JobMon
An Interactive Job Monitor

Conrad Steenberg
California Institute of Technology

Elliot Lipeles, Shih-Chieh Shu, Frank Würthwein
University of California, San Diego
Background

- Thousands of jobs submitted to various clusters in a distributed computing environment
- Presents management and debugging problems for both users submitting jobs and site administrators
- Jobs disappear into a *black box* once they reach worker nodes
- Some tools exist to send monitoring information from individual jobs
- Strong need for interactivity with jobs on worker nodes
Environment

- Multiple clusters
- One or more head node per cluster
- Up to several thousand worker nodes, probably on private network
• For interactivity we need to contact jobs running on worker nodes
• Worker nodes not allowed to run servers or not publicly addressable
• All network connections at FNAL must be strongly authenticated via Kerberos tickets or or X509
Job Initialization

- Job reaches the worker node via batch scheduler
- Includes *JobMonDaemon* wrapper script
- Also unique X509 cert/private key unless credentials can be obtained e.g. Via Kerberos
- *JobMonDaemon* starts up, contacts specified JobMon server, and registers itself
- Retrieves JobMon server configuration
- Either
  - Connects to *JobMonTCP* server
  - Starts UDP listener for callbacks
User Request Handling

- Client calls `query` Clarens method, authenticated
- Message sent to job via `JobMonTCPServer` or UDP
- Job wakes up, calls `getJobToDo` Clarens method
- Job performs requested action, calls `outputJobResult` Clarens method
- Result gets returned to the client
- Job periodically re-registers itself using the `registerJob` Clarens method
User Commands

- `ps`, `head`, `tail`, `cat`, `top`, `kill`, `ls (dir)`
- `jobs`: Job Status and Information
- `log`: Check progress of a job
- `node`: Show node that job is running on
Performance I

- Web services technology with callbacks chosen to avoid persistent TCP connections, for security
- UDP listener configuration more scalable than *JobMonTCPServer*
- Primary performance bottleneck is the periodic re-registration of jobs
- Re-registration involves setting up new HTTPS connection, authenticating and calling *registerJob* method – slow process
Tests show that for a 3000 node cluster the server can handle re-registrations at 4 Hz with 1s latency.

This translates to a re-registration interval of ~12 minutes for each job.

Deployment at CDF Central Analysis facility since March shows ~100 concurrently registered jobs.
Authentication and Authorization

- Each job is assigned an ID, from which an X509 cert/private key pair is generated.
- The subject of the X509 cert belonging to the job owner is of the form
  `/DC=gov/DC=fnal/O=Fermilab/OU=People/CN=<name>`
- The server will allow this user to connect to all jobs with certificate subjects of the form
  `.../O=Fermilab/OU=People/CN=<name>/CN=<jobid>`
- X509 cert/private key sent with job. No more a problem than sending non-encrypted code!
Summary

- JobMon provides a secure, scalable way for interactive job monitoring
  - Re-registration latency main performance bottleneck
- *JobMonDaemon* requires only Python interpreter on worker node
- Successfully deployed and used by CDF CAF
- Part of VDT since version 1.3.8, deployed on numerous US Grid sites
  - [http://jobmon.sf.net](http://jobmon.sf.net)