File Transfer Software SC3

Gavin McCance – JRA1 Data Management Cluster

Service Challenge Meeting
March 15, 2005, Lyon, France

Outline

- Requirements and status
- Design and components
- Testing status

Reliable File Transfer Requirements

- **LCG created a set of requirements based on the Robust Data Transfer Service Challenge**
- **LCG and gLite teams translated this into a detailed architecture and design document for the software and the service**
  - A prototype (radiant) was created to test out the architecture and used in SC1 and SC2
  - gLite FTS ("File Transfer Service") is an instantiation of the same architecture and design, and is the candidate for use in SC3
    - Current version of FTS and SC2 radiant software are interoperable

Requirements Summary

- **From the requirements document:**

Functional	R1.1	R1.2	R1.3	R1.4	R1.5	R1.6	R1.7	R1.8	R1.9	R1.10
	YES	PARTIAL	YES	(NO)	YES	YES	YES	YES	(NO)	PARTIAL

Security	R2.1	R2.2	R2.3	***Scalability***	R3.1	R3.2	R3.3	R3.4
	YES	YES	YES		To demonstrate			

Manageability	R4.1	R4.2	R4.3	R4.4	R4.5	R4.6	R4.7
	YES	To demonstrate		YES	PARTIAL	PARTIAL	YES

Monitoring	R5.1	R5.2	R5.3	R5.4	R5.5
	NO	NO	YES	PARTIAL	YES

Scheduling	R6.1	R6.2	R6.3	R6.4	***User interface***	R7.1	R7.2	R7.3
	YES	(NO)	(NO)	(NO)		YES	YES	PARTIAL

- **Core functionality mostly done: some improvements for SC3**
- **Focus is on monitoring, manageability and service stability**

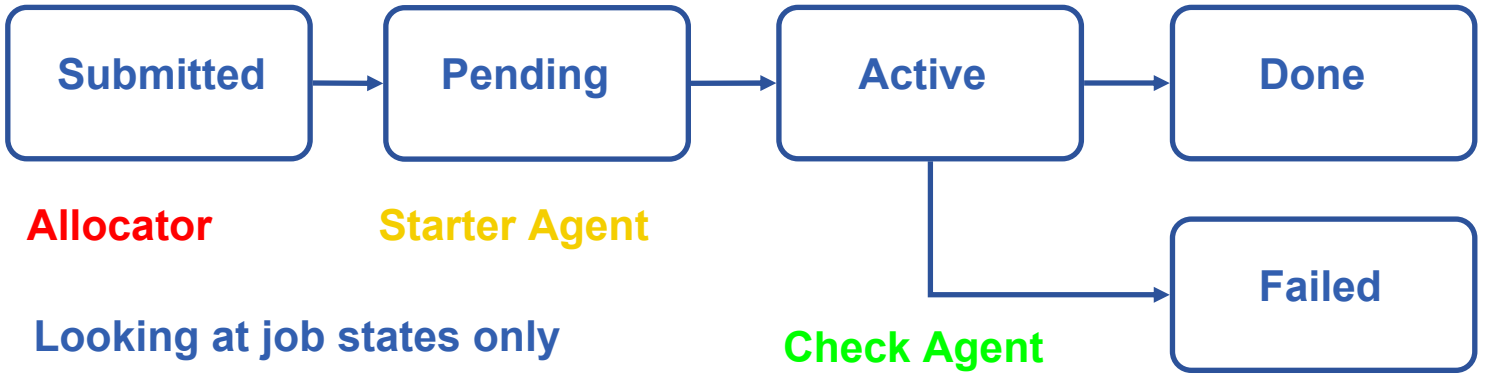
Notes on architecture

- **FTS provides point to point movement of SURLs**
  - Aims to provide reliable file transfer, and that's it!
  - Does not do 'routing' (e.g like Phedex)
  - Does not deal with GUID, LFN, Dataset, Collections
- **These are all higher level concepts and can be dealt with by higher level services**
- **It provides a pluggable agent-based architecture where some of this higher level work could be done, if required**
  - VOs could provide their own agents to do VO specific things (e.g integrated cataloging)
  - gLite will aim to provide agents that integrate the FTS with the rest of the gLite software stack

