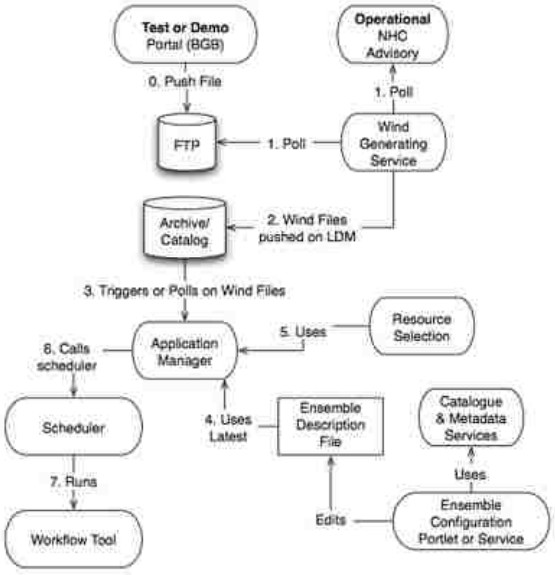

FIGURE 4.5: Flow of data and control through the SCOOP data-aware pipeline. The numbers show the ordering of events as they occur at various components in the pipeline

receive and Catalog the data. The archives hosted at LSU and TAMU receive the wind data and store them locally and register the locations along with additional metadata with the SCOOP catalog service hosted at University of Alabama, Huntsville. The Wind Generating Service also registers the generated wind data with the Catalog service hence all data received at the archives can be validated and data can be retransmitted if necessary. Once the files are cataloged SAM becomes aware of them and can use the additional information it has to attach priority information to the runs that will be sent for scheduling. SAM uses an Ensemble Description File as shown in Listing 4.2 to select the ensemble members and their priority values. This EDF is usually created by the coastal scientist and is used by SAM to set the priority attributes of all possible model runs based on the input data.

Listing 4.2: The Ensemble Description File is used to express the urgency and priority of ensembles. This information is later used to schedule the jobs.

```
<ensembledescription name="Default_Ensemble_2007"
    lastModified="2007-08-28+09:00">
    <storm num="12" name="Katrina" date="2005-08-28+09:00"/>
    <ensemble size="2" creationTime=2011-03-17+12:29"
            lengthForecastHrs="120">
```

```
      <member id="1" priority="0.8">
          <track>e01</track>
          <model>WW3</model>
          <forcing>ANA</forcing>
          <region>Gulf</region>
          <config>0.2</config>
          <hotstart>No</hotstart>
          <comment></comment>
      </member>
      <member id="2" priority="0.5">
          <track>p02</track>
          <model>WW3</model>
          <forcing>ANA</forcing>
          <region>Gulf</region>
          <config>0.2</config>
          <hotstart>No</hotstart>
          <comment></comment>
      </member>
    </ensemble>
</ensembledescription>
```

In the data-driven SCOOP system jobs are sent to the resources as they arrive. Hence a complete set of jobs gets queued at the resource in the order that these jobs arrive every six hours during an active hurricane. Without a sense of urgency and priority all jobs are treated equally and the system makes no use of the information about the jobs. In the hurricane simulation scenario a FCFS scheduling of the jobs causes less important jobs to be scheduled in front of the more important jobs. Since hurricane events are long events a delay introduced in any set of jobs causes the important

set of jobs in the second set to get delayed. Any delay in the chain is propagated downwards and effects the turnaround time of the important jobs namely those in set E_2. The problem is further complicated when events such as more than one hurricane cause the systems to get flooded with jobs and in such a scenario with no proper ordering of jobs it is impossible to preserve supercomputing units and deliver reliable results obtained from running the most important tracks. The existing systems used for storm surge prediction are mostly orchestrated using a set of scripts that coordinate the triggering of runs upon receiving the advisory, submitting the jobs to the supercomputer and then collecting the results and transferring them back for analysis and visualization.

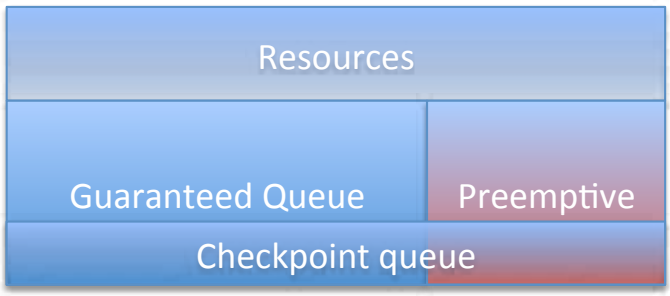

FIGURE 4.6: Preemptive queues where preemptable jobs have access to all resources and non-preemptable jobs have access to a subset of the resources

The SCOOP system uses resources from various different grids such as the Louisiana Optical Network Initiative (LONI) [7], SURAGrid [8] and TeraGrid [6]. All three grids have a very different mode of operation and administration. For instance the LONI grid is centrally administered and all policies are enforced on all resources. SURAGrid allows greater flexibility and control since policy making is in the hands of individual resource providers. The TeraGrid is also composed of resources that are administered by the resource providers. All three grids implement access to on-demand resources in a variety of ways. LONI machines make use of preempt queues to offer on-demand resources. SURAGrid uses Loadleveler supported checkpoint restart mechanisms to provide access to on-demand resources by suspending running jobs. TeraGrid has an entire cluster dedicated for on-demand jobs.

4.2.2 Data-aware Scheduler

The SCOOP end to end system has been implemented as a set of services based on a Service Oriented Architecture described earlier. The key services that will make up the system include the Archive Service, the Catalog Service, the Simulation Application Manager, the Scheduling Service and the Visualization Service.

The Archive Service is used to receive, store and retrieve all data that the system receives from external sources such as NHC and the data that is generated by the system in the form of model outputs, visualization products etc. The Archive Service is responsible for providing URIs for accessing the files and updating them as the locality of data changes. Replication and Location Services are considered part of the Archive Service and the Archive provides a simple API to upload data and retrieve file locations and download data. The Archive allows searching for files based on file name patterns, however searches based on file content will be supported based on the metadata available in the Catalog Service and not by the Archive Service.

The Catalog Service provides access to metadata information about the various resource objects in the SCOOP system. These objects could include model input and output files, model configuration files and parameter values etc. The Catalog services will allow searching and retrieval of objects based on any of these metadata attributes using a simple API to perform these searches. The results returned could include file URIs, or Object names that match the metadata attributes used in the query. Information from the Catalog will also be used to determine other attributes which can form the basis of the scheduling decisions such as Ensemble set membership and model priority.

The Simulation Application Manager as explained earlier is responsible for coordinating the run-time activities of the system and reacting to the events generated by the various components for instance, when the archive reports that a particular input file has been received and the Catalog confirms that it has been cataloged, SAM could evaluate which model runs must be triggered based on the type of file that was received or generated for example ADCIRC and Wave Watch III may be triggered when analytic winds are available. The EDF is also used to create the subset of models to be executed rather than running the whole complement in any arbitrary order. We propose a data-aware scheduling system that takes into account the priority within jobs as perceived by the user

submitting the job into account and use it to minimize the number of high priority jobs that miss their deadlines. The scheduling system consists of a core scheduler responsible for separating and ordering the jobs based on the urgency and priority parameters specified. The scheduler passes the workflow jobs to Condor DAGMan which manages the submission and execution of the tasks on the end resource. Figure 4.7 shows the architecture of the data-aware job submission system that makes

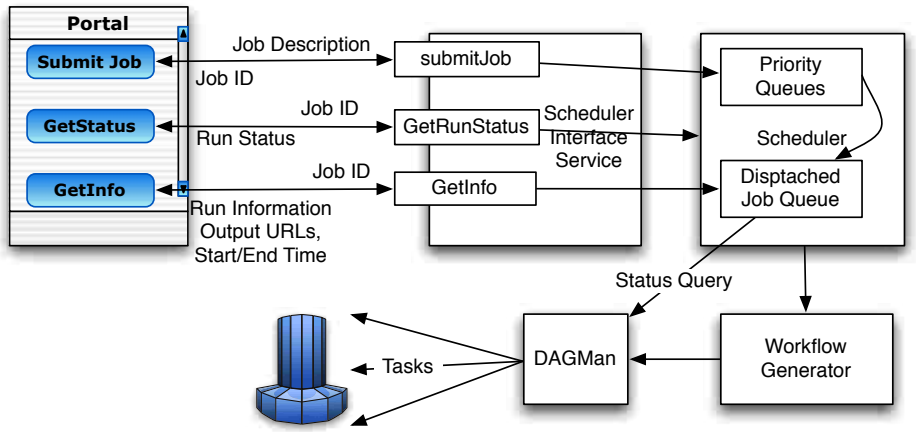

FIGURE 4.7: Simulation Application Manager (SAM) Architecture: Job submission and Management interfaces typically on a portal interact with the SAM Scheduler Interface Service which schedules jobs for dispatch using the SAM Scheduler. Once the scheduler dispatches the job the workflow generator is triggered and the workflow submitted to DAGMan. The individual tasks are submitted to the local schedulers on compute resources using Condor-G and Globus.

use of the data-aware scheduler. The Scheduler Interface Service provides a Web Service interface to the Scheduler and allows interaction with the system to submit jobs, query the status and retrieve the output of the submitted jobs. The scheduler described in greater detail in the next paragraph dispatches the jobs by triggering the workflow generator and submitting the workflow to DAGMan. Condor is used to manage the workflow tasks and Condor-G is used to submit the tasks to Globus based resources for execution.

Figure 4.8 depicts a block diagram of the scheduler's queues and the handling of job requests by the scheduler. The scheduler is a traditional Unix service that listens for requests on a port. A simple language that consists of key value pairs is used to describe the job to the scheduler and the urgency and priority levels associated with the job. A job submission request is shown below:

```
runtype=CFD_BIOTRANSPORT_1.0;
priority=0.8;
variablesFileName=CFD_BIOTRANSPORT_1.0.90edffe3-316c-46b1-9b37-5ba383f038c5.var;
SUBMIT;
CLIENT_SEND_END
```

Similarly a job status query looks like

```
job_id=2005;
QUERY;
CLIENT_SEND_END
```

The `CLIENT_SEND_END` command marks the end of the request. All statements in the request are terminated by a semicolon. The last statement in the request is the command given to the scheduler to perform a specific task. All statements before the command serve to provide data needed by the scheduler to execute the command. The order of these statements is not important. However the request must end with the command followed by a semicolon and the `CLIENT_SEND_END` command.

Incoming jobs are put into one of the three QoS level queues based on the priority value. Each queue is a priority queue with elements sorted on the value of the priority parameter. After a fixed time interval jobs in the three queues are dispatched starting with the on-demand queue and ending with the best effort queue. Once dispatched, the jobs are placed in the dispatch queue. Once here a workflow generator is started for each job and the workflow is submitted to DAGMan. Once a DAGMan DAG Id is received the jobs are moved to the Processed queue where they stay until completion.

Figure 4.8 shows the scheduler's queues and the movement of tasks between the queues indicated by arrows. Although the tasks contain the urgency information and a single queue would suffice to sort and dispatch jobs, having three queues offers an advantage. It is possible to tune parameters

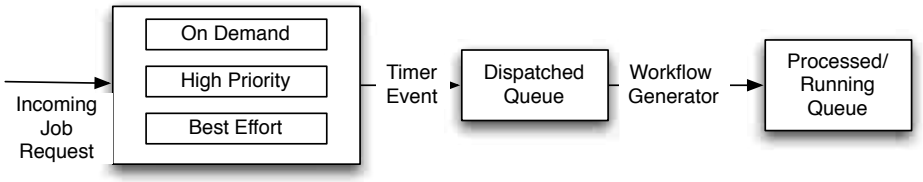

FIGURE 4.8: Scheduler Actions in response to Job submission and Timer Events

such as the dispatch interval of the three queues separately and also add other scheduling parameters if needed.

