The FermiGrid Software Acceptance Process aka "So you want me to run your software in a production environment?"

Keith Chadwick

Fermilab

chadwick@fnal.gov

Work supported by the U.S. Department of Energy under contract No. DE-AC02-07CH11359

Background

FermiGrid has deployed the majority of the services listed in the FermiGrid service catalog in high availability (HA) configurations that are collectively known as "FermiGrid-HA".

The goal for FermiGrid-HA is 99.999% service availability.

- Less than 5m 15s of service downtime in a year.

The measured overall core service availability over the past year is is 99.963%.

- Average of 3h 14m 24s of downtime per service;
- Chief downtime cause was user originated denial of service attacks.
  - The services are being hardened against these attacks.
- Second downtime cause was failure of underlying infrastructure.
  - Additional monitoring of the infrastructure has been deployed.
  - Additional infrastructure has been deployed.

The Service Level Agreement (SLA) for the core FermiGrid-HA services is 99.9%.

- Overall, we are doing (just) better than the SLA.
- Key to meeting the SLA and availability goals is to carefully plan and perform the software deployments and perform continual service improvement.

Core Service Availability Details

Service	Availability	Downtime
VOMS	99.874%	11h 2m 15s
GUMS	100.000%	0h
SAZ	99.887%	9h 53m 56s
Squid	99.858%	12h 26m 21s

The VOMS outages were caused by a failure of the Fermilab Kerberos Certificate Authority (KCA).

- We have deployed additional monitoring of the Fermilab KCA.
- The KCA administrators are in the process of deploying a second KCA as part of the HSM migration.

The SAZ outages have been caused by authorization "tsunamis".

- We have deployed additional SAZ servers "segmented" by client group (CDF, D0, CMS, GP Grid) to isolate the effects of such tsunamis.
- We are also in the middle of a software refactoring of SAZ to eliminate the underlying vulnerabilities.

The Squid outages were caused by a user attempting to download a 1.2 Gbyte file through the squid servers on ~1,700 systems simultaneously.

- We have modified the squid servers configuration to **reject** transfer attempts greater than ~256 Mbytes.

Gatekeeper Availability Details

Gatekeeper	Availability	Downtime
CDF - fcdfosg[1-4]	97.611%	1d 10h 4m 35s
CMS - cmsosgce3	95.680%	18d 10h 25m 55s
D0 - d0osg[1,2]	99.687%	1d 3h 25m 8s
GP Grid fnpc[3,4,5]	99.480%	1d 21h 33m 7s

The gatekeeper availability data includes scheduled outages.

The large downtimes for CDF and CMS were dominated by hardware failures that required extended periods of time for the vendors to provide working replacements.

- CDF (fcdfosg4) – 30d 21h 45m 33s.
- Access to the backend CDF and CMS clusters remained available through other gatekeepers.

Other Services Availability Details

Service	Availability	Downtime
Batch Services	99.635%	1d 6h 23m 50s
Resource Selection Service	99.978%	1h 55m 38s
Gratia Accounting Service	99.792%	18h 13m 15s

The measured batch services availability data includes scheduled downtimes.

The Resource Selection Service has had an availability of 100.000% since it was deployed in the high availability "Ress-HA" configuration three months ago.

Work is actively underway to "HA" the Gratia Accounting Service.

Sources of Software

Scientific Linux 5.3 + Xen Hypervisor

- LVS (linux Virtual Server)
- MySQL 5 with circular replication.

Virtual Data Toolkit (http://vdt.cs.wisc.edu/)

- Globus gatekeepers
- worker node client
- Condor batch system

OSG Enhancements:

- Gratia Accounting Service
- Resource Selection Service

Locally Developed Software:

- jobmanager-cemon
- Site AuthoriZation (SAZ) Service

Software Acceptance Framework

From the beginning of the FermiGrid project, the project operated under an informal software acceptance (development/integration/production) framework.

In February 2008, an initial document was written to describe the software acceptance process.

This version was shared with several of the software providers and their comments and feedback was incorporated.

The "final" version was released in June 2008, and FermiGrid has been operating under this policy since that time.

http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=2684

Reasons for this Process

Effective operation and management of FermiGrid requires a strict software release management and software acceptance process in order to insure operation of the FermiGrid services at levels that meet (exceed) the Service Level Agreement(s).

Additional goals of this process include:

- Effective tests at production scale;
- Ease administration effort by operations group;
- Reduce the need to contact the developer(s).

Support and encourage continual service improvement.

A Change in "Attitude"

In order to support these goals, software project managers (providers) wind up having additional work as part of their projects:

- Incorporate time to transition to operations in WBS.
- Incorporate documentation (install, operate, capture debug information, upgrade).

