FTS administrator tutorial

Server and service admin tutorial

Gavin McCance (gavin.mccance@cern.ch)

www.eu-egee.org

www.glite.org

Information Society

Outline

- **What is it and who uses it?**
- **Component description and deployment**
- **Installation and configuration**
  - Separate slides
- **Troubleshooting / Support**
  - FTS web-service
  - FTA agents
- **GGUS tickets you will see**

FTS overview

- **The File Transfer Service (FTS) is a data movement fabric service**
  - It is a multi-VO service, used to balance usage of site resources according to VO and site policies

- **Why is it needed ?**
  - For the user, the service it provides is the reliable point to point movement of files
  - For the site manager, it provides a reliable and manageable way of serving file movement requests from their experiments
  - For the production manager, it provides ability to control requests coming from his users
    - Re-ordering, prioritization,…
  - The focus is on the “service”
    - It should make it easy to do these things well

Who uses it?

- **FTS is used by experiment frameworks**
  - Typically end-users do not interact directly with it – they interact with their experiment framework
  - Production managers sometimes query it directly to debug / chase problems

- **Experiment framework decides it wants to move a set of files**
  - The expt. framework is responsible for staging-in (for now..)
  - It packages up a set of source/destination file pairs and submits transfer jobs to FTS
  - The state of each job is tracked as it progresses through the various transfer stages
  - The experiment framework can poll the status at any time

Management concept: channel

- **For management ease, the service supports splitting jobs into multiple "channels"**
  - Once a job is submitted it is assigned to a suitable channel for serving

- **A channel may be:**
  - A point to point network link
    - (e.g. we manage all the T0-export links in WLCG on a separate channel)
  - Various "catch-all" channels
    - (e.g. anywhere to me, or me to anywhere)
  - More flexible channel definitions are on the way (but not there yet)

- **Channels are uni-directional**
  - e.g. at CERN we have one set for the export and one set for the import

Management concept: channel

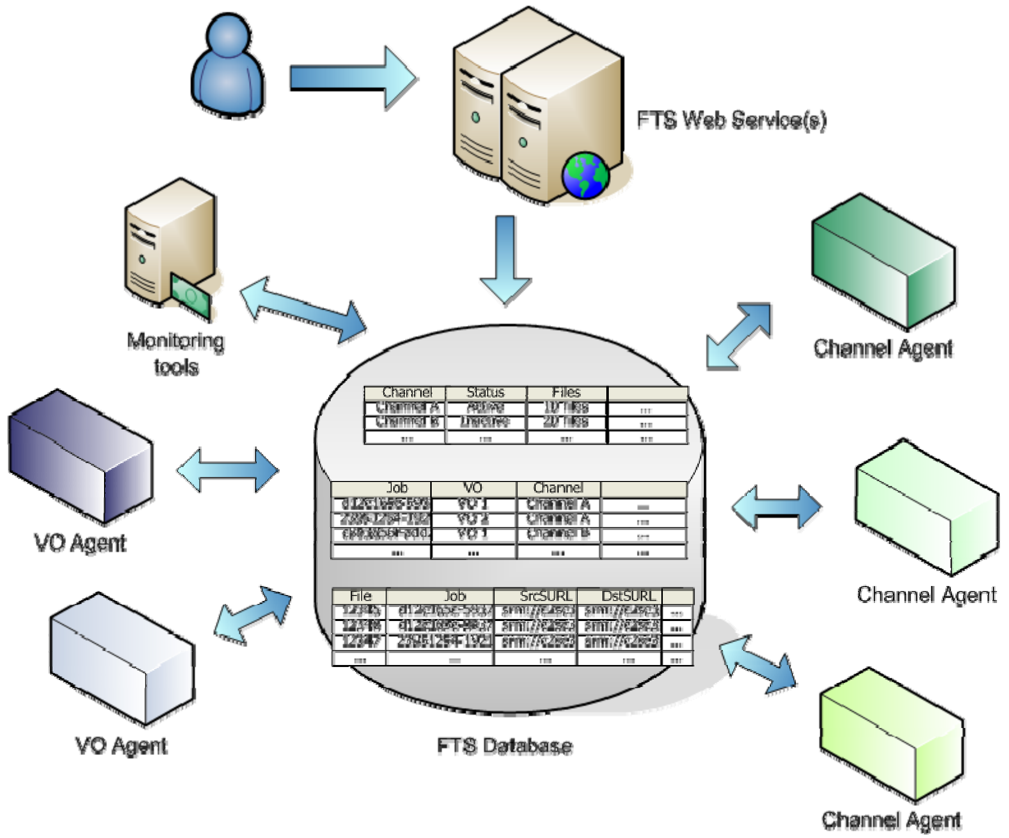
- **“Channel”: it’s not a great name**
  - Isn’t tied to a physical network path
  - It’s just a management concept
  - “Queue” might be better ☺

