Building an Infrastructure for Urgent Computing

Pete BECKMAN [a] Ivan BESCHASTNIKH [b] Suman NADELLA [c] and Nick TREBON [c]

[a] *Mathematics and Computer Science Division, Argonne National Laboratory*
9700 S. Cass Avenue, Argonne, IL 60439
[b] *Computer Science and Engineering, University of Washington*
Box 352350, Seattle, WA 98195
[c] *Computation Institute, The University of Chicago*
5801 S. Ellis Avenue, Chicago, IL 60637

Abstract.
Large-scale scientific computing is playing an ever-increasing role in critical decision-making and dynamic, event-driven systems. While some computation can simply wait in a job queue until resources become available, key decisions concerning life-threatening situations must be made quickly. A computation to predict the flow of airborne contaminants from a ruptured railcar must guide evacuation rapidly, before the results are meaningless. Although not as urgent, on-demand computing is often required to leverage a scientific opportunity. For example, a burst of data from seismometers could trigger urgent computation that then redirects instruments to focus data collection on specific regions, before the opportunity is lost. This paper describes the challenges of building an infrastructure to support urgent computing. We focus on three areas: the needs and requirements of an urgent computing system, a prototype urgent computing system called SPRUCE currently deployed on the TeraGrid, and future technologies and directions for urgent computing.

Keywords. Urgent Computing, On-Demand, Emergency, Disaster Response

Introduction

Since the days of the first supercomputers, computation has been playing an ever-increasing role in science. While the earliest supercomputers were used primarily to increase the speed of basic calculations that could otherwise be performed by hand, modern scientific simulation and modeling programs are extremely sophisticated. Often, large-scale codes represent decades of programmer effort. As the computer models have incorporated better computational methods and improved physics and chemistry, they have become more accurate. Their ability to accurately model and predict complex systems has led to advances in areas ranging from weather prediction to better traffic planning. For example, the latest report from the Intergovernmental Panel on Climate Change [1] relies heavily on computer models to peer into the future and explore what the Earth will be like as warming continues. While decision makers would like simulation results as

soon as possible, there is often little urgency or a deadline to complete the computation. Developing public policy is rarely fast.

However, there is a growing number of problem domains where key decisions must be made quickly with the aid of large-scale computation. In these domains, "urgent computing" is essential, and late results are useless. A computer model capable of determining where tornadoes will form must provide early warning to local residents. A computation to predict coastline flooding or avalanche danger must guide evacuation while there is still time. Furthermore, on-demand computing is often required to take advantage of a scientific opportunity, for example, to process data and steer activities during an experiment or observation of an unpredictable natural event. Without immediate access to large computational resources, the opportunity may be lost.

These on-demand, large-scale computations cannot wait endlessly in a job queue for Grid resources to become available. However, neither can the scientific community afford to keep multimillion dollar infrastructures idle until needed by an urgent computation. Instead, we must develop technologies that can support urgent computation dynamically and yet preserve overall machine utilization and the productivity of the scientists working daily on the systems.

1. The Needs of Urgent Computing

Urgent computing stems from dynamic work-flows and deadline-driven activities. Some computation needs more interactivity and immediate attention, while other computation is more time tolerant and suitable for traditional batch computing environments. This notion is not at all new; and in our daily computing, we have a rich range of models. Many new Web-based applications, from instant messengers to wikis, require users to be connected to the Internet. These applications are tolerant to network delays, however, and response delays can be several seconds before the user experience is significantly impacted. Even more tolerant are mail programs and source code repositories that queue transactions for later. At the other end of the spectrum are applications where users need near-instant feedback, such as voice-over-IP, remotely editing documents, or using a mouse to control the rendering and display of a large parallel data-set. Unfortunately, this rich continuum of responsiveness that we have come to expect from our Internet-based computing is not supported at most high-performance computing (HPC) centers. For most HPC systems, jobs are submitted to a batch queue where they will sit for some unknown length of time, or the user can request a reservation. Reservations fix the start time for the job; but because the scheduler uses worst-case execution times to guarantee the reservation, start times are pessimistic and often worse than what a user would experience by simply submitting to the batch queue.

Urgent computation is event-driven and deadline-based. To build an international infrastructure to support such computation, however, we must address more than the technical challenges. While questions such as "Which applications have execution deadlines not met by standard batch queues?" and "Under what circumstances should executing jobs be preempted?" are difficult to answer, we must begin to build a framework for discussing these topics and formulating fair policies to support the diverse user community.

2. A Framework for Urgent Computing

An urgent computing system requires more than a mechanism for elevating job priority. Some existing queueing systems and schedulers permit users to begin jobs sooner by using extra accounting units or by contacting an administrator for the computing resource. But effective urgent computing must be supported within a larger framework that supports interactions with user communities, applications, job queues, and decision makers. We believe it must also specifically address five important concepts: urgent computing session, priority policies, participation flexibility, allocations, and verification drills.

2.1. Session Activition

We are all familiar with on-demand, urgent situations at work and in our community. From the warning light on an automobile dashboard to an automated phone pager alert that a supercomputer is down, urgent situations begin with a trigger. Based on operating protocols, the event may initiate a response, or "session." Once initiated, the session is not concluded until someone evaluates the circumstances and determines that special actions are no longer required. Likewise, urgent computing jobs must occur within a clearly defined session. For some applications, hundreds or thousands of jobs might be submitted and run during a session. For others, a workflow requiring several machines and instruments might be required. The details surrounding the activation of an urgent session must be carefully logged and recorded. In all cases, an audit trail that can be later scrutinized and used to improve policies and response times must be created. For urgent computing, system administrators must be notified when sessions begin, permitting periods of increased attentiveness and, if needed, human intervention to provide the resources required. When the session completes, everyone must also be notified that operations have returned to regular service.

2.2. Elevated Priority for Resources

A key component to urgent computing procedures is deadline and priority-based resource management. An ambulance with sirens blaring and lights flashing forces non critical vehicles to slow and yield the right-of-way. An urgent computing framework must permit important jobs to gain priority access to the CPU, disk, networking, and other key components required for simulations. Tasks and activities that slow or hamper the speedy completion of computer simulations must be avoided when time is critical. On some systems, deadline-based elevated priority may simply grant shorter wait times in the queue. On others, it may translate to immediate, dedicated access. Furthermore, simulations are always part of a workflow that includes setting up the problem, data acquisition, computation, and analysis of the results. Some applications may require the movement of massive quantities of data, where network bandwidth can become a limiting constraint. In such situations, elevated priority might be implemented with prioritization of data streams in the routers or with guaranteed bandwidth reservations.