FermiGrid also distinguishes between three categories of software releases:

- Emergency bug fixes to restore existing functionality;
- Normal routine bug fixes of existing functionality;
- Extensions to existing functionality.

Developers need to think about running their code on a system that is providing other services and cannot "just" be rebooted.

- The FermiGrid virtualization deployment has somewhat undermined this argument.

Software Packaging

Multiple forms of software packaging/ distribution methods shall be accepted:

- pacman (OSG/VDT "standard");
- RPM (Redhat/SL "standard");
- ups/upd (old Fermilab product distribution method);
- "bare" tarballs.

Software Acceptance Environment

Rule 1:

- In general, there shall be no root access to the production software or service instance by developers (other than explicitly enabled by FermiGrid personnel during incident diagnosis and/or resolution).

Rule 2:

- Refer to Rule 1. ☺

Software Documentation

FermiGrid personnel shall receive a documentation package from the service developers that includes:

- How to install and/or upgrade the service;
- How to revert back to a prior version of the service if necessary;
- How to start the service;
- How to shutdown the service;
- How to test critical service functionality (ideally a full regression test);
- How to monitor critical service functionality;
- A description of the service log locations and logging options;
- A list of information to capture in the event of problems (debug script?) etc.;
- Other information that is necessary to insure service operation.

It is expected that all of the above documents are "living documents" – they will be developed, revised and enhanced as part of the service development and deployment lifecycle.

Required Software Support

The service development team must provide a well-defined and documented process (including version release procedure and regression test suite).

The developers must be able to provide service updates and/or patches for all currently supported versions of their software without requiring FermiGrid to update to the latest version of the service.

The developers must supply a support infrastructure for "expert" level questions.

Development

Development of the service takes place on development system(s) that are either operated by the developers or otherwise operated where the developers have root access.

Development version service passes the "regression test suite" that is defined and provided by the developers:

- This test suite may be a manual process or (ideally) an automated process.
- The regression test suite must be capable of testing at least the critical portions of the service functionality as well as major new features introduced in new releases, and must also provide "load" or "stress" test functions when applicable and/or appropriate.
- If the regression test suite is not automated then it must be explicitly documented to assure that all tests have been performed each and every time that a release is cut.
- If it is not documented, then it never happened.
- Developers are responsible for defining test acceptance criteria.

The developers perform a formal software version cut and provide FermiGrid with a package in the appropriate format (see Introduction).

Integration

FermiGrid personnel install this version on the test/integration system(s) using the instructions provided by the developers. Note: Developers may assist in the installation, but any such "assistance" must be immediately entered into the documentation package by the developers.

Test/integration installation passes regression test suite.

- FermiGrid personnel rerun the regression tests with the assistance of the developers where necessary.
- Any "unusual" assistance by the developers must be immediately entered into the service documentation package.

Once the software passes the integration phase, formal acceptance of version release by FermiGrid management is now considered complete and the proposed changes may be submitted to FermiGrid Change Advisory Board for scheduling the transition of the service.

Production

Service version installation on production system(s) by FermiGrid personnel.

Production installation passes regression test suite.

- In the event of a failure, roll back to previous production version using the developer provided "backout" procedure.
- If the service is deployed in an "HA" configuration – we upgrade one half of the "HA" configuration, verify functionality, and then upgrade the other half of the "HA" configuration.

Formal release to operation.

Software Environment Evolution

Development

Developer may have "root" access.

Software installed by developer as needed.

System either administrated by developer or FermiGrid administrators.

Integration

Developer does not have "root" access.

System administered by FermiGrid administrators

Software installed by FermiGrid administrators based on instructions from developers.

Production

Developer does not normally have any access.

System administered by FermiGrid administrators.

Software installed by FermiGrid administrators when approved by CAB.

ITIL / ISO 20000

In October 2008, the Fermilab Computing Division announced a five phase roadmap for ITIL implementation leading to ISO 20000 certification for the CD.

- Incident Management
- Problem Management
- Change Management
  - Standard change request
  - Minor change request
  - Major change request
  - Emergency change request
- Release Management
- Configuration Management
- Financial Management

Conclusions

The FermiGrid Software Acceptance Process has brought value to both FermiGrid and the Fermilab Computing Division.

The FermiGrid Software Acceptance Process meshes well with the CD ITIL / ISO 20000 roadmap:

- "Wheels within wheels" / "Gears within gears";

Or:

- "It's turtles all the way down".
  - http://www.the-funneled-web.com/Hawking.htm

Fin

Any questions?