Table of Contents

WMS Test Plan

NOTICE: missing tests:

- drain
- use of condor grid_monitor.sh
- osb truncation: MaxOSBSize
- testing more LBs with "LBserver" as a vector
- proxies with long chain (v-p-i -noregen)
- membership of more than one VO in the certificate

Unit tests

N/A

Deployment tests

Generic repository

- epel.repo
- EGI-trustanchors
- sl.repo
- sl-security.repo

Installation test

First of all, install the yum-protectbase rpm: `yum install yum-protectbase.noarch`

Then proceed with the installation of the CA certificates by issuing:

```
yum install ca-policy-egi-core
```

Install the WMS metapackage:

```
yum install emi-wms
```

After the definition of the *site-info.def* file configure the WMS:

```
/opt/glite/yaim/bin/yaim -c -s site-info.def -n WMS
```

At the end of the installation the various init script should be checked with all parameters (start | stop | restart | status | version) (TBD)

Update test

Starting from a *production* WMS add the patch repository then issue:

```
yum update
```

If necessary reconfigure the WMS:

```
/opt/glite/yaim/bin/yaim -c -s site-info.def -n WMS
```

At the end of the update the various init script should be checked with all parameters (start | stop | restart | status | version) (TBD)

Service configuration tests

- Check that the host certificate proxy is properly generated (glite-wms-create-host-proxy.cron).
- Check the glite-wms-check-daemons.cron and that stopped daemons are not restarted etc.
- Check the proxycache cleanup script

Functionality tests

Features/Scenarios to be tested

WMS can be deployed into two modes:

1. Using an LB server installed in the same machine (BOTH mode)
2. Using an external LB server (PROXY mode)

Both scenarios should be tested.

Test job cycle (from submission to output retrieve)

Submit a job to the WMS service and when finished retrieve the output; a the end the final status of the jobs should be *Cleared*.

Submission can be tested using different type of proxy:

- Proxy from different VO (TBD)
- Proxy with different ROLE (TBD)
- Delegated proxy retrieved from a MyproxyServer (TBD)

Different types of CE must be used:

- Lcg CE
- CREAM CE
- ARC CE

Test job submission with the following type of jobs:

Normal Job

- Test the complete cycle submitting to the two types of CE: lcg and Cream **Implemented.**

More different jdls can added in the future. In particular these attributes should be tested:

- DataRequirements (with differents DataCatalogType) (TBD)
- OutputData (TBD)
- InputSandboxBaseURI, OutputSandboxDestURI and OutputSandboxBaseDestURI (TBD)
- AllowZippedISB and ZippedISB (TBD)
- ExpiryTime (TBD)
- ShortDeadlineJob (TBD)

Perusal job

Job perusal is the ability to view output from a job while it is running. **Implemented.**

DAG job

Directed Acyclic Graphs (a set of jobs where the input/output/execution of one of more jobs may depend on one or more other jobs). **Implemented.**

- Also the nodes should be in state *Cleared*

More different jdls can added in the future.

Parametric Job

Multiple jobs with one parametrized description. **Implemented.**

Collection Job

Multiple jobs with a common description. There are two ways to submit collection:

- you can create a single jdl with all the jdls of node **Implemented.**
- you can submit all the jdls stored in a directory (bulk submission) **Implemented.**

Parallel Job

Jobs that can be running in one or more cpus in parallel. **Implemented.**

- One of the charactheristics of this type of jobs is the possibility to pass these parameters directly to the CE:
  - ♦ WHOLENODES
  - ♦ SMPGRANULARITY
  - ♦ HOSTNUMBER
  - ♦ CPUNUMBER

Jdls combining these parameters should be used. **TBD**

Delegation

- There are two types of delegation: the automatic ones or you can delegate before submission. Submit jdls using both methods **Implemented.**

- Make a delegation with an expired proxy. Command should fails. **Implemented.**

- Submit with an expired delegation. Command should fails. **Implemented.**

TEST: https://ggus.eu/ws/ticket_info.php?ticket=79096

Shallow and deep re-submission

There two type of resubmission; the first is defined *deep* occurs when the user's job has stardted running on the WN and then the job itself or the WMS JobWrapper has failed. The second one is called *shallow* and occurs when the WMS JobWrapper has failed before starting the actual user's job. **Implemented.**

