Job Submission to Grid Computing Environments

Richard Bruin
eMinerals project
Departmen of Earth Sciences
University of Cambridge
rbru03@esc.cam.ac.uk

NATURAL
ENVIRONMENT
RESEARCH COUNCIL

eMinerals

Outline..

□ eMinerals project background

□ Submission tools and their requirements

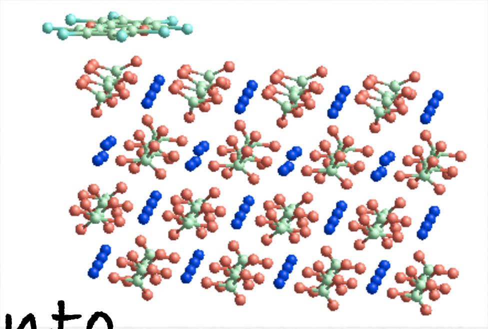
□ my_condor_submit

□ Parameter sweeps (ensemble studies)

The eMinerals project

- Fairly large, NERC funded project
  - 6 Institutions
  - 30 staff - PhD through to Professors
- Wide ranging research interests
  - Scientific modelling
  - ...
  - Grid computing

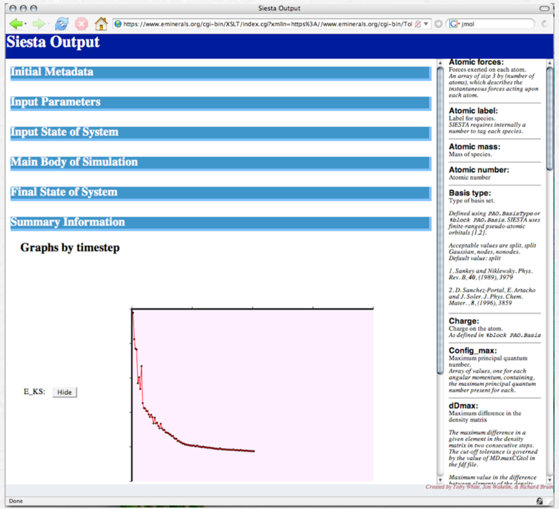
eMinerals Science Research



- □ Pollutants and their adsorption onto minerals
- □ Water (and its influence on adsorption)
- □ Radiation damage / waste containment

eMinerals grid research

- □ Building and configuring grids
- □ Job submission tools
- □ Data management
- □ Metadata management
- □ Data processing / Information extraction
- □ Simulation output visualisation



Siesta Output

https://www.eminerals.org/cgi-bin/XSLT/index.cgi?xmlIn=https%3A//www.eminerals.org/cgi-bin/Tol

jmol

Siesta Output

Initial Metadata

Input Parameters

Input State of System

Main Body of Simulation

Final State of System

Summary Information

Graphs by timestep

Atomic forces:
Forces exerted on each atom.
An array of size 3 by (number of atoms), which describes the instantaneous forces acting upon each atom.

Atomic label:
Label for species.
SIESTA requires internally a number to tag each species.

Atomic mass:
Mass of species.

Atomic number:
Atomic number

Basis type:
Type of basis set.

Defined using PAO.BasisType or #block PAO.Basis. SIESTA uses finite-ranged pseudo-atomic orbitals [1,2].

Acceptable values are split, split Gaussian, nodes, nonodes. Default value: split

*1. Sankey and Niklewsky. Phys. Rev. B, **40**, (1989), 3979*

2. D. Sanchez-Portal, E. Artacho and J. Soler. J. Phys. Chem. Mater., 8, (1996), 3859

Charge:
Charge on the atom.
As defined in #block PAO.Basis

Config_max:
Maximum principal quantum number.
Array of values, one for each angular momentum, containing, the maximum principal quantum number present for each.

dDmax:
Maximum difference in the density matrix

The maximum difference in a given element in the density matrix in two consecutive steps. The cut-off tolerance is governed by the value of MD.maxCGtol in the fdf file.

