Table of Contents

WLCG Operations and Tools TEG - WG4 and WG5

WG4 - Operational Requirements on Middleware

Deliverable

The analysis is summarized in the following document:

- WG4_TEG_OPS.docx

Introduction

Sites

1. The sites have to **operate** middleware services.
    - The services should be as **simple** as possible, with few features that essentially are unused.
    - The services should be as **robust** as possible, i.e. admin interventions should rarely be needed.
2. They need to understand whether the services are **configured** correctly.
    - The **documentation** should explain the expected behavior resulting from the configuration.
    - Any parameters not explicitly configured should have reasonable **default** values where possible.
3. They need to understand whether their services are functioning correctly, e.g. via **monitoring**.
4. When a service is not functioning correctly, the **monitoring** should notify site people of the malfunction in a way that tells them efficiently:
    1. what exactly is malfunctioning
    2. why it is malfunctioning
    3. what steps need to be taken to resolve the situation.
5. If a user **reports a problem** to the site, one of the **requirements on the middleware** is that it should be possible to quickly understand what the real problem is:
    - Good error messages and logging.

Users

1. Users (experiments) have to **use** the middleware clients and services.
2. It should be easy to understand how to **configure** a command or a job to make it do what is wanted, this should be well **documented**.
3. The system should be as **simple** as can be reasonably expected.
    - Standard usage should be straightforward.
4. Clients should be **robust**, e.g. fail over to another instance of a service when possible.
5. Commands should have reasonable **defaults**.
6. If something goes wrong with a user's job or requested operation, a **requirement on the middleware** is that it should be easy to understand what caused the problem:
    - Good error messages.

Model

1. The **model** of clients and services should be as **robust** as possible.
    - Complexity should be reduced where possible.
    - Clear distinction between responsibilities of experiments vs. sites.
    - One site should not be able to cause another to be degraded (cf. FTS channel model).
    - Services should be able to throttle operations in times of congestion and clients need to deal with that through intelligent retry and fail-over logic.
2. **Correlations** between services should be easy to track.
    - Logging and monitoring should facilitate that.

3. The **state** of a service instance should be advertised consistently and **taken into account** by clients.
    - The service itself, the information system, the GOC DB, monitoring pages, ... should agree on the state.
    - Clients should make use of downtime information to help prevent unnecessary tickets.

What Works Well

- much of the middleware works well in most deployment scenarios most of the time, but that fact is overshadowed by the many issues that experiments and sites need to spend a lot of time on, as detailed below

Top Problems

Error logging

- lack of clear error logging (KIT, UNL, Glasgow, Manchester)
- too much clutter in logs (France)
- better log management needed for Incident Detection and Traceability (France)

Robustness

- middleware robustness issues (CSCS, Glasgow, Manchester, Frascati, Milano, Napoli, Roma-1)
- BDII contents are unreliable (CMS)
- lack of cron jobs for disk cleanup etc. (KIT)
- lack of automatic recovery actions for grid services (CERN)
- High Availability solutions do not exist or require local tweaks (CSCS, CERN)
- scalability of various services (e.g. batch systems and their info providers) (PIC, CERN)
- widespread use of unoptimized MySQL DBs (Manchester)

Investigation of failures

- file transfer problems e.g. due to channel model (KIT, UNL)
- job failures and tickets despite CE maintenance announcements (UNL)
- no easy way to correlate batch/CREAM/PanDA job IDs (UVic)

Deployment complications

- some sites have issues with glexec deployment (Manchester)
- the VO sometimes needs to do too much to get some middleware to work at the sites (e.g. glexec) (CMS)
- pool account mapping implementation (CERN)
- banning users across different services and multiple instances (CERN)

Strategic Directions

- improve middleware compatibility with larger set of deployment scenarios (e.g. NAS/SAN) (France)
- compliance with existing standards (France)
- middleware tools ought to be scripts where possible to facilitate debugging (UVic)
- more details in middleware documentation (CSCS, Frascati, Napoli, Roma-1)
- ensure better consistency between GOCDB and BDII to avoid confusion (CSCS)
- better support of middleware integration in experiments and at sites (CMS)

Saving Manpower

- manpower can be saved by addressing the Top Problems and following the Strategic Directions detailed above

Operational middleware requirements of EGI

The open operational middleware requirements of EGI are summarized at:

- https://wiki.egi.eu/wiki/Requirements_2010_2011

Requirements gathering process

EGI periodically collects operational requirements that concern the deployed software, namely ARC, dCache, gLite, UNICORE and Globus. Requirements can be submitted through the EGI Request Tracker system:

- https://rt.egi.eu/rt/index.html ("Requirements" queue)

Read access to Requirements tickets is open.