Job List-match Testing

Test various matching requests **Implemented.**

With data

Test matchmaking using data requests **TBD**

- You need to register a file on an SE, then try a list-match using a jdl like this one (as InputData put the lfn(s) registered before):

```
########################################
#      JDL with Data Requirements      #
########################################

Executable = "calc-pi.sh";
Arguments = "1000";
StdOutput = "std.out";
StdError = "std.err";
Prologue = "prologue.sh";
InputSandbox = {"calc-pi.sh", "fileA", "fileB","prologue.sh"};
OutputSandbox = {"std.out", "std.err","out-PI.txt","out-e.txt"};
Requirements = true;

DataRequirements = {
[
DataCatalogType = "DLI";
DataCatalog = "http://lfcserver.cnaf.infn.it:8085";
InputData = {"lfn:/grid/infngrid/cesini/PI_1M.txt","lfn:/grid/infngrid/cesini/e-2M.txt"};
]
};
DataAccessProtocol = "gsiftp";
```

The listed CEs should be the ones "close" to the used SE

Gang-Matching

If we consider for example a job that requires a CE and a determined amount of free space on a close SE to run successfully, the matchmaking solution to this problem requires three participants in the match (i.e., job, CE and SE), which cannot be accommodated by conventional (bilateral) matchmaking. The gangmatching feature of the classads library provides a multilateral matchmaking formalism to address this deficiency.

Try some listmatch using different expressions of Requirements which use the `anyMatch()` function: TBD

WMS Job Cancel Testing

Test the cancellation of these type of jobs (final status should be *Cancelled*):

- Submit and cancel a normal job **Implemented.**
- Submit a dag job and then cancel it (the *parent*) **Implemented.**
- Submit a dag job and then cancel some of its nodes **Implemented.**
- Submit a collection job and then cancel it (the *parent*) **Implemented.**
- Submit a collection job and then cancel some of its nodes **Implemented.**
- Cancellation of a *Done* job should fails. **Implemented.**

It is important that the cancel is issued at different times from the submission. Id state is submitted, then WMP code is tested, waiting->scheduled wm (+jc), running, wm+lm

Prologue and Epilogue jobs

In the jdl you can specify two attributes *prologue* and *epilogue* which are scripts that are execute respectively before and after the user's job. **Implemented.**

Proxy renewal

- Submit a long job with *myproxyserver* set using a short proxy to both CE (lcg and CREAM). Job should finishes *Done (Success)* **Implemented.**
- Submit a long job without setting *myproxyserver* using a short proxy to both CE (lcg and CREAM). Job should finishes *Aborted* with reason "proxy expired" **Implemented.**

WMS feedback

This mechanism avoid a job to remain stuck for long time in queue waiting to be assigned to a worker node for execution. There are three parameters in the jdl that can be used to manage this mechanism:

1. EnableWMSFeedback
2. ReplanGracePeriod
3. MaxRetryCount

The test should submit a lot of long jobs with short ReplanGracePeriod using a small number of resources, at the end of the test some jobs should be replanned (i.e. reassigned to different CEs). This can be evinced from the logging info of the jobs. (TBD)

A list-match with a jdl where *EnableWMSFeedback* is set to true must return only `CREAM` CE

Limiter mechanism

The WMS has implemented a limiter mechanism to protect himself from overload. This mechanism is based on different parameters anc can be configured inside wms configuration file. All these parameters should be checked and tested. (TBD)

```
Usage:/usr/sbin/glite_wms_wmproxy_load_monitor [OPTIONS]...
      --load1       threshold for load average (1min)
      --load5       threshold for load average (5min)
      --load15    threshold for load average (15min)
      --memusage    threshold for memory usage (%)
      --swapusage    threshold for swap usage (%)
      --fdnum       threshold for used file descriptor
      --diskusage    threshold for disk usage (%)
      --flsize    threshold for input filelist size (KB)
      --flnum    threshold for number of unprocessed jobs (for filelist)
      --jdsize    threshold for input jobdir size (KB)
      --jdnum    threshold for number of unprocessed jobs (for jobdir)
       --ftpconn    threshold for number of FTP connections
       --oper       operation to monitor (can be listed with --list)
      --list       list operation supported
      --show       show all the current values
```

Purging

There are differents purging mechanisms on the WMS:

- SandboxDir purging done by a cron script using command: /usr/sbin/glite-wms-purgeStorage.sh
- Internal purging which means removal of files used by the various daemons as temporary information store. This purging should be done by the daemons themself.
- Proxy purging done by a cron script using the command: /usr/bin/glite-wms-wmproxy-purge-proxycache

All these mechanism should be tested, i.e. check if all the unused files are removed at the end of the job. (TBD)

Configuration file

The file /etc/glite-wms/glite_wms.conf is used to configure all the daemons running on a WMS. A lot of parameters should be set with this file. Almost all these parameters should be checked. (TBD)

It should be verified that in the configuration file `/etc/glite-wms/glite_wms.conf` there are these hard-coded values:

