Experiences with a lightweight GRID infrastructure using AFS

Enabling Grids for E-sciencE

Roger Barlow
Alessandra Forti
Andrew McNab
Sabah Salih
Michael Salt

School of Physics and Astronomy
The University of Manchester

A user has three different ways of running their job on a remote grid cluster

1) Ensure that all the necessary software is installed in the remote environment
   Suitable solution for groups with a large software stack and large influence, e.g. ATLAS and CMS
2) Download everything that is or might be needed in a sandbox
   Can be inefficient if a lot of time is spent copying and anpacking files that may not be needed. Also liable to fail when an unusual outcome requires an unusual file that the user was not aware was needed
3) use afs and gssklog to run the job in their usual working directory
   The subject of this poster

Basic script submitted to the grid

1) Run gssklog.
2) cd to the working directory on the home computer, where the job has been debugged
3) Run the executable as usual

Actual use cases have a few variations on this basic template

afs is not a highly efficient I/O method. But if it is only required for reading control parameters, program files, etc, and if the output volume is low, then this does not matter.

Output can be written locally and staged back at the end, if efficiency is a problem.

gssklog provides a bridge between X509 authentication and kerberos, as used by afs.

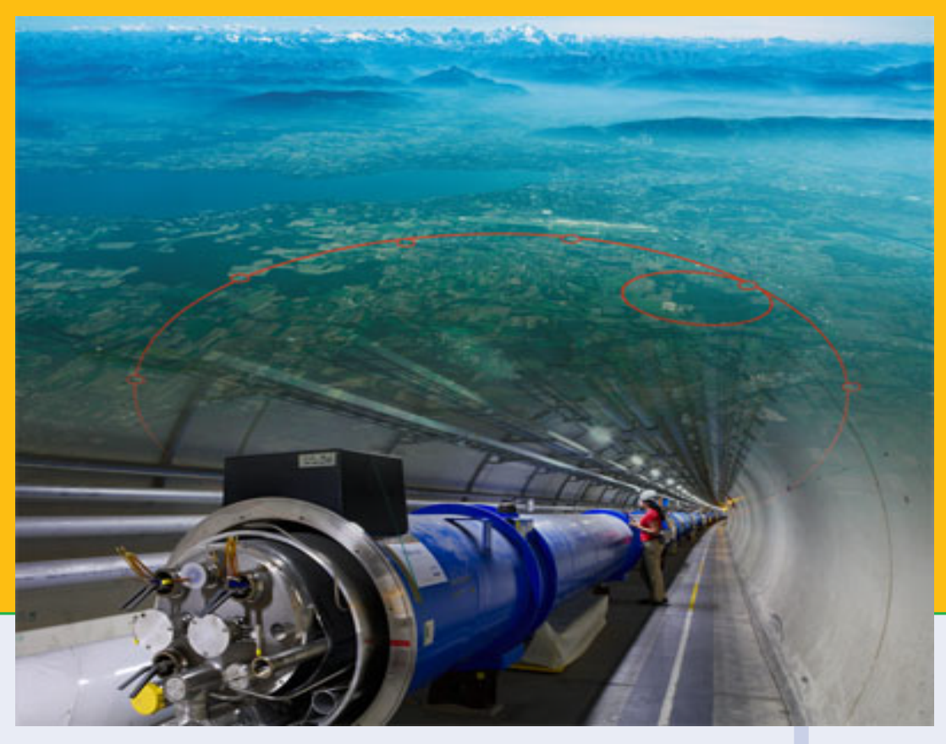
A job on a grid worker node is authenticated by a proxy with a user DN. gssklog informs the afs server that a connection is requested. The worker nodes must run afs client: many do

