SPRUCE: A System for Supporting Urgent High-Performance Computing

Pete Beckman[1], Suman Nadella[2], Nick Trebon[2], and Ivan Beschastnikh[3]

Mathematics and Computer Science Division, Argonne National Laboratory
9700 S. Cass Avenue, Argonne, IL 60439 beckman@mcs.anl.gov
Computation Institute, The University of Chicago/ Argonne National Laboratory
5801 S. Ellis Avenue, Chicago, IL 60637 snadella,ntrebon@uchicago.edu
Computer Science Dept, The University of Washington
Seattle, WA 98195 ivan@cs.washington.edu

Modeling and simulation using high-performance computing are playing an increasingly important role in decision making and prediction. For time-critical emergency decision support applications, such as influenza modeling and severe weather prediction, late results may be useless. A specialized infrastructure is needed to provide computational resources quickly. This paper describes the architecture and implementation of SPRUCE, a system for supporting urgent computing on both traditional supercomputers and distributed computing Grids. Currently deployed on the TeraGrid, SPRUCE provides users with "right-of-way tokens" that can be activated from a Web-based portal or Web service invocation in the event of an urgent computing need. Tokens are transferrable and can be restricted to specific resource sets and priority levels. Once a session is activated, job submissions may request elevated priority. Based on local policy, computing resources can respond, for example, by preempting active jobs or raising the job's priority in the queue. This paper also explores the strengths and weaknesses of the SPRUCE architecture and token-based activation for urgent computing applications.

1 Introduction

Scientific computing is playing an ever-increasing role in making critical decisions. For example, global climate modeling played a key role in influencing the Kyoto Protocol for the reduction of greenhouse gas emissions [1]. Likewise, computer models have helped large metropolitan areas plan for new highways and congestion relief [2]. 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. There are, however, growing sets 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 the flow of airborne contaminants from a ruptured

Please use the following format when citing this chapter:

Beckman, P., Nadella, S., Trebon, N., Beschastnikh, I., 2007, in IFIP International Federation for Information Processing, Volume 239, Grid-Based Problem Solving Environments, eds. Gaffney, P. W., Pool, J.C.T., (Boston: Springer), pp. 295-311.

railcar must guide evacuation while there is still time. These on-demand large-scale computations cannot wait in a job queue for Grid resources to become available. However, neither can the scientific community afford to keep multi-million dollar infrastructures idle until required by an emergency. Instead, we must develop technologies that can support urgent computation. Scientists need mechanisms to find, evaluate, select, and launch elevated-priority applications on high-performance computing (HPC) resources. Those computations might re-order, preempt, or terminate existing jobs to provide the needed cycles in time. SPRUCE, the *S*pecial *PR*iority and *U*rgent *C*omputing *E*nvironment, is a system for providing resources quickly and efficiently to high-priority applications that must get computational power without delay. This paper makes two contributions: it presents and analyzes an architecture for supporting urgent computing across large production Grids, and it provides implementation experiences from the TeraGrid [3] deployment.

1.1 Requirements for Supporting Urgent Computing

Urgent computing can be supported in many ways. Priority queues, administrative intervention, and emergency stand-by resources could all be used to provide compute cycles quickly. While these methods may be effective for some usage scenarios, however, they cannot provide a feature-rich architecture capable of supporting large, distributed Grids. A Grid-based architecture for urgent computing must meet the following design requirements:

- Urgent computing jobs must occur within a clearly defined "session." System administrators are notified when sessions begin, permitting periods of increased attentiveness and, if needed, human intervention to provide the resources required.
- The system must support possibly different urgent computing policy frameworks among Grid resource providers and coexist with ongoing operations. For example, some HPC centers may support preemption for certain applications or priorities, while other HPC centers may provide only "run-next" priority following the normal completion of existing jobs.
- Permission to initiate an urgent computing session must be easily transferrable so team leaders, managers, and senior personnel can respond quickly to emergency situations.
- Application team leaders must be able to quickly assemble and authorize sets of users to submit priority jobs across a Grid that spans multiple sites and administration domains.

1.2 SPRUCE Architecture

Our architecture for urgent computing uses a token-based system to address these requirements. Tokens can have various levels of priority and different sets of resources applicable to them. Initiating an urgent computing session

begins with the initialization, or "activation," of a token via a Web portal or command line. Tokens are simple authorization codes and therefore completely transferrable either electronically or on a printed card. 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 key managers at hospitals, fire departments, government offices, and 911 centers, carry a wallet-sized card with an authorization number. Cardholders can use the code 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, important traffic can get the priority needed. Since tokens are transferrable, users benefit from tremendous flexibility during critical-response situations.