- **All file transfer jobs on the same channel are served as part of the same queue**
  - Intra-VO priorities for the queue (Atlas gets 75%, CMS gets the rest)
  - Internal-VO priorities within a VO

- **Each channel has its own set of transfer parameters**
  - Number of concurrent files running, number streams, TCP buffer, etc

- **Given the transfers your FTS server is required to support (as defined by experiment computing models and WLCG), channels allow you to split up the management of these as you see fit**
  - For WLCG, we do however make recommendations…

FTS architecture

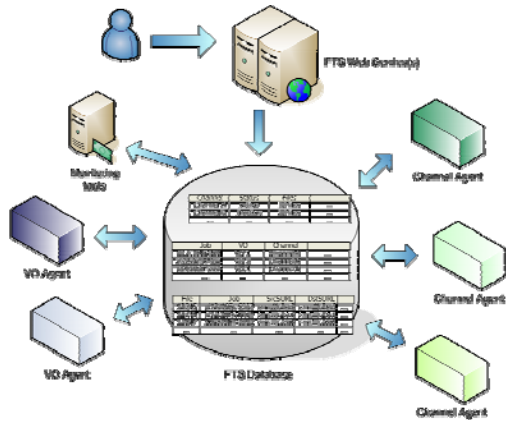


- Experiments interact via web-service
- VO agents do VO-specific operations (1 per VO)
- Channel agents to channel specific operation (e.g. the transfers)
- Monitoring and statistics can be collected via the DB

- All components are decoupled from each other
  – Each interacts only with the database

Deployment models

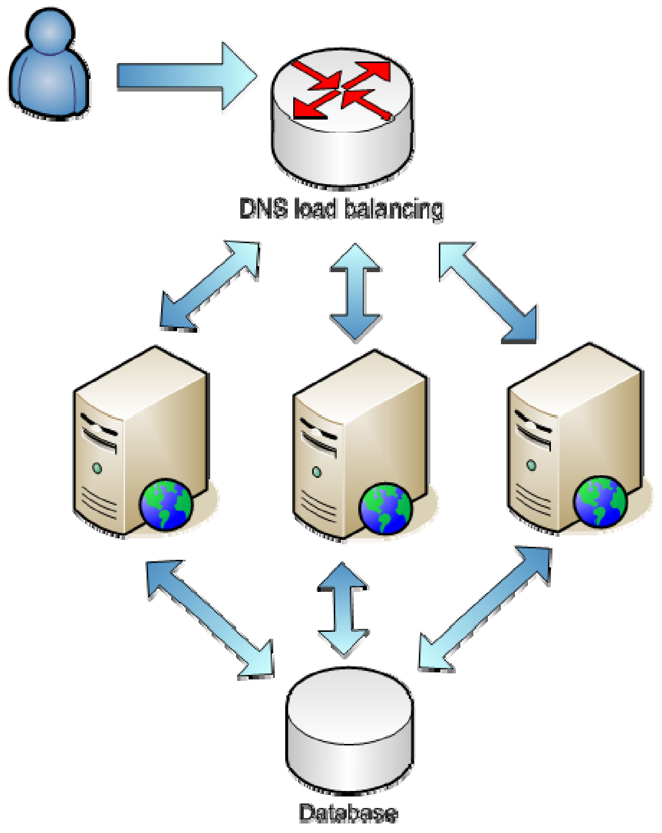
- **For a small test system, all components may be installed on one node**