Design concepts

- **System is driven by state machine and a set of 'agents' working on their own part of the state machine**
  - The state is kept in a database.
  - Agents should be as stateless as possible, and always able to recover the true state should they fail
- **Multiple file transfers grouped into a single job**
  - Overall job state ("is it done yet?") is a function of the individual states of the job's constituent files

Somewhat simplified design



e.g. three components:

- Allocator: looks for jobs in 'Submitted' - assigns a suitable channel to them (based on the files in the job) and then sets the job state to 'Pending'.
- Transfer agents:
  - looks for jobs in 'Pending' where the channel name is one it is responsible for. If resources are available, start the copying: set to Active.
  - Checks 'Active' jobs: set to 'Done' or 'Failed' depending on result of job's transfers

Components

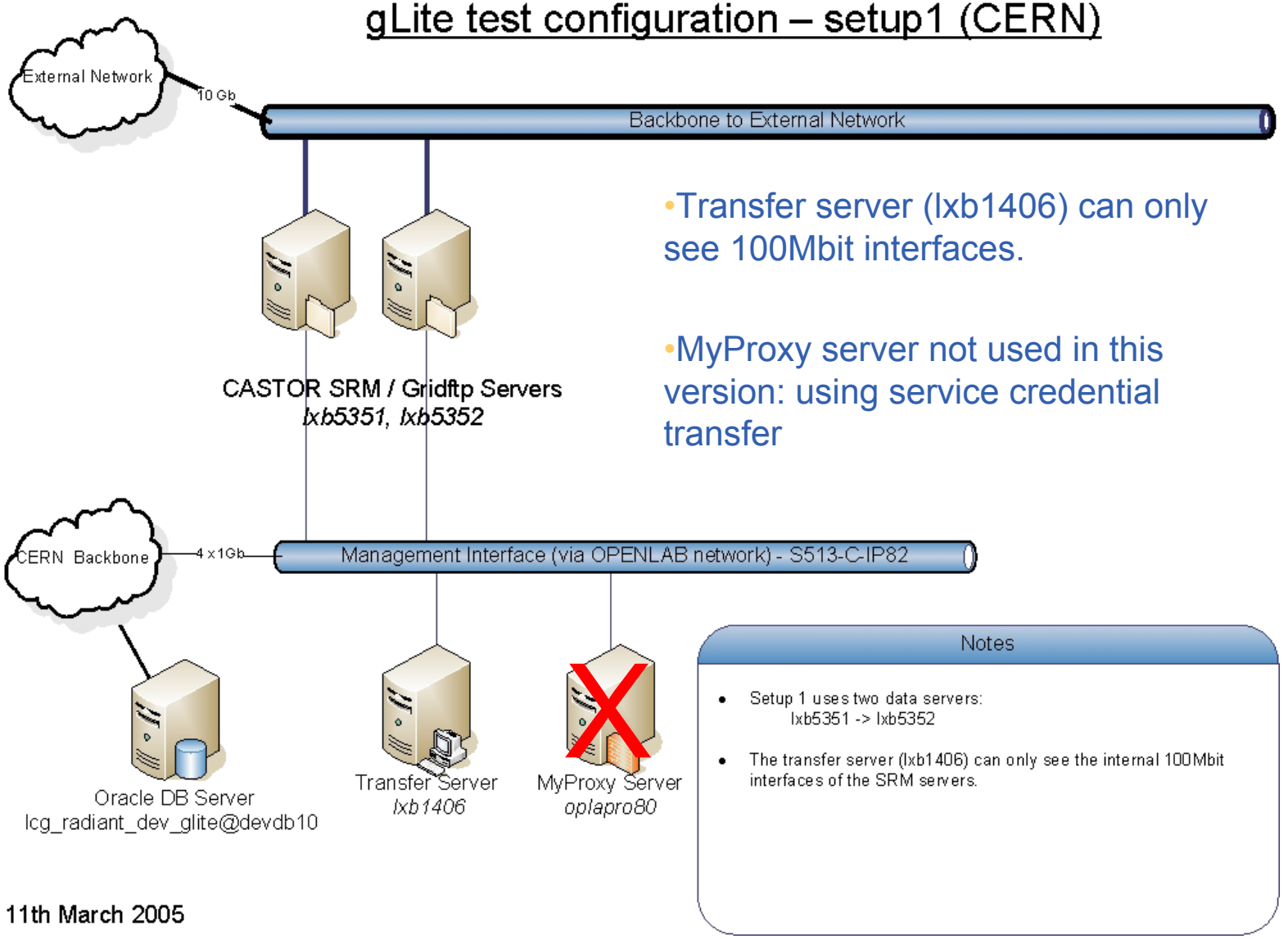
- **Database: stores all the state**
- **Web-service: wrapper over database, secure submissions and status querying via the interface, channel service management**
- **Decoupled ~stateless agents:**
  - Allocator agent
  - Transfer agents
  - Todo: retry agents to decide what to do with failed files (depending on the failure class).
- **Command line: command line clients (job management and service management)**
- **GUI: monitoring GUI running against web-service or DB directly**