To support token-based access to elevated-priority resources, the SPRUCE architecture has three main components: user workflow and client-side job submission tools, a Web service-based user interface to manage tokens, and local resource provider agents that can respond to the request for priority access. The remainder of this paper presents the design and implementation of each of these components, an analysis of the architecture, and our experiences deploying the system on the TeraGrid.

2 SPRUCE User Workflow

The SPRUCE workflow is designed for the application teams that can provide computer-aided decision support. Each application team is organized by its principal investigator (PI). The PI selects the computational "first responders," senior staff who can evaluate the request, initiate a SPRUCE session, and engage the other team members. The first responders hold the right-of-way tokens and decide when they should be used based on the best available information and the policies of the Grid system. The PI also selects an "interpreter" to translate the raw data and simulation output into advice for decision makers. For example, imagine an application that models airflow across a city and can be used to evaluate contamination scenarios. The results of that simulation may have many subtle details that need interpretation and presentation to city managers as they formulate response scenarios.

Figure 1 illustrates how the SPRUCE workflow is initiated. A trigger causes the computational first responders to spring into action. The trigger may be automatic, such as an automated warning message from a tsunami alarm buoy, or human generated, for example by a phone call to the PI. For many applications, the computing request may include a deadline. If the results cannot be provided before the deadline, the window for effective decision-support will have passed. The scientists must choose an appropriate priority level for the situation based on the importance of the job to be submitted. SPRUCE right-of-way token holders must adhere to the policies concerning activation and must use

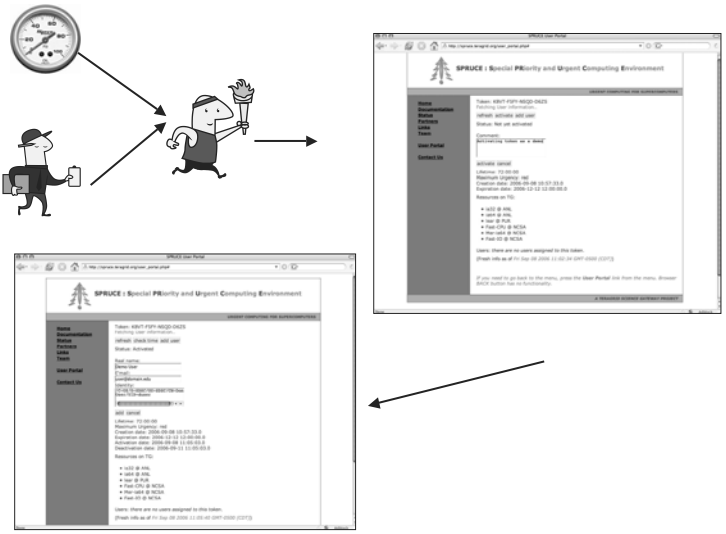

Fig. 1. SPRUCE workflow and token activation

discretion, in the same way that citizens are expected to use good judgment before dialing 911. Users must be aware that misuse can result in the revocation of their tokens.

Interaction with the SPRUCE system starts by activating a right-of-way token. A Web-based portal built on Web services is provided, where the token manipulations can be performed. Additionally, the Web service functionality may be incorporated into automated work-flows, thereby avoiding human intervention in managing tokens. Activation is described in greater detail in Section 3. 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 indicate which scientist or set of scientists will be able to request elevated priority while submitting urgent jobs.

2.1 Choosing Resources

When a scientist wants to choose a resource to run on, two factors must be considered. The application needs to be fine tuned to suit the resource environment, and the policy pertaining to priority access should be functioning correctly.

With the token activated and the application team 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.