- **For any serious production-scale system this is not recommended!**
  - The web-service(s) is put on a separate node(s)
  - The VO agents are put on a separate node
  - The transfer agents are split over multiple nodes
  - The Oracle database should be on a separate node
  - The monitoring tools and web-server should be on a separate node

Web-service deployment

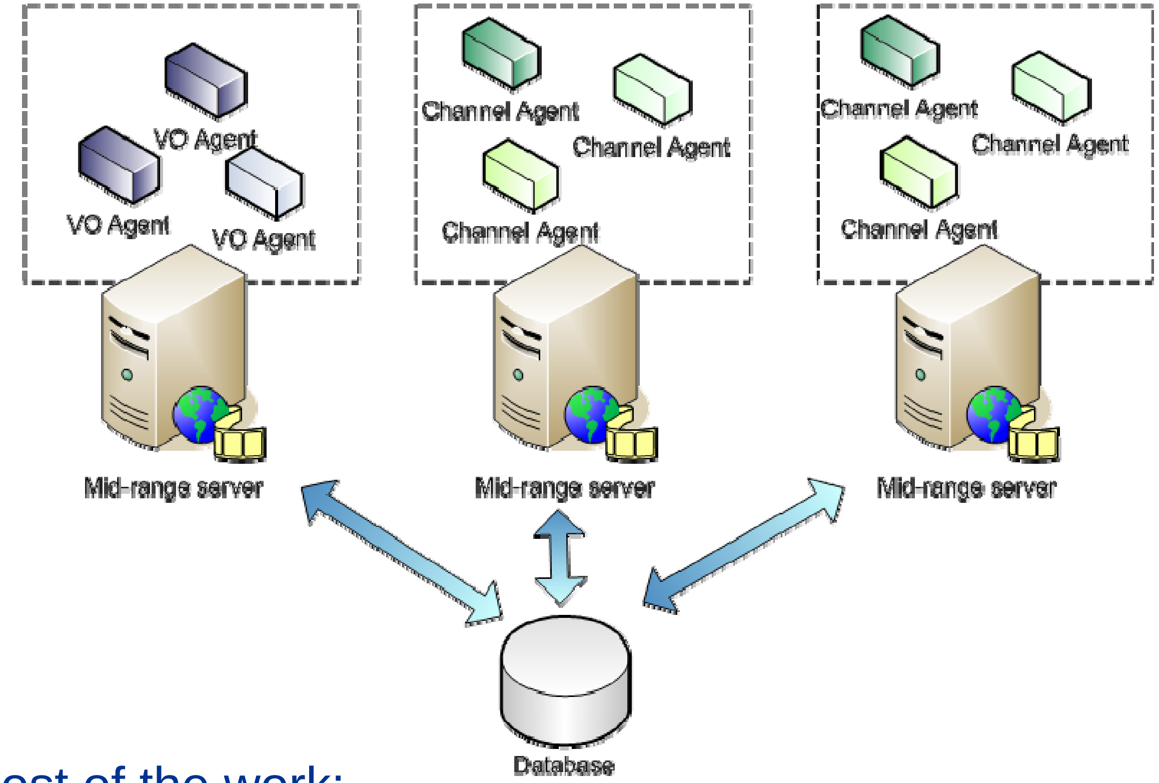
- **The web-service is stateless and is designed to be load-balanced to make it scale**
  - Currently DNS load-balancing is used at CERN
    - (currently over 3 mid-range servers)
  - Investigating apache based load-balancing using mod_jk - this brings a number of benefits
  - Work ongoing to understand best tomcat tuning to improve performance of individual instances