Chapter 5
Related Work

In this Chapter we discuss related work in the various areas relevant to the work such as Scheduling, Metascheduling, Workflows and Application Managers.

5.1 Scheduling

As described in Chapter 2, scheduling has been area of research since the early days of Computing and has several applications ranging from low level operating system task scheduling to high level parallel job scheduling on distributed supercomputers. Operating System task scheduling is a key component of modern day time-shared operating systems that use preemptive scheduling to assign processor time to the various tasks waiting for processor time. Priority is used as a method of controlling the ordering of jobs that receive CPU time. The key difference between modern day desktop operating systems and high performance computing clusters is that desktop operating systems are designed to provide interactive access and hence all tasks waiting for CPU time must receive CPU time frequently to provide a decent multi-tasking experience to the user, in other words it is not acceptable to wait for a task to finish before assigning CPU time to some other task. Operating system scheduling relies on frequent task switching and processors have ways of storing context and switching between processes quickly, which makes preemptive scheduling easy while scheduling tasks.

There has been substantial work in the area of scheduling on supercomputers. Local schedulers such as Maui [37], Moab [38], Loadleveler [39] etc. are designed to create optimal schedules using techniques such as priority inversion and back filling. The local schedulers however are unaware of the application and its characteristics and only react to what is currently present in the queue. This calls for a higher level data-aware scheduling mechanism that can make intelligent decisions to improve the turnaround time for important jobs and allow the science to drive scheduling.

Other metaschedulers such as Condor [15] and Gridway [5] perform scheduling across multiple resources however do not offer the data-aware scheduling as described in this thesis. Condor is a job management middleware that was developed for high throughput computing applications. Condor

is a workload management system that provides various subsystems for managing resources, managing jobs and matching resources with jobs. Condor makes use of ClassAds [15] which are used to advertise resources and describe jobs and thus provide a way to match resources with jobs. Jobs can easily state both job requirements and job preferences using ClassAds. Likewise, machines can specify requirements and preferences about the jobs they are willing to run. These requirements and preferences can be described in powerful expressions, resulting in Condor's adaptation to nearly any desired policy. This offers a great deal of flexibility and can potentially match based on parameters such as urgency and priority, however without any additional knowledge of the jobs and a way to identify resources that provide the capabilities needed to support the additional parameters Condor is only as useful as any other scheduler. The SAM scheduler, as described in 4.2.2, integrates these parameters into the scheduling algorithms and has a way of receiving these as input from the other subsystems.

GridWay is a workload manager that performs job execution management and resource brokering on a Grid consisting of distinct computing platforms that could be dynamically extended with Cloud resources. GridWay allows unattended, reliable, and efficient execution of single, array, or complex jobs on heterogeneous and dynamic Grids. GridWay performs all the job scheduling and submission steps transparently to the end user and adapts job execution to changing grid conditions by providing fault recovery mechanisms, dynamic scheduling, migration on-request and opportunistic migration. GridWay provides decoupling between applications and the underlying local management systems. Gridway uses Globus to provide a unified access to resources under various DRM systems making it extremely flexible and easy to submit jobs to multiple resources. However as is the case with other local schedulers Gridway also does not allow for expressing or taking advantage of the job's properties such as input data or scientific significance.

Services such as the Highly Available Resource Co-allocator (HARC) [12] and GUR [13] are examples of middleware that interact with local schedulers to provide users with control on when their job will be executed using advance reservations to reserve resources based on the user's requirement. These services can be used by our pipeline to provide more ways of obtaining resources that provide high QoS levels.

5.2 Data-aware Scheduling

Data-aware scheduling is not limited to the hurricane scenario but is useful across a wide variety of applications where there is a sense of relative importance of jobs. Application aware scheduling can be applied to regular day-to-day jobs submitted by users. Such scheduling not only helps improve utilization but also ensures that the limited amount of supercomputing time available to the scientists is spent doing more valuable science. Other related work in the areas of workflow management include projects such as Triana [26], Kepler [25] etc. We choose to use DAGMan [17] as we did not need any of the advanced features supported by these and using DAGMan was an obvious solution since the workflow management system was Condor. This keeps the number of software components to install low and keeps the software stack small.

Data-aware scheduling is also often referred to the concept of data location aware computing, which aims at bringing computation to data [21]. This model of computing is extremely popular when data sizes are large and transferring data between the compute units is prohibitively expensive. Schedulers are used to schedule data-transfers optimally such that no transfer is repeated and the computations can be performed on the transferred data without moving it to other locations. Stork [24] is an example of a data-aware scheduler that can perform data-placement optimizations.

5.3 SPRUCE

SPRUCE [22] is a system to support urgent or event-driven computing on both traditional supercomputers and distributed Grids. Scientists are provided with transferable Right-of-Way tokens with varying urgency levels. During an emergency, a token has to be activated at the SPRUCE portal, and jobs can then request urgent access. Local policies dictate the response, which may include providing next-to-run status or immediately preempting other jobs. In the context of this work SPRUCE provides a mechanism to obtain resources that provide the next-to-run and on-demand QoS levels. SPRUCE was integrated into the SCOOP portal [23] to provide a way for users to make use of on-demand resources on the TeraGrid.

5.4 Workflow Systems

Multiple workflow systems such as Kepler [25], Triana [26], etc exist that manage execution of multiple interdependent jobs. These workflow managers are capable of supporting large complex workflows with thousands of nodes and millions of executions. In SCOOP our workflow is simple and limited to three components, preprocessing, model execution and post processing, hence we chose DAGMan as the the workflow manager. DAGMan's close integration with Condor provided us the low level control of tasks that we desired.

5.5 End to End Systems

There have been multiple funded projects, in addition to SCOOP and CERA, to build end to end systems or pipelines for various research areas. The Linked Environments for Atmospheric Discovery (LEAD) [27] project is one such project that makes meteorological data, forecast models, and analysis and visualization tools available to anyone who wants to interactively explore the weather as it evolves. LEAD has several similarities to SCOOP in terms of pipeline architecture, choice of grid middleware etc, but addresses a different environmental research area. The scheduling paradigm presented in this thesis can be applied to pipelines like LEAD to take advantage of QoS levels offered by resources.

Chapter 6
Results

Data-aware scheduling was motivated by its potential to provide optimal execution of time sensitive jobs through its capability to base scheduling decisions on the properties of input data and its attributes. This chapter presents results from a set of experiments which support this case, and show that in the experiments that we performed data-aware scheduling provided improvements of 85% on average on a busy supercomputer, over the data-driven scheduling strategy where jobs are scheduled as they arrive without additional treatment. The experiments were all performed using the SCOOP and CERA scenario for coastal ensemble forecasting described in Section 2.2.

These results show how the SCOOP and CERA scenarios fared when their execution was simulated on a parallel computer workload representative of current day real supercomputers. The experiments simulate the execution of jobs from the various SCOOP ensemble sets E_1 to E_6. The setup for each set of experiments is described at the start of their respective sections and the common setup is described in Section 6.1.

The results presented here were generated using the GSSIM [48] workload scheduling and execution simulator and it is important to justify why these results are valid in the real world. In our simulated environment we model the real world schedulers in such a way that the simulated start times are a conservative estimate of the true start times. For instance, the simulator does not take into account the effect of reduced priority when a common user submits many jobs. This has no effect in the data-aware strategy as the QoS classes are fixed and guaranteed. Also the results presented here do not take into account node failures that are common on large supercomputers and which introduce delays in the schedule. Under the fair assumption that QoS guarantees will be upheld inspite of node failures, the real world makespans for the data-driven strategy can only be larger. The simulator also allowed us to test the strategies at different load levels over the course of a whole year, which is a difficult task in the real world and causes unnecessary interruption to users trying to perform real

work. It must be noted that in the results presented here we use both simulated and real workload data, in order to provide a convincing argument in favor of the data-aware strategy.

This chapter documents the results of the comparison of the Quality of Service based data-aware scheduling strategy to traditional scheduling approaches. The data-aware scheduling strategy was compared to the data-driven strategy where no special treatment was given to any jobs.

6.1 Application Scenario

The results presented in this chapter model the SCOOP and CERA application scenarios that were discussed in Chapter 3. The model run times are based on the true runtimes of SCOOP and CERA models (Table 4.1) and the number of runs for the SCOOP scenario is based on the selection of ensemble sets E_1 to E_6. The results presented here use benchmarking information of Wave Watch III(WWIII) and ADCIRC as presented in Section 4.1.8 to simulate the running times of these models. Two sets of workload data, simulated data and real data from Queenbee, were used for generating the results presented here.

TABLE 6.1: Description of experiments performed on real and simulated workload data

Exp. ID	Model Type	Workload Type	Runtime (procs)	Ensemble Size	Next-to-run jobs	On-demand jobs	Cycles
1	SCOOP WWIII 6.4	Simulated	1200s(64)	88	11	5	20
2	SCOOP ADCIRC 6.5	Queenbee	1200s(32)	88	11	5	24
3	SCOOP WWIII 6.6	Queenbee	1200s(64)	88	11	5	96
4	CERA ADCIRC 6.7	Queenbee	6120s(256)	5	1	4	96

The simulated (synthetic) workload was generated by a workload generator program that models the workloads on modern parallel computers and the real data was obtained from workload information based on the jobs that were submitted to and run on the LONI Queenbee cluster during the year 2008. Table 6.1 shows the various experiments that were conducted. Queenbee is a 680 node cluster with each node having 8 cores and is a medium sized supercomputer that represents

most of the heavily used academic supercomputers very well. According to the statistics published by TOP500 [46], machines of the size of Queenbee make up 58% of all machines by size and 78.4% of all machines by processor architecture, in the TOP500 list.

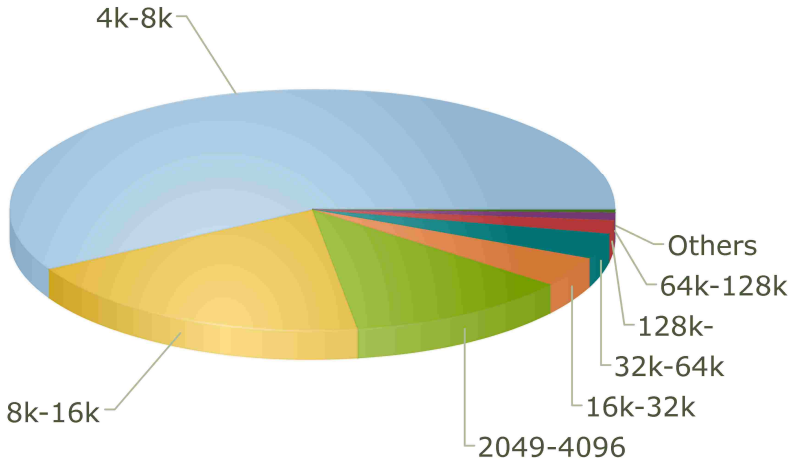

FIGURE 6.1: TOP500.org chart showing the breakup of machines based on number of processor cores. Machines like Queenbee enjoy a comfortable majority. [46]

The simulator is configured by adding resources and specifying the sizes of the various processor pools to be used by jobs from the different QoS classes. 48 nodes were placed in the preemptive queue for the data-aware runs for use by the on-demand runs. The next section describes how the GSSIM simulator environment was configured to produce the results presented in this chapter and the tools and data used for the experiments. We obtained Queenbee's workload information for the year of 2008 and split it into 12 parts, one file for each month. Queenbee was commissioned at the start of 2008 and usage at the beginning was low and picked up as the months passed. We have chosen 2008 specifically to demonstrate the impact of the strategy at various load levels. Results were also generated using simulated workload data generated by the Feitelson workload model [45] described in Section 6.2.