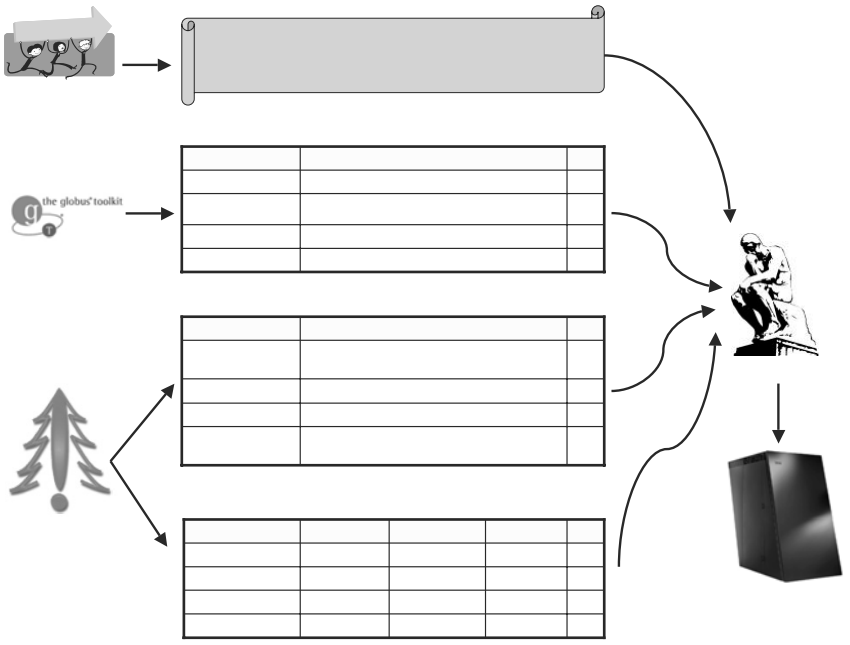

Fig. 2. The SPRUCE advisor helps choose the best resource

Applications must be prepared for immediate use—they must be in "warm standby." All of the application development, testing, and tuning must be complete before 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 require programmer resources, and often application teams narrow their efforts to a handful of favorite platforms and sites. Additionally, these sites should have their priority policy in place and all the hooks needed to implement it immediately when needed. 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 recorded.

From this pool of Grid resources where the validated application awaits in warm standby, the team must select where to submit their jobs. This process can be challenging. In a distributed Grid linking resources provided by independent resource providers, different urgent computing policies will exist. One site may provide only a slightly increased priority to SPRUCE jobs, while another site

may kill running jobs and hand the entire supercomputer to extremely urgent computations. On resources where existing jobs are not killed or preempted, current job load will affect resource selection. Data movement may also constrain resource selection. To support finding and choosing resources, SPRUCE users may select resources manually or may use an automated SPRUCE "advisor" (see Figure 2).

The proposed SPRUCE advisor needs four pieces of data to recommend where jobs should be submitted: the deadline for results, the estimated running time of the job, each site's urgent computing policy expressed as a scheduling algorithm for SPRUCE jobs, and the current status of job queues at the resource sites, which can be provided via MDS4 [5] on the TeraGrid. By combining these parameters with the application validation histories for each resource, several resources can be recommended or even automatically selected. If manual selection of the resource is preferred, the user may analyze the job queue status reports from MDS4 and examine previous warm standby results from the SPRUCE database to make a decision. Once the resource is selected, the user submits the job with a designated urgency.

2.2 Prioritized Job Submission

SPRUCE provides support for both Globus-based urgent submissions and direct submission to local job-queuing systems. Currently all the major resource managers such as Torque, LoadLeveler, and LSF and schedulers such as Moab, Maui, PBS Pro, and Catalina are supported. 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 [6] for Grid computing provides the TeraGrid with uniform tools for authentication, job submission, file transfer, and resource description. Users can submit remote jobs to any of the TeraGrid platforms. By extending the Resource Specification Language (RSL) [7], which is used by Globus to identify user-specific resource requests, we give the user the ability to indicate a level of urgency for jobs. A new "urgency" parameter is defined for three levels: *critical* (red), *high* (orange), and *important* (yellow). Urgency levels are used in two places. Gridwide policies and guidelines can help scientists organize and differentiate potentially competing jobs by urgency. On the back end, the resource provider can enable site-local response protocols according to urgency.

The urgency can be specified within a Globus submission job script. Figure 3 shows an example. The site-local job manager agents check for validity of the request based on the token attributes applicable to that particular user and respond accordingly.

Unlike the Globus RSL, local job queue submission interfaces, such as the PBS command *qsub* [8], are often not trivially extended to accept new parameters. To specify the urgency level when submitting directly to a computer's

```
              _test.rsl
+
(&
(resourceManagerContact =
site-contact.teragrid.org/jobmanager-spruce)
(rsl_substitution =
(HOMEDIR "/soft/spruce/examples"))
(executable = $(HOMEDIR)/mpihello)
(jobType = mpi)
(host_types = ia64-compute)
(host_xcount = 4)
(urgency = red)
(stdout = $(HOMEDIR)/globus_stdout)
(stderr = $(HOMEDIR)/globus_stderr)
)
> globusrun -o f globus_test.rsl
Jobnumber.resource.teragrid.org
```

Fig. 3. *globusrun* usage with additional job submission parameters

local job queue usually requires a modified job submission command or a wrapper script. SPRUCE provides a *spruce_sub* script that accepts the additional command line parameter, which can be yellow, orange, or red depending on the required priority.

From the user's perspective, with the job submitted, the final step to the workflow is waiting for the job to be run and the data analyzed. If the resource does not launch the job as expected or if there are run-time errors, the job can be killed or dequeued by using the normally supported tools for the system. Behind the scenes, during the submission process the job is also checked against activated right-of-way tokens. Jobs for users without valid session tokens or unauthorized urgency levels will not be queued. They are rejected immediately. The authorization mechanism is described in the next section.