- **Load-balancing also allows rolling software upgrade with zero user-visible downtime and permits graceful failover if one node dies**

Agent deployment

- **The configuration allows the agents daemons to be split arbitrarily over server nodes**
  - VO agents don’t do much just now: suggestion is to put them all on one node



  - Transfer agents do most of the work: split them over multiple nodes as necessary
- **At least one spare node is needed**

Example: CERN

- **CERN tier-0 export / import FTS deployment**
  - All machines are mid-range server
- **Database on separate RAC cluster**
- **10 mid-range servers:**
  - 3 load-balanced web-service instances
  - 24 channel agents split over 3 servers
  - All 6 VO agents together on one server
  - Monitoring tools and web-server deployed on one server
  - 2 pre-configured spare servers

Global Inter-Site Rates - Megatable

Centre	*T0->T1*	*T1->T2*	*T2->T1*	*T1<->T1*
	Predictable – Data Taking	*Bursty – User Needs*	**Predictable – Simulation**	**Scheduled Reprocessing**
IN2P3, Lyon	168.9	286.2	85.5	498.0
GridKA, DE	179.3	384.9	84.1	395.6
CNAF, Italy	214.7	321.3	58.4	583.8
FNAL, USA	110	415.0	52.6	417.0
BNL, USA	186.5	137.7	24.8	358.0
RAL, UK	111.1	108.3	36.0	479.4
NIKHEF, NL	107.0	34.1	6.1	310.4
ASGC, Taipei	72.7	126.5	19.3	241.2
PIC, Spain	55.3	167.1	23.3	294.5
Nordic Data Grid Facility	41.8	-	-	62.4
TRIUMF, Canada	19.2	-	-	59.0

Service availability

- **Essentially, you should think of FTS as two distinct services with regard to availability**

- **The web-service**
  - MUST stay up at all times
  - Otherwise users “see” downtime – their commands are rejected
  - Restarts can be hidden by DNS load-balancing
  - Evaluating apache2 (with gridsite) front-tier that would allow transparent tomcat restart (as well as address some scaling issues)

- **The agents**
  - Can tolerate a bit more flexibility.. But..
  - User will “see” downtime if the rate drops
  - If you stop the VO agents for too long, the channel agents will become starved of work (and the transfer rate will drop)
  - If you stop the channel agents, no new files will start to transfer (but currently running ones will continue)

FTS web-service

FTS web-service

- **The FTS web-service runs inside the tomcat container**
  - Tomcat5 runs as a multi-threaded “j2sdk” process owned by the tomcat4:tomcat4 user
  - Logs are in /var/log/tomcat5/
  - Tomcat itself logs into catalina.out
    - Not so much useful there…
  - The application logs into:
    - org.glite.data – debug information
    - org.glite.data.transfer.fts-calls – one call per line logging client DN and IP and call information for standard FTS operations
    - org.glite.data.transfer.channeladmin-calls – one call per line logging client DN and IP and call information for channel control operations

FTS w/s: what to monitor

- **Things to monitor**
  - That the tomcat5 daemon is running
  - That it's listening on *:tcp/8443 and localhost:tcp/8005
  - That the application running inside it is not stuck
    - e.g. test glite-transfer-channel-list and alarm if it doesn't answer
  - Check that there are ESTABLISHED connections to the database nodes
  - Check logfile for "ORA-xxxxx" errors (and associated stacktrace)

- **Other experiences / advice would be good!**
  - Ron's session this afternoon

FTS w/s: logfiles

- **org.glite.data: main thing is Oracle errors if something bad happens on DB or the connection**
- **The stacktrace is not interesting – the interesting message is at the top of it (telling you what ORA error you got and sometimes why)**
  - It's worth asking your DBA about these – they often indicate a problem on the DB

```
2006-09-07 19:49:43,280 WARN  [http-8443-Processor13]  Caught SQLException. Exiting:  -
OracleFTSDBHelper.listChannels:1474
java.sql.SQLException: ORA-02396: exceeded maximum idle time, please connect again

        at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:112)
        at oracle.jdbc.driver.T2CConnection.checkError(T2CConnection.java:672)
        at oracle.jdbc.driver.T2CConnection.checkError(T2CConnection.java:598)
```