Test setups

- **The goal was to demonstrate ~ 1 week unattended continuous running of the software for the SC meeting**
  - Focus was on demonstrating service stability rather than maximum throughput

- **Test setup was created at CERN**
  - 1st setup: low bandwidth
  - 2nd setup: higher bandwidth, newer version of software

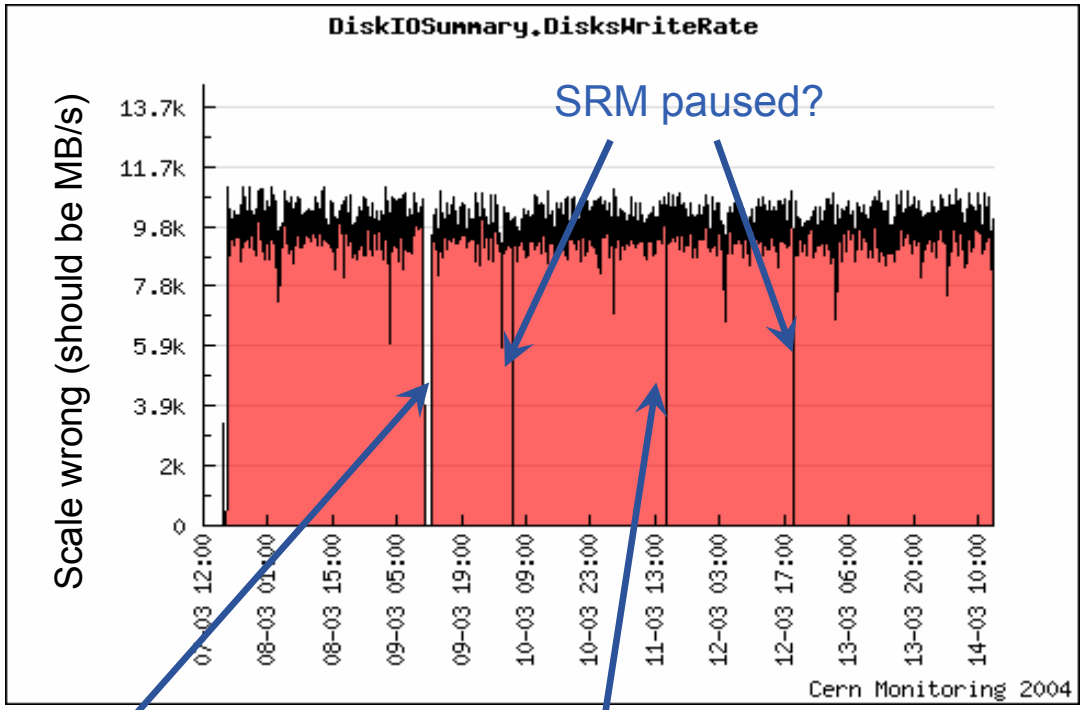
Test setup 1



- Transfer server (lxb1406) can only see 100Mbit interfaces.
- MyProxy server not used in this version: using service credential transfer