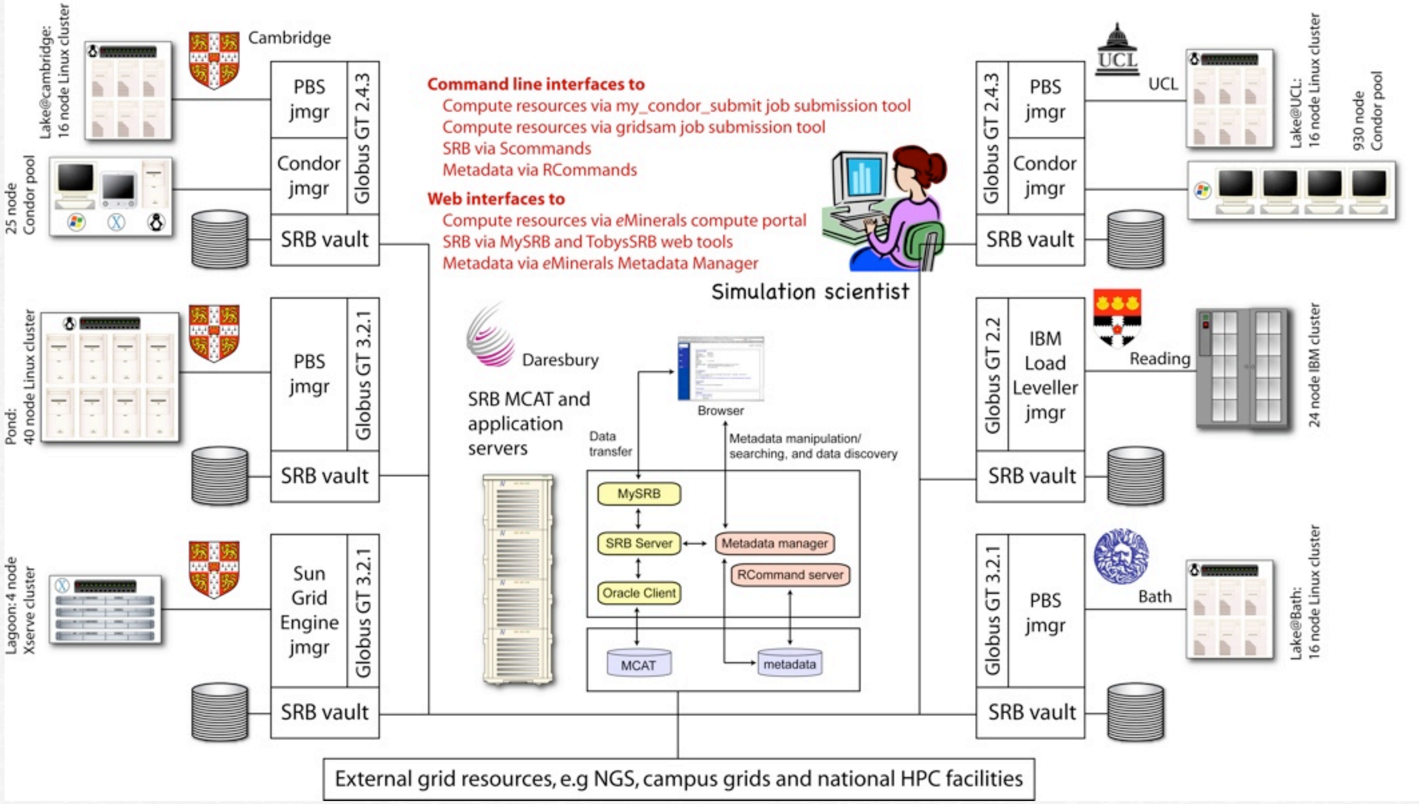
Maximum value in the difference

Created by Toby White, Jon Wakelin, & Richard Bruin

Done

The eMinerals minigrid

- □ A prototype heterogeneous, integrated grid infrastructure



Command line interfaces to

Compute resources via my_condor_submit job submission tool
Compute resources via gridsam job submission tool
SRB via Scommands
Metadata via RCommands

Web interfaces to

Compute resources via eMinerals compute portal
SRB via MySRB and TobysSRB web tools
Metadata via eMinerals Metadata Manager