  - The most common is the "idle timeout" message – this is transient but causes a single client call to fail
    - We're looking at ways of fixing this…

How to handle DB problems

- **If there are DB problems causing downtime: transient**
  - Most problems the service can recover without a restart
  - The agents tend to be more resilient in this regard

- **If problems persist:**
  - Fix the problem on the DB (ask your DBA), switch off agents and web-service if necessary
  - Restart the service (agents and/or web-service)
  - No state will be lost (provided the problem on the DB wasn’t terminal)

If the tomcat daemon is up but the web-service isn’t responding

- **This can happen if it runs out of memory**
  - Default heap is just 64MB
- **Restart the tomcat daemon “fixes” the problem**
  - (service tomcat5 restart)
- **See installation guide (slides) on how to add more memory for tomcat**
- **Caveat: possible leak under investigation…**

If the tomcat daemon starts but the web-service doesn't

- This is usually a DB connection problem
- Scan org.glite.data for FATAL errors which should tell you what the problem is

If the tomcat daemon doesn't even start

- **Occasionally I've seen this happen**
  - Usually on test boxes I've been 'playing on'
  - I don't bother trying to debug it

- **Scrub reinstall of tomcat does it**
  - rpm –qa | grep tomcat | xargs rpm –ev … or similar
  - rm –rf /usr/share/tomcat
  - rm –rf /etc/tomcat
  - rm –rf /var/log/tomcat
  - apt-get install tomcat5
  - [YAIM reconfig of FTS]

FTS web-service: other things to do

- **Zip and archive the logs (/usr/share/tomcat5/logs)**
  - They are rotated, but not zipped and not deleted
  - Will be added to YAIM soonish…