2.3. Flexible Participation Policies

Unless deployed on a set of resources owned and managed by a single stakeholder, an urgent computing system must be able to accommodate many types of resources and

policies. The NASA Columbia supercomputer was designed to provide simulation and modeling support for active launch missions [2]. As such, operational policies and expectations can be designed from the top down. To build an international network of diverse resources to support urgent computation, however, participation policies must be flexible. Furthermore, the system must coexist with ongoing operations. For example, some HPC centers may support preemption for certain applications or priorities, while other HPC centers may provide only "next-to-run" priority following the normal completion of existing jobs. There may also be regional influences based on stake-holders. For example, California supercomputer centers may link all of their resources to support urgent modeling of infrastructure, such as roadways and gas lines, immediately following an earthquake, while supercomputer centers in the Rocky Mountains may work together to build a capability that can quickly predict where a wildfire might travel based on current weather data and fuel models of the terrain.

2.4. Allocation and Usage Policies

The first question that arises when discussing priority access for resources is "Who gets it?" Naturally, a key element to supporting on-demand and urgent computing is creating a clear set of policies for both allocation and activation. At the Urgent Computing Workshop held at Argonne National Laboratory in April 2007 [3], participants suggested that science teams needing urgent computing should quantify their requirements, from the frequency of need to the size and length of jobs, so HPC centers can better plan for their response. They should also explain their need, so requests can be prioritized. Similarly, even after an allocation for urgent computing is provided, clear activation policies must be formulated. A scientist given an urgent computing allocation for calculating flood levels should not use it to meet a paper deadline. Since urgent computing is a relatively new concept, clear policies for allocation and usage are needed to allay user concerns of favoritism or misuse.

2.5. Practice and Verification Drills

No system, regardless of its level of technical sophistication, can operate reliably without careful testing and verification. A system to support urgent computation must include support for testing and verification of components as part of the core infrastructure, not as an afterthought. In the same way that office buildings must periodically perform fire drills to test both the methods for alerting occupants as well as their training to evacuate quickly and calmly, an urgent computing framework must have periodic tests of the procedures, policies, and technology. We refer to urgent computing applications that have been periodically tested and their correctness verified as being in "warm standby." They are ready to run as soon as inputs are provided. Scientists must know both when the last successful test was performed and how often the application runs correctly and to completion on a particular platform. These values can help in the prediction and selection of computational resources.

3. SPRUCE

SPRUCE, the *S*pecial *PR*iority and *U*rgent *C*omputing *E*nvironment, is our implementation of the underlying components required to build an urgent computing framework.

SPRUCE uses a token-based authorization system chosen to facilitate allocation and tracking of urgent sessions. As a raw technology, there are no dictated policies within SPRUCE; resource providers have full control and flexibility to choose the policies they are comfortable with and implement them as they see fit. To build a complete solution for urgent computing, SPRUCE must be combined with allocation and activation policies, local participation policies for each resource, and procedures to support "warm standby" drills. These application drills not only verify end-to-end correctness, they also generate performance and reliability logs that can aid in resource selection.

3.1. Right-of-Way Tokens

Many possible authorization mechanisms could be used to let users initiate an urgent computing session, including digital certificates, signed files, proxy authentication, and shared-secret passwords. In time-critical situations, however, simpler is better. Relying on complex digital authentication and authorization schemes could easily become a stumbling block to quick response. Hence, simple transferable tokens were chosen for SPRUCE. This design is based on existing emergency response systems proven in the field, such as the priority telephone access system supported by the U.S. Government Emergency Telecommunications Service within the Department of Homeland Security [4]. Users of the priority telephone access system, such as officials at hospitals, fire departments, government offices, and 911 centers, carry a wallet-sized card with an authorization number. Cardholders can use the number to place high-priority phone calls that jump to the top of the queue for both land- and cell-based traffic even if circuits are completely jammed because of a disaster.

The SPRUCE tokens (see Figure 1) are unique 16-character strings that are issued to scientists who have permission to initiate an urgent computing session. The token need not be physical and could be hidden behind a Web portal or other middleware and used by automated systems. When a token is created, several important attributes are set.

- Resources that can be used for urgent jobs
- Maximum urgency that can be requested on any job
- Session lifetime (period for which urgent jobs may be submitted)
- Expiration date of the token
- Project name
- People to be notified if the token is used (e.g., local administrators)

A token represents a unique "session" that can include multiple jobs and lasts for a clearly defined period. A token can also be associated with a large group of collaborators; once the token is activated, anyone in the group can submit jobs with elevated priority. Users may be added or removed from the token at any time, providing flexible coordination. We emphasize that the right-of-way token is not related to machine access or authentication. Users must already have an account and be able to log on and authenticate in the traditional manner. The token allows the users only to begin and end an urgent computing session, during which they may request elevated priority for jobs. Without an active session, requests for elevated job priority are logged and ignored. Moreover, job priority is not elevated by default. Only jobs submitted with special, urgent job parameters, described in Section 3.4, receive unique treatment.

To support token-based session activation and access to elevated-priority resources, the SPRUCE architecture has three main components: user workflow and client-side job

Figure 1. SPRUCE "right-of-way" token

submission tools, the SPRUCE portal for token management and urgent request authentication, and local resource provider agents that respond to priority requests.

3.2. SPRUCE User Workflow

The SPRUCE workflow is designed for application teams that provide computer-aided decision support or instrument control. Each application team is organized by a principal investigator (PI). The PI selects the computational "first responders," senior staff who may initiate an urgent computing session by activating a token. First responders are responsible for evaluating the situation in light of the policies for using urgent computing. They must decide whether usage of the urgent resources is indeed warranted and meets the guidelines. The team must also have analysts or interpreters who can translate the raw data and simulation output into advice for decision makers or possibly feedback to instruments or controls. An application that models airflow across a city to evaluate contamination scenarios may have many subtle details that need interpretation and presentation to city managers as they formulate response scenarios.

Figure 2 illustrates how the SPRUCE workflow is initiated. The workflow begins as the result of a trigger, which may be automatic (e.g., an automated warning message from a weather advisory RSS feed parser) or human-generated (e.g., a phone call to the PI). SPRUCE token holders are expected to use tokens with discretion and according to coordinated policies, similar to the way that citizens are expected to use good judgment before dialing 911. Token usage will be monitored and reviewed. Administrators can revoke tokens at any time.