Job submission tools

- Standard tool requirements:
  - Simple to use
  - Non-intrusive to the user
  - Allow the user to do what they want
- Grid tool requirements:
  - Appropriate data and metadata handling
  - Ability to metaschedule across any resource
  - Automated as far as possible

my_condor_submit (MCS)

□ Single job submission per invocation

□ condor_g style interface

□ Metascheduling across any Globus resource

□ Metadata storage (RCommands)

□ Information extraction (AgentX)

□ Data handling / archiving (SRB)

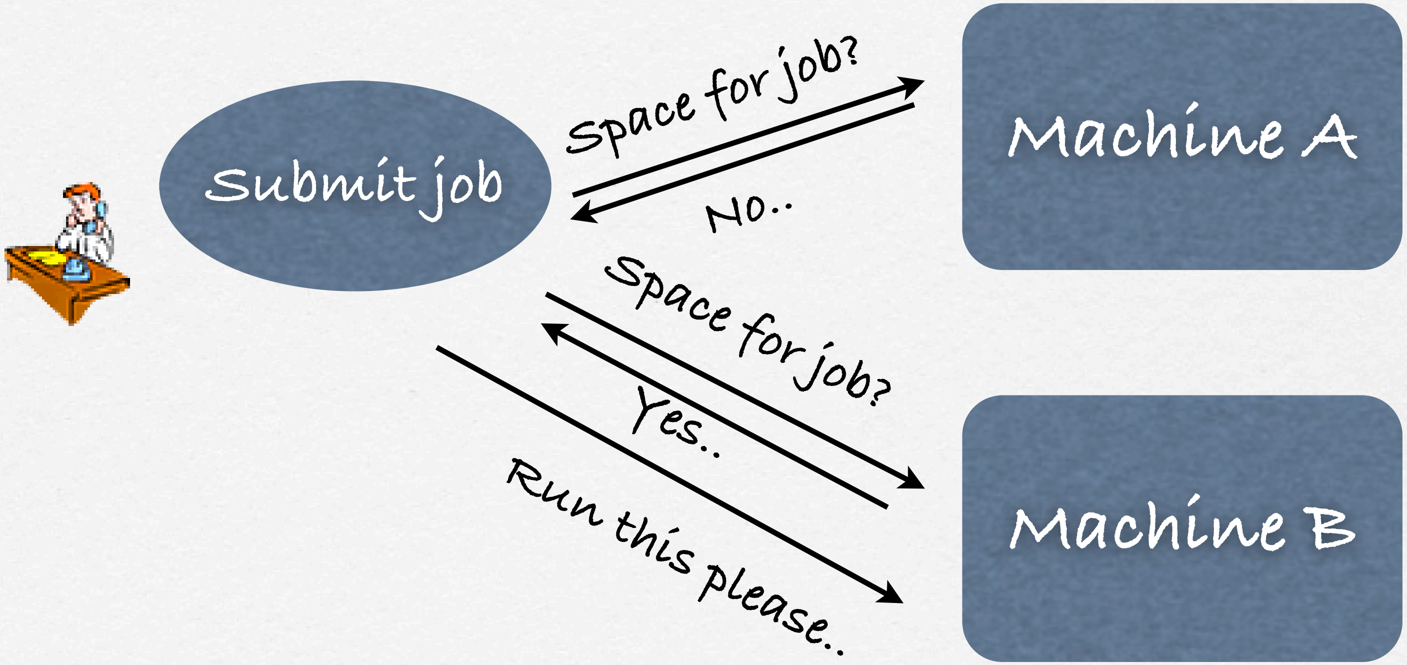
MCS input file

```
# Specify the name of the executable to run
Executable      = gulp

# Specify where the executable should get stdin from and put stdout to
GlobusRSL = (stdin=andalusite.dat)(stdout=andalusite.out)

# Specify an SRB collection to get the relevant executable from
pathToExe        = /home/codes.eminerals/gulp/

# Specify a metadata dataset to create all metadata within
RDatasetId   = 55

# Specify a directory to get files from, put files to and relate to
# metadata created below
Sdir     = /home/user01.eminerals/gulpminerals/
Sget     = *
Sput     = *
# Creates and names a metadata data object
Rdesc    = "Gulp output from andalusite at ambient conditions"
# Specify metadata to get from files with Agent-x - get environment
# and default metadata only
AgentXDefault = andalusite.xml
GetEnvMetadata= True
```

MCS metascheduling

- Relatively simple round-robin algorithm
- Allows user to limit machines to schedule across
- Supports different architectures (including multi-processor and multi-core machines)
- Supports serial and parallel jobs
- Automatic load balancing across all resources

MCS metascheduling cont.



