BOINC extensions in the SZTAKI DesktopGrid system

József Kovács
smith@sztaki.hu

BOINC Workshop, Grenoble, 10/09/2008
SZTAKI Desktop Grid: BOINC project

Number of users: 22947  Number of hosts: 31004
Active hosts in last 48 hours: 1479
Estimated performance of last 48 hours: 755,531 GFlop/s  Peak performance: 1.5 TFlop/s
Workunits processed in last 48 hours: N/A

Join SZTAKI Desktop Grid
Rules and policies [read this first]
Getting started
More information on desktopgrid.hu
Wikipedia article about SZTAKI Desktop Grid
Frequently Asked Questions (FAQ)
Create account
Applications
Returning participants
Your account - view state, modify preferences

Application current
SZTAKI Desktop Grid project systems
Description on the SZTAKI Desktop Grid project

User of the day
Ariana L.

BOINC Manager
Choose a project
To choose a project, click its name or type its URL below:
World Community Grid
Malawi Strange
Project@Home
Project@Home
GPU Grid - ISGB Grid
SuperK@Technion
The Lattice Project
Rosetta@Home
Nowruz@Home

BOINC project URL: 

http://szdg.lpds.sztaki.hu/szdg
SZTAKI Desktop Grid is a collection of various developments towards Desktop Grid direction, based on BOINC:

- Debian package of the BOINC server
- application programming interface: DC-API
- integration with various backends: e.g. Condor
- supporting various application types on the client side: e.g.: Java, MPI
- aggregating the power of different BOINC projects: hierarchically connected DGs
- ease the application porting under BOINC client: genwrapper
- improving security: introducing certificates in BOINC, sandboxing under BOINC client
- generalise job creation/handling: queuemanager on BOINC server

Most of them can be downloaded from [www.desktopgrid.hu](http://www.desktopgrid.hu), others are under development, ⇒ [desktopgrid@lpds.sztaki.hu](mailto:desktopgrid@lpds.sztaki.hu)
CancerGrid EU FP6 project

- Grid Aided Computer System For Rapid Anti-Cancer Drug Design
  - January 1, 2007 – December 31, 2009
  - Developing focused libraries with a high content of anti-cancer leads, building models for predicting various molecule properties
  - Developing a computer system based on grid technology, which helps to accelerate and automate the in silico design of libraries for drug discovery processes

Bag of tasks: cmol3d, mopac, mdc, fmt, fma, etc.
- Fortran, C, C++
- Processing/Memory requirements
- Multi-binary applications, Libraries
- For some apps source is not available
- Config file preparation before execution
- Pure logging/debugging information
The CancerGrid architecture

DG Server Components

- BOINC Server
- Scheduler
- Data server
- BOINC Task DB
- Job Database (Description of Jobs: Apps, Args, I/O files)
- Queue Manager
- Scheduling policy
- Batch creation
- DC-API master
- DG Submitter
- gUSE DG Submitter
- Job descr.

WS-PGRADE (User IF)

- (WF representation)

- gUSE Storage
- gUSE Local Submitter
- gUSE WS Submitter
- gUSE (Workflow enactor)

BOINC client

- BOINC client
- DC-API cli
- GenWrapper for batch execution
- Legacy Application

BOINC client

- BOINC client
- DC-API cli
- GenWrapper for batch execution
- Legacy Application

The CancerGrid architecture
Generic Wrapper (GenWrapper)

• Why did we developed?
  – The features of BOINC wrapper is not enough (e.g. patching config files on client machines, generating extra messages, independent jobs in a WU, etc.)
  – Wanted to be prepared for unknown requirements might be raised by future Cancergrid applications
  – We did not want to extend the BOINC wrapper to make it an XML-based programming language, we choose to BOINCify an existing language -> Bourne shell

