The tale of the UFRC Galaxy

Oleksandr Moskalenko, Ph.D.
UFIT Research Computing

SURAGrid Call on September 16th, 2013

Galaxy goals

- Provide an easy to use interface for entry level analyses at first, expand the mission as the Galaxy's capabilities grow
- Provide a collaborative environment
- Provide a teaching environment
- Supplant the iNquiry instance to be retired from the Genetics Institute

UF Galaxy History and stats

- Deployment history
- Development and deployment setup
- Production configuration

History

- 1st prototype – KVM Virtual Machines – crash and burn
- 2nd prototype – real head node, Lustre FS, dedicated Torque/MOAB reservation
- 1st production setup – (10 months)
  - Head node (4cores, 16GB RAM)
  - Cluster (SGE – 4 12-core/96GB RAM nodes)
  - Lustre FS (60TB)
  - 1Gbit network, Infiniband from nodes to storage
  - Software provided by the modules system

Initial production setup (2012)

- Number of active users (last 6mo) - ~150
- Size of the database/files – 5TB
- Jobs run: over 11000
- Hardware:
  - Head node – 4 cores, 16GB RAM (32GB upcoming)
  - Cluster
    - 7000 cores, 2GB/core average RAM
    - 120, 500GB RAM bigmem nodes (1TB upcoming)
  - Storage
    - NexentaStor ZFS/NFS – slow (/galaxy)
    - HA NexentaStor ZFS/NFS – fast (/bio, ref data)

Current Production (2013) ~3x/yr

- Number of active users (<6mo) - still ~150
- Size of the database/files – 18TB + >6TB refs.
- Jobs run: over 11000 - > 31300, ~1,700/Mo
- Over 500 tools - biology, chemistry (gaussian)
- Hardware:
  - Head node – 8 cores, 32GB RAM
  - Cluster (old cluster, preparing a move to HiPerGator)
    - 7000 cores, 2GB/core average RAM
    - 500GB RAM, 1TB bigmem nodes
  - Storage
    - NexentaStor ZFS/NFS (/galaxy)
    - HA NexentaStor ZFS/NFS for reference data (/bio)

Deployment and development

- 3 head nodes: production, staging, development
  - Development:
    - Up-to-date galaxy-central code
    - Wrapper development
    - Hacking on the galaxy core
  - Staging:
    - Pull galaxy-dist releases
    - Pull and update a copy of the production database
    - Clean and stabilize
    - Set up tools and local modifications and wrappers
    - Replace the production code

Production Configuration

- Authentication – Remote User
  - Apache mod_auth_tkt, LDAP back-end.
- Software – mix of modules and tool shed
- Database node – 4 core, 32GB RAM, PostgreSQL
- Had a local (campus) Tool Shed, removed for security considerations

Initial Hurdles

- Torque - python_pbs memory leaks
- Head node overloaded
- Poor visualization capabilities
- No soft restart capability
- Upgrades – tool cleanup, maintaining hacks
- Providing consistent environment (drmaa.py / modules)
- No job accounting
- “I can haz moar wrapperzzz”

Early Hacks

- Watcher cron job for python_pbs
- Run **all** jobs on the cluster, even the upload
- Soft restart init scripts
- Use two RCS – mercurial and git to manage upgrades
- Hacked drmaa.py to load modules
- Unified the tools used on the cluster and in Galaxy via the environment modules system
- Unified reference databases
- Set up big data import

User Issue Report Categories

- Inappropriate job resource requests hard-coded into the tool runner URIs
- Tool failures because of bad data or tool bugs
- Poor understanding of what tools are needed
- Help with analyses
- Reference dataset issues (dbkey)

"Real User Jobs" – 2012

- Jobs at UFRC must run under the real user's id
  - Very powerful feature, but opens many cans of worms
- Move from SGE to Torque/MOAB
  - No green banner of death
  - Strict resource request handling
  - Finding a drmaa library that works - http://apps.man.poznan.pl/trac/pbs-drmaa
- Introduction of the Tool Shed
  - Cultivate new wrapper contributors
  - Very unstable at the moment
  - Divergence from the env. modules in version handling
- Dynamic job runner
  - Almost unlimited possibilities in with the real userid jobs

Recent Galaxy improvements

- Some progress on the reference dataset handling automation with data tables
- Going from one to three visualization and visual analysis tools
- Job handling configuration completely redone recently (xml-based now, decoupled from the main configuration)
- Intelligent shed tool upgrade handling and warnings to the users
- Dynamic banner for user communication
- Warning/error messages to the users from the job handler

The Wish List

- Universal job resource request interface mechanism
- More flexible output file handling
- No more hard-coding tool arguments
- Reference dataset handling automation
- More documentation and more capabilities in the tool definition file logic
- More published workflows and docs

Finding Help

- Galaxy wiki is a great resource for experienced minds
- http://gmod.827538.n3.nabble.com/Galaxy-Development-f815885.html (galaxy-dev list archive on Gmod) and
- http://gmod.827538.n3.nabble.com/Galaxy-Users-f815892i36.html (galaxy-user list archive on Gmod) are invaluable
- http://seqanswers.com/ - treasure trove of information on tools when a bug report calls.
- Read the code
- IRC (#galaxyproject) – mostly quiet, but getting a core developer to commit a new feature to galaxy-central after a 3 minute discussion is priceless when it happens
- Galaxy Community Conference
- Blogs

Questions?