

The QCDGrid Software

Advanced Administration

- ▶ Administration of main QCDgrid systems
- ▶ Component architecture
 - Some 3rd party dependencies
 - Relevant implementation details
- ▶ Deployment information
 - Installation
 - Configuration

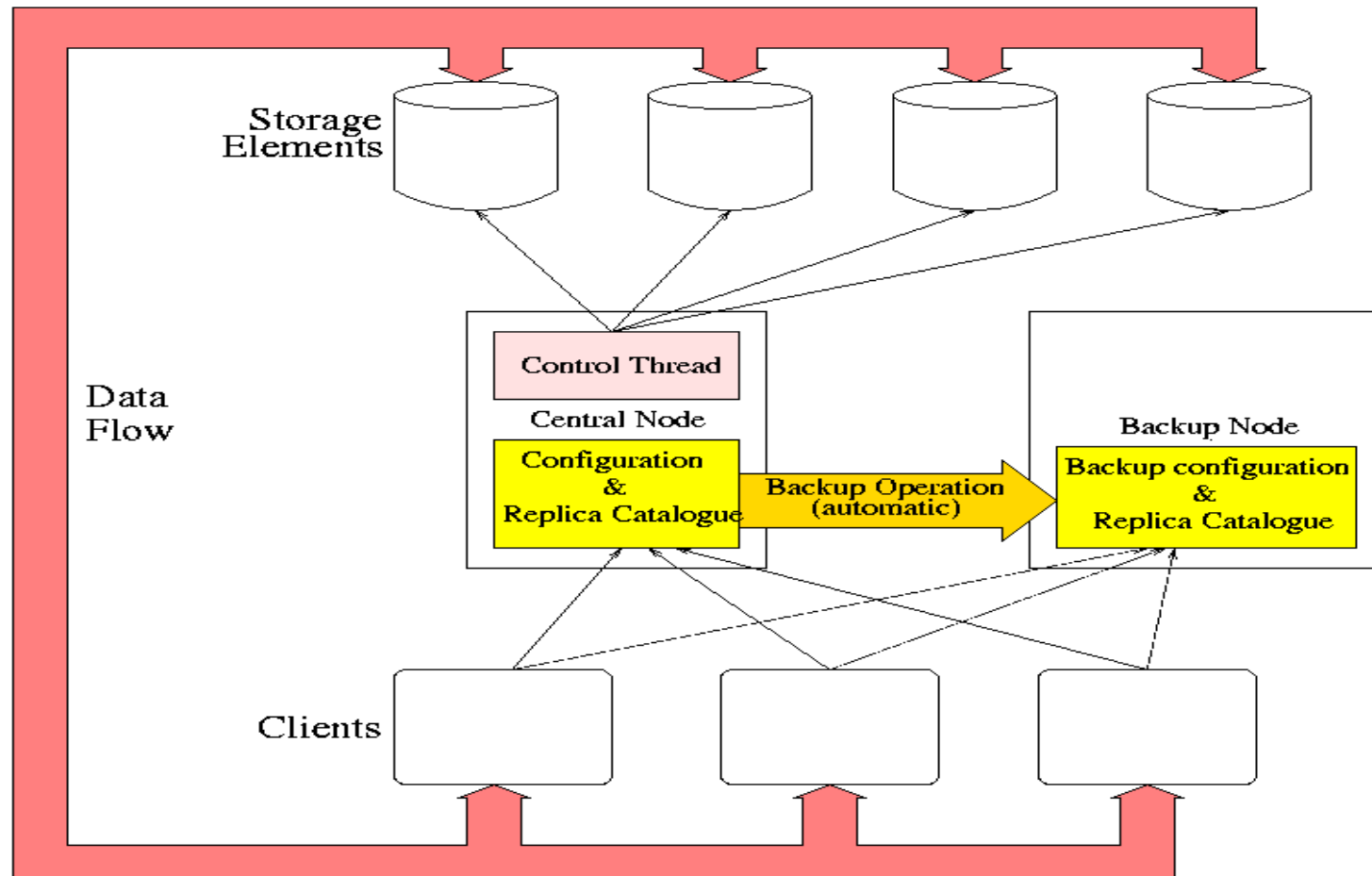
- ▶ Data grid
- ▶ Metadata catalogue and browser
- ▶ Job submission grid
 - Less used

Data grid administration

- ▶ Store binary data files
 - Result of QCD simulations, but potentially any files
 - Eventually store large numbers of these file
- ▶ Store files:
 - Reliably
 - Duplicates
 - Transparently
 - Files have a logical filename – that is all clients need to know
 - A distributed filesystem
 - Securely
- ▶ Retrieve files
- ▶ Implemented using a mix of C and Java
 - Originally pure C
 - Now some JNI bridging to C code