Test setup 1

- **Monitoring on SRM destination node**

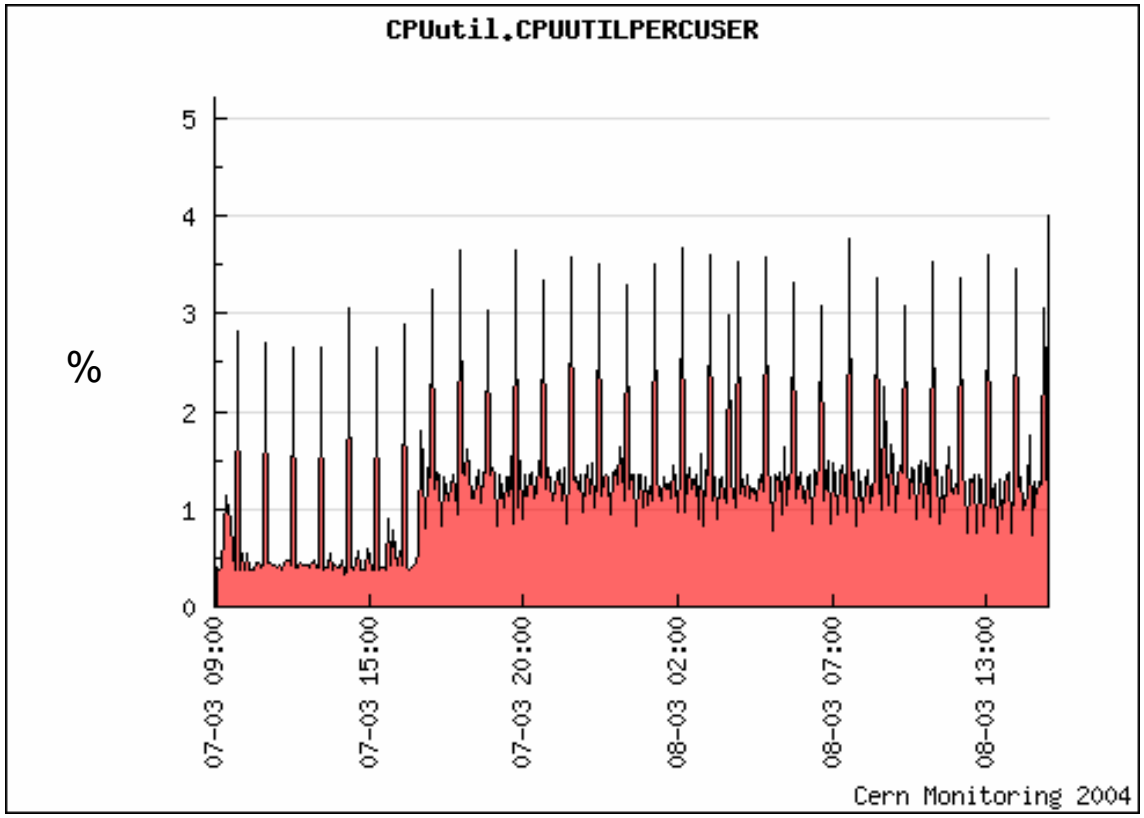


- **Disk write rate on destination SRM**
  - ~10MB/s limited by network card
- **> 6000 1-gig files transferred**
- **40 failed**
  - Failed file transfer are not yet retried

4 concurrent transfers (1 stream each)

