

# Automatic Software Updates on Heterogeneous Clusters with STACI

Michael Shuey  
Linux Developer and Administrator

LCI: The HPC Revolution  
May 19, 2004



## Outline

- Introduction
- Common maintenance problems
- STACI
- Usage experiences
- Future Directions



- We operate a variety of HPC resources
  - All resources are 24x7, used by local researchers nearly at capacity
  - Clusters becoming pervasive
  - New nodes, clusters appear often
- We're a PBS shop
  - Jobs last up to 30 days
  - Several servers, about 1 per resource
- We support a LOT of applications
  - Frequently adding more on request



# Problem Child 1



- “recycled” cluster
  - Older machines, usually desktops
  - Cheap to build, easy to assemble
  - Random nodes die!
  - VERY large, heavily used – scheduling maintenance times problematic



## Problem Child 2



- 100-node dual Athlon cluster
  - Maintained for prominent researcher
  - Nodes die frequently due to hardware flaws
  - Keeping software stack consistent across nodes is challenging...



## Large Cluster Headaches

- Frequent software changes require frequent maintenance windows, hampering long jobs
- Frequent hardware updates cause administrator oversights
- Dead nodes make software updates tedious
  - Difficult to ensure consistent software stack
  - Automated tools often choke on dying nodes
- Multiple clusters, multiple architectures
  - Multiple software update mechanisms
  - Multiple admins do updates concurrently...



# Possible solutions

- Reload nodes on job completion
  - Easy to automate
  - Hard to do with multiple jobs on SMP node
  - May be possible to reload image during image update...
- Schedule maintenance time
  - Nearly impossible to do all at once
  - Too much bookkeeping to do piecemeal
  - Causes large, user-visible disruption
- Write something new...



# Goals

- Must be fully automated
  - running rsync is a waste of time
- Must provide minimal user disruption
  - Shouldn't change software during active jobs!
  - Shouldn't reserve large chunks of the cluster(s)
- Must handle multiple clusters
- Must handle multiple architectures, operating systems
- Should keep administrators from stepping on each others' toes
- Needs to get list of nodes from job scheduler
  - Only scheduler will know current node (and job) status
- Should allow different nodes to have different software



- Node software lives in chroot-able image
  - Images have version numbers
  - Must be locked (and mounted rw) for edits
  - Software is delivered via rsync over ssh
- Two processes continually update nodes
  - nodescanner – marks nodes to update
  - rolling\_upgrade – does the heavy lifting
- Rely on the PBS server(s) for info
  - PBS always knows node availability, state, etc.
  - Job scheduler can be told to keep work off a node until it can be upgraded
- All other info (per-node config data, image applied on node, etc.) is stored in a ;-delimited “nodes” file



- lock\_img
  - prepare image for edits, create lockfile
- unlock\_img
  - removes lockfile, increments versions number
- cimage
  - sends out image irrespective of lock, compatible with c3 tools
- cmark
  - forcibly mark nodes for upgrading, compatible with c3 tools



- /etc/staci/clusters.conf
  - Lists available clusters, scheduler types
- /etc/staci/staci.conf
  - Various configuration parameters
  - PBS install root, delay between scans, etc.
- /var/lib/mgmt/nodes
  - Lists all nodes, image names, other details
  - ;-delimited
- /var/lib/mgmt/postinstall
  - Postinstall script (invoked after imaging), lives on nodes
  - Can parse nodes file for node details
- Image directories (and lockfiles) live in /var/lib/mgmt/images



- lock\_img <directory>
- chroot <directory>
- <fiddle with things>
- optional – use cimage for test deploys
- unlock\_img <directory>

Sit back, STACI will handle deployment.



## STACI's behavior (1)

- nodescanner:
  - contacts each node from each server
  - compares /etc/image\_version on node with version in image
  - marks node for upgrade
    - » usually involves PBS comments, adding reservation to scheduler
  - after unlock\_img nodescanner will immediately notice new version numbers and act accordingly



## STACI's behavior (2)

- rolling\_upgrade:
  - activates periodically (default: 30 min), scans for nodes to upgrade
  - selects a group of like-minded nodes
    - » same image, limited number
  - pushes out upgrade(s) via parallel ssh
    - » rsyncs image
    - » runs postinstall script on nodes
    - » reboots nodes
  - removes comments, reservations – node can be used again



## Usage experiences (1)

- Reduces admin time spent pushing out changes
- Removes the need to schedule maintenance time for most changes
  - Security updates can still cause issues!
- Updates appear instantaneous (to users)
  - Deploy new software to head node, image
  - Mark appropriate nodes as needing upgrade
  - New jobs will only run on upgraded nodes



## Usage experiences (2)

- Drives up load on image server like crazy
  - Not bad for 10 or 20 machines, but for 100...
  - Lesson learned: use a big server, or set an upper bound on simultaneous updates
- ssh overhead is definitely not a factor...
  - Nodes are idle HPC resources, ssh is trivial
- ...network bandwidth can be an issue
  - Image server needs decent network feed



## Future directions (1)

- No sense of indirect clusters from c3
  - 3<sup>rd</sup> party machine could act as image server
  - Reduces load on main server
  - Potentially less network congestion
  - Can be used to update firewalled clusters
- Deploying to more platforms
  - Several Linux distributions now
  - AIX, Solaris soon



## Future directions (2)

- Integration with non-PBS schedulers
  - PBSPro (with Maui) supported now
  - Integration with SGE, LSF, etc. feasible
- Better fault detection and error reporting
  - The nodescanner visits all nodes regularly
  - Very easy to page sysadmin on timeouts...
  - Would be better to invoke problem recovery scripts, avoid manual intervention if possible