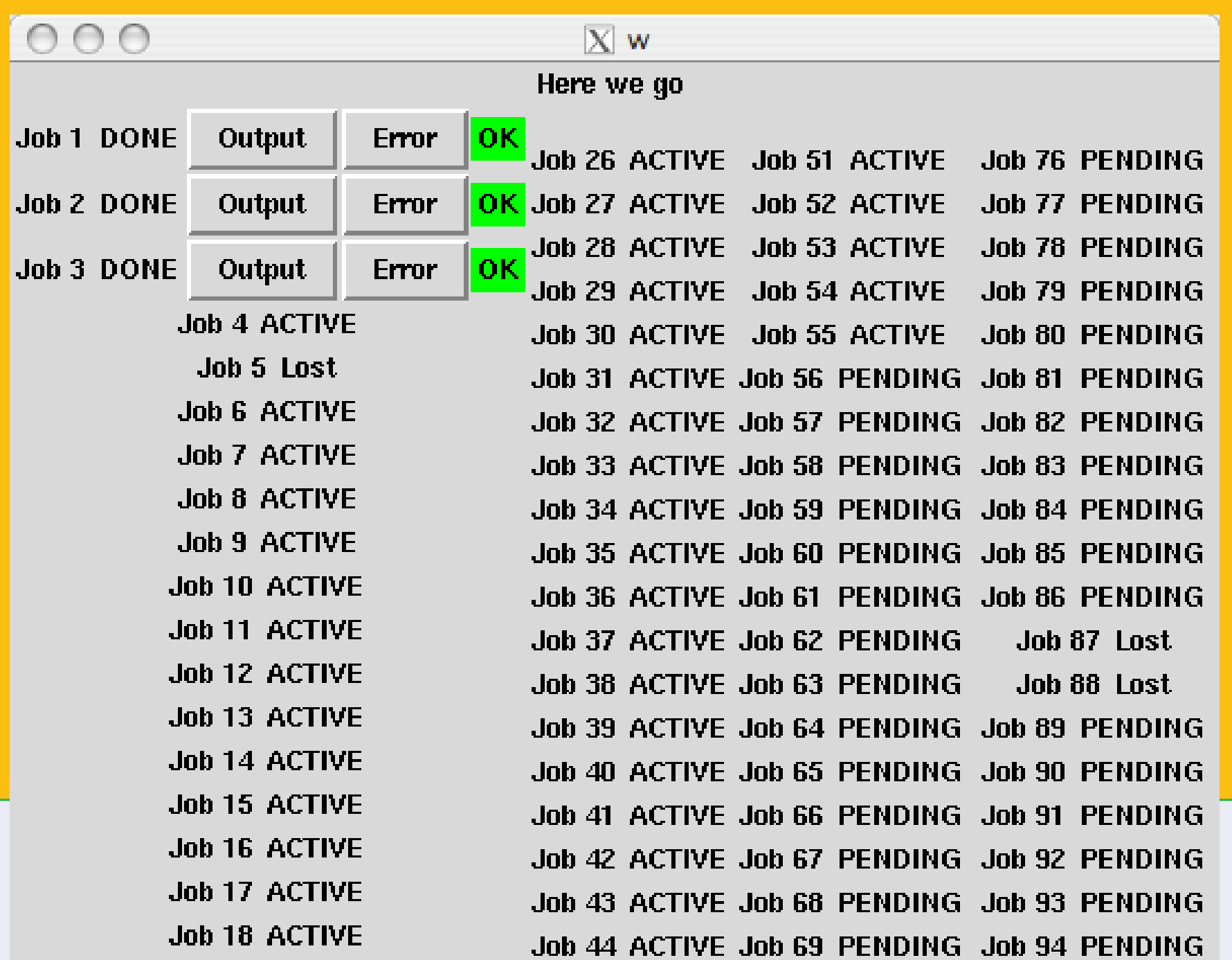
The local mapfile must be set up to map the user DN to the afs identifier. This one-off action is required on the user's machine, not the remote centre.

Use Case 1: MERLIN. Simulating the LHC collimation system (Program written by Nick Walker. See http://www.desy.de/~merlin)

Bunches of 100,000 particles tracked round the LHC accelerator for 200 turns, to study where the tails of the bunch hit the collimation system.
Requires ~2 hours. Good statistics for low-probability outcomes needs ~100 jobs
A simple tcl/tk is used for tracking progress

