19th International Conference
ENGINEERING MECHANICS 2013
Svratka, Czech Republic, May 13 – 16, 2013

USE BOINC FOR VOLUNTEER COMPUTING

J. Nosek

Abstract: *GRID is a utilization of heterogeneous computing facilities interconnected with the fast Ethernet. The main advantage is GRID availability, since all offices are nowadays equipped with powerful PCs. There is no need for so-called volunteer computing like the famous SETI@home project which uses a computing power offered by anonymous users on internet. Any project can be implemented in a framework of a scientific group through the BOINC middleware which was developed during the SETI@home project. Our contribution shows the possibilities of such computing environment on an example of structural sizing optimization that needs an enormous computational power.*

Keywords: ***GRID, BOINC, parallel computing, volunteer computing.***

1. Introduction

Many tasks (in our case structural size optimization) need enormous computing time that is not usually easily accessible. The solution can be distributed computing. In other words, divide a large problem into small jobs and giving these small jobs to many computers to solve them and then gather jobs' outputs. Aggregation of outputs then forms the solution of the original "difficult" task.

The potential of volunteer computing have been demonstrated by a project like SETI@home (once of BOINC projects). This project gets computing power from over 100, 000 desktops and notebook PCs volunteered by their owners. The huge population of PCs (more than 1 billion) and still increasing speed offers huge power for volunteer computing. Considerable advantage is much lower cost than dedicated supercomputer. By the way, directly involving the public in science, volunteer computing can increase public knowledge of scientific goals and processes.

Most applications of volunteers[*] computing are based on software called BOINC. The BOINC is an open source middleware based on client – server technology. There are other project like "Bayanihan" (Sarmenta & Satoshi, 1999) witch use ubiquitous and easy to use technology such as web browsers and Java. We focused on BOINC despite of more difficulties for usage but great possibilities. The BOINC project is based at the U.C. Berkley Space Sciences Laboratory and has been funded by the National Science Foundation since its start in 2002.

For you idea, BOINC is used by more than 70 projects, to which over 2 500 000 volunteers (Upton, 2013) and 6 600 000 computers (many volunteers use more than one computer) are connected. Computers supply 2 petaFLOPS of processing power (wikipedia, 2012). It is nice idea, isn't it?

2. Volunteer computing

Volunteer computing and Grid computing share the goal of better utilizing existing computing resources. However, there are basic differences. Grid computing involves organizationally-owned resources: supercomputer, clusters, PCs owned and maintained by university or another organization. All these resources are managed by professional employees (IT specialists). Devices are connected to high-bandwidth network links, and dedicated. There are relationships between organizations. Each of them can use or provide the resources. In contrast, volunteer computing involves an asymmetric relationship between volunteers and projects. Usual project is a small or medium size academic research group with a limited computer power as well as limited budget. Most volunteers are individuals who owns (or uses) Windows (the vast majority), Linux or Mac PC. Computers are often

[*] [*] Ing. Josef Nosek: Faculty of Civil Engineering, Czech Technical University in Prague; Thákurova 7, 166 29 Praha 6 - Dejvice; CZ, e-mail:pepa nosek@seznam.cz

behind NAT or firewall, power of, or disconnected from internet. Volunteers are not computer experts and participate on the project when they are interested in and receive reward like credit or screen-saver. Project has no control over volunteer, cannot make deterministically prediction and cannot exclude malicious behavior.

I write small crawler program to download information about hosts from (BOINCSTAT, 2013). The page contains information about more than 6 million hosts. As a sufficiently large sample, only information about half-million host has been downloaded. Table 1 shows the percentage of OS on host computers. Most volunteers use Windows based computer. This is important information for developers.

Tab. 1: distribution of OS

Operating system	% in sample
Windows 8	1,74
Windows 7	47,9
Windows Vista	8,1
Windows XP	19,1
Windows 2003	0,1
Windows 2000	0,2
Windows Server (all versions)	5,2
Windows (unspecifed)	0,7
All Windows based	83,1
Sun OS	0,01
OSX	5,6
Linux	11,1
Other	0,25

Accordingly there are different requirements on middleware for public resource computing than for Grid computing, e.g. redundant computing or anti-cheat technology. However one of key challenges is to find volunteer. How you do it depends on you and your possibilities. You can ask university employees (for example CVUT about 3,000) or students (CVUT roughly 25,000).