Test setup 1

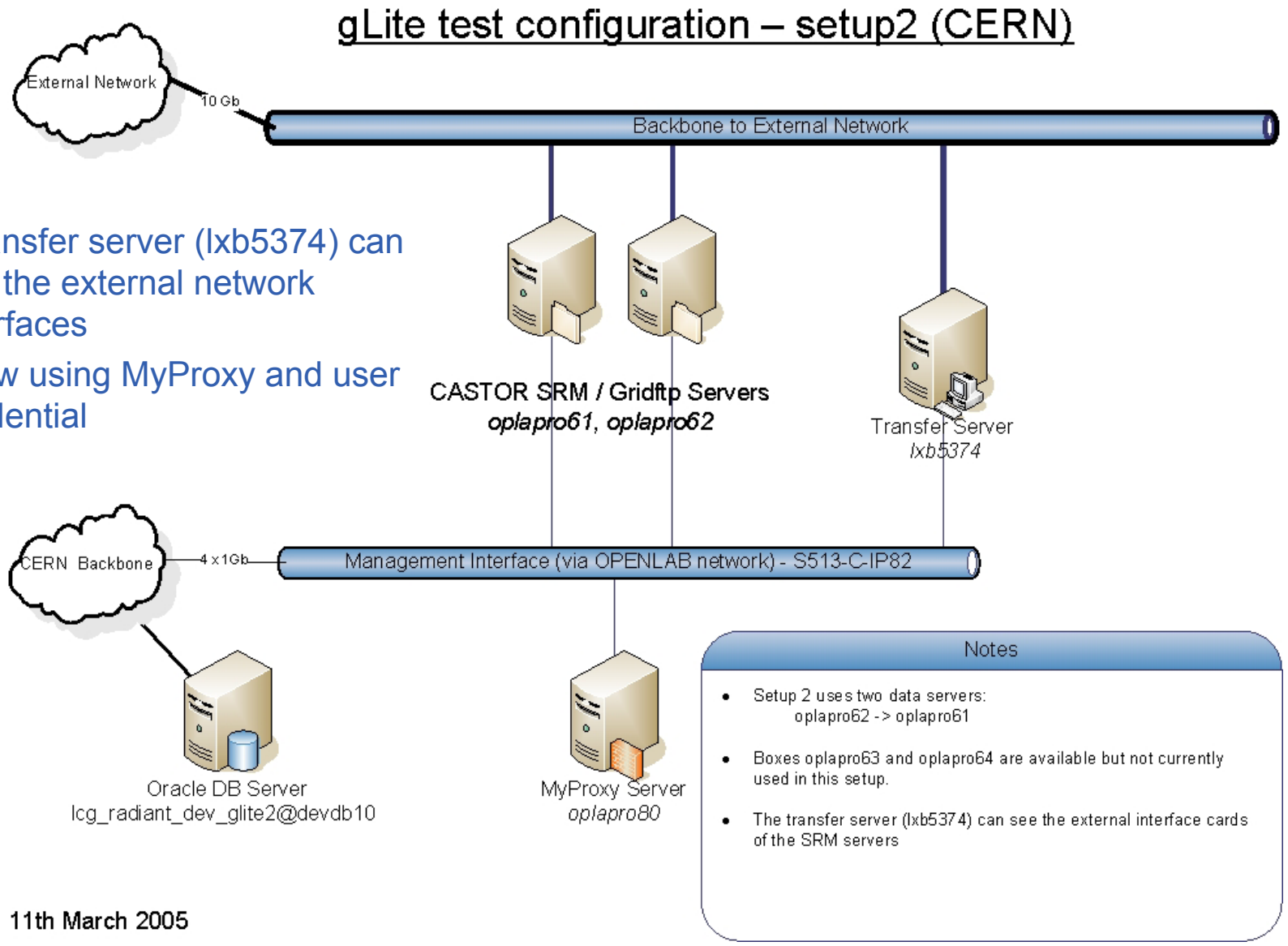
- **Monitoring on transfer server node (lxb1406)**
  - Helps considerably with the debug effort