In this application a master script creates a subdirectory for each job, and results are subsequently merged





```
Here we go
Job 1  DONE  Output  Error  OK
Job 2  DONE  Output  Error  OK
Job 3  DONE  Output  Error  OK
Job 4  ACTIVE
Job 5  Lost
Job 6  ACTIVE
Job 7  ACTIVE
Job 8  ACTIVE
Job 9  ACTIVE
Job 10  ACTIVE
Job 11  ACTIVE
Job 12  ACTIVE
Job 13  ACTIVE
Job 14  ACTIVE
Job 15  ACTIVE
Job 16  ACTIVE
Job 17  ACTIVE
Job 18  ACTIVE

Job 26  ACTIVE   Job 51  ACTIVE    Job 76  PENDING
Job 27  ACTIVE   Job 52  ACTIVE    Job 77  PENDING
Job 28  ACTIVE   Job 53  ACTIVE    Job 78  PENDING
Job 29  ACTIVE   Job 54  ACTIVE    Job 79  PENDING
Job 30  ACTIVE   Job 55  ACTIVE    Job 80  PENDING
Job 31  ACTIVE   Job 56  PENDING   Job 81  PENDING
Job 32  ACTIVE   Job 57  PENDING   Job 82  PENDING
Job 33  ACTIVE   Job 58  PENDING   Job 83  PENDING
Job 34  ACTIVE   Job 59  PENDING   Job 84  PENDING
Job 35  ACTIVE   Job 60  PENDING   Job 85  PENDING
Job 36  ACTIVE   Job 61  PENDING   Job 86  PENDING
Job 37  ACTIVE   Job 62  PENDING   Job 87  Lost
Job 38  ACTIVE   Job 63  PENDING   Job 88  Lost
Job 39  ACTIVE   Job 64  PENDING   Job 89  PENDING
Job 40  ACTIVE   Job 65  PENDING   Job 90  PENDING
Job 41  ACTIVE   Job 66  PENDING   Job 91  PENDING
Job 42  ACTIVE   Job 67  PENDING   Job 92  PENDING
Job 43  ACTIVE   Job 68  PENDING   Job 93  PENDING
Job 44  ACTIVE   Job 69  PENDING   Job 94  PENDING
```

Environment variables set by the master script

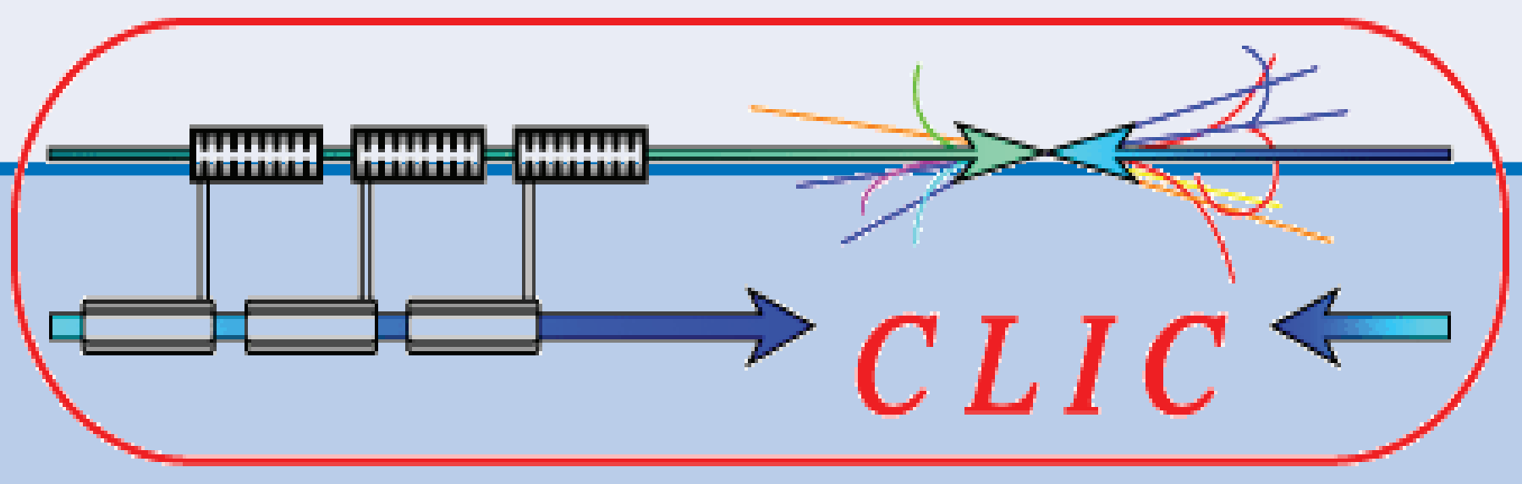
```
/usr/bin/gssklog -cell hep.man.ac.uk -principal $afsname -server afs1.hep.man.ac.uk -port 5750
cd $homedir/$DIR
export ROOTSYS=../../myRoot/root
export LD_LIBRARY_PATH=../../../lib:../../myRoot/root/lib:$LD_LIBRARY_PATH
ln -s ../../LHCB19.tfs LHCB19.tfs
ln -s ../../../Merlin/MerlinExamples/Wakefields/Data Data
../../example411 $iseed >run.out >run.err
```

In the interactive login script, but needs to be done explicitly in batch

Set by the master script to ensure each job is different

The program expects these in its local directory. As subdirectories are being used for each job they need to be set uxplicitly