3. BOINC

3.1. Introduction to BOINC

A BOINC project has two basic parts, server and client (host). Client downloads executables and data files from servers, carries out the task (by running applications with specific data files) and uploads output files to server. Server accepts files, validate and assimilate results and (in case of successfully validation) grant credit to a client.

3.2. BOINC client

The BOINC client software consists of several parts (see Figure 1) (paddysinspace, 2013)

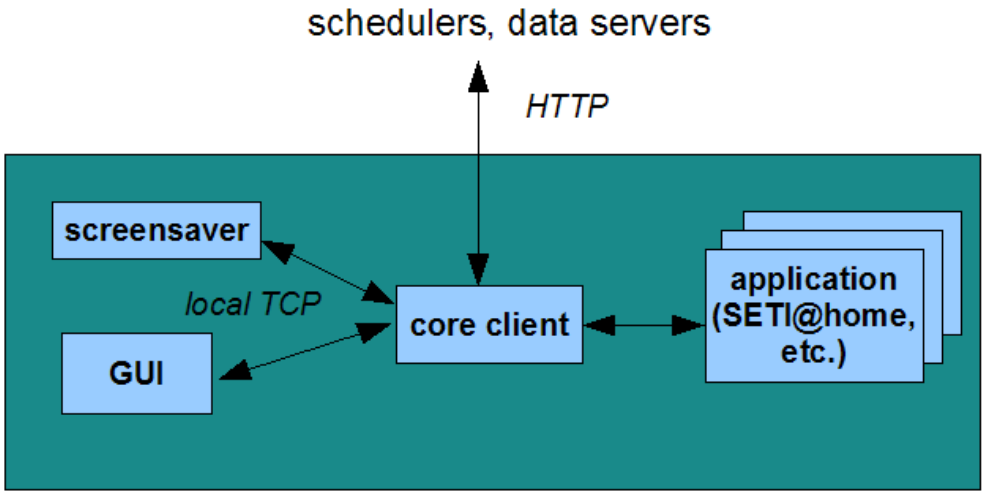

Figure 1 BOINC client

Core client – communicates with external servers via HTTP communications protocol. Program communicates with schedulers, uploads and downloads files and executes applications.

Applications – programs for scientific computation. They can consist of a single process or dynamic set of multiple processes.

The GUI – (also called manager) provide graphical interface that lets you control the core client. GUI communicates with the core client by TCP connection. Usually its local connection is possible to control core remotely.

The Screensaver – runs when you are away from PC. It does communicate with the core by local TCP. Not all projects provide screensaver graphic. However it is visual communication channel which should be interesting for a volunteer, shows the progress and meaning of applications.

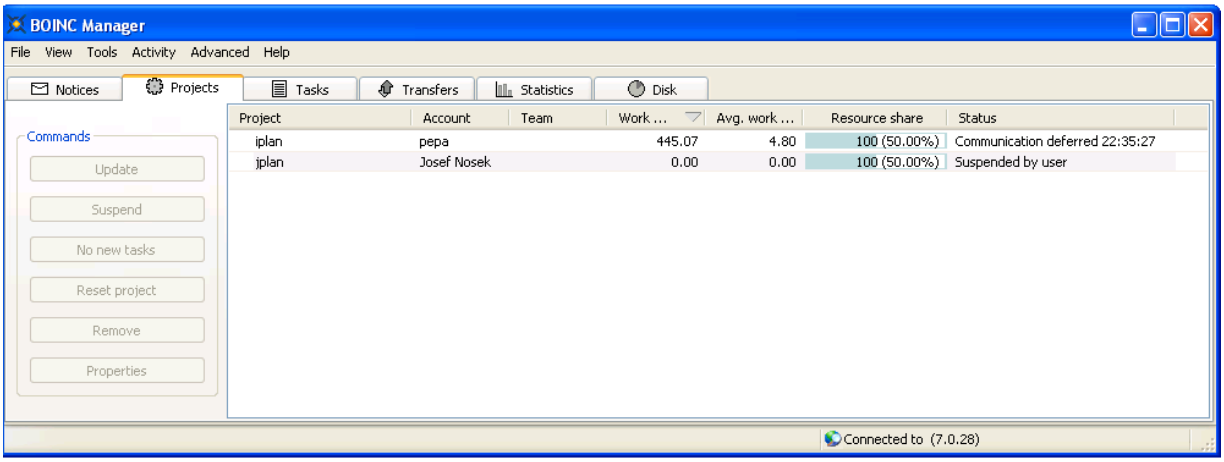

Figure 2.- The BOINC Manager

The core client schedules application. Multiprocessors system with *n* CPUs, can attempts to run *n* applications at once. Each task has project-specified memory limit and computation limit (a number of floating point operations).