6.2 Simulator Environment

The comparison of the data-aware and data-driven scheduling strategies was performed using both real and simulated data. The simulated data was generated using workload generation models, specifically the Feitelson workload model [45]. The workload model reflects real workloads by capturing

the discrete nature of the distribution of degrees of parallelism, the correlation between parallelism and runtime, and the repeated executions of the same jobs. The specific version of the model that was used to generate the workload data was capable of modeling arrival times as well. The workload model was based on statistical observations that were made on data from supercomputers at NASA, Argonne, SDSC, LLNL, CTC and ETH, Zurich. The model takes into account the dominance of small jobs even on machines with hundreds of nodes. The second is the discrete nature of the distribution, with some sizes preferred over others. The most popular sizes are powers of two, even on machines where there is no architectural reason for using such sizes. The model for sizes is based on a harmonic distribution of order 1.5 (the probability for size n is proportional to $1/n^{1.5}$), which is then hand-tailored to further emphasize small sizes and powers of two. The model for runtimes is a hyperexponential distribution, so that the coefficient of variation is larger than one. In addition, a linear relation is made between the job size and the probability of using the exponential with the higher mean. Thus there is actually a different distribution for each size, and the mean runtime for larger sizes is higher as observed on the workloads. Repeated executions were taken into account when determining run length and it was determined that run lengths followed a Zipf distribution. The model was therefore that the probability of a run length of n is proportional to $n^{-2.5}$. The arrival process used in this model is Poisson. However, the fact that each job may be repeated multiple times changes this, because the repetitions are generated immediately after the previous run completes, simulating the submission of a bunch of identical jobs.

The workload information used for the simulation is described using the Standard Workload Format (SWF) [47]. SWF is governed by the following rules

- Each workload is stored in a single ASCII file.
- Each job is represented by a single line in the file.
- Lines contain a predefined number of fields, which are mostly integers, separated by whitespace. Fields that are irrelevant for a specific log or model appear with a value of -1.
- Comments are allowed, as identified by lines that start with a ';'. In particular, files are expected to start with a set of header comments that define the environment or model.

Changes were made to the SWF format to accommodate the Quality of Service parameters for jobs. The userid field was used to indicate the QoS parameter, with -1 being best effort, 1 being high priority and 2 being urgent.

In order to simulate the workload the GSSIM simulator was used. GSSIM [48] has been developed at the Poznan Supercomputing and Networking Center and is a simulation framework which enables easy-to-use experimental studies of various scheduling algorithms. Its goal is also to allow researchers to move the implementations of scheduling algorithms between simulation environments and real systems. GSSIM supports multilevel scheduling architectures with plugged-in algorithms both for Grid and local schedulers.

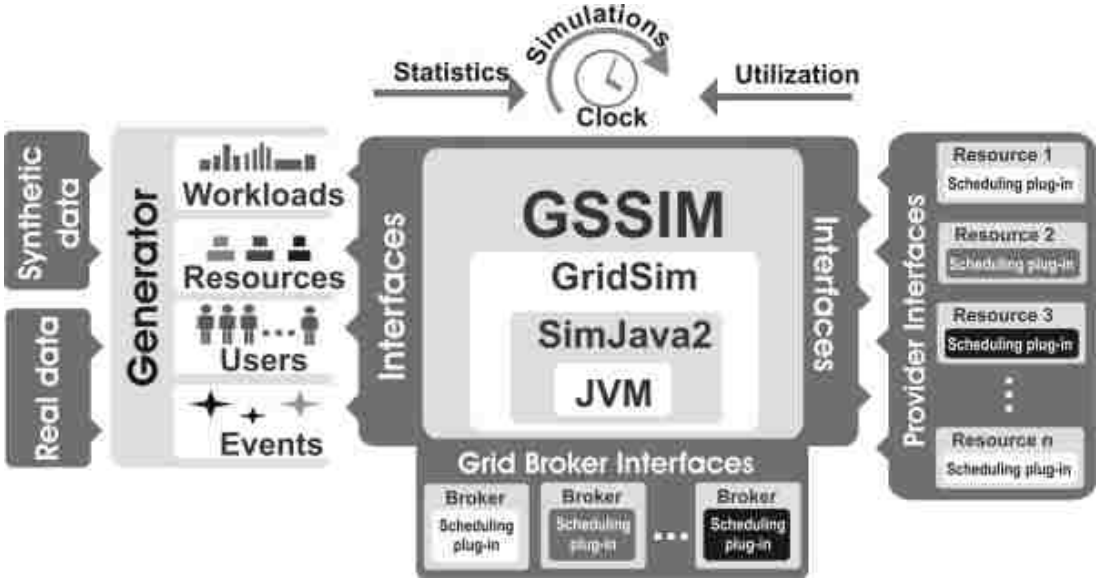

FIGURE 6.2: GSSIM architecture [49]. At the heart of the GSSIM simulator is the GridSim simulator, which is based on SimJava2. GSSIM uses a plugin model to allow different Grid and local scheduling algorithms to be simulated on both synthetic (simulated) and real workload model.

To apply GSSIM to this problem we needed to do the following:

- Develop a scheduling plugin for handling QoS parameters for resources.
- Generate simulated workload data to represent supercomputer workloads.
- Gather real workload information and convert it to the Standard Workload Format.
- Modify the workload information to specify the QoS requirements for the jobs.

A scheduling plugin was implemented to handle the QoS parameters specified in the SWF file and treat the jobs accordingly. GSSIM's native FirstFit algorithm was used to model the Queenbee moab scheduler with backfill with modifications to handle the QoS parameters. For the simulations we did not implement priority inversion and per user queue limits, as these only make the data-driven strategy perform poorly and do not affect the results that we are presenting.

6.3 Overall Results

Figures 6.3 show the percentage of jobs that missed the deadline every month on the Queenbee workload. As can be seen from the graphs the data-aware strategy, shown by dark lines, performs

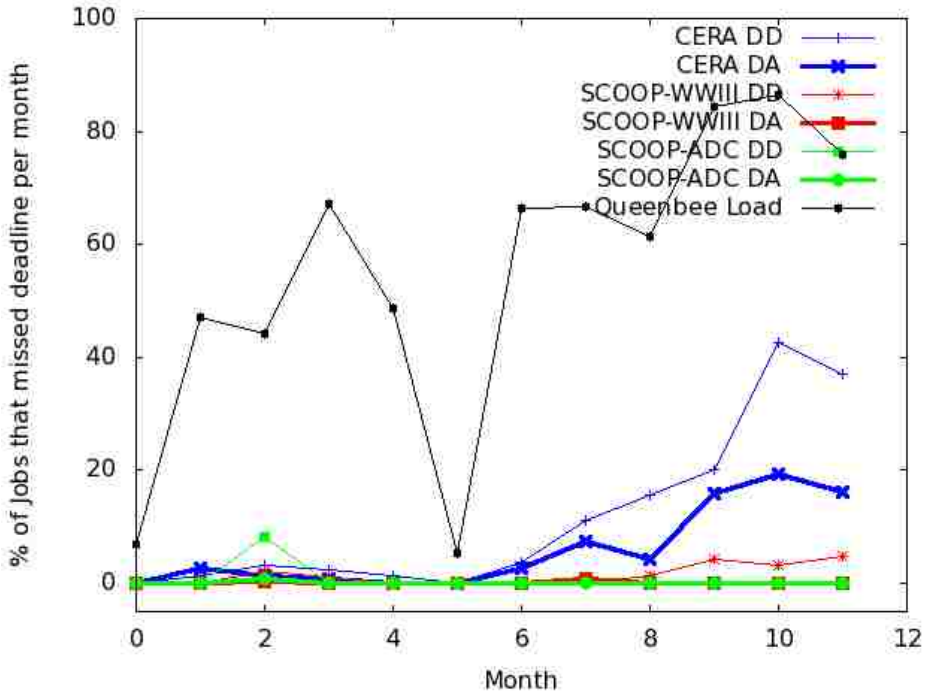

FIGURE 6.3: Number of jobs that missed the deadline every month in the various experiment scenarios: SCOOP WWIII, SCOOP ADCIRC and CERA ADCIRC

better than the data-driven strategy when dealing with important jobs.

6.4 Simulated Workload Runs

As described in Section 6.1, the results presented here are based on GSSIM simulations on real and simulated workload data. The generated workload consisted of 3000 jobs and simulated 15 days of real time with 20 cycles of runs, with the first cycle starting at 3200 seconds after the start of the simulation in real time. These 20 cycles model a 5 day hurricane. Each cycle consists of 88 jobs

corresponding to SCOOP ensemble set E_6, with 5 on-demand jobs (E_1) and 11 next-to-run jobs ($E_2 - E_1$). Figure 6.4 shows the makespan of each cycle of next-to-run and on-demand jobs on a 680 node machine. The SCOOP jobs simulated took 8 nodes and ran for 1200 seconds, which represent a WWIII job. The graphs show that the data-aware strategy performed significantly better than the data-driven strategy for all except one cycle. Cycle 8 shows that the data-aware strategy performed poorly for next-to-run jobs. This is due to the fact that the data-aware strategy uses a pool of 48 nodes to run on-demand jobs, causing the pool of machines where next-to-run jobs are submitted to shrink. In some cases where long running jobs take up a large part of the non-preemptive pool (all but 48 nodes), some next-to-run jobs had to wait for running jobs to finish and other jobs in the same cycle had to wait for the next-to-run jobs submitted before them to complete.

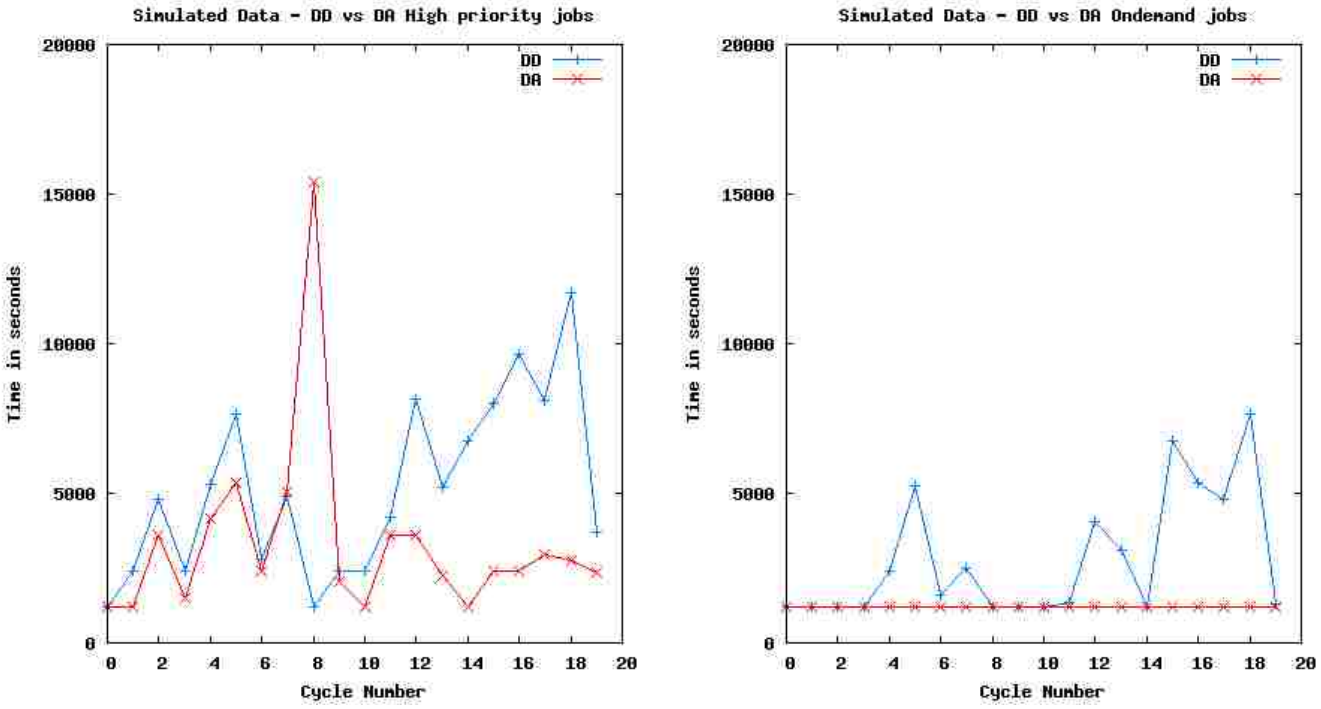

FIGURE 6.4: Makespan for each cycle of next-to-run (left) and on-demand (right) WWIII jobs using data driven (DD), shown in blue, and data-aware (DA), shown in red, strategy on simulated workload of a 680 node machine.