Use Case: BDSIM Simulating Electromagnetic Shower Evolution in the CLIC Extraction Line for Background Estimations



Due to the low probability of a backscattered photon incident on the interaction point, it is necessary to track 3,000,000 post collision particles to provide sufficient statistics for estimating the flux. BDSIM uses GEANT4 to model the interaction of particles with masks, magnets and dumps in the extraction line. Even with leading particle biasing a job life of 7 days is needed, achievable on the GRID through delegated proxies, allowing for large batches to be prepared in advance and processed in one job. To achieve this, the input and output files were split, and a script written to recombine the output files.

```
First of all,  initialise myProxy, with a 7 days lifetime...
myproxy-init –n –d
Then generate a VOMS proxy...
voms-proxy-init –voms vo.northgrid.ac.uk
Delegate the 7 day proxy, with the identifier 'mick'
glite-wms-job-delegate-proxy –d mick
Submit the job, using delegated proxy (-d option), appending the job identifier to the file 'jobid' to keep track
glite-wms-job-submit -d mick -o jobid --config autowms.conf -r ce02.tier2.hep.manchester.ac.uk:2119/jobmanager-lcgpbs-long bdsim-test-northgrid4.jdl
```

```
/usr/bin/gssklog -cell hep.man.ac.uk -principal mick -server afs1.hep.man.ac.uk -port 5750
/bin/echo -e "\nthe exit status of the gssklog is $?"
#
export BDSIM_BASE_DIR=/afs/hep.man.ac.uk/u/mick/Programs/BDSIM
export ROOTSYS=/afs/hep.man.ac.uk/u/mick/Programs
export CLHEP_BASE_DIR=/afs/hep.man.ac.uk/u/mick/Programs
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$BDSIM_BASE_DIR/parser:$ROOTSYS/lib/root:$CLHEP_BASE_DIR/lib
export PATH=$PATH:$BDSIM_BASE_DIR/bin/Linux-g++
#
cp /afs/hep.man.ac.uk/u/mick/tointdump/coll12_geomlist.sql .
#... several other copies done here
cp /afs/hep.man.ac.uk/u/mick/tointdump/cohminus_2708.dat .
#
bdsim --batch --file=run_cohminus_2708.gmad --output=root --outfile=output_2708 > log_2708.txt
/usr/bin/gssklog -cell hep.man.ac.uk -principal mick -server afs1.hep.man.ac.uk -port 5750
cp output_2708_0.root /afs/hep.man.ac.uk/u/mick/tointdump/
cp log_2708.txt /afs/hep.man.ac.uk/u/mick/tointdump/
#
cp /afs/hep.man.ac.uk/u/mick/tointdump/run_cohminus_2707.gmad .
cp /afs/hep.man.ac.uk/u/mick/tointdump/cohminus_2707.dat .
bdsim --batch --file=run_cohminus_2707.gmad --output=root --outfile=output_2707 > log_2707.txt
/usr/bin/gssklog -cell hep.man.ac.uk -principal mick -server afs1.hep.man.ac.uk -port 5750
cp output_2707_0.root /afs/hep.man.ac.uk/u/mick/tointdump/
cp log_2707.txt /afs/hep.man.ac.uk/u/mick/tointdump/
```