For the common section:

```
DGUser = "\${GLITE_WMS_USER}"
HostProxyFile = "\${WMS_LOCATION_VAR}/glite/wms.proxy"
LBProxy = true
```

For the JobController section:

```
CondorSubmit = "${CONDORG_INSTALL_PATH}/bin/condor_submit"
CondorRemove = "${CONDORG_INSTALL_PATH}/bin/condor_rm"
CondorQuery = "${CONDORG_INSTALL_PATH}/bin/condor_q"
CondorRelease = "${CONDORG_INSTALL_PATH}/bin/condor_release"
CondorDagman = "${CONDORG_INSTALL_PATH}/bin/condor_dagman"
DagmanMaxPre = 10
SubmitFileDir = "${WMS_LOCATION_VAR}/jobcontrol/submit"
OutputFileDir = "${WMS_LOCATION_VAR}/jobcontrol/condorio"
InputType = "jobdir"
Input = "${WMS_LOCATION_VAR}/jobcontrol/jobdir/"
LockFile = "${WMS_LOCATION_VAR}/jobcontrol/lock"
LogFile = "\${WMS_LOCATION_LOG}/jobcontoller_events.log"
LogLevel = 5
MaximumTimeAllowedForCondorMatch = 1800
ContainerRefreshThreshold = 1000
```

For the NetworkServer section:

```
II_Port  = 2170
Gris_Port = 2170
II_Timeout = 100
Gris_Timeout = 20
II_DN = "mds-vo-name=local, o=grid"
Gris_DN = "mds-vo-name=local, o=grid"
BacklogSize = 64
ListeningPort = 7772
MasterThreads = 8
DispatcherThreads = 10
SandboxStagingPath = "${WMS_LOCATION_VAR}/SandboxDir"
LogFile = "${WMS_LOCATION_LOG}/networkserver_events.log"
LogLevel = 5
EnableQuotaManagement = false
MaxInputSandboxSize = 10000000
EnableDynamicQuotaAdjustment = false
QuotaAdjustmentAmount = 10000
QuotaInsensibleDiskPortion = 2.0
```

```
DLI_SI_CatalogTimeout = 60
ConnectionTimeout = 300
```

For the LogMonitor section:

```
JobsPerCondorLog = 1000
LockFile = "${WMS_LOCATION_VAR}/logmonitor/lock"
LogFile = "${WMS_LOCATION_LOG}/logmonitor_events.log"
LogLevel = 5
ExternalLogFile = "\${WMS_LOCATION_LOG}/logmonitor_external.log"
MainLoopDuration = 5
CondorLogDir = "${WMS_LOCATION_VAR}/logmonitor/CondorG.log"
CondorLogRecycleDir = "${WMS_LOCATION_VAR}/logmonitor/CondorG.log/recycle"
MonitorInternalDir = "${WMS_LOCATION_VAR}/logmonitor/internal"
IdRepositoryName = "irepository.dat"
AbortedJobsTimeout = 600
GlobusDownTimeout = 7200
RemoveJobFiles = true
ForceCancellationRetries = 2
```

or the Workloadmanager section:

```
PipeDepth = 200
WorkerThreads = 5
DispatcherType = "jobdir"
Input = "${WMS_LOCATION_VAR}/workload_manager/jobdir"
LogLevel = 5
LogFile  = "${WMS_LOCATION_LOG}/workload_manager_events.log"
MaxRetryCount = 10
CeMonitorServices = {}
CeMonitorAsynchPort = 0
IsmBlackList = {}
IsmUpdateRate = 600
IsmIiPurchasingRate = 480
JobWrapperTemplateDir = "${WMS_JOBWRAPPER_TEMPLATE}"
IsmThreads = false
IsmDump = "${WMS_LOCATION_VAR}/workload_manager/ismdump.fl"
SiServiceName = "org.glite.SEIndex"
DliServiceName = "data-location-interface"
MaxRetryCount = 10
DisablePurchasingFromGris = true
EnableBulkMM = true
CeForwardParameters = {"GlueHostMainMemoryVirtualSize","GlueHostMainMemoryRAMSize","GlueCEPolicyM
MaxOutputSandboxSize = -1
EnableRecovery = true
QueueSize = 1000
ReplanGracePeriod = 3600
MaxReplansCount = 5
WmsRequirements  = ((ShortDeadlineJob =?= TRUE) ? RegExp(".*sdj$", other.GlueCEUniqueID) : !RegEx
```