The first responder begins interaction with the SPRUCE system by initiating a session. Token activation can be done through a Web-based user portal or via a Web service interface. Systems built from the Web service interface can be automated and incorporated into domain-specific toolsets, avoiding human intervention. Activation is described in greater detail in Section 3.5.2. Often, running a large simulation involves numerous scientists who are responsible for tasks ranging from acquiring the most recent data-set to producing a visualization for analysis. The initiator of the SPRUCE session can in-

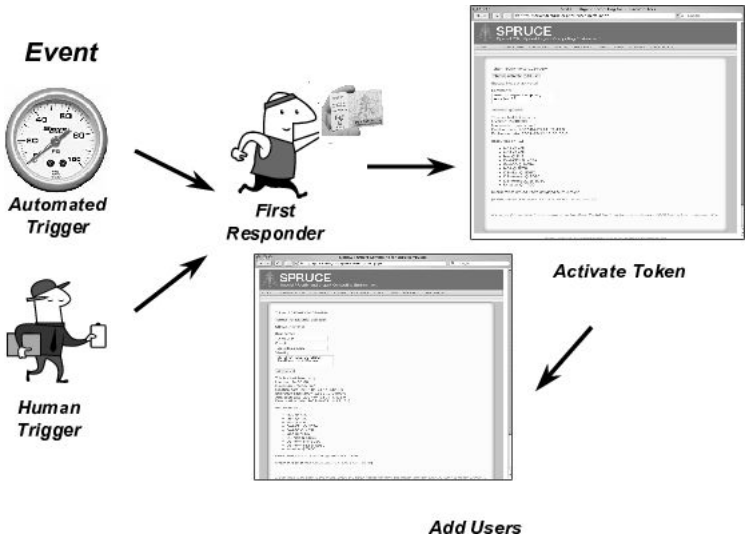

Figure 2. SPRUCE token activation

dicate which scientist or set of scientists will be able to request elevated priority while submitting urgent jobs. This set may later be augmented or edited.

3.3. Resource Selection

Once a token is activated and the application team has been specified, scientists can organize their computation and submit jobs. Naturally, there is no time to port the application to new platforms or architectures or to try a new compiler. Applications must be prepared for immediate use—they must be in "warm standby." All of the application development, testing, and tuning must be complete prior to freezing the code and marking it ready for urgent computation. Grids such as the TeraGrid have dozens of large-scale computational resources. The SPRUCE architecture supports large, diverse Grids; but ultimately, the science teams must select the best resources for their application. Maintaining and validating the accuracy of a simulation requires programmer resources, and often application teams narrow their efforts to a handful of favorite platforms and sites. Additionally, these sites should have their urgent computing priority policy in place and clearly defined. In the same way that emergency equipment, personnel, and procedures are periodically tested for preparedness and flawless operation, SPRUCE proposes to have applications and policies in warm standby mode, being periodically tested and their date of last validation logged (see Figure 3).

From this pool of warm standby Grid resources, the team must identify where to submit their urgent jobs. In a distributed Grid of independent resource providers, different urgent computing policies will exist. For example, one site may provide only a slightly increased priority to SPRUCE jobs, while another site may kill all the running jobs and allow an extremely urgent computation to use an entire supercomputer. On resources where existing jobs are not killed or preempted, current job load will affect resource selection. For urgent applications that produce or consume massive amounts of data, data movement may also place constraints on resource selection. Moreover, how a given

Platform	App.	Validated	Reliability	...
Site 1::Resource 1	Tornado	8 days ago	95%	...
Site 1::Resource 2	City Airflow	14 days ago	98%	...
Site 2::Resource 1	City Airflow	45 days ago	78%	...
Site 2::Resource 1	Influenza	30 days ago	59%	...

Figure 3. Example of the warm standby log, which tracks the validation history of urgent applications on specific resources

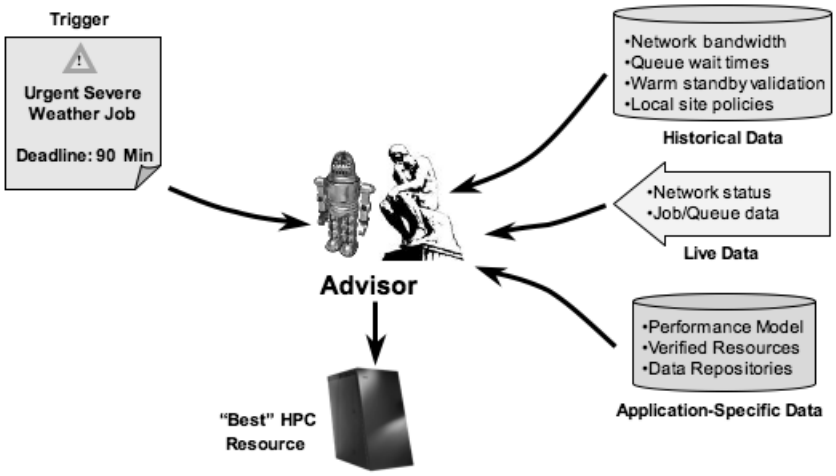

Figure 4. The SPRUCE advisor helps choose the best resource.

application performs on each of the computational resources must also be considered. To support resource selection, SPRUCE users may select resources manually or may use an automated SPRUCE "advisor" (see Figure 4).

The SPRUCE advisor, currently under development, must determine which resources offer the greatest probability to meet the given deadline. To accomplish this task, the advisor must consider a wide variety of information, including the deadline, historical information (e.g., warm standby logs, local site policies), live data (e.g., current network/queue/resource status), and application-specific data (e.g., the set of warm standby resources, performance model, input/output data repositories). To determine the likelihood of an urgent computation meeting a deadline on a given resource, the advisor must calculate an upper bound on the total turnaround time for the job (i.e., the amount of time from when a job is submitted to when the output is ready to be analyzed). Generally, the turnaround time can be divided into three parts: the data movement for input and output data-sets, the resource acquisition delay before the computation begins execution, and the time executing the computation.

The data movement delay includes the time to stage data in as well as the time it takes to transfer output to a destination for analysis. There are several approaches to predicting large file transfer delay by combining sporadic, historic transfer delay data with frequent, lightweight network probe data [5,6]. Additionally, recent work [7] incorporated into the Network Weather Service (NWS) [8] implements two strategies for determining an upper bound on the expected network bandwidth. Both approaches make

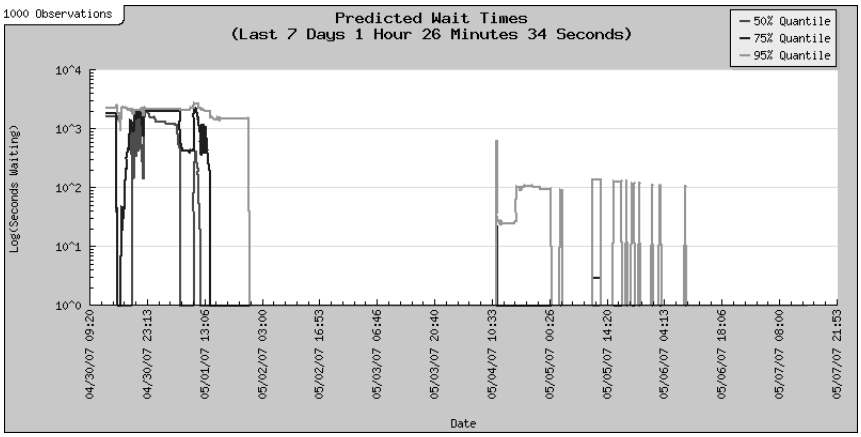

Figure 5. Time series display of the queue delay predictions for next-to-run jobs requesting 17 to 32 nodes on the UC-TeraGrid IA-64 cluster (image courtesy Rich Wolski, UCSB).

forecasts based upon a historical time series, generated via suitably sized NWS network probes that estimate the available bandwidth between the source and destination.