initialisation

BDSIM runs for 2 sets of input in this job

BDSIM runs for days. The kerberos token has to be renewed

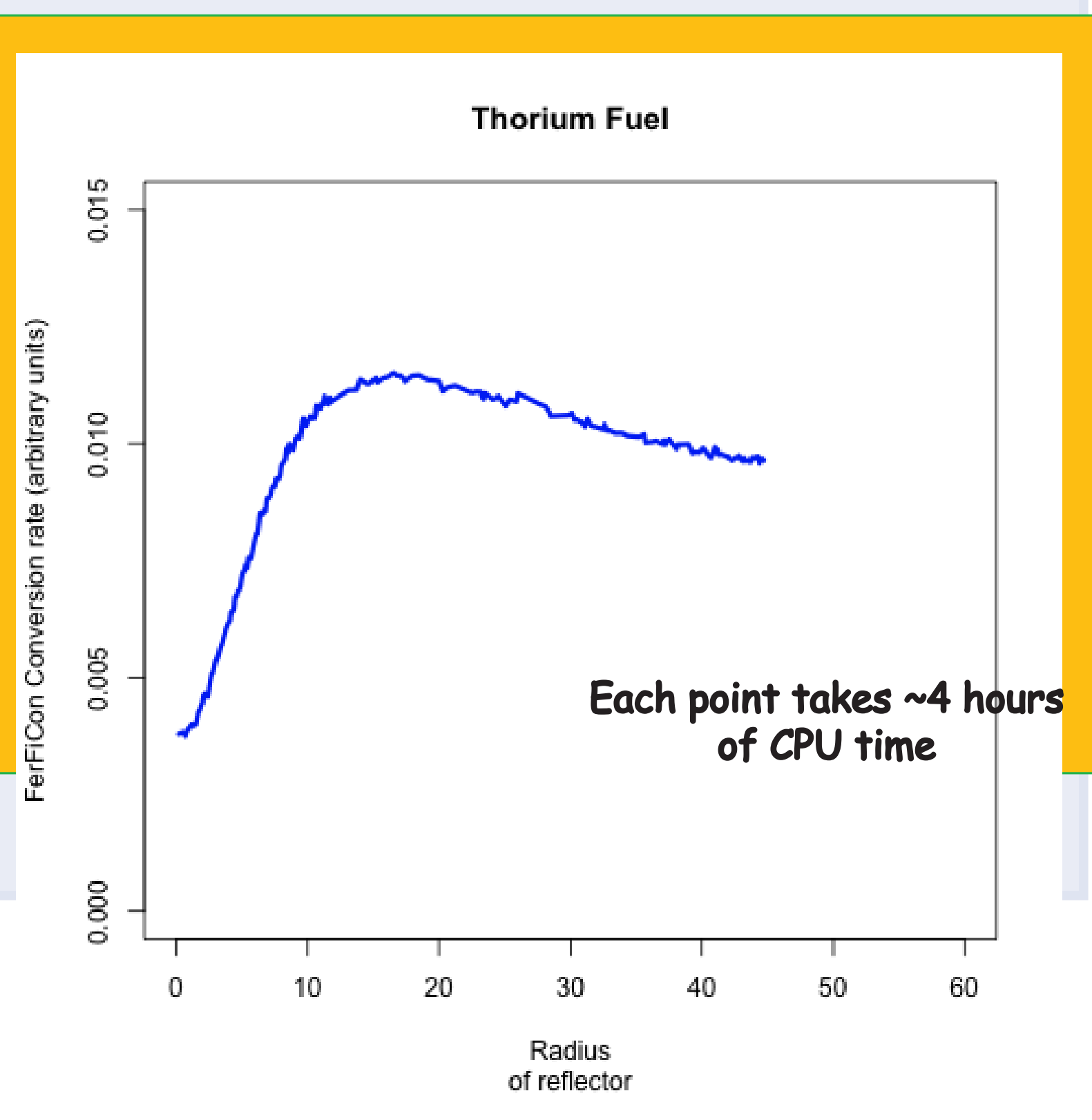
Use Case: MCNP: converting ^{232}Thorium to ^{233}Uranium

Fertile Thorium can be converted to fissile Uranium, and used to power nuclear reactors. As part of the process we simulate a Thorium fuel rod in a heavy water moderator, surrounded by a lead reflector, and we need to optimise the radius of the relector. Each point requires ~100000 neutron cascades, which takes several hours.
We submit 500 jobs, scanning distances up to 50 cm in mm steps
The conversion probability appears as a number written in a standard text file: a simple script combines the results.
Another script checks for missing outputs and resubmits jobs for them

Separate subdirectory per job as in Case 1

```
gssklog -cell hep.man.ac.uk -principal $afsname -server afs1.hep.man.ac.uk -port 5750
cd $homedir/$DIR
echo Spallation and rod
RAD=`echo .1*$iseed| bc`
expot RAD
cat >>temp <<EOF
Spallation in fuel rod
...
 1 CZ 0.5
2 CZ  $RAD
3 PZ -200
4 PZ 200
 ...
EOF
 export DATAPATH=/afs/hep.man.ac.uk/g/accelerators/sw/MCNP/MCNP_DATA
/afs/hep.man.ac.uk/g/accelerators/sw/MCNP/MCNP5/bin/Linux/mcnp5_i386 n=temp
```

Job sequence number divided by 10 to get reflector radius



Each point takes ~4 hours of CPU time

The EGEE project is building a Grid infrastructure for the scientific community. Grids are networks of computers spread across many sites but able to act together to provide a range of large scale facilities, from incredible processing power and mass storage to a platform for international collaboration.

www.eu-egee.org

e-infrastructure

EGEE-III is co-funded by the European Commission under contract number INFSO-RI-222667