FTA agent daemons

FTA agents

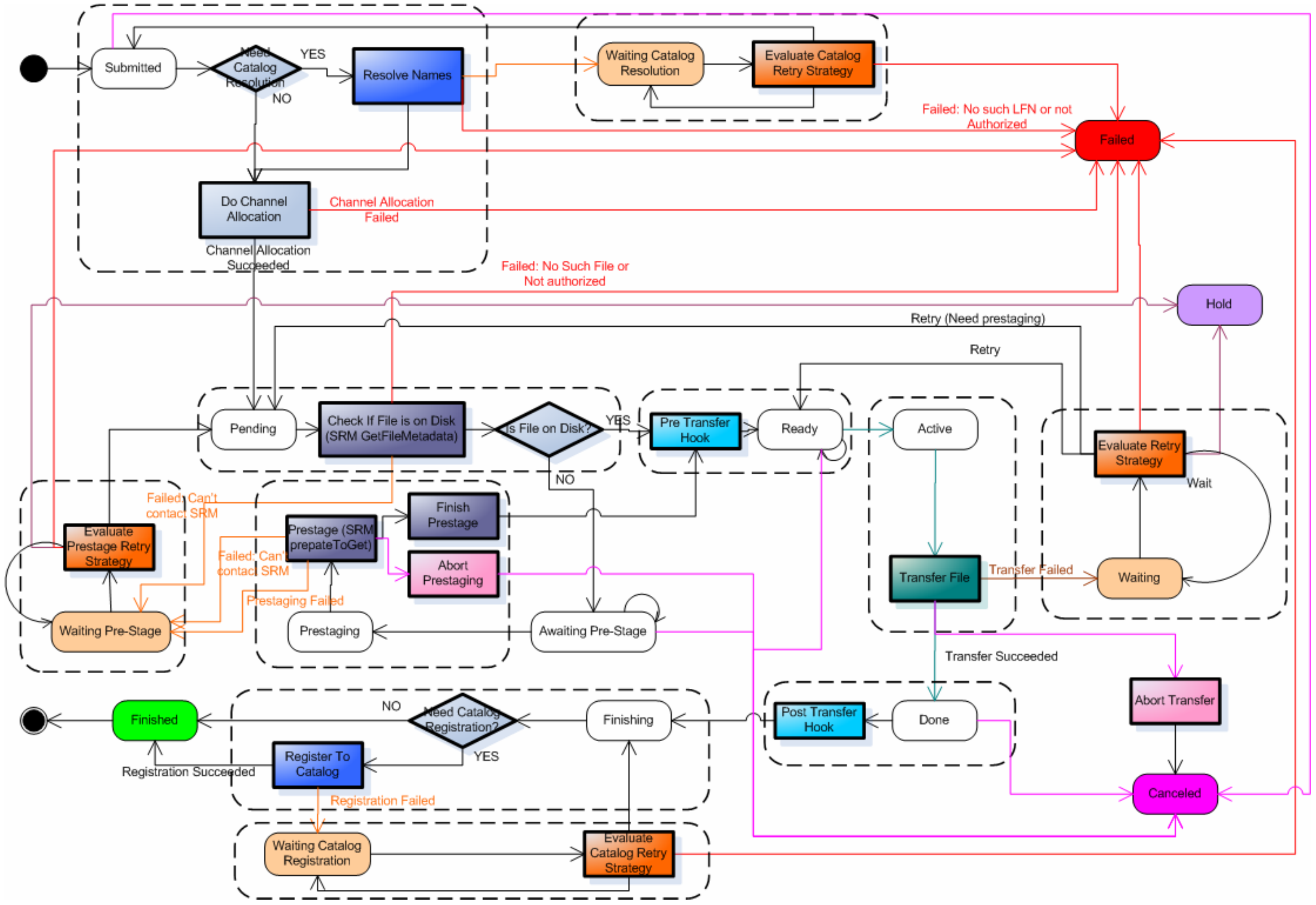
- **What do they actually do?**

- **Each FTA daemon runs a set of independent ‘tasks’**
  - Cron style: each task is run periodically
  - Each task operates on some part of the FTS file state machine, i.e. it’s looking for files or jobs in a particular state so it can do something with them
    - If it doesn’t find any, it skips this iteration, and the next task runs

- **We ‘bundle’ the tasks together into agent daemons**
  - Tasks which may behave differently for different VOs are put in the VO agent (1 per VO)
    - e.g. files in “Waiting” state, awaiting retry: different VO policy
  - Tasks which are channel specific are put in the channel agents (1 per channel)
    - e.g. files in “Pending” state, awaiting transfer

Couldn't have an FTS presentation without it…

- **Details aren't so important here…**



FTA agents

- **The daemons are independent: you can stop one agent without affecting the others**
  - e.g. if you stop all the VO agents because you need to move them to a new box, the channel agents will continue to run and process jobs
    - …but, if you take too long, the channel agents will run out of work, since it’s the VO agent that assigns new jobs to a given channel for processing

- **For practical purposes, it means that if you need to reconfigure (and restart) the agents for maintenance you can freely do this at any time without affecting the service at all**
  - Just don’t take too long…

- **Also note that stopping the agents does not affect the web-service**
  - Users can continue to submit new jobs and poll the status of existing ones via the web-service
  - …their new jobs just won’t be processed until the agents are running

FTA transfer processes: URLCOPY

- **When there is a spare 'slot' on a channel, the agent forks a process to handle the transfer**
  - The process renames it's /proc/?????/cmdline so it's readily visible with ps aux. The basic process name is still glite_data_config
  - It's double forked (init looks after it)
  - It writes it's log into /var/tmp/glite-url-copy-edguser/CHANNEL_%DATE%_%mktmp% which matches the process name
  - It creates a socket file with the same name + .sock
  - The agent process that created the job communicates via the .sock file