We found that on average across all cycles the data-aware strategy performed 46% better than the data-driven strategy, if cycle 8 is ignored, 30% better if cycle 8 is not ignored, and 76% better in the best case. The on-demand runs all ran to completion as soon as they were submitted.

TABLE 6.2: Best and worst times and averages in seconds for the data-driven and data-aware strategy for the simulated workload data

Strategy	Best Time	Worst Time	Average Time	Best Cycle(s)	Worst Cycle
Data driven	1200	11696	4677	1,2,9	19
Data-aware	1200	15395	3232	1,2,11,15	9

6.5 ADCIRC Runs

This section describes the results generated by using Queenbee's workload for the year 2008 and adding ADCIRC jobs corresponding to 6 days with 4 cycles per day to the workload after one simulated day was elapsed. Each cycle consists of 88 jobs representing the SCOOP set E_6. 5 of these corresponding to the SCOOP set E_2 are submitted to the on-demand resources and the 11 corresponding to jobs that are in E_4 but not in E_1, are submitted as high priority jobs. The graphs show all the ADCIRC jobs submitted during each month and compares the makespan of on-demand (E_2) jobs and next-to-run jobs ($E_4 - E_2$) in each cycle when run using the data-aware strategy and when run without it.

The results are split into four figures, Figure 6.5 shows results for the months of January to March 2008 for the Queenbee workload, Figure 6.6 for the months of April to June 2008 for the Queenbee workload, Figure 6.7 for the months of July to September 2008 for the Queenbee workload and Figure 6.8 for the months of October to December 2008 for the Queenbee workload. The left column in all figures shows the graphs plotting the makespan of jobs from the E_4 for each cycle and the right column shows the makespan of jobs in E_2 for each cycle in the month. Each row in all figures corresponds to graphs for a month of data from the Queenbee workload for 2008. As seen in Figures 6.3, the average monthly load has a general increasing trend from January 2008 to December 2008. Comparing this to the ADCIRC results, the data-aware strategy proves very valuable in the high load months from July to December.

6.6 Wave Watch III Runs

This section describes the results generated by adding WWIII jobs to the workload at different points in time, simulating hurricane events, on the Queenbee 2008 workload. In the results shown here we simulate hurricane events starting at 3200 second (53 minutes) after the start of the workload and

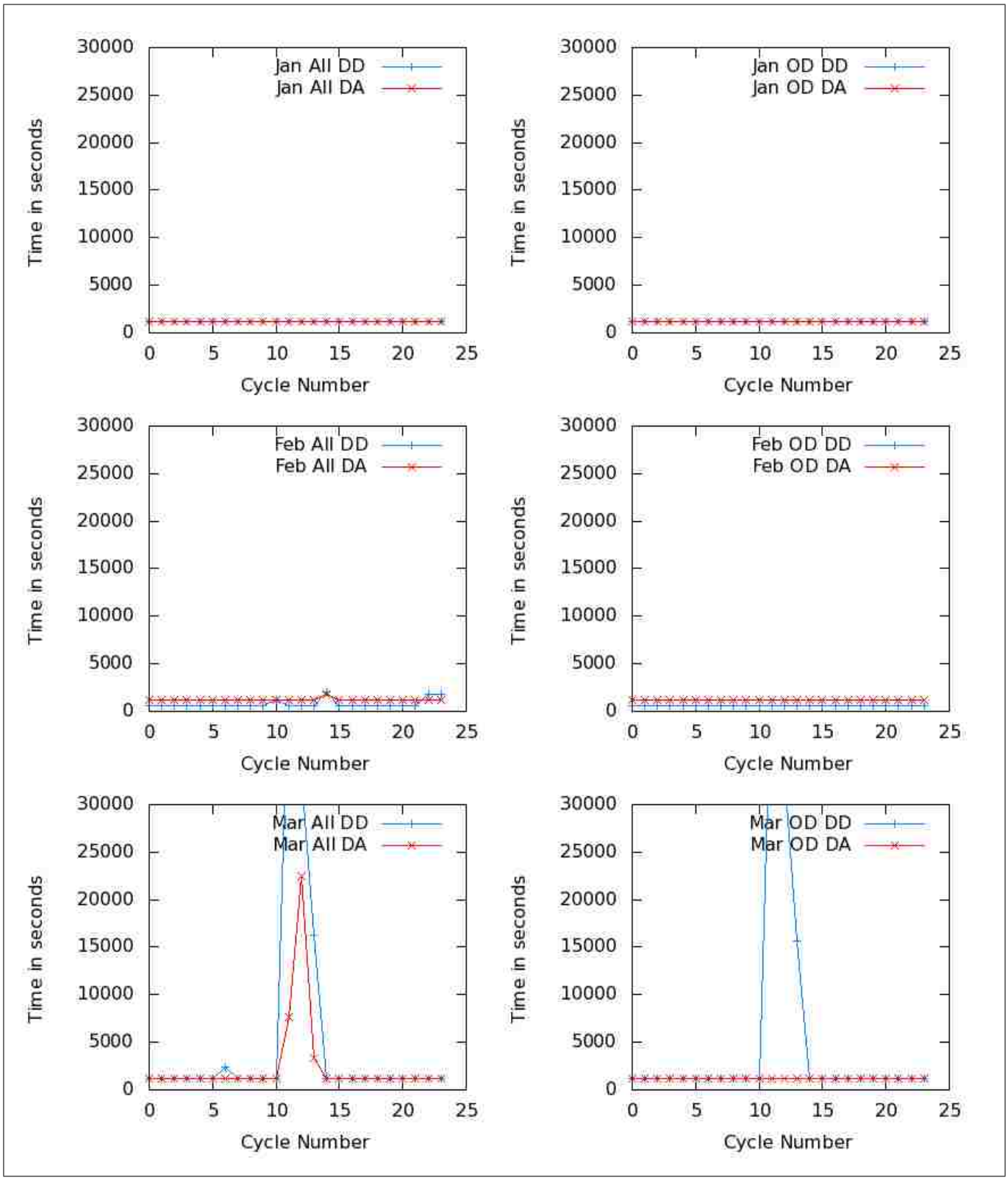

FIGURE 6.5: Makespan for next-to-run (left column) and on-demand (right column) ADCIRC jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of January, February and March 2008 (rows)

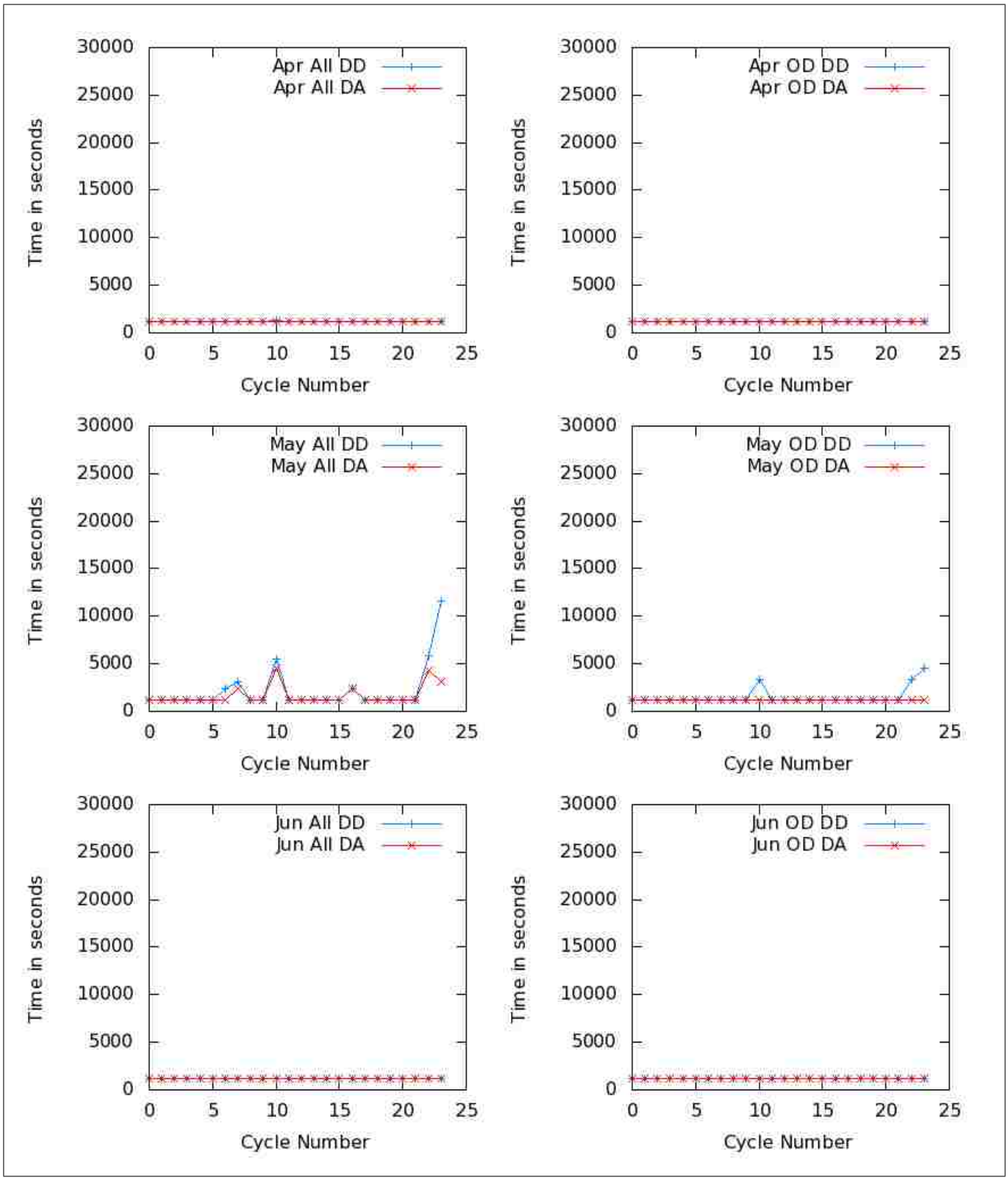

FIGURE 6.6: Makespan for next-to-run (left column) and on-demand (right column) ADCIRC jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of April, May and June 2008 (rows)