In general, the resource acquisition delay represents the time a job waits prior to execution. For SPRUCE, this delay entirely depends upon the requested urgency and the local policy for each resource. Since each resource provider implements a local policy for handling urgent computation, it becomes a key variable for determining the bound on the queue delay. A prototype implementation of a bounds-predictor for next-to-run jobs has been created by the NWS team. The predictor analyzes the batch queue logs of a particular resource and generates a bound with a given level of confidence on the amount of time a next-to-run job will wait in the queue. An example of these bounds is depicted in Figure 5. Current research is geared toward determining similar methods for bounds calculations for other resource policy choices, such as preemption and checkpoint/restart.

Ideally, the execution time will be predicted from a performance model of the application on a given resource. If no performance model is available, a prediction may be generated by the warm standby validation logs that track the performance of certain configurations of the application on all warm standby resources.

The total turnaround time can then be bounded by combining the three predicted delays. Additionally, the advisor can consider other aspects that affect the turnaround time, for example, “Can the likelihood of meeting a deadline be improved by increasing or reducing the number of CPUs the urgent job requests?” Increasing the number of CPUs used in the computation should decrease the execution time but may result in an increase in the queue delay.

The turnaround time is not the only factor a user must consider when selecting a resource. The reliability of the application on a given resource is also important. For instance, the user may choose a resource that has a slower predicted turnaround time but on which the application has performed more reliably. As part of the warm standby mechanism, SPRUCE can create a reliability metric based on historical validation tests. Also, the current queue state for each resource (which can be provided via MDS4 [9] on the TeraGrid) can be analyzed. The current queue delay methodology predicts bounds based on historical queue data for each resource, and not on the current state of the queue.

However, the user may be able to detect certain situations that may impact their decision, such as an idle resource or a resource that just began executing a long-running job that is consuming the entire resource.

An important question is how to handle job failures. When successful completion of an urgent job is paramount, it may be prudent to simultaneously submit the job to multiple resources. Not only is the likelihood of successful completion increased, but the results may also be compared for validation. In the event that a resource fails while a job is in the queue, the job should be automatically routed to another resource. This resource may be selected by the advisor or specified a priori by the user. If the job fails or the machine goes down during execution, the user should be supplied with a list of alternate resources. We propose this approach rather than automatic rerouting, for two reasons. First, enough time might have elapsed that it is no longer possible to meet the deadline on any available resource; as a result, the computation would be wasted. Second, the execution may have produced partial results that could be used to guide a subsequent computation.

3.4. Prioritized Job Submission

SPRUCE provides support for both Globus-based urgent submissions and direct submission to local job-queueing systems. Currently SPRUCE supports all the major resource managers such as Torque, LoadLeveler, and LSF and schedulers such as Moab, Maui, PBS Pro, and Catalina. The system can be extended to any scheduler with little effort. Authorized users who have active tokens need only to specify an additional "urgency" parameter when submitting their jobs.

The Globus Toolkit [10] for Grid computing provides the TeraGrid with uniform tools for authentication, job submission, file transfer, and resource description. By extending the Resource Specification Language (RSL) [11], which is used by Globus to identify user-specific resource requests, the ability to indicate a level of urgency for jobs is incorporated. A new "urgency" parameter is defined for three levels: *critical* (red), *high* (orange), and *important* (yellow). These urgency levels are guidelines that help resource providers enable varying site-local response protocols to differentiate potentially competing jobs. Users with valid SPRUCE tokens can simply submit their original Globus submission script with one additional RSL parameter (of the form 'urgency = <level>'), to gain priority access.

Unlike the Globus RSL, local job queue submission interfaces, such as the PBS command *qsub* [12], are often not trivially extended to accept new parameters. Specification of urgency level when submitting directly to a resource's local job queue typically requires a modified job submission command or a wrapper script. SPRUCE provides a *spruce_sub* script that accepts an additional command line parameter specifying the job's requested urgency level.

From the user's perspective, the next step after submitting the job is to analyze the results once the job successfully completes execution. On the computing resource, behind the scenes, the job script is parsed, and a check is performed with the SPRUCE server to verify the user has permission to run urgent jobs before taking the corresponding action (see Section 3.5.3).

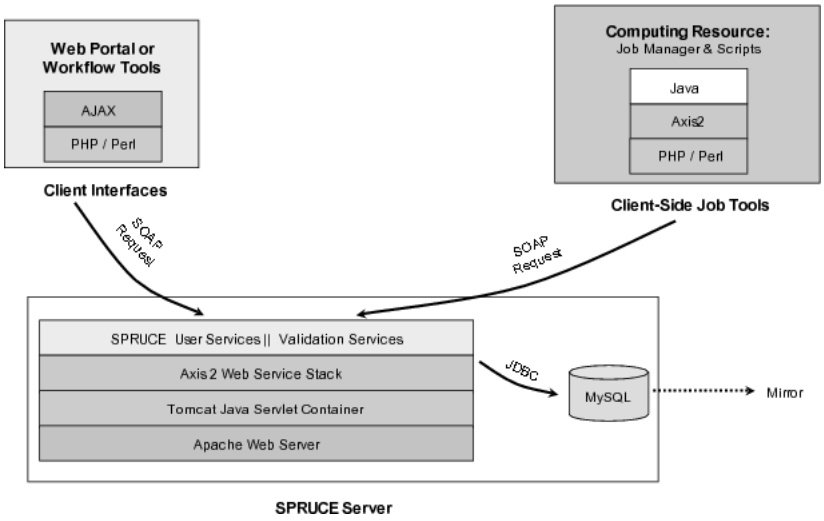

Figure 6. Internal configuration of the SPRUCE portal

3.5. SPRUCE Portal

The SPRUCE portal provides a single-point of administration and authorization for urgent computing across an entire Grid. It consists of three parts:

- The Web-based administrative interface allows privileged site administrators to create, issue, monitor, and deactivate right-of-way tokens. It features a hierarchical structure, allowing management of specific sub-domains.
- The Web service-based user interface permits token holders to activate an urgent computing session and to manage user permissions.
- The authentication service verifies urgent computing job submissions. A local site job manager agent queries the remote SPRUCE server to ensure that the submitting user is associated with an active token that gives permission to run urgent jobs on the given resource at the requested urgency.

