SPRUCE
Special PRiority and Urgent Computing Environment
http://spruce.uchicago.edu

Pete Beckman
Argonne National Laboratory
University of Chicago
I Need it Now!

- Applications with dynamic data and **result deadlines** are being deployed
- Late results are useless
  - Wildfire path prediction
  - Storm/Flood prediction
  - Influenza modeling
- Some jobs need priority access “Right-of-Way Token”
How can we get cycles?

• Build supercomputers for the app
  † Pros: Resource is ALWAYS available
  † Cons: Incredibly costly (99% idle)
  † Example: Coast Guard rescue boats

• Share public infrastructure
  † Pros: low cost
  † Cons: Requires complex system for authorization, resource mgmt, and control
  † Examples: school buses for evacuation, cruise ships for temporary housing
Introducing SPRUCE

• The Vision:
  - Build cohesive infrastructure that can provide urgent computing cycles

• Technical Challenges:
  - Provide high degree of reliability
  - Session management, elevated priority
  - Resource selection, data movement

• Social Challenges:
  - How will emergency use impact regular use?
  - Decision-making, workflow, and interpretation
Existing “Digital Right-of-Way” Emergency Phone System

GETS is a "ubiquitous" service in the Public Switched Telephone Network...if you can get a DIAL TONE, you can make a GETS call

Calling cards are in widespread use and easily understood by the NS/EP User, simplifying GETS usage

GETS priority is invoked “call-by-call”

Dial 1-710-NCS-GETS (627-4387)
At the tone, enter your PIN.
When prompted, dial your destination number (area code + number).
If you cannot complete a call, use a different long distance carrier:

- AT&T: 1010 + 288 -or- 1-888-288-4387
- MCI: 1010 + 222 +1-710-627-4387 -or- 1-800-900-4387
- Sprint: 1010 + 333 -or- 1-800-257-8373

From a Wireless Priority Service enabled device:
Dial *272 before any call, including a GETS call.

Assistance: For help or to report trouble, dial 1-800-818-GETS (4387) or 1-703-818-GETS (4387).
Test Calls: Make periodic GETS calls to 703-818-3924.

US GOVERNMENT PROPERTY. If found, return to:
NCS (N3), PO Box 4502, Arlington, VA 22204-4502
WARNING: For Official Use Only by Authorized Personnel.
SPRUCE Architecture Overview (1/3)

Right-of-Way Tokens

Event

1. Automated Trigger
2. First Responder

Human Trigger

Right-of-Way Token

SPRUCE Gateway / Web Services
SPRUCE Architecture Overview (2/3)

Submitting Urgent Jobs

1. User Team
2. Urgent Computing Job Submission
3. Choose a Resource
4. Authentication
5. Priority Job Queue
6. Supercomputer Resource

- Conventional Job Submission Parameters
- Urgent Computing Parameters
- Local Site Policies
SPRUCE Architecture Overview (3/3)

Analyzing Urgent Jobs

1. Supercomputer Resource
2. Results
3. Domain Specialist Interpreter
4. Decision Maker
Summary of Components

- Token & session management
  - admin, user, job manager

- Priority queue and local policies

- Authorization & management for job submission and queuing
Internal Architecture

Web Portal or Workflow Tools
- AJAX
- PHP / Perl

Client Interfaces
- SOAP Request

SPRUCE User Services || Validation Services
- Axis 2 Web Service Stack
- Tomcat Java Servlet Container
- Apache Web Server

Computing Resource:
- Job Manager & Scripts
  - Java
  - Axis2
  - PHP / Perl

Client-Side Job Tools
- SOAP Request

MySQL

JDBC

Mirror
Site–Local Response Policies: How will Urgent Computing be treated?

- “Next-to-run” status for priority queue; wait for running jobs to complete
- Force checkpoint of existing jobs; run urgent job
- Suspend current job in memory (kill -STOP); run urgent job
- Kill all jobs immediately; run urgent job

- Provide differentiated CPU accounting
  - “jobs that can be killed because they maintain their own checkpoints will be charged 20% less”
- Other incentives
Emergency Preparedness Testing: “Warm Standby”

• In urgent computing situation, there is no time to port applications
  - Applications must be in “warm standby”
  - Verification and validation runs test readiness periodically (Inca, MDS)
  - Reliability calculations
  - Only verified apps participate in urgent computing

• Grid-wide Information Catalog
  - Application was last tested & validated on <date>
  - Also provides key success/failure history logs
Selecting a Resource

Trigger

- Urgent
- Severe
- Weather Job

Deadline: 90 Min

Advisor

Historical Data

- Network bandwidth
- Queue wait times
- Warm standby validation
- Local site policies

Live Data

- Network status
- Job/Queue data

Application-Specific Data

- Performance Model
- Verified Resources
- Data Repositories

“Best” HPC Resource

Advisor
Deployment Status

- Deployed and Available on TeraGrid –
  - UC/ANL
  - NCSA
  - Indiana
  - Purdue
  - SDSC
  - TACC
- Ongoing work
  - LSU
  - Virginia Tech
- Integration into LEAD Portal –
  - First user–customer
What About “Capacity” or High Throughput Computing?

- SPRUCE works well with “capability” computing:
  - Interface to small set of large resources
- However, imagine a much larger set of smaller resources?
  - Condor integration and management?
  - Sensors and Databases
  - How would designs change?
  - How do we integrate?
  - Real on-demand servers (Amazon, etc)?
The Future

- Automatic restart tokens
- Flexible tokens – aggregation, extension
- “Start_By” deadlines
- Encode (and probe for) local site policies
- Automated ‘advisor’
- Warm standby integration
- Data movement
- Failover / Mirroring (SDSC, others)
- Redundancy to avoid portal downtime
- Integration of VGrADS performance models
- Easily transportable computation (VMs?)
- Integration with analysis and decision-making
Imagine…

- A world-wide system for supporting urgent computing on supercomputers
- *Slow, patient* growth of large-scale urgent apps
- Expanding our notion of priority queuing, checkpoint/restart, CPU pricing, etc
- A standardized set of web services for “request VM”, including all the complicated small bits
  - DHCP, VLANS, DNS, local storage, remote storage
- For Capability: 10 to 20 supercomputers available on demand
- For Capacity: Condor Flocks & Dynamic VMs provide availability of 250K “node instances”