The Old Infrastructure

• 2009
  • 1 bare metal app db/server
  • 15 compute nodes (8 cores, 4 GB/core)
  • 48 TB storage server
• 2010
  • 3 more nodes (4 cores, 8 GB/core)
  • Additional JBOD for the storage server
  • 128 cores for NGS
• 2011
  • 3 storage servers
  • 5 arrays (48 TB each)
• 2013
  • 120 TB storage server
The New Infrastructure

- 2 (8 core, 2 GB/core) VMs for application
- 2 (8 core, 2 GB/core) VMs for database
- 16 (16 core, 128 GB/core) compute
  - 10 reserved for NGS
  - 6 for general purpose jobs
- Infinite (ha ha) space on Corral
Dataset Storage

- “Nearline” large volume filesystem
- NFS via multiple gateways, but GPFS possible
Networking

- 10 Gb/s connection to XSEDE via PSC
- Lit a new $\lambda$
- Galaxy: The first entity on the XSEDE network not an XSEDE member institution
Data Transfer

- 10 Gb/s link does not equal 10 Gb/s data transfers
- Tune endpoint TCP/IP stacks, jumbos on every point on the route between endpoints
- Link Speed: 10000; Peak Tx % bandwith: 90.52
Data Transfer Mechanisms

Globus Online
- Hands-off big data transfer
- Easy, fault-tolerant
- Performance auto-tuning
- Limited to 3 simultaneous transfers

Globus GridFTP
- The transfer technology underlying Globus Online
- Not as resilient as rsync or Globus Online

rsync over HPN-SSH
- Massive improvement over standard SSH
- rsync capable of wire speed(!)
Galaxy Improvements

- uWSGI replaces Paste#https\server
- One uwsgi server per app VM
- Preforks Galaxy web processes
- Restart Galaxy: clients block
- Slurm enhancements (but works “out of the box” with DRMAA)
- iRODS Object Store plugin
- Hierarchical Object Store plugin
- Unrelated but so awesome I have to mention it: new upload on Galaxy Test
Online Backups

- Replacement for tape archiver
- Powered by SLASH2

www.psc.edu/DataSupercell/
quipu.psc.teragrid.org/slash2/
Tricks and Technologies

• Run web processes from local disk to ensure network filesystem performance does not impact UI
• Run handlers from shared filesystem
• Process management via supervisord
• Use Nagios to check individual handlers
• Use sentry to aggregate tracebacks
• Use config management
Config Management

- Formerly: CFEngine
- Now: Ansible
  - Dependencies: sshd, Python
  - No infrastructure required!
  - Descriptions in YAML!
  - Modules in Python!
  - Happily coexists with TACC’s Puppet
TODO

- PostgreSQL streaming replication
- Optimize uWSGI usage
  - Offload file serving
- Remote site job running/data transfer
  - SLASH2? iRODS? Ceph?
- …or Galaxy Object Store API
SLASH2
quivu.psc.teragrid.org/slash2/

- SLASH2 is a network filesystem
- Fully WAN capable
- Network via Lustre LNET
- IO servers can live on any POSIX-like filesystem
- **Transparent replication**
Data available everywhere as soon as the file is written

Performance improves over time

Successfully ran jobs on Blacklight from Penn State
CLI Job Runner

- Interacts with DRM systems (PBS, SGE, Slurm) with command line tools
- Can remotely access these systems via SSH, GSI-SSH to a remote host