Submitting and managing distributed computations
The researcher's interface to a BOINC project

Toni Giorgino, I. Buch, K. Sadiq, M. J. Harvey, G. De Fabritiis

Computational Biochemistry and Biophysics Lab / GRIB-IMIM
Barcelona Biomedical Research Parc (PRBB)
www.gpugrid.net

Thursday, 22 October 2009
GPUGRID.net - or -
Remotely using BOINC as a "virtual supercomputer"
for molecular simulations
Our aim: bridge the “scale gap” between biological models and the most detailed atomistic representations available
- One of the first BOINC projects to exploit accelerated processors
- Highly optimized CUDA-only application

Each work-unit simulates $O(1\text{ns})$
- Requires parameters, initial state, simulation protocol, etc.
- Returns final state, logs, etc.
- ~5 PFLOP/WU, 8h, 30 MB
Workunit turnaround
Submission, in practice - or -
Where do these WUs come from?
BOINC as a *virtual supercomputer*: overview

- We have *scientists* to create work-units
  - They do not access the server directly
  - Not necessarily BOINC- or even coding- experts
  - Could be at remote sites

- **Unrestricted access to server is ruled out**

- A *remote boinc client* allows them to
  - *submit* their computations
  - *retrieve* results
  - *manage* and *monitor* progress of WUs
Remote scientist interface: overview

Alice’s PC

Rboinc client

Input files

Remote-enabled BOINC server

Temporary storage

N steps

create_work

Clients

Submit

Input files

Submit

N steps
Remote scientist interface: overview

Remote-enabled BOINC server

Alice’s PC
- Rboinc client
  - Input files
  - Result files

Otto’s PC

Clients
- create_work

N steps

Submit
Retrieve
Submission, in practice

- Alice can start an hypothetical simulation with this command line

```
rboinc_submit
 -url http://big.edu -auth ...
 -app md
 -help_parameters
```
Submission, in practice

boinc_submit.pl \
  -url http://www.ps3grid.net:8383/rboinc_cgi \
  -app meta -help_parameters

Remote application queue `meta'
Description: Standard ACEMD run with optional DCD and PLUMED
Application on server: `acemd'

Options defined for this application queue:
  -pdb_file    PDB structure
  -coor_file   Binary coordinates
  -hills_file  (optional) PLUMED metadynamics restart
  -idx_file    (optional) (undocumented)
  -metainp_file (optional) PLUMED metadynamics configuration
  -conf_file   ACEMD input file
  -par_file    CHARMM parameters
  -psf_file    PSF topology
  -vel_file    Binary velocities
Alice can start an hypothetical simulation with this command line:

```
rboinc_submit
  -group AQUAPORIN -name SEQ1
  -url http://big.edu -auth ...
  -app md -num_steps 20 -metadata ...
  -pdb_file aquaporin.pdb
  -coor_file initial.coor [...]
```
A look behind the scenes

**Rboinc client**
- Initial option checking
- Parse application-specific options
- Upload files
- Show outcome

**Rboinc server**
- Authenticate
- Fetch application’s template files
- Pool received files
- Start workunit
- HTTP

create_work
Retrieving, in practice

- Now Alice wants to download the results computed so far
  - Files will be removed from the server

```bash
rboinc_retrieve
  -group AQUAPORIN [-name SEQ1]
  -url http://abigserver.edu -auth ...
```
Monitoring

- Each WU is associated with a *scientist* and a *group*
- Useful for accounting and error-checking
  - How many credits did I “consume”? 
  - Faulty WUs can be traced back
- Reports on demand
  - Which WUs am I running?
- Nightly reports
  - Who is running, how much, and at what error rate?
## Nightly report: example

### Per-user summary

<table>
<thead>
<tr>
<th>scientist</th>
<th>sent</th>
<th>unsent</th>
<th>day_suc</th>
<th>day_unsuc</th>
<th>day_credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALICE</td>
<td>1527</td>
<td>1453</td>
<td>868</td>
<td>336</td>
<td>5134196</td>
</tr>
<tr>
<td>BOB</td>
<td>2305</td>
<td>447</td>
<td>1466</td>
<td>390</td>
<td>7704984</td>
</tr>
</tbody>
</table>