Submit job

Space for job?

No..

Machine A

Space for job?

Yes..

Run this please..

Machine B

MCS job execution workflow

- Three stages, handled by Condor DAGman:
  - Pre script: Stage in executable and data from the SRB
  - Run job
  - Post script: Stage out data, collect and store metadata

MCS data handling

- Data staged in and out from the SRB
  - Transfers to / from any number of collections
  - Support for wildcard file specifications
  - Use of recursion allowed

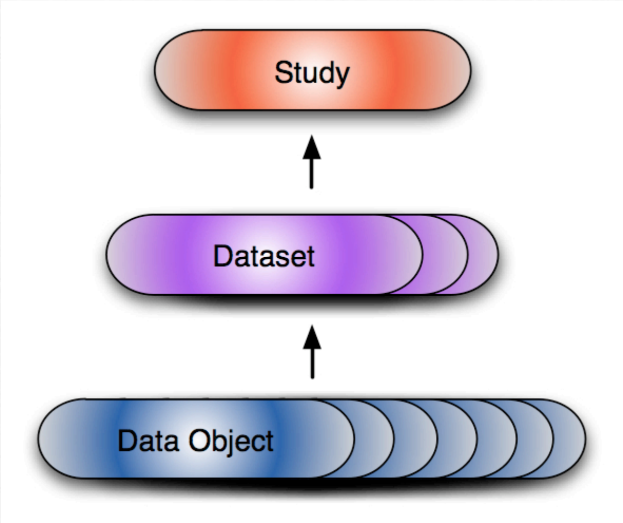
MCS information extraction

- Using AgentX
    - Ontology based system - logical querying, hiding filesystem structure
    - User specifies simple XPath-like query:

        -
          ```
          AgentX = finalEnergy,
          chlorobenzene.xml:/Module[last]/
          PropertyList[title = ‘Final Energy’]/
          Property[dictRef = ‘siesta:Etot’]
          ```

MCS metadata storage

- □ uses RCommands for storage and structuring
  - □ Subset of the CCLRC Metadata model
  - □ Simple binary command line web service clients
- □ Three types of metadata collected:
  - □ 'Environment' metadata
  - □ 'Default' metadata
  - □ 'User specified' metadata



MCS supported machines

- Currently tested list:
    - Each type of eMinerals minigrid machine (PBS, SGE, Condor, Loadleveler)
    - NGS core nodes (PBS)
    - NW-Grid clusters (SGE)
    - ...
- Basically anything with Globus installed!

Parameter sweeps

- MCS designed for single job per invocation, need something to handle large ensemble runs. Including:
  - Simulation code input file creation
  - Submission tool input file creation
  - Any necessary data staging
  - Simple manner to submit this many jobs
  - Ways to manage and monitor these jobs
  - Ways to collate and view the output from these jobs

Job creation

- Single command and a config file
  - Specify:
    - string to find / replace in template input file
    - Start and end values for sweep
    - Number of steps in between
  - Input files created and uploaded to the SRB
  - MCS input files created for later use

Job submission

- Single command:
  - Walks through created jobs
  - Submits each found job using MCS
  - Keeps track of job directories and IDs for monitoring tools
  - Commands to check that all jobs submitted and resubmit any failed submissions