• How does it work?
  – a shell interpreter (gitbox - a variant of busybox) is started instead of the real application
  – it executes an application script, that
    • realizes boincification through script commands
    • may run legacy applications in any way (e.g.: multiple input process)
    • may perform any preparation on input-, output files, environment, etc.
    • may do whatever you can do by a script
Application 1

Application 1

application1_1.01_windows_intelx86.exe
application1_1.01_windows_intelx86.zip

slot dir

1. unzips %BASENAME%.zip to slot dir
2. executes \texttt{dc\_init()} or \texttt{boinc\_init()} [ and \texttt{dc\_finish()} or \texttt{boinc\_finish()} at the end ]
3. creates a script that:
   - sources the Profile Script
   - starts the Application Script
4. starts GenWrapper

5. executes the generated script
   - compound BOINC application

   - handles architecture/platform dependent pre-run tasks for the application

Input Files
command line parameters
Application Script
Output Files

Workunit

GenWrapper

Legacy Executable 1..n (LE)

Legacy Executable Dependencies

Profile Script

uncompress
Sample GenWrapper Script

```
1. IN=`boinc resolve_filename in`
2. OUT=`boinc resolve_filename out`
3. NUM=`cat ${IN}`
4. PERCENT_PER_ITER=$((100000 / NUM))
5. for i in `seq $NUM`; do
6.   PERCENT_COMPLETE=$((PERCENT_PER_ITER * i / 1000))
7.   boinc fraction_done_percent ${PERCENT_COMPLETE}
8.   echo -e "I am ${PERCENT_COMPLETE}% complete." >> ${OUT}
9.   sleep 1;
10. done
```

- shell script contains the BOINC commands
- every filename needs to be resolved
- status: on-going development, still missing some features (CPU time calculation, signal handling, background process, checkpointing, etc.)
Integration of the web-portal to DG

source plug-in

WS-PGRADE/gUSE web based portal plugin

Queue Manager
Scheduling policy

Job Database
(Description of Jobs: Apps, Args, I/O files)

Batch creation
DC-API master

destination plug-in

BOINC plug-in

WS-PGRADE/gUSE

source plug-in

BOINC

BOINC

CancerGrid
Batching in QM

- Substrings like "%{<word>}" are substituted with the appropriate value. Unknown substitutions are left alone and copied as-is.
- 3 template scripts must be prepared for every app.
  - Head template: extracts %{inputs}.
  - Per-job templates:
    - All input files are under %{input_dir} (relative to the directory where the script is started).
      - Moves the input files to appropriate location if necessary.
    - Calls “application %{args}”
    - Moves all output files to %{output_dir} (relative to the directory where the script is started).
  - Tail template: packs the directory %{output_dir} as %{outputs}. 
Example templates

• Example head template:
  
  set +e
tar xzf %{inputs}
  BASEDIR=`pwd`

• Example per-job template:
  
  cd $BASEDIR/%{input_dir}
  $BASEDIR/app %{args} >stdout 2>stderr
  mv out_file stdout stderr $BASEDIR/%{output_dir}
  cd -

• Example tail template:
  
  cd $BASEDIR/outputs
tar czf $BASEDIR/%{outputs} *
Workflow development & configuration

Workflow execution

Algorithms configuration

Integrated components of CancerGrid portal

Molecule database browser
Conclusion

• Any community that has a class of workflow type applications requiring bag of task type of components can easily use a BOINC system:
  – the community can create its own non-public BOINC project
  – can easily map the bag of task components into BOINC applications
  – can easily combine these components into more complex workflow applications

• Such a system
  – has been prototyped for the Cancer Research community within the CancerGrid projects
  – will be available as production system in Q4 of 2008

• Within the EDGeS project we would like to support other communities with this technology
If you need more detailed (technical) information, email to desktopgrid@lpds.sztaki.hu or visit www.desktopgrid.hu

Thank you for your attention!

Questions?

Acknowledgement:
CancerGrid EU FP6 project (FP6-2005-LIFESCTHTALTH-7)
http://www.cancergrid.eu