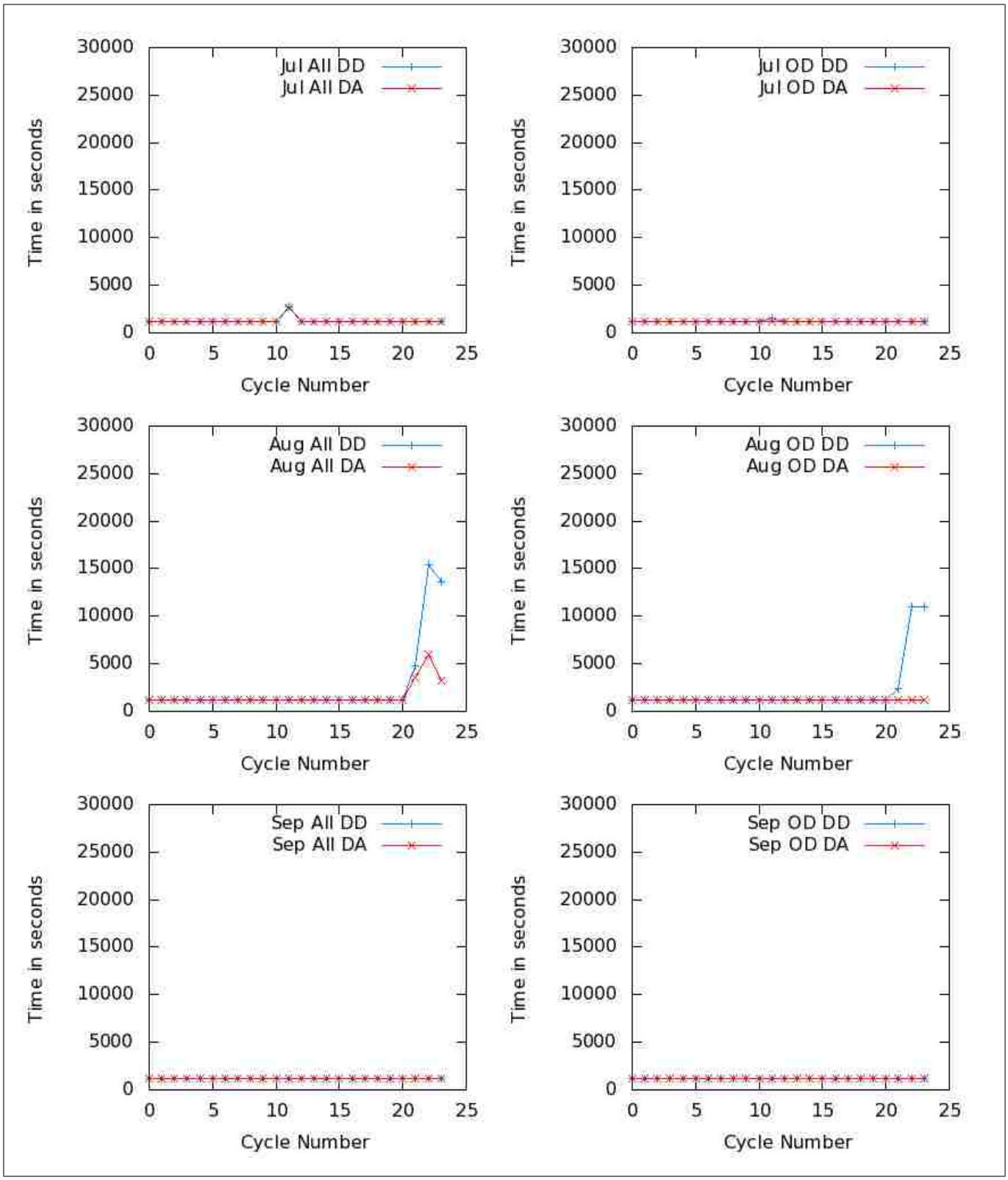

FIGURE 6.7: Makespan for next-to-run (left column) and on-demand (right column) ADCIRC jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of July, August and September 2008 (rows)

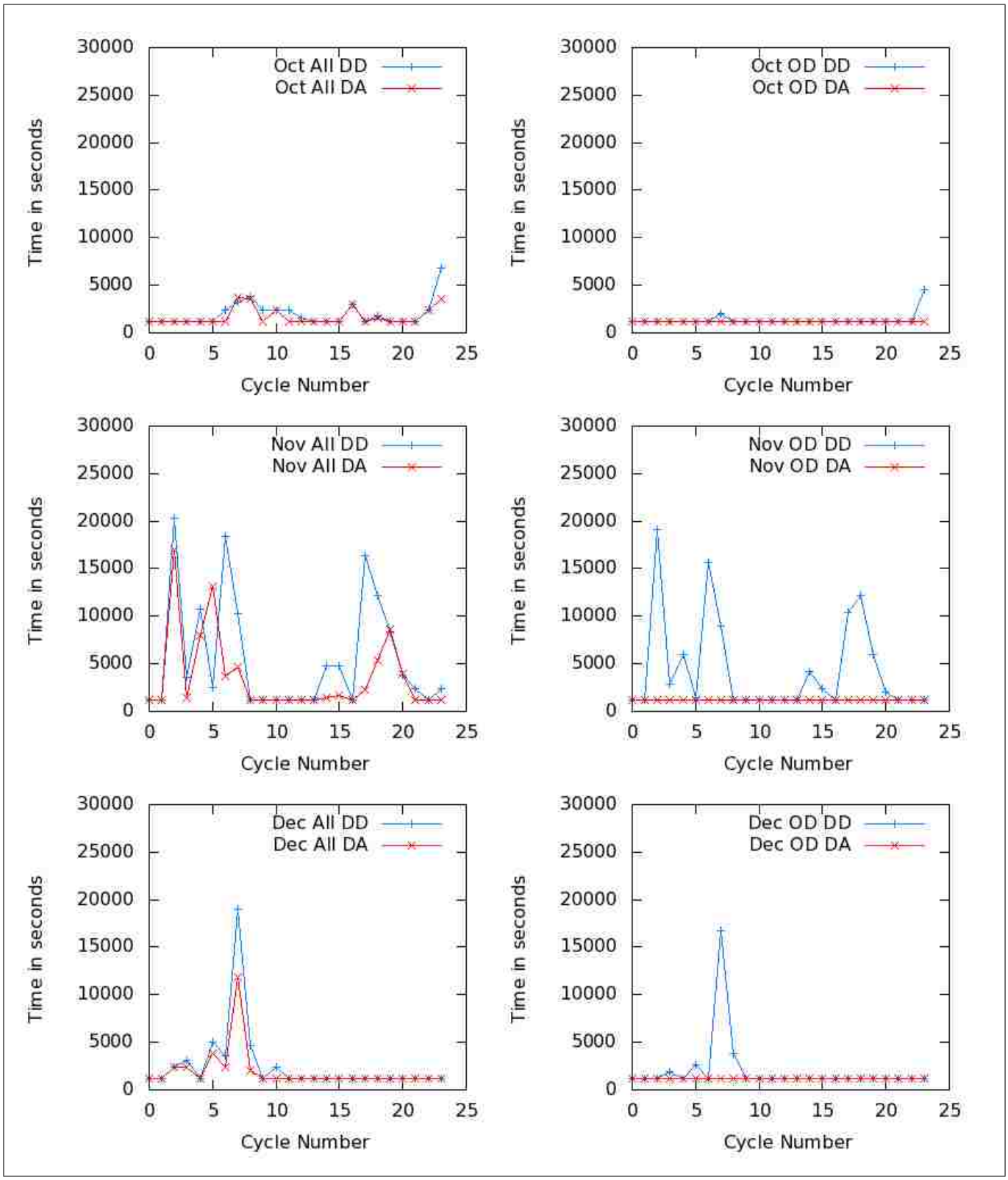

FIGURE 6.8: Makespan for next-to-run (left column) and on-demand (right column) ADCIRC jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of October, November and December 2008 (rows)

lasting for 6 days, which is the usual length of a tropical storm event. The start of the hurricane was chosen to be 53 minutes to allow the queues to fill up before submitting the hurricane jobs. Each day consists of 4 cycles with 88 jobs each. The top 5 of these corresponding to set E_2 are submitted to the on-demand resources and the next 11, corresponding to set $E_4 - E_2$, are submitted as next-to-run jobs. The graphs in Figures 6.9, 6.10, 6.11 and 6.12 show the makespan of each cycle of E_2 and E_4 jobs, when run using the data-aware strategy and when run without it. As described earlier the job cycles are separated by 6 hours, hence we will use 6 hours as the deadline for completion of jobs after the start of a cycle.

Figure 6.9 shows the makespan for every cycle run on Queenbee's workloads from January to March. For the month of March, the number of E_4 jobs that missed the deadline in the DD strategy was 22 and 3 in the DA. 10 jobs from E_2 did not finish within the deadline in the data-driven strategy and all jobs from E_2 finished within the deadline in the data-aware strategy. Figure 6.10 shows the makespan for every cycle when run on Queenbee's workload of April, May and June 2008. In the April runs, none of the E_4 jobs missed the 6 hour deadline in both cases, however the longest makespan in the data-aware strategy was 6471 seconds compared to 11589 seconds in the DD strategy, which is a 44% improvement.

The respective averages for the April workload were 1332 seconds and 1426 seconds respectively. None of the on-demand jobs missed their deadline and also had no wait times. This corresponds to a low load on Queenbee in April. Also since the WWIII jobs are small 8 node jobs that take only 20 minutes to run, the likelihood of finding backfill windows is very high, when load is low. Similarly for May and June since the workload was low all jobs completed without any delay as seen in Figure 6.10. As seen in Figure 6.12 when the workload increased more delays were seen in the jobs with some jobs coming close to the deadline and some others missing the deadline. Also noticeable is the fact that in some cases the DA strategy performed poorly, when compared to the DD strategy. This is expected in some rare cases, when dedicating a part of the resource to on-demand jobs means that the next-to-run jobs have to wait until long running jobs, which were using the rest of the resource, have to finish before the high priority job can start, since it was submitted to the section of the machine that does not allow preemption. This is the tradeoff for providing QoS guarantees to some