The internal architecture of the SPRUCE portal is depicted in Figure 6. Both the user interface and the authentication service communicate with the SPRUCE server via a Web services interface. The user interface is implemented on top of Apache Axis 2 [20], while the portal is implemented in PHP and uses MySQL. External portals and workflows can become SPRUCE-enabled simply by incorporating the necessary Web service invocations.

SPRUCE users can interact with the system using a Web browser, requiring only minimal additional training, and making SPRUCE appropriate for emergency situations. Likewise, administrators will find the interface easy to navigate and use regardless of their environment.

3.5.1. Administration Interface

Distributed Grids typically span multiple administration domains. For the TeraGrid, the Grid Infrastructure Group (GIG) coordinates the software infrastructure, allocation, us-

age reporting, user support, and Grid security. While each resource provider, such as the San Diego Supercomputer Center or the University of Chicago, has a defined "service level agreement" for participation in the TeraGrid, they are nevertheless independent organizations. To support multiple administrative domains and virtual organizations, SPRUCE maintains a hierarchical Web-based administration interface, organized into three domains. Ordered by increasing privileges, these domains are the site (Grid resource provider), virtual organization, and root administrator.

All administrators have the ability to create and distribute tokens that are limited to particular resources and levels of priority within their domain, along with the privilege to revoke any tokens they created. They also have access to the token activation and urgent job submission history, user logs, and related statistics for their respective domain.

To permit distinct resource and management policies at each site, SPRUCE maintains sites within a virtual organization as independent entities. This strategy enables the site administrator to use SPRUCE in the wider context of a large multisite Grid, as well as privately for local machines and users. A virtual organization, such as the TeraGrid, spans multiple sites. An administrator in this domain may issue tokens that are valid across any sites that are a part of the virtual organization. Additionally, virtual organization administrators can add new sites. The most privileged administrative domain is the SPRUCE super-user, who is responsible for managing the virtual organizations. We note that SPRUCE does not support tokens that span multiple virtual organizations. The reason is that one of the defining features of a virtual organization is that their users are uniquely identified (e.g., certificate-based Distinguished Name, UNIX user name). However, sites belonging to distinct virtual organizations that utilize different user identity mechanisms and wish to allow cross-site tokens may form a new virtual organization with a shared identity mechansim.

3.5.2. User Interface

The intended users of the SPRUCE Web interface are scientists responsible for organizing the application team. Their tasks include monitoring the status of tokens, activating sessions, and organizing the team that will participate in an urgent computing session. This interface must be simple, fast, and modeled after the workflow described earlier in Section 3.2.

The Web services architecture enables SPRUCE integration with existing scientific Web portals and workflows. Users who prefer to use a Web-based interface can use the SPRUCE user portal. When a token is activated, the urgent computing session begins immediately; it terminates once the lifetime of the token has lapsed. We note that the session lifetime refers only to the period when urgent jobs can be submitted. Once the session ends, jobs that are currently executing will be allowed to continue uninterrupted. For convenience, members of the team who can submit jobs can be organized (added or removed) before the token is activated or any time during an active session. Changes to the set of users associated with a token are propagated without delay. All SPRUCE users may monitor basic statistics such as the remaining lifetime of the token and can query SPRUCE to find out the tokens with which they are currently associated.

3.5.3. Job Authentication Interface

At the core of the SPRUCE architecture is the invariant that urgent jobs may be submitted only while a right-of-way token is active. In order to support this invariant across a

distributed Grid system, a remote authentication step must be inserted into the job submission tool-chain for each resource supporting urgent computation. Since the SPRUCE portal contains the updated information regarding active sessions and users permitted to submit urgent jobs, it is also the natural point for authentication.

When an urgent computing job is submitted via Globus or the local queue system, the urgent priority parameters trigger authentication. This authentication is not related to a user's access to resource, which has already been handled by the traditional Grid certificate or by logging into the Unix-based resource. Rather, it is a "Mother, may I" request for permission to enqueue a high-priority job. This request is sent to the SPRUCE portal, where it is checked against active tokens, resource names, maximum priority, and associated users. Permission is granted if an appropriate right-of-way token is active and the job parameters are within the constraints set for the token. All transactions, successful and unsuccessful, are logged.

3.6. Resource Providers

To provide urgent computing capabilities for a supercomputer with SPRUCE, the resource providers must take three actions: register with the SPRUCE portal, formulate a resource specific policy for responding to urgent computing requests, and install SPRUCE components that interface with the job manager and the queueing system.

3.6.1. Portal Registration

Sites that use SPRUCE need an administrative account on the SPRUCE portal. Using this account, administrators can specify the details for each of the computational resources supporting urgent job submissions. The site administrator will also provide important contact information that can be used for emergency notifications when tokens are activated or critical errors occur. Once a site is registered, the administrator may begin generating and issuing right-of-way tokens. If the site is a member of a larger distributed Grid system that is already a part of SPRUCE, it may be incorporated into the corresponding virtual organization.

3.6.2. Responding to Urgent Computation

The SPRUCE architecture does not define or assume any particular policy for how sites respond to urgent computing requests. This approach complicates some usage scenarios, but it is unavoidable given the way we build Grids from distributed resources of independent autonomous centers, and the diversity of resources and operating systems available for computing.

When small-memory vector computers were the standard for HPC computing, preempting jobs was natively supported. Long-running jobs were routinely suspended, not to support urgent decision calculations, but to permit shorter jobs to achieve fast turnaround times during compile or debug sessions. Unfortunately, almost all modern supercomputers have lost this key feature, and therefore the SPRUCE architecture cannot simply standardize the strategy for responding to urgent computation as immediate preemption. Instead, we are left with many possible choices for supporting urgent computation depending on the systems software and middleware as well as on constraints based on accounting of CPU cycles, machine usability, and user acceptance. Given the current

technology for Linux clusters and more tightly integrated systems such as the Cray XT3 and the IBM Blue Gene, the following responses to an urgent computing request are possible:

- Scheduling the urgent job as "next-to-run" in a priority queue. This approach is simple and is highly recommended as a possible response for all resource providers. All modern queueing and job management systems support priority queues that are used for selecting the next job to run. No running computation is killed; and from the perspective of the user community, the impact on normal use is low. The urgent job will begin when all of the running jobs complete for a given set of CPUs.
- Suspending running jobs and immediately launching the urgent job. Some systems allow jobs to be suspended but remain resident in memory (via STOP signal). Running the urgent job will then force some memory paging, but the suspended job could be resumed later. Some applications that use external data sources and network connections may fail (connections time out and reset) if they are suspended. If a node crashes, both the suspended and the urgent jobs will be lost. The benefit of this policy is that urgent computation will begin almost immediately, making this option very attractive in some cases.
- Forcing a checkpoint/restart of running jobs and enqueueing the urgent job as the next to run. This response is similar to the previous response but safely moves the checkpoint to a location where it can then be used to restart on alternate resources, and is safe from node reboots. Some architectures support system-based checkpoint/restart; and, where it is reliable, it could be used to support urgent jobs. The urgent job begins execution when the checkpoint completes, which for large-memory systems could take 30 minutes or more depending on I/O and disk rates.
- Killing all running jobs and enqueueing the urgent job as next to run. Clearly this response is drastic and frustrating to the users who will lose their computation. Nevertheless, for extremely urgent computation, what user would demand that a black-hole simulation complete before launching an emergency hurricane or flood modeling scenario? In this response, an urgent job would begin immediately after running jobs are killed. Often, the delay to start is only several minutes.