For the WorkloadManagerProxy:

```
SandboxStagingPath = "${WMS_LOCATION_VAR}/SandboxDir"
LogFile = "${WMS_LOCATION_LOG}/wmproxy.log"
LogLevel = 5
MaxInputSandboxSize = 100000000
ListMatchRootPath = "/tmp"
GridFTPPort = 2811
LBLocalLogger = "localhost:9002"
MinPerusalTimeInterval = 1000
AsyncJobStart = true
EnableServiceDiscovery = false
LBServiceDiscoveryType = "org.glite.lb.server"
ServiceDiscoveryInfoValidity = 3600
WeightsCacheValidity = 86400
```

```
MaxServedRequests = 50
OperationLoadScripts = [
jobRegister = "${WMS_LOCATION_SBIN}/glite_wms_wmproxy_load_monitor --oper jobRegister --load1 22
jobSubmit = "${WMS_LOCATION_SBIN}/glite_wms_wmproxy_load_monitor --oper jobSubmit --load1 22 --lo
RuntimeMalloc = "/usr/lib64/libtcmalloc_minimal.so"
]
```

For the ICE section:

```
start_listener  =  false
start_lease_updater  =  false
logfile  =  "${WMS_LOCATION_LOG}/ice.log"
log_on_file = true
creamdelegation_url_prefix  =  "https://"
listener_enable_authz  =  true
poller_status_threshold_time  =  30*60
ice_topic  =  "CREAM_JOBS"
subscription_update_threshold_time  =  3600
lease_delta_time  =  0
notification_frequency  =  3*60
start_proxy_renewer  =  true
max_logfile_size  =  100*1024*1024
ice_host_cert  =  "${GLITE_HOST_CERT}"
Input  =  "${WMS_LOCATION_VAR}/ice/jobdir"
job_cancellation_threshold_time  =  300
poller_delay  =  2*60
persist_dir  =  "${WMS_LOCATION_VAR}/ice/persist_dir"
lease_update_frequency  =  20*60
log_on_console = false
cream_url_postfix  =  "/ce-cream/services/CREAM2"
subscription_duration  =  86400
bulk_query_size  =  100
purge_jobs  =  false
InputType  =  "jobdir"
listeneristener_enable_authn  =  true
ice_host_key  =  "${GLITE_HOST_KEY}"
start_poller  =  true
creamdelegation_url_postfix  =  "/ce-cream/services/gridsite-delegation"
cream_url_prefix  =  "https://"
max_ice_threads  =  10
cemon_url_prefix  =  "https://"
start_subscription_updater  =  true
proxy_renewal_frequency  =  600
ice_log_level  =  700
soap_timeout  =  60
max_logfile_rotations  =  10
cemon_url_postfix  =  "/ce-monitor/services/CEMonitor"
max_ice_mem = 2096000
ice_empty_threshold = 600
```

It should then be verified that:

- The attribute `II_Contact` of `NetworkServer` section matches the value of the yaim variable `BDII_HOST`
- The attribute `WMExpiryPeriod` of `WorkloadManager` section matches the value of yaim variable `WMS_EXPIRY_PERIOD`
- The attribute `MatchRetryPeriod` of `WorkloadManager` section matches the value of yaim variable `WMS_MATCH_RETRY_PERIOD`
- The attribute `IsmIiLDAPCEFilterExt` of `WorkloadManager` section is `(|(GlueCEAccessControlBaseRule=VO:vo1)(GlueCEAccessControlBaseRule=VOMS:/v`
- The attribute `LBServer` of the `WorkloadManagerProxy` section matches the value of yaim variable `LB_HOST`

Performance tests

Collection of multiple nodes

Submit a collection of **n** (a good compromise should be 1000) nodes. (TBD)

Stress test

Stress tests can parametrized some features: (partially implemented)

- Type of submitted job (e.g. Collections, normal, dag, parametric)
- Submission frequency (i.e. number of submissions for minute)
- Number of submission (i.e. duration of test)
- Number of parallel submission threads (i.e. each one with a different user proxy)
- With or without automatic delegation
- With or without resubmission enable
- With or without proxy renewal enable
- Jdl description
    - with or without sandbox
    - with or without cpu computation
    - executable duration
    - with or without data transfer
    - etc...

This could be an example of stress test

- 2880 collections each of 20 jobs (2 days of test)
- One collection every 60 seconds
- Four users
- Use LCG-CEs and CREAM-CEs (with different batch systems)
- Use automatic-delegation
- The job is a "sleep random(666)"
- Resubmission is enabled
- Enable proxy renewal

Regression tests

Complete list of Rfc tests

Nagios probe test

For tests about Nagios probes see here

Note

Implemented. means that an automatic test exists. Otherwise test must be developed and or execute by hand.

This topic: WMS > TestPlan
Topic revision: r18 - 2013-05-02 - MarcoCecchi