- **When the job is finished (Done or Failed), the agent:**
  - scans the log for the FINAL message and some stats
  - Moves the file to CHANNELfailed or CHANNELcompleted
  - Updates the state in the DB

FTA transfer processes: SRMCOPY

- **When there is a spare 'slot' on a channel, the agent forks a process to handle parts of the SRM copy**
  - The process is very short lived – each SRM copy operation (submit or poll) is done separately
  - It writes it's log into /var/tmp/glite-url-copy-edguser/CHANNEL_%DATE%_%mktmp%
  - It creates a memory mapped file to hold its state
  - The agent process that created the job communicates via mapped file

- **When the job is finished (Done or Failed), the agent:**
  - scans the mapped file for the FINAL message and some stats
  - Moves the file to CHANNELfailed or CHANNELcompleted
  - Updates the state in the DB

- **Plans to refactor the URLCOPY code to use the memory mapped file approach**

FTA: what to monitor

- **Things to monitor**
  - Look for configured agents and checks they are running
    - There is a script (not yet YAIMed-up) that does this and writes an alarm file that your local monitoring can react to
  - Check that for every busy URLCOPY channel, there are some forked processes running (there should always be there if the service has work to do)
  - Check that none of the log-files, memory files or sock files in /var/tmp/glite-url-copy-edguser/ are too old (like >3 hours)
  - Check for “ORA-xxxxx” errors in logfiles

- **Other experiences / advice would be good!**
  - Ron’s session this afternoon

Moving an agent: how to drain

- **Why drain?**
  - Because some of the transfer state is in the log-file / memory mapped file of the running transfers
  - You should allow them all the finish so that the agent can ‘sweep’ this final state into the DB
- **Set the channel of the associated agent(s) Inactive**
  - glite-transfer-channel-set –S Inactive CERN-TEST
- **Wait until there are no CERN-TEST transfer processes**
  - ps aux | grep CERN-TEST_
- **Stop the agent**
  - service transfer-agents stop –instance glite-transfer-channel-agent-urlcopy-CERN-TEST
- **Rerun YAIM to remove the agent from the old machine**
- **Wait (~3 minutes)**
- **Rerun YAIM on the new machine to add the new agent**
- **Bring it up on another machine**
  - service transfer-agents start –instance glite-transfer-channel-agent-urlcopy-CERN-TEST
  - glite-transfer-channel-set –S Inactive CERN-TEST
- **N.B. VO agents do not need to be drained**
  - Just stop it on one machine and start on another (after 3 minutes)

If the agents do not start

- **Check there is no other agents for the given channel or VO running (on this node and all other agent nodes)**
  - Only one must run per channel or per VO
  - The mutex logic is not bad, but isn't yet perfect…
  - This is particularly important if you've just changed an agent from "urlcopy" to "srmcopy" or vice-versa

- **If it fails, check the logfile (/var/log/glite/) for DB connection errors**

If the agents do not stop

- **If this command fails:**
  - service transfer-agents stop –instance glite-transfer-channel-agent-urlcopy-CERN-TEST
- **The agent may be busy…**
  - It won't stop until it finishes it's current task
  - Wait a while and try again
- **It may be stuck (e.g. bug 19539)**
  - service transfer-agents kill –instance glite-transfer-channel-agent-urlcopy-CERN-TEST

FTA: other things to do

- **Things to do (that YAIM doesn’t yet do..)**
  - Archive the /var/tmp/glite-url-copy-edguser/CHANNEL*/ directories regularly

Transfer-scripts

- **For next release…**
  - Available unpackaged now

- **History cleanup**
  - You MUST cleanup old jobs (otherwise each task query takes longer and longer as old finished jobs pile up)
  - There is a PL/SQL tool in transfer-scripts to help you do this
  - It archives jobs over 7 days old, by default
- **Admin tools**
  - Channel renaming
  - Channel sitename changing
- **Monitoring tools**
  - To be decided at this workshop!