Another factor in choosing the response policy is accounting and stakeholder accountability. Certain machines are funded for specific activities, and only a small amount of discretionary time is permitted. In some cases, there may be no specific "charge code" for urgent computing cycles. Furthermore, in order to improve fairness, some form of compensation could be provided to jobs that are killed to make room for an urgent job. For example, users could be refunded their CPU hours, given extra time for their trouble, and rescheduled with higher priority in order to avoid being relegated to the back of the job queue.

Another idea is to provide discounted CPU cycles for jobs that are willing to be terminated to make room for urgent computations. Some users have extremely robust and well-integrated problem solving environments that can perform checkpoint/restart easily. Some users design their software so only one or two hours of work is lost should a CPU fail or the entire system go down. Such users should be rewarded, and a discounted rate would allow them to recover lost work and run more inexpensively.

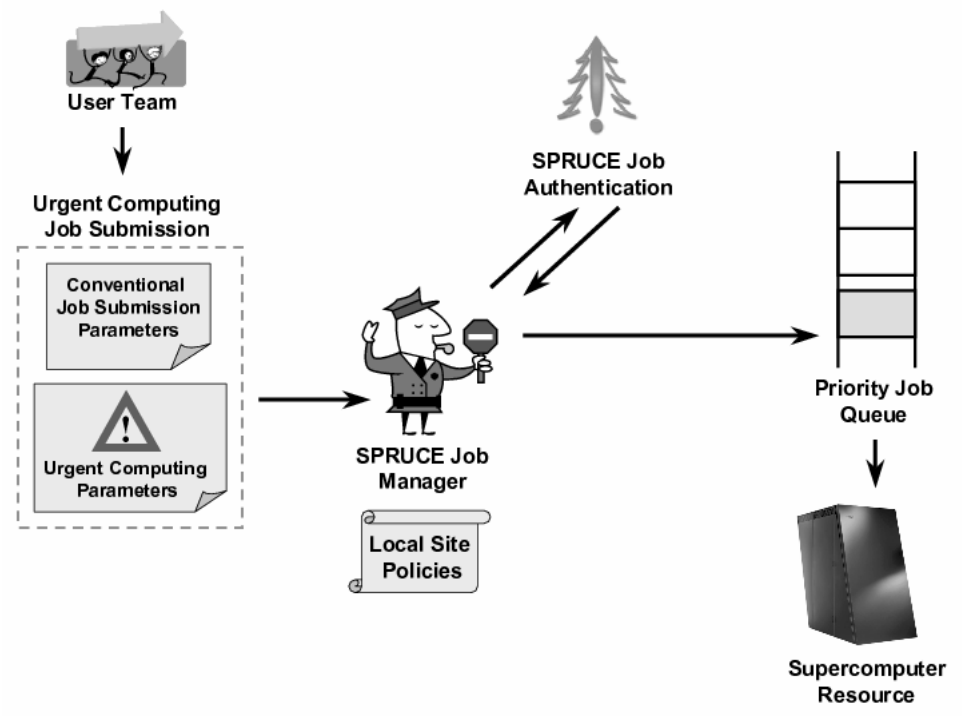

Figure 7. Resource provider architecture

The calculation of "maximum time to begin" (i.e., resource acquisition delay) discussed in Section 3.3 may play an important role in choosing a response strategy. For checkpoint/restart and killing of running jobs, the maximum time to begin can be bounded, possibly on the order of a few minutes to tens of minutes. If it is easy to calculate or determine, it can be used in conjunction with the computation deadline for selecting resources. Unfortunately, jobs with next-to-run priority could wait hours or days before running jobs complete. In any case, resource providers are encouraged to map all three levels of urgency—critical, high, and important—to clearly defined responses.

3.6.3. Handling Urgent Job Submissions

Figure 7 gives an overview of how the job requests are handled at the resource provider. In the Globus architecture, incoming jobs are routed through a job manager. When SPRUCE is installed on the resource, a job manager tailored to handle the additional urgency job parameters is added. This generates a job script dynamically when the urgent job is submitted and passes it to the native resource manager such as PBS Pro or Torque. It then authenticates the request against the SPRUCE portal (see Section 3.5.3). For example, in the case of the Torque scheduler, a submit-filter [13] script specific to SPRUCE is run every time a job is submitted. This filter authenticates the job script and confirms that it was prepared by the job manager rather than a malicious user attempting to submit a job into the high-priority queue without SPRUCE validation. If the user does not have sufficient permission, the request is rejected. If the request passes the verification stage, the actions needed to grant urgent access are performed based on the local site policy

and the requested priority level. After verification, the native job scheduler sends the job ID back to Globus, and when the requested resources become available, the queued job is launched.

Local sites can also support the command line version of the urgency job submission mechanism in the form of *spruce_sub*. Submission requests of this type are also routed through the SPRUCE job manager; the implementation mechanism remains the same. The only difference between these two submission methods is the interface.

3.7. Future Work

The SPRUCE system is currently running in production on many sites. Our current research focus is the automated "advisor" (see Section 3.3). This capability will need some job-specific information such as running time, data dependencies, and other possible computer-specific characterizations that can be collected periodically via verification drills. Work is under way to create a framework that can automatically handle these warm standby runs to verify application and policy readiness. We plan to leverage the INCA monitoring system [14] and MDS4 of the Globus Toolkit for this purpose. An important challenge is to be able to "encode" local site policies in such a way that they can be probed by the advisor.

Another goal is to incorporate more flexibility into job submissions and tokens. Currently, users can specify only the urgency level in their RSL. The inclusion of more information such as deadline can help the advisor select a suitable resource for the job to run. Some applications, such as hurricane and tornado modeling, have a lead time before they will need priority access. In such cases, the policy would need to be able to clear off the queues within the lead time, rather than providing next-to-run or preemptive access. Additionally, token holders might want the ability to aggregate tokens, rather than manage back-to-back separate sessions.