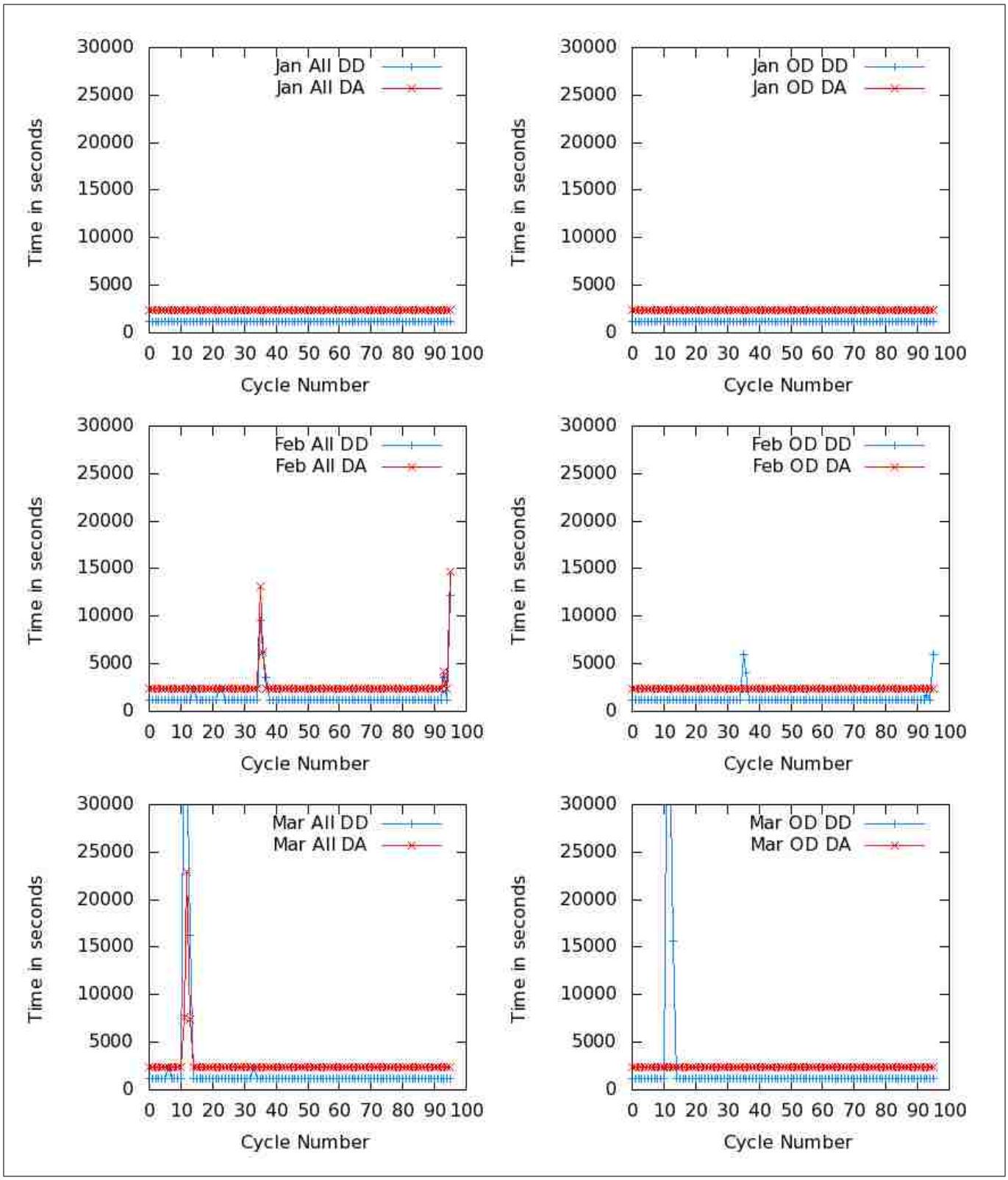

FIGURE 6.9: Makespan for next-to-run (left column) and on-demand (right column) WWIII jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of January, February and March 2008 (rows)

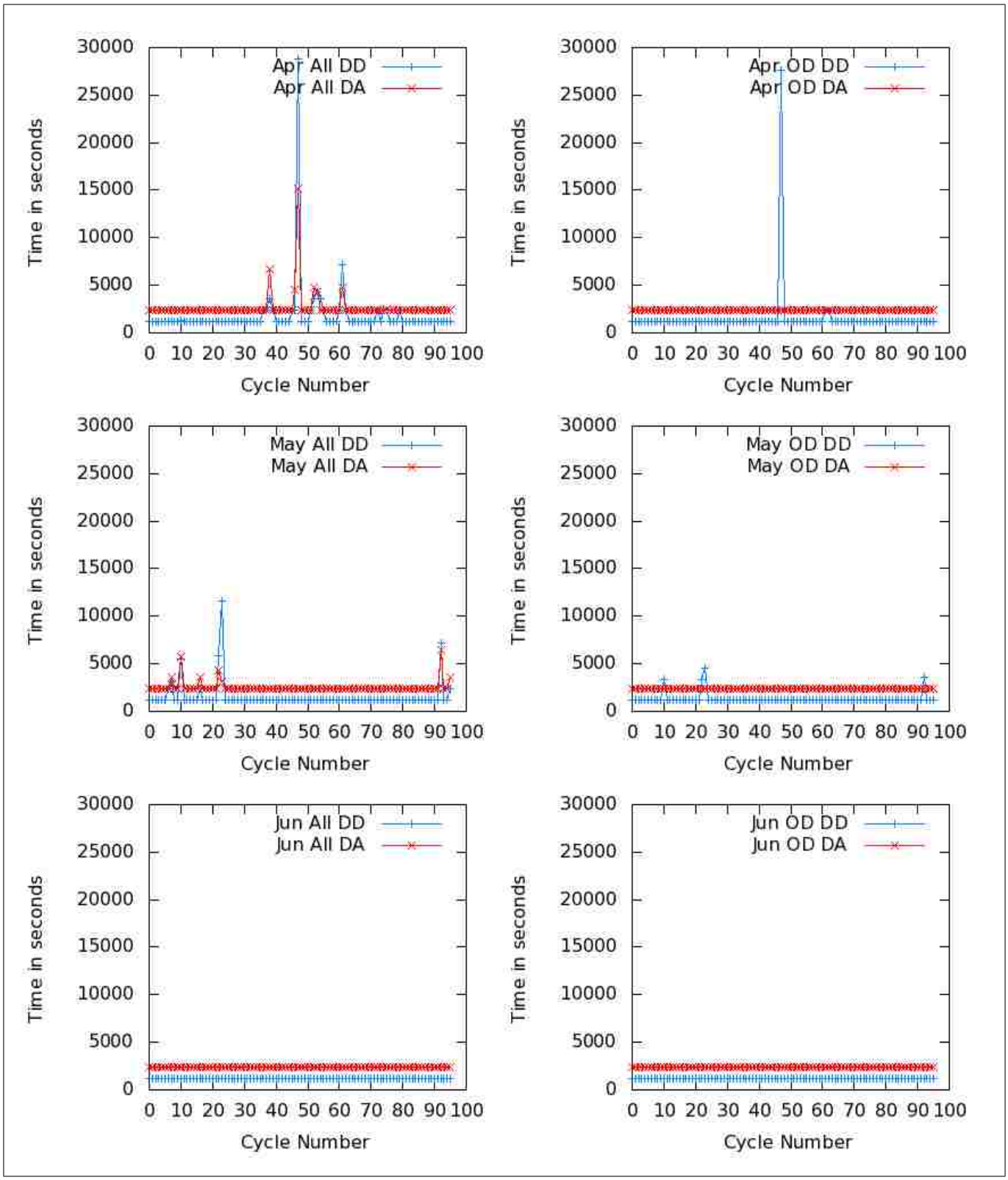

FIGURE 6.10: Makespan for next-to-run (left column) and on-demand (right column) WWIII jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of April, May and June 2008 (rows)

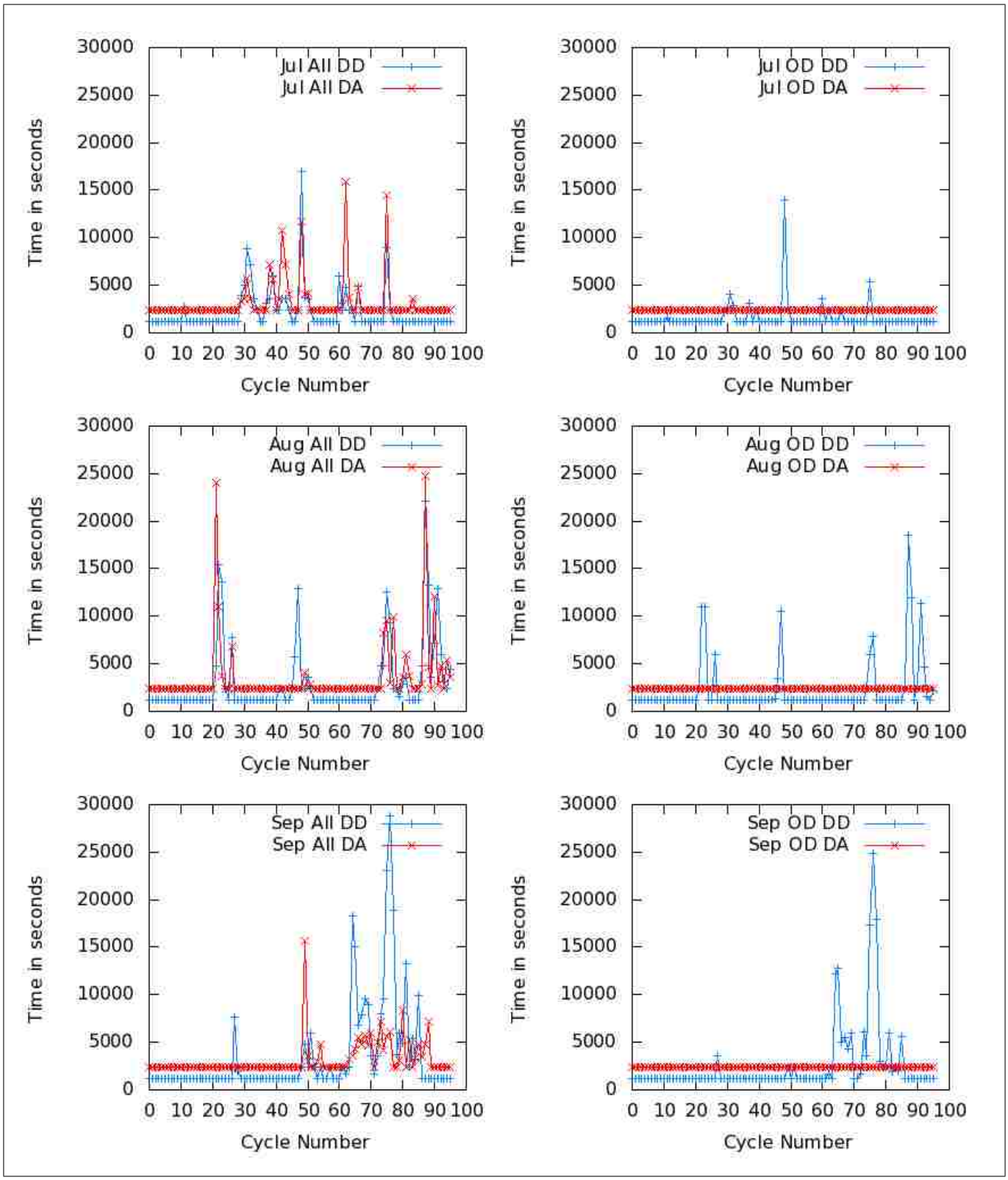

FIGURE 6.11: Makespan for next-to-run (left column) and on-demand (right column) WWIII jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of July, August and September 2008 (rows)