- ▶ In deployment there are three types of grid node:
 - Control node (sometimes called central node)
 - Single point of failure
 - Backup node
 - Adds some redundancy
 - Storage element node(s)
- ▶ Also client machines
 - May also be one of the above types, but can be standalone
- ▶ Control node controls entire system
 - Keeps track of file locations
- ▶ Backup node
 - In the event of failure of central node, allows read access to data grid
- ▶ Storage nodes
 - Provide storage space on local filesystem

- ▶ Access to nodes secured via Globus GSI
 - UK eScience certificates
 - gridmap-file entries
- ▶ It's possible to use EDG VO software to store central list of users
 - For ease of administration
 - List periodically updated on client machines
 - This is the configuration used on the UKQCD data grid
 - RPMs are no longer available to download, but QCDgrid can provide them



- ▶ Again preferable to have a dedicated `qcdgrid` account
- ▶ Download `QCDGRID_LINUX_CLIENT_STORAGE_BIN_1_X_Y` package
- ▶ Unzip in desired directory (usually `~qcdgrid`)
- ▶ If using a different architecture, can build from source
 - `make storagenode`
 - `make client (optional)`

- ▶ Certificate of central node must be in gridmap file
 - Central node must be able to run GRAM jobs on storage nodes
 - Obtain from central node/data grid administrator
 - Can be ignored if using EDG/pooled accounts
 - However the control node certificate **MUST** be mapped to an account that has read and write access to the data storage directories

- ▶ Globus ports must be open on firewall
 - 2811
 - 2119
 - `$GLOBUS_TCP_PORT_RANGE`

▶ Configuring data storage

- Choose storage location (e.g. `/raid/qcdgrid`)
- Ensure this directory is owned by the `qcdgrid` account
- Create a directory called `NEW` (again owned by the `qcdgrid` account, writable)
- Create link in `qcdgrid` home directory to this storage

```
ln -s /raid/qcdgrid data
```

▶ Can add multiple storage locations

- Don't need `NEW` directory
- Identified by `data`, `data1`, `data2` etc

▶ Upgrading

- Usually just unzip and build new version of the software

▶ Disabling for maintenance

- If you are a grid administrator, run
`disable-qcdgrid-node fqdn.of.node.ac.uk`
- Otherwise ask an administrator to do this for you
- Node is placed in a “disabled” state – cannot read or write but it is expected that the node will come back up at a later time
- Use `enable-qcdgrid-node` to re-enable a grid node

▶ Removing totally

- Use `retire-qcdgrid-node` to permanently remove a node (need to be a grid administrator)

▶ Coordinate data storage

- Maps logical file names to physical locations
 - Uses Globus RLS
- Ensures duplication of files i.e. two (or more) copies of each exists at all times
- Ensures consistency of data on the grid
 - i.e. verifies no corruption has occurred

▶ Keep track of storage nodes

- Manage disabled nodes (temporarily offline storage nodes)
- Manage node retirement (permanently removed storage nodes)

- ▶ Executable called background
 - Files `background.c`, `background-delete.c` and `background-new.c`
 - Run via `central_node.sh`
 - Does `grid-proxy-init`
 - Renews every 12 hours as proxy certificate runs out
- ▶ Uses Globus Replica Location Service (RLS) to map logical filenames (LFNs) to physical locations
- ▶ Connects to locally running RLS
 - Wrapped in `rls.c`
- ▶ Carries out node scanning/interaction by submitting executables to the nodes for execution
 - Via Globus GRAM
- ▶ Some functionality carried out via normal Globus protocols
 - E.g. disk space request

- ▶ Automatic events periodically
 - Checks messages – from client, storage nodes (via Globus Socket)
 - Checks all nodes alive (GRAM job)
 - Moves new files on the grid to their permanent home
 - Checks the disk space on each node
 - If low, alert administrator
 - Checks that two copies of each file it knows about are on grid
 - Replicates if necessary
 - Verifies some data files – checksum based comparison
 - Verifies the contents of the catalogue are actually there
- ▶ Direct response to requests
 - E.g. `qcdgrid-ping`

- ▶ Install Globus RLS (as discussed earlier)
 - Ensure it is running
- ▶ Ensure that the `mail` command works
 - For warnings etc.
- ▶ Create a `qcdgrid` user account
 - Not strictly necessary, can run as any user, but recommended
- ▶ Machine certificate is required
 - E.g. UK eScience CA issued machine certificate
- ▶ Download the latest version of the software from NeSCForge
 - <http://forge.nesc.ac.uk/projects/qcdgrid>
 - Available from “Files” tab
 - Download package tagged `QCDGRID_PLATFORM_CLIENT_STORAGE_1_N_N`
 - (precompiled binaries)
- ▶ Unzip in `qcdgrid` home directory

▶ Compiling from source

- Unzip distribution (all distributions have source code)
- Edit Makefile values
 - Globus location
 - Globus flavour (we use gcc32dbg)
 - Compiler
- Type 'make server'