Yet another aspect that we are considering is providing a priority resubmit coupon to users whose jobs have been preempted during an urgent session. This coupon is similar to a token but would be valid for a single submission that is constrained to be similar to the preempted job (e.g., similar number of requested CPUs, requested wallclock time, etc.). The coupon would allow users to return to the top of the queue rather than the bottom, where they may wait days before getting back on the machine. This strategy provides a straightforward bound on the delay incurred when users are preempted - they will be inconvenienced no longer than is required for the urgent job to complete.

Currently, SPRUCE does not provide tools for prioritized data movement, which is crucial for many high-performance computations. Existing data movement strategies and tools such as the GridFTP project [15] will facilitate SPRUCE in the future. In addition, network provisioning that reserves bandwidth may be needed for applications with large data requirements [16].

One drawback of the current design is that there exists a single point of failure - the SPRUCE portal. If the portal is experiencing downtime or the user fails to access it, the only alternative is to abort an urgent submission. In order to counteract this weakness, the portal and the backend database will require redundancy and remote fail-over locations. The existing version of the portal is also subject to the same variety of attacks as other Internet Web servers, including denial of service, spoofing, and abuse of software vulnerabilities. These and other research topics have received considerable attention; we hope to take advantage of these efforts in our future work.

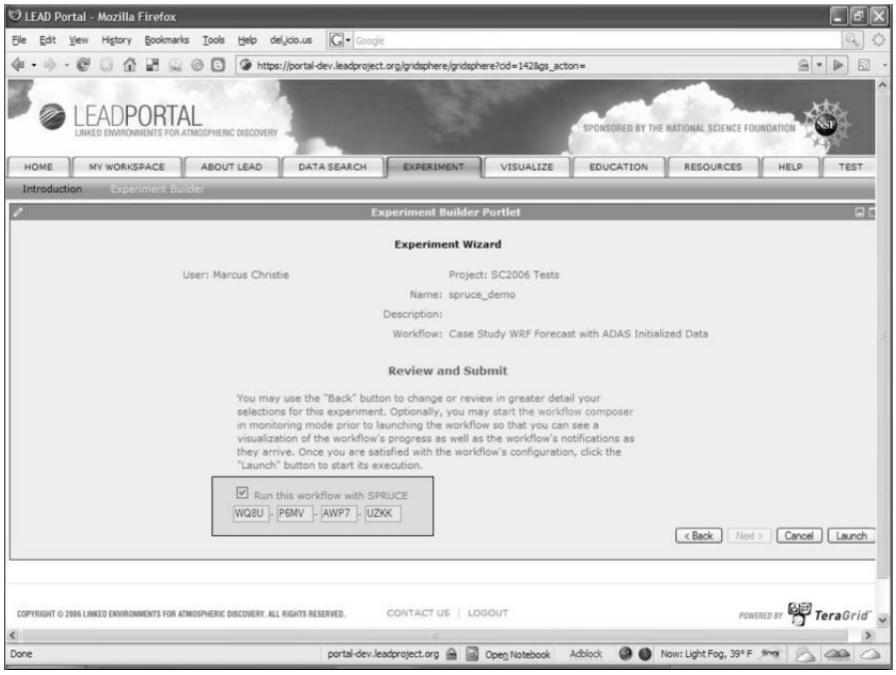

Figure 8. Screenshot of SPRUCE integrated into the LEAD automated workflow

3.8. Experiences

Currently, SPRUCE is deployed at the University of Chicago/Argonne National Laboratory (UC/ANL), Purdue University, Texas Advanced Computing Center (TACC), San Diego Supercomputer Center (SDSC), and the National Center for Supercomputing Applications (NCSA) and is being deployed at Indiana University. Virginia Tech University is one of our early non-TeraGrid adopters planning to use this system for simulating epidemiological spread patterns as a part of EPISIMS [17]. Louisiana State University is also considering installing SPRUCE on it's local cluster for the SCOOP [18] project.

Collaboration between the LEAD weather forecasting and analysis project [19] and the UC/ANL TeraGrid resources resulted in real-time, on-demand severe weather modeling and forecasting during May and June 2007. Additionally, the UC/ANL IA64 machine currently supports preemption for the highest priority urgent jobs. As an incentive to use the platform even though jobs may be killed, users are charged at a rate of 90 percent of the standard CPU service unit billing. Deciding which jobs are preempted is determined by an internal scheduler algorithm that considers several aspects, such as the elapsed time for the existing job, number of nodes and jobs per user. The complete policy mapping on the IA64 UC/ANL resource is as follows.

- Yellow - elevated priority
- Orange - next-to-run status
- Red - preemptive access

The LEAD collaboration clearly demonstrates the capability that SPRUCE can provide to existing workflows. LEAD was given a limited number of tokens for use through-

out the tornado season. Urgent runs are triggered automatically by parsing the RSS feed of advisories published by National Weather Service for possible warning flags. A right-of-way token at the required urgency level is activated based on the warning, and a list of users is added onto the token automatically via the Web service interface to the SPRUCE server. Alternatively, community users can activate tokens given to them from the LEAD portal directly, without logging into the SPRUCE portal (see Figure 8).

A significant challenge involved allowing local sites to establish their own policies while keeping the SPRUCE installation as simple as possible. Each site needs a customized version of the job manager (dependent on the site policy and scheduler), which cannot be bundled into a common distribution. Hence, site administrators must make minor modifications to the distributed SPRUCE job manager to integrate SPRUCE into their systems. All these changes are well documented.

4. Discussion

Apart from the framework, urgent computing needs to focus on three key areas in the long run: Policy, Planning and Technology.

4.1. Policy

Most resource providers and users support the basic ideas of on-demand and urgent computing. However, the realities of resource utilization, user expectations, and current policies make supporting urgent computing challenging. At this time, funding agencies such as the NSF and DOE evaluate HPC centers on utilization and delivered flops per dollar. Viewed in this narrow scope, support for urgent computing has few benefits to the participants in this market. Warm standby computing drills and eager scheduling of urgent jobs depress resource utilization. Urgent runs can disrupt running jobs and delay those waiting in the queue. Drill runs to provide application warm standby are also costly, with no clear answer as to which allocation should be charged for the continual testing of an urgent application. However, these issues stem from the priorities and metrics used by the stakeholders. Traditionally, the General Accounting Office has asked large government-funded supercomputer centers for measures including system utilization, cost, and average time waiting in a queue. Hand-in-hand with building support for urgent computing we must create new metrics and policies to reflect priorities other than peak flops per dollar.

Because urgent computing is by its very design disruptive to the normal operation of a center, incentives and pricing models must be explored to encourage user participation and adoption. For example, urgent computing can create incentives for application developers to develop codes that can be checkpoint/restarted or preempted. Providing rewards or CPU-pricing incentives to use special "interruptible" job queues can help free more resources for urgent computing. In return, jobs from these queues will be offered a discounted rate. There is also an implicit benefit for applications to be compatible with urgent computing because it encourages development of code that is robust to interruptions and failures.