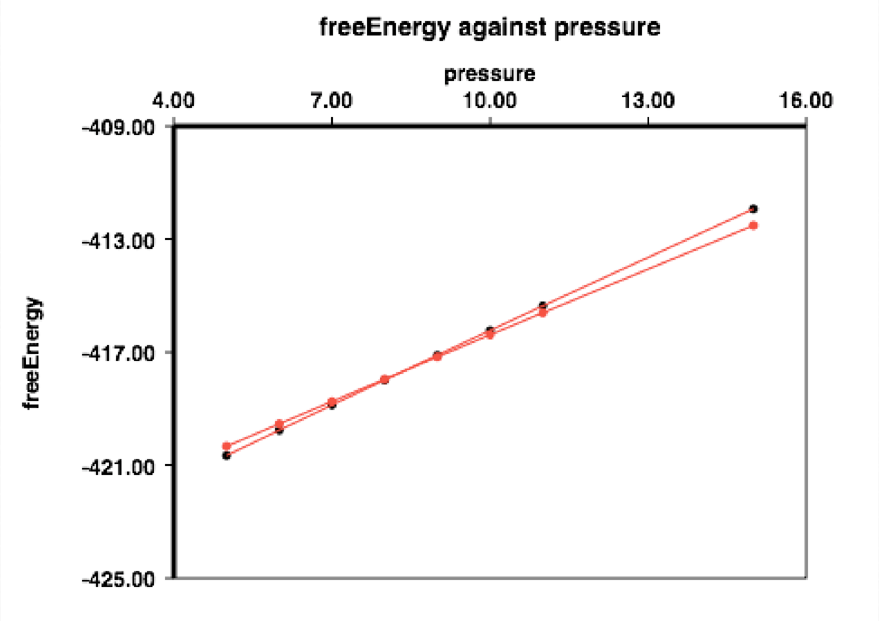
Job monitoring

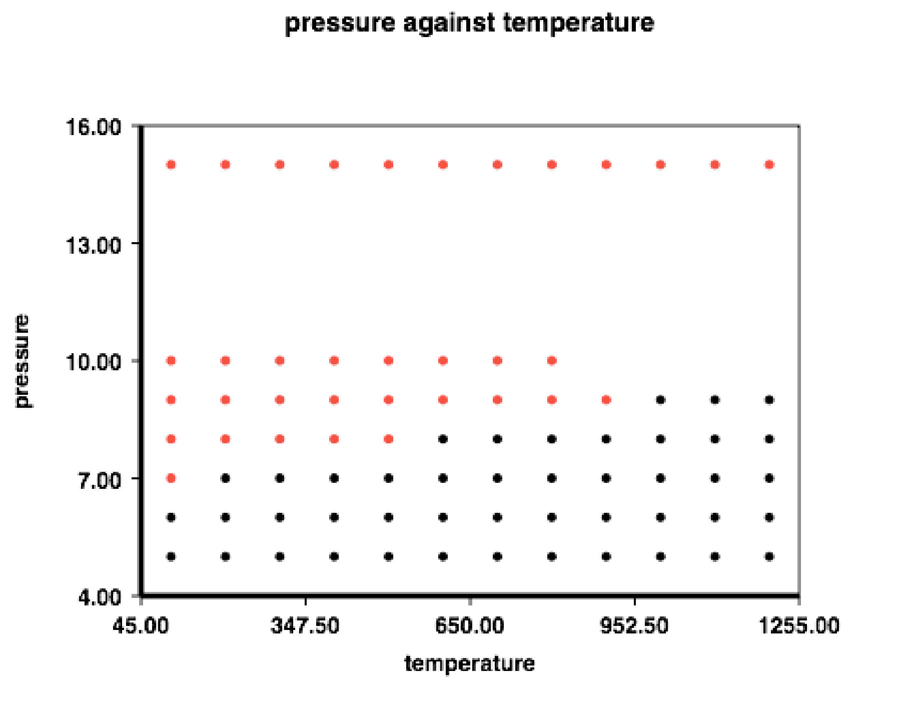
- Standard condor_q command to see what's running
- Additional command checks whole set of jobs, informing user if any still running

Processing sweep output

- User provides, entries in configuration file and name of files to process.
- Our simulation codes use CML, this means we can, with one command:
    - Combine the files together
    - Extract relevant information from them
    - Translate the CML into SVG, drawing pretty graphs

Pretty graphs...





Conclusion...

- Example uses given in the paper (no time to show here I’m afraid)
- Any questions?