- **Various statistics**
  - CPU
  - Memory, swapping
  - Disk I/O rates
  - Network rates
- **Helps us monitor the performance of the transfer software and catch problems early**

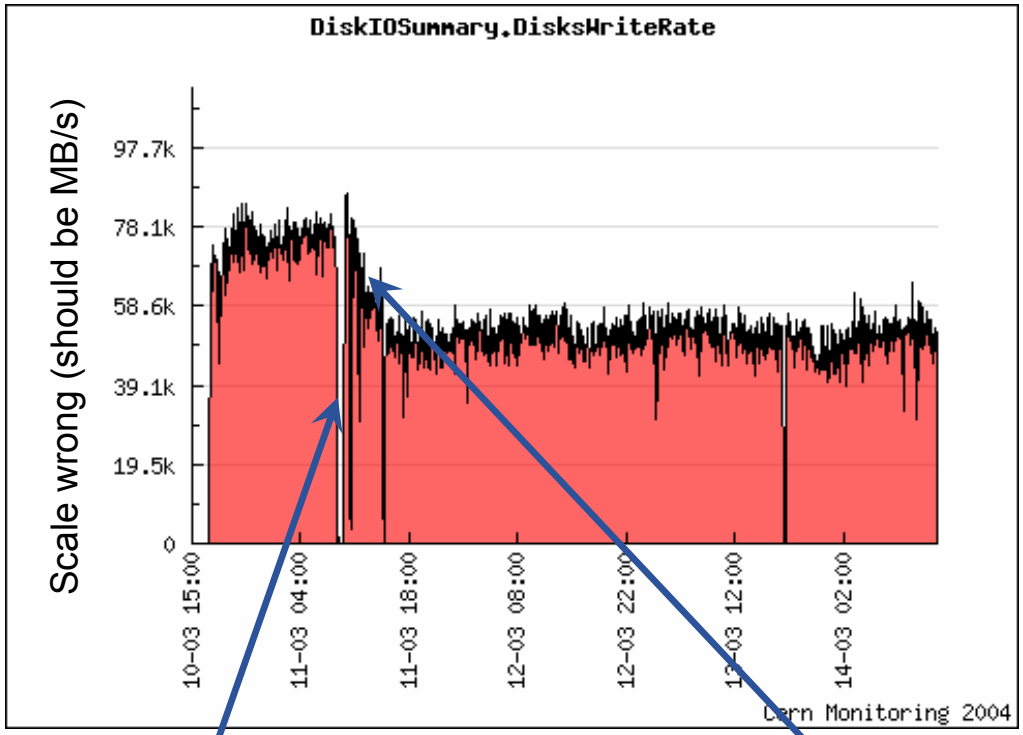
Test setup 2

- Transfer server (lxb5374) can see the external network interfaces
- Now using MyProxy and user credential



Test setup 2

- **Monitoring on SRM destination node**



Critical bug found in transfer code (the new MyProxy part):

Bug-fixed version was deployed and service restarted.

Transfer request distribution changed

- **Disk write rate on destination SRM**
  - Both source and destination systems have 3 disks
  - Achieved rate is very sensitive to distribution of files on disk
- **> 14 000 1-gig files transferred OK**
- **22 failed**
  - + others as result of the bug we found

4 concurrent transfers (1 stream each)

Did we do it?

- **The goal was to demonstrate ~ 1 week unattended continuous running of the software for the SC meeting**
  - On setup 1: demonstrated 1 week ~continuous running
  - But, 2 interventions were necessary
  - Setup 2: higher bandwidth, newer version of software
  - After false start has been stable for a few days
    - 1 intervention necessary (same cause as on setup1)
- **Issues uncovered and bugs found (and fixed!)**
- **We now have a test setup we can use to keep stressing the software**