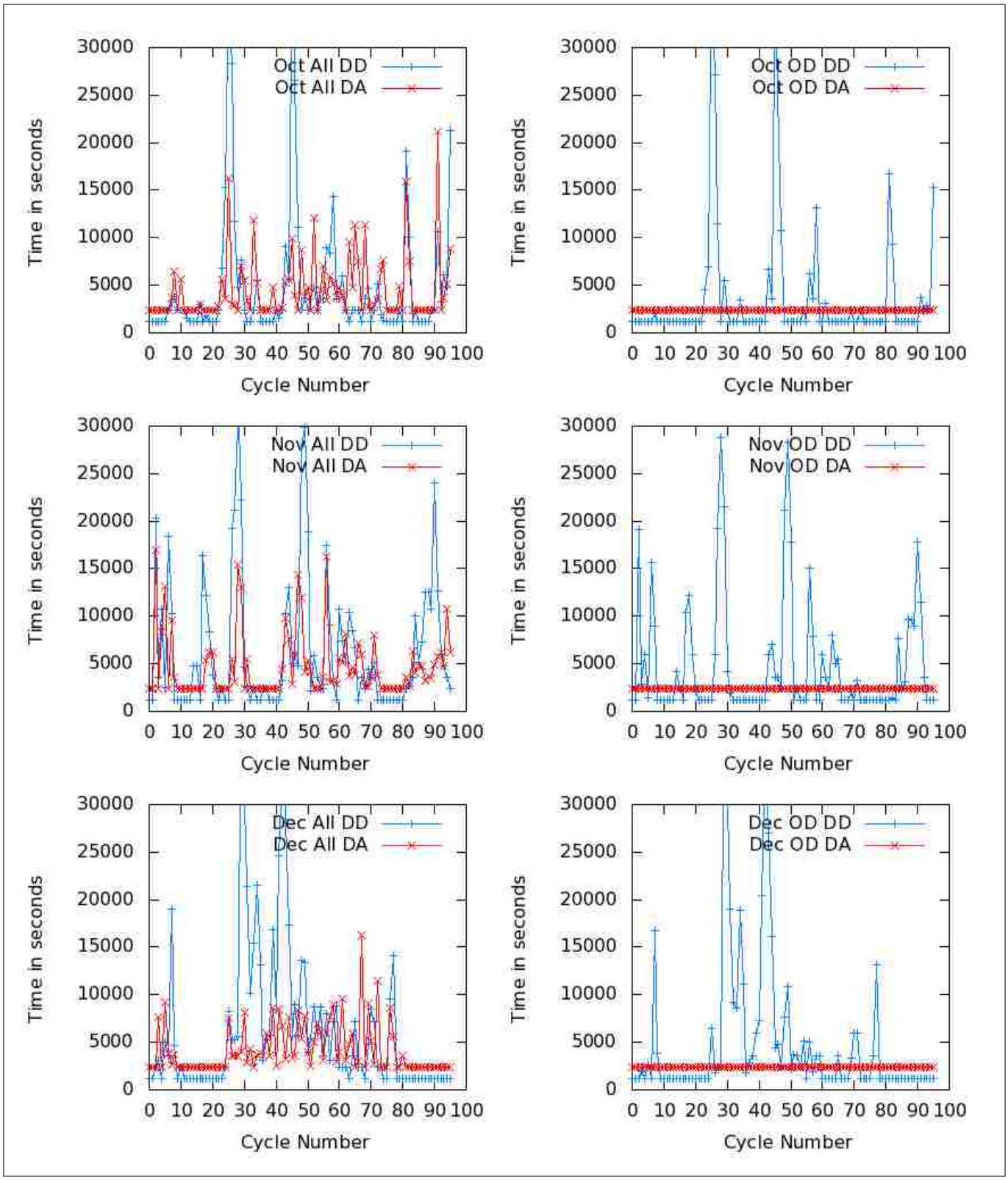

FIGURE 6.12: Makespan for next-to-run (left column) and on-demand (right column) WWIII jobs using data driven (blue) and data-aware(red) strategy on Queenbee workload of October, November and December 2008 (rows)

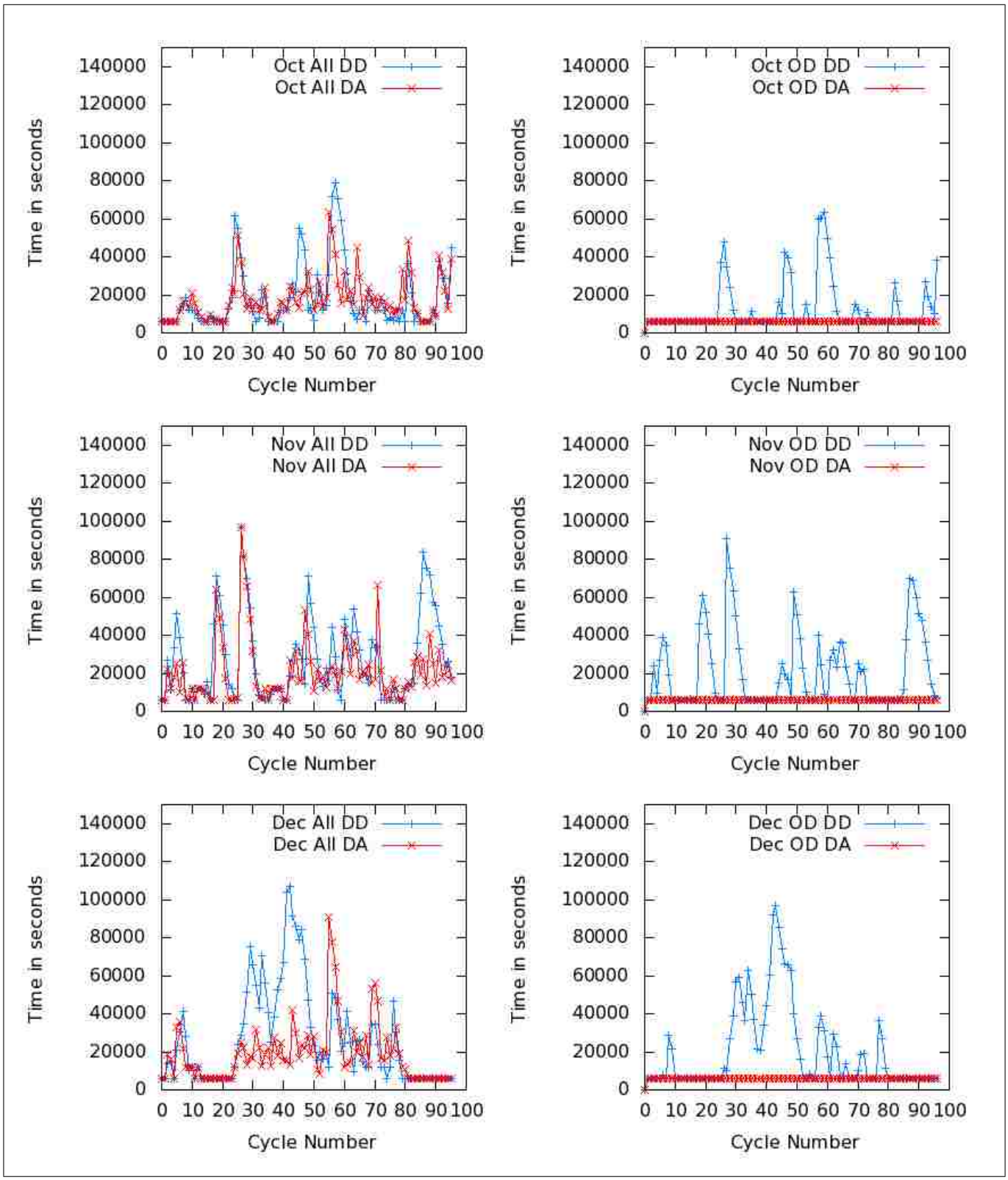

FIGURE 6.13: Rowwise from top: Showing the makespan of each cycle for a CERA scenario run on the Queenbee workload from October to December 2008

jobs. Having the whole machine available for scheduling provides better opportunities for running the jobs in E_4, but it has the risk that on-demand jobs may preempt these jobs, which is wasteful and also undesirable.

6.7 CERA ADCIRC Results

The CERA scenario was composed of 5 large ADCIRC runs that represent the original storm track and 4 different perturbations of the original track. This scenario was simulated using GSSIM on the Queenbee workload data for the months of October 2008 to December 2008 and the results are shown in Figure 6.13. We focus on the results from October to December as these months represent the load level that is common for shared supercomputers in academic environments. As can be seen, the QoS based Data-aware strategy of scheduling did significantly better that the data-driven approach with no special consideration given to important jobs. 37 jobs from E_2 missed their deadlines in the runs for Queenbee's November 2008 workload and 168 jobs from $E_4 - E_2$ missed their deadlines when using the data-driven strategy, while no jobs from E_2 and only 93 jobs from $E_4 - E_2$ missed their deadline. For 96 cycles, this corresponds to less than 1 job per cycle.

It is clear that using QoS based data-aware scheduling is capable of prioritizing the important jobs in any given cycle and ensuring that supercomputer time is put to its best use by running the most important jobs in a timely fashion.

Chapter 7
Conclusions and Future Work

In this thesis we studied the handling of time sensitive simulations on shared high performance computing resources. We developed a new scheduling paradigm that can be used to make scheduling decisions based on attributes of the input data and termed it data-aware scheduling. We established Quality of Service levels necessary for handling time sensitive simulations on HPC resources and showed in the context of the SCOOP scenario, how the relative priority of jobs can be mapped to these QoS levels. We provided a general analysis of QoS based data-aware scheduling and then presented the specific algorithm used in the SCOOP scenario.

At the start of the thesis we had raised some key questions that needed to be answered to provide a solution to the problem of scheduling time sensitive simulations on shared resources. Through this thesis we have answered all of those questions. In Chapter 6 we analyzed the performance of the QoS based data-aware scheduling strategy in comparison to the non-QoS data-driven strategy. In all of the results we saw considerable improvement in the makespan of the time sensitive jobs and also saw that using Quality of Service parameters prevented many important jobs from missing their deadline. For example, as shown in Figure 6.13, in the results for the December workload the on-demand jobs missed their deadline in 30 cycles when not using the data-aware strategy. In the event of a real hurricane this is a serious delay and cannot be tolerated.

We discussed the various requirements that time sensitive simulations have and the limitations of supercomputer batch queues in handling these efficiently, thus establishing the need for a new scheduling paradigm. We also presented a detailed description of the middleware that we developed to support the new scheduling paradigm and its interaction with other coastal pipeline components. We formalized the communication of the requirements of time sensitive simulations to the scheduling system and described in detail the format and how it could be used for any application that has time sensitive scheduling requirements.

The data-aware strategy is not limited to hurricane simulations and can be extended to any application that can make use of QoS classes based on relative importance of data or job attributes. For example the One Degree Imager (ODI) astronomy pipeline [51] is used to process images captured by a telescope and scientists use these images to look for interesting astronomical phenomena. For some studies the number of white pixels that indicate celestial objects in processed images, can be used to prioritize jobs that use these images as input files.

New models of computing such as cloud computing are becoming prevalent and are now being used for scientific research. Traditional supercomputers are beginning to provide interfaces that allow them to be used like cloud computing environments. Cloud computing in the context of execution environments is a way of providing on-demand access to compute resources, so from the perspective of our solution cloud computing provides a new QoS class that can be used to serve urgent jobs. Future work in this area may involve a way to provide cloud platforms as a possible alternative or in addition to preemptive scheduling.

Future state of the art supercomputers such as the Blue Waters machine [50] scheduled to be deployed in 2011 will contain over 400,000 compute units and will achieve a peak performance of around 10 PetaFlops. This growth in capacity poses new challenges for scheduling jobs across these resources for example guaranteeing a fair mix of different job sizes. There are also policy issues, for example, large resources typically prioritize long running large parallel jobs and starve smaller jobs even if resources are available. Also on these large general purpose supercomputers that are used for a wide variety of compute tasks, expressed simply as a request for a certain number of processors for a certain amount of time, it is not possible to convey specific job requirements such as deadlines to the scheduler. This is an issue that is unlikely to be solved as resources become bigger and our solution will also be required on these large resources.

Bibliography

[1] F. Darema, "Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements", International Conference on Computational Science, pp. 662-669.