Requirements can be submitted by anybody: site administrators, Operations Centers etc. The submitter needs a valid SSO account. Anybody can get an SSO account on the EGI Single Sign-On system:

- https://www.egi.eu/sso/

Requirements can be characterized by adding extra attributes to the RT ticket, which qualify the identity of who is submitting (member of a user community, project member etc.) and the type of problem (middleware distribution affected, type of capability, etc.).

Detailed instructions on how to open a RT requirement ticket are provided at:

- https://wiki.egi.eu/wiki/Mw-requirements.html

The same process can be followed to submit requirements that concern tools maintained by EGI, namely: accounting, accounting portal, GGUS, GOCDB, Operations portal and dashboard, SAM. The list of open tool requirements is available at:

- https://rt.egi.eu/guest/Dashboards/1039/Operational%20Tools

Middleware requirements are collected, discussed and prioritized at the Operations Management Board (OMB), which meets monthly:

- https://www.egi.eu/indico/categoryDisplay.py?categId=19

Operational tool requirements are collected, discussed and prioritized at the Operational Tool Advisory Group (OTAG):

- https://www.egi.eu/indico/categoryDisplay.py?categId=4

J. Shiers and M. Girone are both in the OMB and OTAG.

Requirements that are approved are then either submitted to the Technology Provider such as EMI through GGUS (in case of enhancements) or discussed at the Technology Coordination Board - where EMI is

represented - in case of major requirements with architectural implications or involving multiple products.

Sustainability, Manpower

- Joint EGI and WLCG task forces in requirements gathering
- Exploit more the existing EGI resources for these matters

WG5 - Middleware Configuration, Deployment, Distribution

Deliverable

The analysis is contained in the following document:

- WG5_TEG_OPS.docx

Source material

- WW: works well
- TP: top problem
- SD: strategic direction
- SM: saving manpower

Middleware Configuration

General

- **WW:** ARC has a simple configuration which could serve as a model

- **SD:** simplify configuration using rpm (13)
- **SD:** reconfiguration and rollback should be supported in any future solution (16)
- **SD:** simplify configuration by providing better documentation (17)

- **TP:** middleware should be configurable with a single file which the service can reread when necessary.

- **SM:** site config takes 60% of time. A config tool which doesn't only do middleware would be awesome (4.1)
- **SM:** A combined tool for middleware and system configuration would be great, but a general solution for all sites is probably not viable (16)
- **SM:** Middleware configuration is one of the activities which occupies most time (18)

YAIM & Quattor

- **SM:** A yaim reconfiguration is an expensive process
- **SM:** Lack of support from middleware developers makes reconfiguration very expensive in time and effort

- **SD:** Shifting effort from yaim to puppet would benefit big sites at the expense of small ones (which were specifically targeted by yaim) (17)
- **SD:** what effort is actually available for yaim support (by INFN?) Should we at least assess the cost of cleaning it up and introducing required features? (17)

- **WW:** yaim (1) concerned about puppet
- **WW:** yaim/quattor combo is useful in some cases (8)
- **WW:** Mware integration using Quattor (long-term support essential) (4.5)
- **WW:** Yaim's lack of dependencies argues in its favour (17)

- **TP:** yaim makes configuration obscure, why no use site-info.def directly? (4.1)
- **TP:** Yaim is flimsy and has a difficult interaction with fabric management. Yaim performs actions which should be done in the packaging (5)
- **TP:** Yaim is incomplete and difficult to integrate with quattor (5)
- **TP:** reverse engineering of yaim scripts to understand what has changed in Mware configuration
- **TP:** no clear directions: yaim or not? it is not desirable... (14)
- **TP:** improved yaim configuration (14)
- **TP:** for a big site yaim can't cover all needs (4.2)