3 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 lets privileged users use a standard Web browser to create, issue, monitor, and deactivate right-of-way tokens. It also allows SPRUCE administrators to manage the portal, including adding other administrators, registering new resources, and changing notification email addresses.
- The Web service-based user interface permits token holders to activate an urgent computing session and manage user permissions. Additional features include monitoring session and user information.

– The authentication service verifies jobs. Local site job manager agents ask the remote SPRUCE portal to validate urgent computing jobs. Provided that the user is associated with an active urgent computing session for the local resource and the requested level is within bounds, the portal approves the request.

3.1 Right-of-Way Tokens

Many possible implementations exist for authorizing an emergency computation, ranging from digital certificates, signed files, and proxy authentication servers to shared-secret passwords. In emergency situations, however, simpler is better. Relying on complex digital authentication and authorization schemes could easily become a stumbling block to timely response. Instead, SPRUCE uses simple, transferrable right-of-way tokens (see Figure 4). Tokens are unique 16-character strings that are issued to scientists who have permission to request urgent priority. When a token is created, several important attributes are set: the resource list of included machines, maximum allowable priority, lifetime (period for which elevated priority jobs may be submitted), and expiration date of the token.

Sites can enforce their own policies for each of the allowed priorities. The intent is to have jobs with higher priority displace lower-priority jobs if resources are limited during instances of simultaneous requests. By carefully selecting the attributes of tokens when they are created, local site administrators can make decisions regarding the relative importance of projects and the resources they may use for urgent computation. SPRUCE can support distributed computation in the form of "cross-site" tokens for which resources at multiple sites, or even all resources in the Grid, can be utilized. After a token's lifetime has run out, another token must be activated if additional priority computation is required.

Fig. 4. SPRUCE "right-of-way" token

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 user only

to request elevated priority for job submission. Without the token, requests for elevated job priority are simply logged and then ignored. Moreover, after activating the token, only jobs submitted with special, elevated-priority job parameters, described in Section 2.2, will receive unique treatment.

3.2 Administration Interface

Distributed Grids have multiple administration domains. Some Gridwide policies and procedures can be set for all participating resource providers. For the TeraGrid, the Grid Infrastructure Group (GIG) coordinates the software infrastructure, allocation and usage reporting, user support, and Grid security. While each resource provider, such as the San Diego Supercomputer Center, the University of Chicago, or the Texas Advanced Computing Center, has a loosely defined "service level agreement" for participation in the TeraGrid, they are nevertheless independent organizations. To support multiple administration domains and virtual organizations across multiple overlapping Grid systems, SPRUCE maintains a hierarchical Web-based administration interface, which is organized into three levels, each granted powers within its respective domain. Ordered by increasing privileges, these are the site (Grid resource provider), virtual organization, and root administrative domains. Figure 5 illustrates the hierarchical nature of the admin interface.

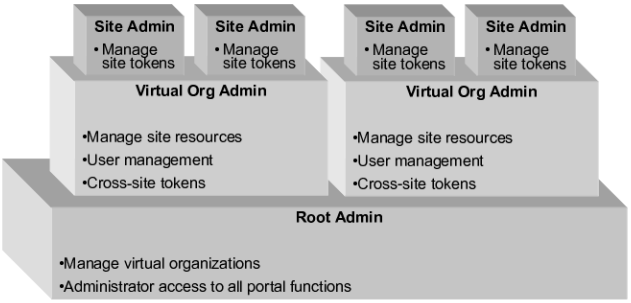

Fig. 5. SPRUCE portal hierarchy of functionality

To permit possibly differing resource and management policies at each site, SPRUCE supports multiple sites under a virtual organization. A token created by a site administrator may be used only for resources present at that site. 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. The administrator may create and distribute tokens that are limited in scope to the computers the site operates. For example, a local earthquake-modeling team working with an HPC center can be presented tokens that are valid only on the

specific supercomputer designated for that research. The local administrator is also responsible for managing the identities of local users as well as the list of machine hosts supporting SPRUCE.

The next level of administration is for the virtual organization spanning several sites, such as the TeraGrid. The administrator for the TeraGrid GIG may issue tokens for resources at multiple sites, in accordance with the policies and service level agreement and management structure of the Grid system. Activating a cross-site token provides users a large collection of machines spread across multiple sites for their jobs. At this level the administrator for the virtual organization can also add new sites.

Administrators also have access to the logging and status information maintained by the portal. Included among this data are the token activation statistics and monitoring of failed attempts to use elevated privilege.

3.3 User Interface

The users of the SPRUCE user interface are the scientists responsible for organizing the application team. Their tasks include monitoring the status of tokens, activating sessions, and organizing the team that may participate in an urgent computing session. The interface for this user community must be simple, fast, and modeled after the workflow described earlier in Section 2. The user services are specifically designed as Web services in order to enable incorporation into existing scientific Web portals and work flows. Users who prefer to use a Web-based interface, can do so at the SPRUCE user portal, which is built on top of these services.