[2] DDDAS: Dynamic Data Driven Applications Systems, http://www.nsf.gov/cise/cns/dddas/DDDAS_Appendix.jsp

[3] S.S. Iyengar, S. Mukhopadhyay, C. Steinmuller, X. Li, "Preventing Future Oil Spills with Software-Based Event Detection", Computer, vol. 43, no. 8, pp. 95-97, Aug. 2010, doi:10.1109/MC.2010.235

[4] I. T. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid - Enabling Scalable Virtual Organizations", CoRR cs.AR/0103025: (2001)

[5] E. Huedo, R.S. Montero and I.M. Llorente, "A Framework for Adaptive Execution on Grid", Software - Practice and Experience 34 (7): 631-651, 2004

[6] TeraGrid, https://www.teragrid.org/

[7] Louisiana Optical Network Initiative, http://www.loni.org

[8] SURAGrid, http://www.sura.org/programs/sura_grid.html

[9] S. H. Clearwater, and S. D. Kleban, "Relaxation Phenomena in Supercomputer Job Arrivals", ArXiv Condensed Matter e-prints 2002.

[10] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, A. Savva, "Job Submission Description Language (JSDL) Specification", Version 1.0

[11] J. Y-T. Leung, Handbook of scheduling: algorithms, models, and performance analysis,

[12] J. MacLaren, "HARC: The Highly-Available Resource Co-allocator", in Proceedings of GADA'07, LNCS 4804 (OTM Conferences 2007, Part II), Springer-Verlag, 2007. pp. 1385 to 1402.

[13] K. Yoshimoto, P. Kovatch, P. Andrews, "Co-scheduling with User-Settable Reservations", Lecture Notes in Computer Science, 2005, NUMB 3834, pages 146-156.

[14] I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", International Journal Supercomputer Applications, 11(2):115-128, 1997.

[15] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003. ISBN: 0-470-85319-0

[16] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Journal of Cluster Computing volume 5, pages 237-246, 2002.

[17] P. Couvares, T. Kosar, A. Roy, J. Weber and K. Wenger, "Workflow in Condor", in In Workflows for e-Science, Editors: I.Taylor, E.Deelman, D.Gannon, M.Shields, Springer Press, January 2007

[18] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. Su, K. Vahi, M. Livny, "Pegasus : Mapping Scientific Workflows onto the Grid", Across Grids Conference 2004, Nicosia, Cyprus

[19] Report on Meta-schedulers, TeraGrid, April 2007, http://www.teragridforum.org/mediawiki/images/b/b4/MetaschedRatReport.pdf

[20] SCOOP, http://scoop.sura.org

[21] T. Kosar and M. Livny, "A framework for reliable and efficient data placement in distributed computing systems", Journal of Parallel and Distributed Computing, 2005.

[22] P. Beckman, S. Nadella, N. Trebon, and I. Beschastnikh, "SPRUCE: A System for Supporting Urgent High-Performance Computing", Pg 295-316 in Grid-Based Problem Solving Environments by Springer Press. (IFIP 2.5 Conference Proceedings).

[23] G. Allen, P. Bogden, T. Kosar, A. Kulshrestha, G. Namala, S. Tummala, E. Seidel, "Cyberinfrastructure for Coastal Hazard Prediction", CTWatch Quarterly, Volume 4, Number 1, March 2008.

[24] T. Kosar, "A New Paradigm in Data Intensive Computing: Stork and the Data-Aware Schedulers", In Proceedings of Challenges of Large Applications in Distributed Environenments (CLADE 2006) Workshop, Paris, France, June 2006, in conjunction with HPDC 2006

[25] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows", (2004) Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on. Pages: 423–424.

[26] I. Taylor, M. Shields, I. Wang, and A. Harrison, "The Triana Workflow Environment: Architecture and Applications", Workflows for e-Science, pages 320-339. Springer, New York, Secaucus, NJ, USA, 2007

[27] D. Gannon, B. Plale, M. Christie, S. Marru, G. Kandaswamy, L. Fang, Y. Huang, S. Lee-Palickara, S. Jenson, N. Liu, S. Shirasuna, Y. Simmhan, A. Slominski, R. Ramachandran, R. D. Clark, K. Lawrence, and I. H. Kim, "The LEAD Science Portal Problem Solving Environment", Preprints, 23rd Conference on Interactive Information Processing Systems for Meteorology, Oceanography and Hydrology, San Antonio, TX, American Meteorology Society, 2007.

[28] CERA, http://www.cera.lsu.edu

[29] J. MacLaren, G. Allen, C. Dekate, D. Huang, A. Hutanu and C. Zhang, "Shelter from the Storm: Building a Safe Archive in a Hostile World" , in OTM 2005 Workshops, LNCS 3762, Springer-Verlag, pp. 294-303

[30] H. Conover, M. Drewry, S. Graves, K. Keiser, M. Maskey, M. Smith, P. Bogden, L. Bermudez,"SCOOP Data Management: A Standards-based Distributed Information System for Coastal Data Management", Standards-Based Data and Information Systems for Earth Observations, New York

[31] Unidata Local Data Manager, http://www.unidata.ucar.edu/software/ldm/

[32] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, "Data management and transfer in high-performance computational grid environments", Parallel Computing, v.28 n.5, p.749-771, May 2002

[33] Ensemble Prediction Systems - A basic training manual targeted for operational meteorologists, http://www.hpc.ncep.noaa.gov/ensembletraining

[34] Ensemble Forecasting, http://en.wikipedia.org/wiki/Ensemble_forecasting

[35] SCOOP File Naming Convention, http://scoop.sura.org/documents/naming_convention_final_5-3-06.pdf

[36] A. Kulshrestha and G. Allen, "Service Oriented Architecture for job submission and management on grid computing resources", International Conference on High Performance Computing (HiPC), 16-19 Dec. 2009, 2009.

[37] B. Bode, D. M. Halstead, R. Kendall, and Z. Lei, "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters", Proceedings of ALS'00 Proceedings of the 4th annual Linux Showcase & Conference - Volume 4

[38] Moab Workload Manager User's manual, http://www.adaptivecomputing.com/resources/docs/mwm/moabusers.php

[39] A. A. Prenneis Jr, "LoadLeveler:Workload Management for Parallel andDistributed Computing Environments", IBM POWER parallel Development, Poughkeepsie, N.Y., U.S.A

[40] Real Time Computing, http://en.wikipedia.org/wiki/Real-time_computing

[41] H. Conover, B. Beaumont, M. Drewry, S. Graves, K. Keiser, M. Maskey, M. Smith, P. Bogden, J. Bintz, "SCOOP Data Management: A Standards-based Distributed System for Coastal Data and Modeling", Proceedings of 2006 IEEE International Geoscience & Remote Sensing Symposium (IGARSS '06), pp. 317-320

[42] H. L. Tolman, "User manual and system documentation of WAVEWATCH III version 3.14", NOAA / NWS / NCEP / MMAB Technical Note 276, 194 pp.+ Appendices

[43] "WAMDIG 1988: The WAM model - A third generation ocean wave prediction model", Journal of Physical Oceanography, 18, 1775-1810

[44] "Technical Summary of the National Hurricane Center Track and Intensity Models", http://www.nhc.noaa.gov/pdf/model_summary_20090724.pdf

[45] D. G. Feitelson, "Packing schemes for gang scheduling". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1996, Lect. Notes Comput. Sci. vol. 1162, pp. 89-110

[46] TOP500 Statistics, http://www.top500.org/charts/list/36/procclass

[47] S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby, "Benchmarks and Standards for the Evaluation of Parallel Job Schedulers". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1999, Lect. Notes Comput. Sci. vol. 1659, pp. 66-89.

[48] K. Kurowski, J. Nabrzyski, A. Oleksiak, J. Weglarz, "Grid Scheduling Simulations with GSSIM", SRMPDS'07 (in conjuction with ICPADS'07)

[49] GSSIM, http://www.gssim.org

[50] Blue Waters Project, http://www.ncsa.illinois.edu/BlueWaters/

[51] G. H. Jacoby, J. L. Tonry, B. E. Burke, C. F. Claver, B. Starr, A. Saha, G. A. Luppino, and C. Harmer, "The WIYN One Degree Imager (ODI)", ASP Vol. 399, Panoramic Views of Galaxy Formation and Evolution, p. 489, 2008.

Vita

Archit Kulshrestha is currently a principal software research engineer on the FutureGrid and Science Gateway projects at Indiana University. He completed his Master of Science in Systems Science at Louisiana State University and is about to receive a doctorate in computer science from LSU. His research interests include distributed systems, high performance, grid and cloud computing. Before joining Indiana University, Archit was a grid architect and project manager of the SURA Coastal Ocean Observing Program(SCOOP) at Louisiana State University, where his key contributions included developing an end to end solution for running hurricane workflows on the Grid. Archit has over 20 publications in the fields of grid and cloud computing, with a focus on service oriented architecture, grid APIs and cloud computing.

Publications, Posters and Demos

- Archit Kulshrestha and Gabrielle Allen, Service Oriented Architecture for job submission and management on grid computing resources, International Conference on High Performance Computing (HiPC), 16-19 Dec. 2009, 2009.

- Marlon Pierce, Suresh Marru, Raminder Singh, Archit Kulshrestha, and Karthik Muthuraman, Open grid computing environments: advanced gateway support activities. In Proceedings of the 2010 TeraGrid Conference (TG '10). ACM, New York, NY, USA, , Article 16, 2010.

- Gabrielle Allen, Philip Bogden, Tevfik Kosar, Archit Kulshrestha, Gayathri Namala, Sirish Tummala, Ed Seidel, Cyberinfrastructure for Coastal Hazard Prediction, CTWatch Quarterly, Volume 4, Number 1, March 2008.

- Prasad Kalghatgi, Jerina Pillert, Bharadhwaj Thalakokkula, Sumanta Acharya, Gabrielle Allen, Peter Diener, Archit Kulshreshta, Sirish Tummala, Design And Implementation of Application Portal for CyberTools Science Drivers, Louisiana RII CyberTools and Science Drives Symposium, May 2009.

- Jiri Denemark, Archit Kulshrestha and Gabrielle Allen, Deploying Legacy Applications on Grids, Proceedings of the 13th Annual Mardi Gras Conference, Frontiers of Grid Applications and Technologies, 3-5 February 2005, Baton Rouge, pp 29-34, (2005)

- Andrei Hutanu, Gabrielle Allen, Stephen D. Beck, Petr Holub, Hartmut Kaiser, Archit Kulshrestha, Milos Liska, Jon MacLaren, Ludek Matyska, Ravi Paruchuri, Steffen Prohaska, Ed Seidel, Brygg Ullmer, Shalini Venkataraman, Distributed and collaborative visualization of large data sets using high-speed networks, Future Generation Computer Systems, Volume 22, Issue 8, p 1004-1010, 2006.

- Zhou Lei, Dayong Huang, Archit Kulshrestha, Santiago Pena, Gabrielle Allen, Xin Li, Richard Duff, Subhash Kalla, Chris D. White, John R. Smith, Leveraging Grid Technologies For Reservoir Uncertainty Analysis, in Proceeding of High Performance Computing Symposium (HPC 2006), April 3- 6, 2006, Huntsville, AL, 2006.

- SCOOP and Lake Pontchartrain Forecasting System Demonstration, Supercomputing 2008, Austin, TX
  Demonstrated the SCOOP dynamic data-driven system.

- SCOOP Demostration, Supercomputing 2007, Reno, NV
  Demonstrated the SCOOP portal and use of HPC Resources.
- SCOOP Demostration, Grid 2007, Austin, TX.
  Demonstrated the SCOOP on-demand system using Quality of Service parameters.
- SCOOP Demostration, Supercomputing 2006, Tampa, FL.
  Demonstrated the porting of coastal applications to HPC resources within a workflow.