Troubleshooting

- Or…
- “GGUS tickets you may see”
- “GGUS tickets you ~~may~~ will see”
- “GGUS tickets you ~~may~~ will have seen”

GGUS tickets you will see 1

- **“Our jobs are all stuck in ‘Submitted’ state”**
  - The associated VO agent is down

- **“All our jobs are stuck in the ‘Pending’ state on channel X”**
  - The associated channel agent is down
  - The channel is simply busy (check for active URLCOPY jobs in the process) – i.e. be patient
  - The VO share is too low for the given VO

GGUS tickets you will see 2

- **“My jobs failed with “No channel found or VO not authorized”**
  - Either:
    - The user has submitted a job which has matched a source and dest site, but there is no channel setup to serve this site
    - In this case, consider whether your site should be supporting channels between these two sites (i.e. according to the WLCG model that we will agree here)
    - If not, direct the user to the correct site according to the agreed model
    - DON’T just set up a channel for it or you may break the discovery tools ☺ [to discuss later on today…]
  - Or:
    - There is a channel, but no VO share defined on it for your VO
    - If required, you can add a share for this VO

GGUS tickets you will see 3

- **My jobs failed with “No site found for host RANDOM-T2”**
  - Rerun the services.xml generator from the central BDII
  - https://twiki.cern.ch/twiki/bin/view/LCG/FtsServerServicesXml15
  - Recommendation for production is:
    - DO NOT add resources by hand into the file: use only SRMs that publish correctly into the information system

GGUS tickets you will see 4

- **Cannot contact web-service**
  - Various TCP connect messages, e.g.
    - Failed to determine the interface version of the service: getInterfaceVersion: SOAP fault: SOAP-ENV:Client - CGSI-gSOAP: Could not open connection ! (TCP get host by name failed in tcp_connect())
  - Web-service is down
  - Bad host in connection or bad endpoint
    - ask user to run CLI with –v option to check

- **Web-service doesn’t respond**
  - It’s stuck – restart Tomcat and look at adding more memory to the java heap (see Installation and Configuration slides)

GGUS tickets you will see 5

- **"My job X failed with:" e.g.**
  - "SRM_DEST: Failed on SRM put: Failed SRM put on httpg://dcsrm.usatlas.bnl.gov:8443/srm/managerv1 ; id=-2142137570 call. Error is
  - RequestFileStatus#-2142137569 failed with error:[GetStorageInfoFailed : file exists, cannot write]"
  - "TRANSPORT: Transfer failed. ERROR the server sent an error response: 451 451 Local resource failure: malloc: Cannot allocate memory."

- **The FTS has "correctly failed"™ the job**
  - i.e. after 3 retries, it still didn't work
    - Chase the problem with the SRMs (either your SRM or the other site's SRM)
    - If it's a big problem, set the channel Inactive while you fix it
    - The experiment frameworks all retry externally

GGUS tickets you will see 6

- **“Your FTS is broken / none of my files transfer”**
  - Try to disentangle what the user actually means, remembering the two distinct services
    - FTS web-service
    - Agents
  - Can you submit a job?
  - Can you query a job?
  - Do you mean your job (after submission) doesn’t process (i.e. is stuck in one state?)
    - Which state is it stuck in?
  - Do you mean the jobs “correctly fails”™ ?
    - i.e. an SRM isn’t working somewhere

Reference

- **All the tutorial material is at:**
  - https://twiki.cern.ch/twiki/bin/view/LCG/FtsTutorial
- **All release information and guides:**
  - https://twiki.cern.ch/twiki/bin/view/LCG/FtsRelease15
- **FTS procedures (upgrading, moving, cleaning)**
  - https://twiki.cern.ch/twiki/bin/view/LCG/FtsProcedures15
- **FTS FAQ**
  - https://twiki.cern.ch/twiki/bin/view/EGEE/DMFtsSupport
- **Mailing list**
  - fts-support@cern.ch for support (closed support)
  - fts-users@cern.ch for community support