Another challenge is the human response to disruptions caused by urgent computing. For example, SPRUCE changes the established model of resource control. In hand-

ing out a SPRUCE token to a resource, the administrator is making a promise to a potential future computation. The administrator is also delegating the decision to preempt jobs to someone else. It removes some of the hands-on control administrators are used to. Outstanding SPRUCE tokens increase the probability of an urgent computing job disrupting regular use of the resource. Since urgent computations are unplanned, they also jeopardize resource utilization in the long run. To participate in an urgent computing network, the rewards for the resource provider must outweigh the reduced utilization and decreased predictability of the system.

4.2. Planning

During security breaches, professionals rely on playbooks to dictate a swift course of action. Likewise, emergencies require fast and thoughtful response based on early planning and organization. Playbooks, or step-by-step instructions on a course of action during emergency scenarios, are especially valuable in this regard. Urgent computing is not something that can be organized on the fly, and we believe that resource providers, urgent application programmers, scientists, and decision makers should all maintain urgent computing playbooks for preparedness. Besides thorough instructions, an urgent computing playbook can contain essential contact information for all the participants in a scenario response to mitigating the overhead of organizing a team during an emergency. Playbooks can also increase awareness of how applications and their output contribute to a coherent response plan that may span multiple organizations, and many time zones.

For many situations where urgent computing could provide insight, the set of relevant codes can be organized in advance. However, there will always be situations outside the planned scenarios or playbooks. Locating application codes and science teams relevant to a particular scenario can take a prohibitively long time. During emergencies, officials should know what relevant applications and scientists might be able to help guide a response scenario. To facilitate this process, we envision a database that aggregates information about applications that could be useful for constructing a dynamic response.

4.3. Technology

Virtualization technologies such as VMWare [21] and Xen [22] hold considerable promise for urgent computing. Virtualization has the benefit of standardizing a runtime environment that can be transported to a resource without worrying about an application's library and architecture dependencies. Urgent codes maintained in virtual machines can be deployed to the resource and begin execution immediately without any overhead of manual configuration. This will lower turnaround time and simplify application maintenance and drill procedures. Another benefit of using virtual machines is that they minimally impact already running virtualized applications through graceful degradation. In a virtualized environment, checkpointing or killing a running job is not necessary to simultaneously execute an urgent job, since the urgent job can time share the CPU and other resources. Or, the virtual machine for a preempted job could be moved to a smaller or slower resource. One drawback to using virtual machines, however, is their large memory and disk footprint. A virtual machine image needs to include libraries, configuration settings, and other data that is usually not bundled with the application.

In building and deploying SPRUCE we have concentrated our efforts primarily on federally sponsored supercomputing centers. However, there are commercial alternatives

to computing and storage resources. For example, Amazon provides the EC2 (Elastic Compute Cloud) service [23] and S3 (Simple Storage Service) [24] frameworks. Although they are "best effort" resources, these commercial services can be potential resources during urgent need, especially in cases when other infrastructure is unavailable. Amazon's services are competitive alternatives because, unlike other resource providers, they fully support virtualization and can prioritize and coordinate large reservations for customers who are willing to pay the premium.

5. Conclusions

Urgent computing is a new and evolving field made possible by the improved fidelity and utility of high-performance computing to decision making and near-real-time instrument steering and control. To support this new field, scientists must work together to develop not only the technology but also the policies to support research and development for emerging urgent computing applications. The community must organize policies that can be used to help guide answers to the questions of "who," "why," and "under what circumstances" application teams will receive permission to initiate urgent computing sessions. To build an international infrastructure to support urgent computing, we must solve both challenges: political and technical. We believe that by deploying prototype systems such as SPRUCE, the community can test and explore this new way to use applications and resources.

6. Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

References

[1] "IPCC Summary for Policymakers," http://www.ipcc.ch/SPM040507.pdf.

[2] National Aeronautics and Space Administration, "High-End Computing at NASA," Technical Report, 2006.

[3] "Urgent Computing Workshop 2007," http://spruce.uchicago.edu/workshop/urgent07.php.

[4] "Telecommunications Service Priority (TSP) program," http://tsp.ncs.gov.

[5] S. Vazhkudai and J. Schopf, "Using Regression Techniques to Predict Large Data Transfers," in *International Journal of High Performance Computing Applications*, 2003, pp. 249–268.

[6] M. Swany and R. Wolski, "Multivariate Resource Performance Forecasting in the Network Weather Service," in *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002, pp. 1–10.

[7] R. Wolski, "Experiences with Predicting Resource Performance On-Line in Computational Grid Settings," *ACM SIGMETRICS Performance Evaluation Review: SPECIAL ISSUE: Special section on Grid computing*, 2003, pp. 41–49.

[8] R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *Future Generation Computer Systems*, pp. 757–768, 1999.

[9] J. Schopf, M. D'Arcy, N. Miller, L. Pearlman, I. Foster, and C. Kesselman, "Monitoring and Discovery in a Web Services Framework: Functionality and Performance of the Globus Toolkit's MDS4," Argonne National Laboratory, Preprint ANL/MCS-P1248-0405, April 2005.

[10] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," in *IFIP International Conference on Network and Parallel Computing*, 2005, pp. 2–13.
[11] "Globus Resource Specification Language," http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html.
[12] "PBS 'qsub' Job Submission Tool," http://www.clusterresources.com/torquedocs21/commands/qsub.shtml.
[13] "Torque Submit Filter," http://www.clusterresources.com/products/torque/docs20/a.jqsubwrapper.shtml.
[14] "Test Harness and Reporting Framework (INCA)," http://inca.sdsc.edu.
[15] "GridFTP Project," http://www.globus.org/grid_software/data/gridftp.php.
[16] L. Gommans, F. Travostino, John, Vollbrecht, C. de Laat, and R. Meijer, "Token-based Authorization of Connection Oriented Network Resources," in *GRIDNETS Conference Proceedings*, October 2004.
[17] "EPISIMS and Simfrastructure," http://ndssl.vbi.vt.edu/episims.html.
[18] "Sura Coastal Ocean Observing and Prediction," http://www.scoop.lsu.edu/gridsphere/gridsphere.
[19] "Linked Environments for Atmospheric Discovery (LEAD)," http://lead.ou.edu/.
[20] "Apache Axis 2," http://ws.apache.org/axis2/.
[21] "VMWare,", http://www.vmware.com/.
[22] "Xen Virtualization Software," http://www.xensource.com/.
[23] "Amazon EC2, Amazon Elastic Compute Cloud," http://aws.amazon.com/ec2.
[24] "Amazon S3, Amazon Simple Storage Service," http://aws.amazon.com/s3.