The first step for a computational first responder is to activate a token by entering its 16-digit code via a Web service call either from a local workflow or from the SPRUCE portal. At that instant, the urgent computing session begins. We expect typical token lifetimes to range from 4 to 24 hours, during which the submission of priority jobs to the resources is permitted.

With the token activated and the session begun, the next step in the workflow is user management. For convenience during an emergency, the users who will be running the jobs can be "preloaded" onto a token, if known prior. If not, the participants of the session can be added after token activation. Changes to the user associations of a token are propagated without delay. After a participant is associated with an active token, urgent computing jobs will authenticate correctly. Token holders can also remove participants as needed. All SPRUCE users may monitor basic statistics such as the remaining lifetime of the token.

Since SPRUCE supports urgent computing for Grid users as well as traditional supercomputing users, the portal maintains two methods for specifying the participants in an active session. For those users with Grid credentials, the Distinguished Name (DN) for the user is appropriate. Sites without Grid support can use the Unix username of the participant for the resource.

3.4 Job Authentication

At the core of the SPRUCE architecture is the notion that only while a right-of-way token is active may urgent jobs be submitted. In order to support this notion across a distributed Grid system, a remote authentication step must be inserted into the job submission toolchain 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 triggers authentication. Remember, this authentication is not for the user, which has already been handled by the traditional Grid certificate or by logging into the Unix-based resource, but is really a "Mother, may I" request for permission to enqueue a high-priority job. That request is sent via the network to the SPRUCE portal, where it is checked against active tokens, resource names, maximum priority, and associated users. If a right-of-way token has already been activated and the other parameters for the job request are within the constraints of the token, permission is granted. All transactions, successful and unsuccessful, are logged.

4 Resource Providers

To support urgent computing for supercomputers via the SPRUCE system, the resource provider 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 queuing system.

4.1 Portal Registration

Sites participating in SPRUCE need an administrative account on the SPRUCE portal. From that account, administrators can provide the details for each of the computational resources that will support urgent jobs. The site administrator will also provide important contact information that can be used for emergency notification, for example when tokens are activated or critical errors occur. Once that preliminary information has been set up, 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 merged with the corresponding virtual organization.

4.2 Responding to Urgent Computation

The SPRUCE architecture does not define or assume any particular policy for how sites must respond to urgent computing requests. This approach complicates the architecture and usage scenarios, but it is unavoidable given the

current state of systems software for supercomputers. 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 simply to permit shorter jobs to achieve fast turnaround times during compile or debug sessions. Unfortunately, almost all modern supercomputers have lost this once 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 for 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 some of the possibilities:

- 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 queuing and job management systems support priority queues that will used for selecting the next job to run. No existing 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 existing jobs complete for a given set of CPUs.
- Suspending existing jobs and immediately launching the urgent job. Some systems allow jobs to be suspended but remain resident in memory (sig STOP). Running the urgent job will then force some memory paging, but the suspended job could be restarted 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, the suspended and the urgent job will be lost. The urgent compuitation 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. Some architectures support system-based checkpoint/restart. Where it is reliable, it could be used to support urgent jobs. Jobs will begin when the checkpoint completes. For large-memory systems, it could be 30 minutes or more depending on I/O and disk rates.
- Killing all running jobs and enqueuing the urgent job as next to run. Clearly this response is drastic and frustrating to users, who will lose their computation. Nevertheless, for extremely urgent computation, what user would demand a black-hole simulation complete before launching an emergency hurricane flood modeling scenario? Urgent jobs could begin immediately after existing jobs are killed.

Another factor in choosing the policy for response is accounting and stakeholder accountability. Certain machines are funded for specific activities, and only small amounts of discretionary time are 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 one. For example, users could be refunded their CPU hours, given extra time for their trouble, and rescheduled with higher priority. They could then get back on the machine quickly after the urgent job completes, rather than 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 computation. 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 are lost should a CPU fail or the entire system go down. Such users should be rewarded. A discounted rate would allow them to regain their lost work and run more inexpensively.

The calculation of "maximum time to begin" may play an important role in choosing a response strategy. For machines that support checkpoint/restart or simply killing existing 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 existing jobs complete. In any case, resource providers are encouraged to map all three levels of urgency—critical, high, and important—to clearly defined responses.

Once the resource provider has decided on a policy and has installed SPRUCE, token holders can activate tokens and associate users, who will then submit urgent priority jobs in one of two ways. Jobs can request priority access by specifying urgency parameters either in Globus job submissions or by using a stand-alone command line *spruce_sub* as described in Section 2.2. These requests are processed by the SPRUCE Job Manager component, which verifies the job request and implements the local policy.