▶ Some issues

- Need to have ALL Globus RLS libraries in `LD_LIBRARY_PATH` (client and server)
- Need to have all other Globus libraries in the `LD_LIBRARY_PATH`
- Limited success building for native 64-bit mode

- ▶ Five main configuration files
 - `qcdgrid.conf`
 - `mainodelist.conf`
 - `deadnodes.conf`
 - `disablednodes.conf`
 - `retiringnodes.conf`
- ▶ The last three of these should be created empty
 - Managed by the software itself
- ▶ All live in the `qcdgrid` home directory

▶ `qcdgrid.conf`

– Stores details that the control node needs to operate

- Ports used
- Number of replicated copies
- Disk panic thresholds
- Administrator details

▶ Typical file looks like

```
rc_port=3890
QCDGRID_port=51000
min_copies=2
disk_space_low=64000000
disk_space_panic=64000000
administrator=/Some/certificate/subject
administrator=/Some/other/subject
admin_email=qcdgrid@epcc.ed.ac.uk
```

- ▶ `mainnodelist.conf`
 - Stores details of the machines on the grid
 - Machine names
 - Disk free
 - Site
 - Path to software
 - Loaded by the client tools to figure out the state of the grid
- ▶ Typical entry

```
node=pytier2.swan.ac.uk
site=Swansea
disk=389111796
path=/home/qcdgrid
```

▶ Copy server certificate into `~qcdgrid/.globus`

▶ Ensure certificate file permissions are correct

```
-r--r--r-- qcdgrid qcdgrid      2397  usercert.pem  
-r----- qcdgrid qcdgrid      1863  userkey.pem
```

▶ Use the supplied `control_thread.sh` script to start the control thread

- Usually you'll want to redirect `stderr`, `stdout` to a log file
- Use the `--verbose` flag to enable logging

▶ Commonly run in the background

- ▶ Contents typically include debugging information, replication notices
- ▶ Can be used to see what control thread was doing if it fails
- ▶ Emails sent for major events
 - E.g. disc panic

- ▶ Keeps a current copy of the replica catalogue on the central node
 - Replication via `mysql replication`
- ▶ Run the secondary executable via the `backup-thread.sh` script
 - Copies the `mainodelist.conf` file over regularly

- ▶ Same install as for central node
 - i.e. `make server`
- ▶ Need to set up replication of database used by RLS (in this case MySQL)
 - Advanced topic beyond scope of this course
 - Full details are available from <http://dev.mysql.com/doc/refman/5.0/en/replication.html>
 - Further details available in the QCDgrid setup guide
- ▶ Start `backup-thread.sh`
 - Again, may wish to redirect output to a log file

▶ Submit file to grid

- `put-file-on-qcdgrid` obtains `mainnodelist.conf` from the control node
- Uses this file to get the amount of free space on each storage node
- Selects suitable node
- Gridftps the file to that node
- Sends message to control node
- Replication will take place under via the control node

▶ Get file from grid

- `get-file-from-qcdgrid` contacts central node to find out where copies of a file are stored
- Selects a node and gridftps file

Metadata catalogue administration

- ▶ Metadata in form of XML documents
- ▶ Stored in eXist database
- ▶ eXist runs as a web application in the Tomcat application server
 - Easy (in theory) to install
- ▶ Many client tools (not just browser) now interact directly with eXist server as well as control thread

▶ Install Tomcat

- Usually just a case of downloading from Apache site and unzipping
- Start with `$TOMCAT_HOME/bin/startup.sh`

▶ Install eXist

- We use version 0.91beta just now
- Obtain from QCDgrid team (no longer available)
- Unzip in `$TOMCAT_HOME/webapps`
- Restart Tomcat

▶ Generally, but not necessarily, runs on same machine as control thread

- ▶ Searches on DB via XQuery/XPath
 - Have to know how to write queries – bad for usability
- ▶ Metadata browser takes away this pain
 - GUI for entering search parameters
 - Creates query, dispatches, gets result
 - Allows user to download data file(s) – via the qcdgrid client software
- ▶ “Smart” browser – knows about metadata schema
 - Auto-generates GUI fields for search parameters
- ▶ Handled in packages:
 - `uk.ac.ed.epcc.qcdgrid.metadata` - queries
 - `uk.ac.ed.epcc.qcdgrid.datagrid` – wrapper to data grid client functionality

- ▶ `qcdgrid-job-submit` executable
- ▶ Implementation in `js/src` subdirectory
- ▶ Simple wrapper over Globus GRAM
- ▶ Designed for small post-processing jobs

- ▶ Globus 2.4
 - Installed via the VDT distribution
 - Very, very easy via PacMan:
 - `pacman -get VDT:VDT`
- ▶ GridPP VO system
 - Makes security easier
 - Centralised gridmap file reduces administrative burden
- ▶ Make from source, or copy binaries
- ▶ Configure depending on node type

- ▶ Same as for `qcdgrid` client software
- ▶ Java Virtual Machine