Other configuration solutions

- **WW:** Puppet is one of the best invests we made recently (3)

- **TP:** relative merits of yaim, puppet, cfengine (mailing list discussion)

- **SD:** vote for puppet (4.2)
- **SD:** Supplement of the middleware releases with appropriate puppet cfengine templates (yaim should be kept). (4.4)
- **SD:** Puppet may well become less attractive once it's filled with logic for all the ugly corner cases (17)
- **SD:** Consider supporting puppets beside to or in place of yaim (18)
- **SD:** Provide post-installation self-tests for middleware services (18)

Middleware deployment

Pre-release validation, staged rollout

- **WW:** staged rollout. It is working as community effort and hence sustainable by definition, and it proved to be necessary (14)

- **TP:** Need closer involvement of experiments in pre-release validation (10)
- **TP:** Experiments should enter in the middleware distribution/deployment only at the end for the validation of their systems (e.g. verify that a new version of dCache does not break CMSSW) (9)

Site deployment

- **WW:** virtualisation essential (3)
- **WW:** yum (1)

- **TP:** updates=problems (2)
- **TP:** Automation and streamlined mechanisms for server re-configuration and post-install tools is essential to save time (2)
- **TP:** WLCG middleware should be as simply installable as possible and coexist with other stuff on shared infrastructure (think glexec) (1)
- **TP:** The upgrade path from gLite -> EMI is unsupported (17)

- **SD:** Is there still a need for a WN distribution given experiment integration of clients and the possibilities offered by CVMfs? (17)

Packaging and Dependencies

- **WW:** packaging (4.1)
- **WW:** native packaging (rpm) is a good thing (13)

- **TP:** understanding dependencies (4.3)
- **TP:** packaging / deps are problematic but improving (5)
- **TP:** understanding the Middelware dependencies (4.5)
- **TP:** Services should create all the logs files under /var/logs, and the temporary files under /tmp. (14)

- **SD:** rpm packaging is the way to go (13)

Particular components

- **TP:** glexec still unpopular (1) -> security TEG
- **TP:** can SRM be dropped (1) -> DM TEG

Middleware distribution

Release Management

- **TP:** The Development-Certification-integration-Deployment chain is too slow and decoupled from the experiments (10)
- **TP:** WLCG baseline not enforced (10)
- **TP:** quick feedback and availability of bug fixes to middleware packages (11)

- **SD:** a single channel for release announcements and documentation (4.3)
- **SD:** A clear and single channel to announce a new release and documentation (4.5)
- **SD:** will need SL6 soon (ed. this is not just dependent on middleware, applications must be ready)

Applications Area

- **?:** fast software deployment, both for middleware and experiment application software. It happens e.g. for experiment software that a package malfunctions on the grid and a quick update is needed. (11)

- **?:** what's the cost of the AA process?

Repositories and Distributions

- **TP:** repository proliferations, repo configuration, dependencies (4.2)
- **TP:** repo proliferation, version, EMI/UMD (5)
- **TP:** The fact that we have too many release flavours (EMI/UMD/gLite) is confusion (9)

- **SD:** we prefer EPEL (5)

References

- **# sites**
- 1 UK
- 2 Jeff T2 CH
- 3 Jeff PIC
- 4 Site mailout wlcg-teg-ops-wg3wg5@cernNOSPAMPLEASE.ch
- 4.1 CSCS-LCG2
- 4.2 ASGC
- 4.3 IN2P3-LAPP
- 4.4 KIT
- 4.5 French sites
- 5 PES

- **# software providers**
- 6 EMI
- **# experiments**
- 7 Atlas
- 8 CMS
- 9 Jeff CMS
- 10 Jeff Atlas
- 11 Jeff LHCb
- 12 Jeff Alice
- **# infrastructures**
- 13 OSG
- 14 EGI
- 15 Jeff EGI
- **# later additions**
- 16 Ian Collier
- 17 TEG meeting 28/11/11
- 18 Italian Sites

This topic: LCG > WLCGTegOperationsWG4WG5
Topic revision: r22 - 2012-01-27 - OliverKeeble