Figure 6 gives an overview of how the job requests are handled at the resource provider.

4.3 Handling Urgent Job Submissions

In the Globus architecture, incoming jobs are routed to a job manager. A job manager tailored to support SPRUCE handles the additional job parameters. When an urgent SPRUCE job is submitted, a job script is dynamically assembled and passed to the native resource manager such as PBS Pro or Torque. It then authenticates the request against the portal (see Section 3.4). This filter makes sure that all job scripts were prepared by the job manager rather than a user attempting to sneak a job into the high-priority queue without SPRUCE validation. In the case of the Torque scheduler, a submit-filter [9] script specific to SPRUCE is run every time a job is submitted. If the user does not have sufficient permission, the request is rejected. If the request passes the verification

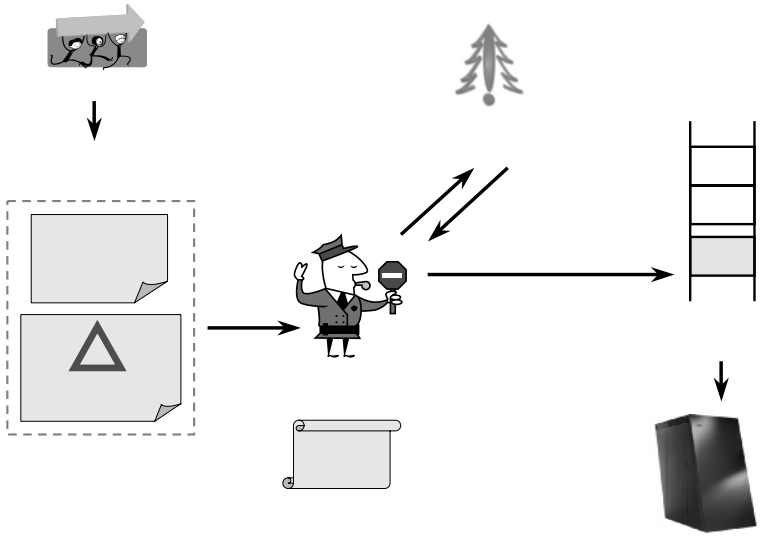

Fig. 6. Resource provider architecture

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; hence the implementation mechanism remains the same. The only difference between these two submission methods is in the interface.

5 Experiences and Analysis

Currently, SPRUCE is deployed on University of Chicago/Argonne National Laboratory (UC/ANL), Purdue University, Texas Advanced Computing Center (TACC), San Diego Supercomputing Center (SDSC), National Center for Supercomputing Applications (NCSA) TeraGrid resource providers and is in the process of being deployed at Indiana University. Louisiana State University is one of our early non-TeraGrid adopters to use the system for the coastal modeling project SCOOP [10]. We are working with the LEAD Project [11] as they gear up to run severe weather simulations in response to real-time weather data. Tokens are distributed to key members who will act as test users of the

system. Efforts are ongoing to configure the applications for periodic warm-standby tests.

The existing implementation of the system encapsulates all the basic framework necessary to allow urgent job submissions. Team first responders can activate and associate user identities with tokens. Team members can then submit jobs with next-to-run priority. The hierarchical administrative domains described in Section 3.2 allow site administrators to manage local tokens.

At the moment, Globus submissions are restricted to the use of the *globusrun* command, and the direct submissions tool *spruce sub* does not handle command line PBS options. All three priority levels map to the next-to-run policy implemented as a priority queue. There are customized distributions to work with most of the popular resource managers and schedulers, as mentioned in Section 2.2. All of them are compatible with pre-Web services versions of the Globus Toolkit. Work is under way to extend the flexibility of the submission tools, enable multisite submissions, and provide extended policy support.

The SPRUCE architecture is designed to work independently or as a part of the Globus installation. The biggest strength of the design is its flexibility: the ability to adapt to any environment it might be interfaced with. The user functionality is implemented completely as Web services using Apache AXIS 2 [12]. External portals and workflows can simply use the Web service interfaces from within their applications. The portal is implemented in PHP and MySQL, uses the underlying Web services, and runs on the Apache Web server. Using a simple Web browser, SPRUCE users can interact with the system. Only minimal additional training is needed, making SPRUCE appropriate for emergency situations. Likewise, administrators will find the interface easy to navigate and use regardless of their environment.

One drawback of the current design of the architecture is that there exists a single point of failure in the form of the portal. If the SPRUCE portal goes down or the user cannot access it, there is no way other way to route the urgent jobs. In order to counteract this weakness, the portal 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 exploits are current research topics and have received considerable attention; we hope to take advantage of these efforts in our future work.