### Per-user per-error

<table>
<thead>
<tr>
<th>Code</th>
<th>Error</th>
<th>ALICE</th>
<th>BOB</th>
<th>[...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-177</td>
<td>RSC_LIMIT_EXCEEDED</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-185</td>
<td>RESULT_START</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-186</td>
<td>RESULT_DOWNLOAD</td>
<td>12</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>-187</td>
<td>RESULT_UPLOAD</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-197</td>
<td>ABORTED_VIA_GUI</td>
<td>17</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>-198</td>
<td>INSUFFICIENT_RESOURCE</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-226</td>
<td>TOO_MANY_EXITS</td>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>-233</td>
<td>UNSTARTED_LATE</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Exit code by app)</td>
<td>151</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>37</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other (Any other exit code)</td>
<td>111</td>
<td>130</td>
<td></td>
</tr>
</tbody>
</table>

### Per-group summary

<table>
<thead>
<tr>
<th>group_name</th>
<th>sent</th>
<th>unsent</th>
<th>day_suc</th>
<th>day_unsuc</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALICE_BIND_11</td>
<td>81</td>
<td>20</td>
<td>51</td>
<td>21</td>
</tr>
<tr>
<td>ALICE_BIND_166_119</td>
<td>422</td>
<td>81</td>
<td>266</td>
<td>94</td>
</tr>
<tr>
<td>BOB_TEST_2</td>
<td>83</td>
<td>16</td>
<td>51</td>
<td>20</td>
</tr>
</tbody>
</table>

[...]
Remoting files -or- the remote application description
Chains as dependent steps

Initial state → Parameters → Step 1 → Out 1 → In 2 → Step 2 → Out 2 → In N → Step N → Out N

Result 1 → "Chaining" → Result 2 → ... → Result N

One work-unit
Let’s focus on one step: the application description

- Arrows are the file interface of a WU
- We have to expose input and output files over the network to Alice’s PC
- Akin to staging
- Accomplished cleanly, extending BOINC templates
Extension of *wu_template*

- *Input files* are enabled by just adding the *rboinc* tag

```xml
<file_ref>
    <file_number>5</file_number>
    <open_name>structure.pdb</open_name>
    <rboinc parameter_name="pdb_file" parameter_description="PDB structure"/>
</file_ref>
```

[...]
Extension of *result_template*

- *Outputs* are enabled similarly, and optionally chained with the *chain* attribute.

```xml
  [...]  

  <file_info>
    <name><OUTFILE_1/></name>
    <generated_locally/>
    <upload_when_present/>
    <max_nbytes>50000000</max_nbytes>
    <url><UPLOAD_URL/></url>
    <rboinc aliases=". coor" chain="2"/>
  </file_info>

  [...]  
```
File pooling

- Usually, simulations share a lot of files
- We “pool” files within the same group
  - Equal files are reused
  - All files are automatically indexed by content: pooling is transparent for users
  - Saves $O(1000)$ disk space
  - Files in upload and download directories are also linked to the pool, rather than copied
Clean storage layout

- On the server, groups stay organized in a natural hierarchy

```
Workflow/
 ALICE_GRAMICIDIN/
  SEQ1/
   (inputs for seq1)
  SEQ2/
   (inputs for seq2)
 pool/
   (pooled file storage)
 (results)
 process*
 ALICE_AQUAPORIN/...
```
Load balancing

- Sometimes, one is willing sequences to proceed *on par*
- Load balancing raises the priority of “late” steps when they are generated
Job layout -or- example of gpugrid.net at work
**Group**: an homogeneous set of simulations

- A scientist has administration rights on the *groups* he/she submitted
  - Create, then
  - Stop/delete them
  - Retrieve results
  - Be accounted for credits consumed…
  - …and errors
Computation chains

- Each group contains one or more computation chains
- Each chain consists of an ordered sequence of steps
- Steps are dependent: output of one is required to start the next
Example: one group, several chains

Same system in multiple conditions
Computation chains

- Users can manage computations as groups or specific chains

- Group attributes:
  - Submitter, application description, pooled storage, load balancing, ...

- Per-chain attributes:
  - Metadata, priority, resource limits, ...
Status & outlook
Conclusions

- RBoinc provides a *generic, simple* and *efficient* remote interface for BOINC projects
  - Researchers become its *users*
  - Any BOINC project can be exposed as a “service” or even a “virtual supercomputer”
- Implementation stable and virtually complete
  - Currently used for **ALL** of gpugrid.net’s work
- To do
  - Test-run WUs before sending into the wild
  - Test it outside the group