Test setups

- **Plan is to maintain these test setups 24/7**
  - They've already been vital in finding issues that we could not find on more limited test setups
- **Test 1 (low bandwidth) should be used for testing the new version of the software**
  - Make sure no bugs have been introduced
  - Try to maintain stable service level, but upgrade to test new versions
- **Test 2 (higher bandwidth) should be used to demonstrate stability and performance**
  - Maintain service level
  - Upgrade less frequently, only when new versions have demonstrated stability on smaller test setup
  - Ramp up this test setup as we head to SC3
    - More machines
    - More sites
    - Define operating procedures

Current status and issues

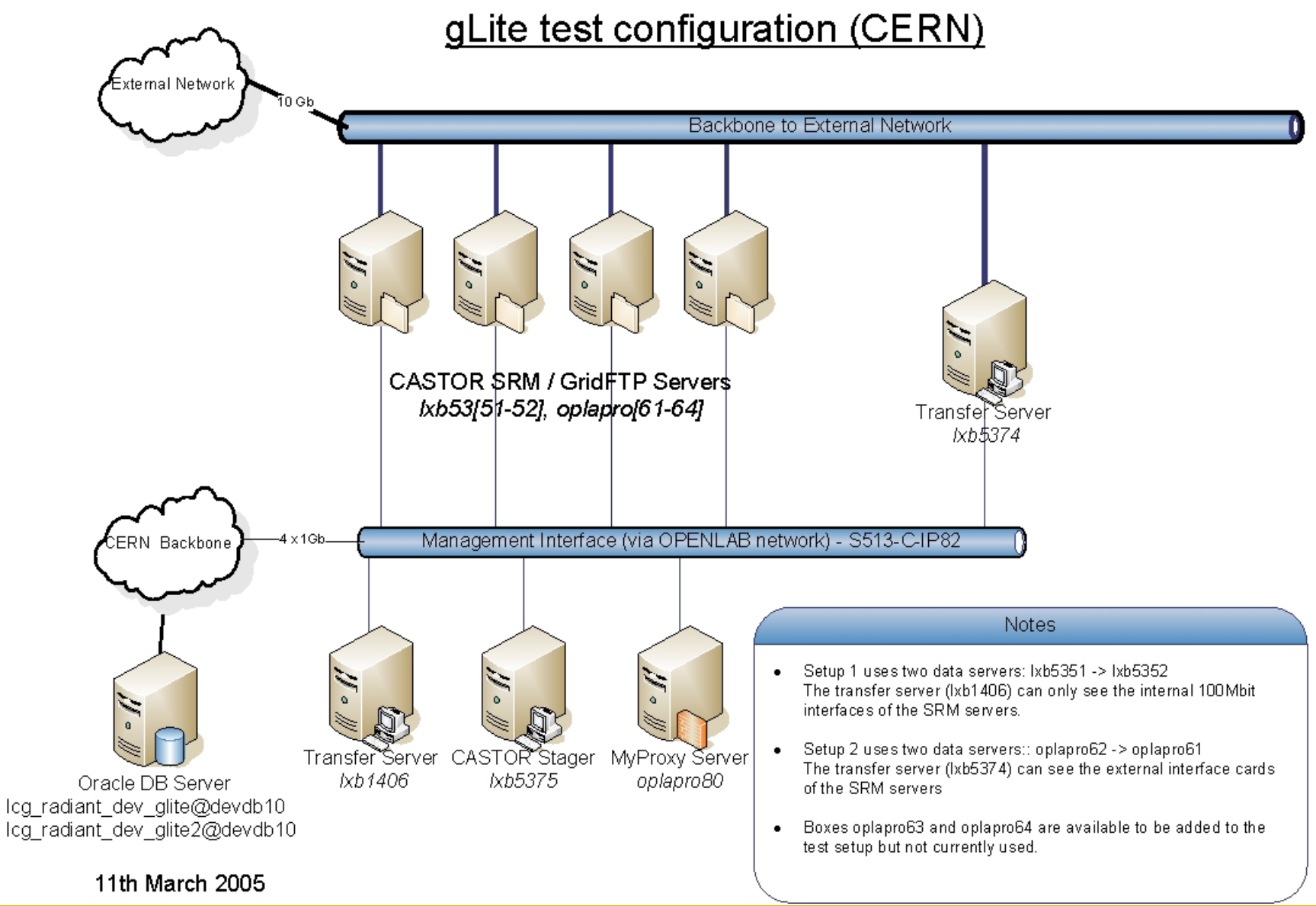
- **Status of work is tracked**

- **http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/transfer/index.htm**
  - With links to the various requirements and design documents