Another challenge is to allow local sites to establish their own policies while keeping SPRUCE installation as simple as possible. Each site needs a customized version of the job manager depending on 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 work with their systems.

6 Future Work

Two of the most attractive and challenging components of the architecture remain to be implemented: the advisor and automated warm-standby testing. SPRUCE jobs are emergency codes that require active maintenance and have certain dependencies that must be taken into account by the SPRUCE advisor before suggesting possible scheduling scenarios to the user. Job-specific information such as running time, data dependencies, and other possible computer-specific characterizations must be collected periodically to ensure that the most recent information is used by the advisor. Warm standby will automate what many users currently do by hand and will ensure the reliability of the monitored emergency codes. It will also help us validate the policy enforcement from time to time. Work is in progress to implement both features by using the INCA monitoring system [13] and MDS4 of the Globus Toolkit. Warm-standby applications require substantial programmer effort, CPU time dedicated to periodic test runs, and fast data transfer. Since the user applications can best be tested for their readiness in the actual environment of the person who will be submitting the job, INCA needs to be customized to submit jobs as the user or collect the data on behalf of the user.

We also plan to incorporate more flexibility to job submissions. Token holders should be able to aggregate tokens and submit jobs from the portal directly. Such flexibility will make SPRUCE a one-stop place to get priority access, select a resource, and submit urgent jobs.

The presented SPRUCE architecture doesn't deal with data movement, which is crucial for most high-performance computations. The advisor will need to take into account transfer delays associated with data movement before advising on the best set of resources. Warm-standby jobs will also have data dependencies that must be resolved automatically. Existing data movement strategies and tools such as the GridFTP project [14] will facilitate SPRUCE. Token-based network authorization would also be a good fit for network provisioning [15].

7 Conclusions

In this paper we have presented the architecture of SPRUCE, a token-based service for providing urgent access to high-performance computing resources modeled after the U.S high-priority telephone system in wide deployment today. The use of tokens allows urgent access to be physically transferrable, and the tokens have restricted access that is encoded prior to their issue. Tokens have expiration dates and lifetimes and may be redeemed only on resources that have been previously encoded into the token. A Web service-based user interface lets scientists manage their tokens easily and efficiently. Moreover, tokens are thoroughly tracked, and all user actions may be monitored by a three-tier hierarchy of administrative domains allowing site, virtual organization, and SPRUCE administrators to enforce policies relevant to their administrative domain.

We have also shown our initial results and an analysis of the architecture for our first TeraGrid customers at the University of Chicago, TACC, SDSC, Purdue University, and NCSA. Currently, we are working on integrating into LEAD portal so scientists can use SPRUCE within their infrastructure. We anticipate significant new developments as more TeraGrid sites and user communities bring SPRUCE support online, including an advanced warm-standby system for periodic testing of emergency codes and policies, a resource selection advisor, and extensions to provide an urgent data movement capability.

References

1. C. D. Keeling, R. B. Bacastow, and T. P. Whorf, "Measurements of the concentration of carbon dioxide at Mauna Loa Observatory, Hawaii," *Carbon Dioxide Review*, pp. 377 – 385, 1982.
2. K. Nagel, R. Beckman, and C. Barrett, "Transmins for transportation planning," in *6th Int. Conf. on Computers in Urban Planning and Urban Management*, 1999.
3. "TeraGrid Project," http://www.teragrid.org.
4. "Telecommunications Service Priority (TSP) program," http://tsp.ncs.gov.
5. 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, Tech. Rep., 2005.
6. I. Foster, "Globus toolkit version 4: Software for service-oriented systems," in *IFIP International Conference on Network and Parallel Computing*, 2005, pp. 2–13.
7. "Globus Resource Specification Language," http://www.globus.org/toolkit/docs/2.4/gram/rsl spec1.html.
8. "PBS 'qsub' Job Submission Tool," http://www.clusterresources.com/products/torque/docs20/commands/qsub.shtml.
9. "Torque Submit Filter," http://www.clusterresources.com/products/torque/docs20/a.jqsubwrapper.shtml.
10. "Sura Coastal Ocean Observing and Prediction," http://www.scoop.lsu.edu/gridsphere/gridsphere.
11. "Linked Environments for Atmospheric Discovery (LEAD)," http://lead.ou.edu/.
12. "Apache AXIS 2," http://ws.apache.org/axis/.
13. "Test Harness and Reporting Framework (INCA)," http://inca.sdsc.edu.
14. "GridFTP Project," http://www.globus.org/grid software/data/gridftp.php.
15. 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.

Q&A – Suman Nadella

Comment: Craig Douglas

This is highly dangerous from an abuse and political viewpoint. The right approach is to get a fundamental piece of operating systems (checkpoint/restart) implemented universally.

Suman Nadella

Clearly, having system-based checkpoint/restart (CPR) will greatly help the implementation of SPRUCE. For years, CPR was available on many systems, and it permitted very flexible scheduling. However, with the community's move toward clusters with advanced compute node features, system-based CPR is a long way off. There are many Grid applications where compute nodes run TCP/IP services, making automatic system-level check pointing without application code cooperation impossible. While we certainly would benefit from a good CPR system, it is nevertheless important that the framework for supporting Urgent Computing proceed as rapidly as possible, using whatever capabilities exist on each machine. We do not believe, in general, that Urgent Computing poses a significant abuse risk. Scientists who use national supercomputing resources understand how their behavior affects their funding, reputation, and career. Given clear guidelines, we do not believe Urgent Computing will be misused, and should it be misused, the remedy is fast, simple, and effective -- revoke all tokens and kill the Urgent Computing job.

Comment: Gabrielle Allen

I think there are ways to set policies that can be used with on-demand computing, for example, charging less for use of preempt queues.

Suman Nadella

Yes, we believe that with a bit more planning, it may be possible to use a system of rewards and discounts to encourage good behavior and flexible scheduling.

Questioner: Sebastian Goasguen

Why not use standard ACL on a high priority queue?

Suman Nadella

SPRUCE can use whatever means are available to restrict usage on the high priority queue. Some systems provide ACLs. On such machines, SPRUCE can certainly use ACLs to further restrict submissions to only select users. However, we feel it is important to use a token-based system because of the inherent flexibility for Grid and multiple-resource computing as well as the very clear act of "activating a token". Activating a token is meant to signal an

Urgent Condition and alert people even before any ACL or queue is touched. After that happens, ACLs can indeed limit access.

Questioner: Sebastian Goasguen

Why not use the attribute authorization systems like gridshib to 'push' tokens to the resource, i.e., how different is SPRUCE from attribute authorization based system?

Suman Nadella

SPRUCE only uses the tokens to elevate priority, not authorize logins, accounts, or shells. We have specifically stayed clear of all authorization and authentication since each site manages those issues quite differently, and with different tools. Instead, SPRUCE assumes that accounts and the ability to submit jobs are already in place, and the only thing SPRUCE can do is request elevated priority after a token has been activated.

Comment: Brian Ford

This gives us an opportunity to take a responsible position as members of society, to show as scientists we understand the order of issues and have a sense of responsibility and order. It can help us to reposition the evaluation of scientists in the mind of the community. We are not just white coats, bringing doom and gloom from our science. We are responsible individuals who understand the prior need of community for urgent address of immediate emergencies.

Suman Nadella

Yes, thank you for the comment. We too believe that the science community should step forward and offer assistance in several areas where ongoing research can help, such as wildfire and storm prediction. If we begin to work out some of the mechanisms and ideas now, over the next 5 years we will be able to slowly move some of the science apps that could benefit society into a position to actually help.

Comment: Brian Ford

Second point - It is good to think and talk about these issues. We have to start at the beginning and think through the issues. This is what this talk and our reactions are about. Perhaps Nobel Prize winners need to recognize (if they don't already) that there are activities - particularly at specific moments and in special circumstances - that are more important than their individual research (valuable as that may be). There is an order in which computing resources should be used in an emergency. We need to consider and seek to derive that order. Pat Gaffney has a counter view.

Comment: Pat Gaffney

Utilities like water, electricity, and railways should NOT be privatized. They are the responsibility of the Government. Emergencies/ response / analysis

etc are the responsibility of Government. Therefore, they should have dedicated resources for this purpose.

Suman Nadella

We understand your concern, but think that the issue is largely about cost, not privatization. Currently, the government uses existing commercial radio stations to broadcast alerts to the public concerning severe weather and abducted children. The US government could maintain a completely independent set of broadcast towers and infrastructure that could then saturate the airwaves with warnings in the event of an emergency. However, everyone realizes that the cost of maintaining completely duplicate infrastructure, only to be used once or twice a month is not cost effective. What SPRUCE does is simply permit supercomputer resources to be used for Urgent Computation. Could the US build a supercomputer center specifically designed for Urgent Computation? Yes. Would it be the best way to use government money? That's a policy question that machine stakeholders need to address. We believe that the responsible use of large-scale resources can be a community service.

Questioner: Mary Thomas

What are your plans to adopt this system to sensor networks or FPGA's?

Suman Nadella

We currently have no plans for integrating sensor networks or FPGA except as they may trigger Urgent Computing jobs. For example, the tornado modeling folks have radar data that can be used to trigger a full-scale Urgent Computing job. Sensor networks may also play an important role in triggering a large computation. FPGAs may be a place where specialized computation can run, but we have not explored that area yet.