- **http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/transfer/issues.htm**
  - Status of tests, current bugs and issues

Backup slides

Overall test setup



Functional Requirements

#	Requirement Description	Current status
1.1	Asynchronously and reliably copy files	YES
1.2	Recovery from partial/failed transfers	PARTIAL Improvements for SC3.
1.3	Report to user the current progress	YES
1.4	Allow user to prioritise their requests	NO. Not for SC3.
1.5	Multi-file requests as single atomic unit	YES
1.6	Access files over a cluster of data servers	YES
1.7	Handle files on disk, accessing via gridFTP	YES
1.8	Handle files via SRM (gridFTP as TURL)	YES
1.9	Other protocols should be usable	NO. Not for SC3.
1.10	Interact with mass storage when scheduling requests	PARTIAL Improvements for SC3.

Security Requirements

#	Requirement Description	Current status
2.1	Require user to be authenticated to use service	YES
2.2	Require user to be authorized to use service	YES
2.3	Use user's proxy to do the transfer	YES

Scalability Requirements

#	Requirement Description	Current status
3.1	Scale to 500MB/s continuous point-to-point	Still to demonstrate
3.2	Scale to 1GB/s peak point-to-point	Still to demonstrate
3.3	Scale to simultaneous transfer of 100 files point-to-point	Still to demonstrate
3.4	Be able to store 1000 multi-file requests in the “pending transfer” queue per point-to-point connection	Still to demonstrate

- Test setup is currently running which is incrementally stressing the transfer software
- Setup is as close to SC2 production setup as possible, and will be stepped up in preparations for SC3

Manageability Requirements

#	Requirement Description	Current status
4.1	Less than 1 day to setup and configure software at a site	YES
4.2	Require less than 1 FTE per 100 servers to manage	To be demonstrated
4.3	Be able to run for weeks unattended	To be demonstrated
4.4	Be able to be restarted and carry on with previous or ongoing jobs	YES
4.5	Require X interventions per month where X ~ 0 (related to #1.2)	PARTIAL. Some automated recovery for SC3.
4.6	Admins should be able to manage channels easily (pause, throttle, change params, …)	PARTIAL. Improvements for SC3.
4.7	System should be deployable on production setup (managed hardware, managed DB, …)	YES

Monitoring Requirements

#	Requirement Description	Current status
5.1	Bandwidth monitoring	NO. To do for SC3.
5.2	Channel performance monitoring	NO. To do for SC3.
5.3	Information for individual jobs	YES
5.4	Dynamic information on a monitoring web page.	PARTIAL. Improvements for SC3.
5.5	Information available as command-line tools	YES Improvements for SC3.

Scheduling Requirements

#	Requirement Description	Current status
6.1	Support simple FIFO scheduling	YES
6.2	More complex scheduling algorithms	NO. Not for SC3. (although it will be able to use srm-cp).
6.3	Allow for bandwidth quotas (MB/s , GB/day)	NO. Not for SC3.
6.4	Allow for per-VO bandwidth quotas	NO. Not for SC3.

User Interface Requirements

#	Requirement Description	Current status
7.1	Allow for user-level submission, monitoring, management and monitoring of transfer requests via command line.	YES Improvements for SC3.
7.2	Allow for user-level submission, monitoring, management and monitoring of transfer requests via a web-based GUI.	YES Improvements for SC3.
7.3	Allow for monitoring and management of the service via a GUI.	PARTIAL Improvements for SC3.