Detailed description of BOINC client is not a goal of this paper. An interested reader can find more information in (Anderson & David, 2006)

3.3. BOINC server

An easiest way how to set-up BOINC server is to download a prepared image for Virtual Machine. It can be immediately run and used. However, it is recommended to carefully read a manual at first.

Every BOINC server maintains a Project. Each of these servers can handle one or more Projects. But you should be careful. Volunteers are joined to the Project not to the server. If you change a Project you need new volunteers. Every volunteer can cooperate on many projects however each Project must manually add to BOINC client. Projects handle Applications (one or more). Hence if you have volunteers connected to your Project you should maintain it very well.

Relationship between components can be seen in figure 3.

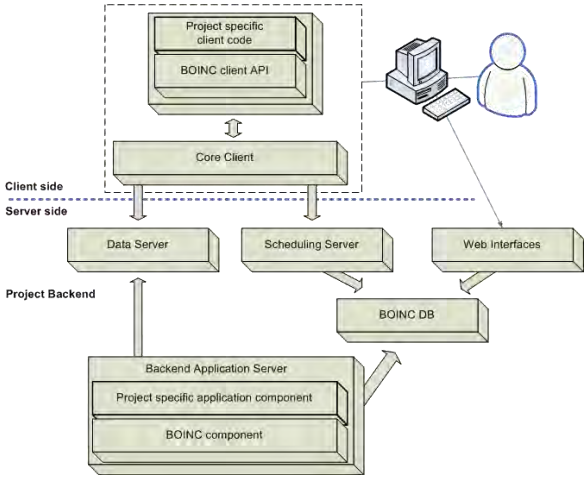

Figure 3. BOINC relationship

BOINC server includes these parts: daemons, schedulers, databases, file servers.

Daemons

BOINC servers use daemons to manage and keep track of their jobs, in BOINC terms work units (WU).

Work generator – generate work unit and corresponded input files

Validator – validate the results of each WU and its redundant copies

Assimilator – copy files from BOINC upload directory to the permanent location, or parse output files and insert results to a database

File delete – check for completed assimilated WUs and then delete input and output files on the server.

Feeder – enhance the schedulers performance and to reduce queries to the database. It does so by placing WUs, from the database into shared memory.

Transitioner – handle state transitions of work unit and results. It generates initial result for work units and generates more WU when timeouts or errors occur.

Database purge – removes records from the BOINC Database (and write it to xml file) when they are no longer needed. Work units are purged only when their input files have been deleted. When are WU purged, all result and output files are purged too.

Schedulers

Determines what work is going to be sent to (or received) from a client on demand. If you disable scheduler you cannot receive any result.

Database

The BOINC database is a MySQL that stores information. Main tables are:

Platform – compilation targets of the core client and application.

App – application

App_version – version of application, include MD5 checksum of executable file.

User – describes user and registration information

Host – describes host and their computer

Work unit – Describes WU. The input file descriptions are stored in an XML document in a blob field. Includes counts of the number of results linked to this work unit, and the numbers that have been sent, that have succeeded, and that have failed.

Result - Describes results. Includes a 'state' (whether the result has been dispatched). Stores a number of items relevant only after the result has been returned: CPU time, exit status, and validation status.

BOINC server directory structure (BOINC, 2013)

```
PROJECT/
    apps/
    bin/
    cgi-bin/
    log HOSTNAME/
    pid HOSTNAME/
    download/
    html/
        inc/
        ops/
        project/
        stats/
        user/
        user profile/
    keys/
    upload/
```

For our purposes, the important folder ‘apps’ contains applications and versions. Folder *download* contains ‘flat’ directory structure to increase CPU time. Directory has set of 1024 subdirectories, named 0 to 3ff. Input files are hashed into these subdirectories.

3.3. Building BOINC application

Even if you will not write a program for BOINC, it is good to have an idea how BOINC works and how the work is handled.

For beta testing of parallel computing using BOINC it is suitable to use BOINC wrapper. Wrapper runs application as subprocess and handless all communication with the BOINC client as show in figure 3 (BOINC, 2013). You can download a source code or a precompiled version for frequently used OS.

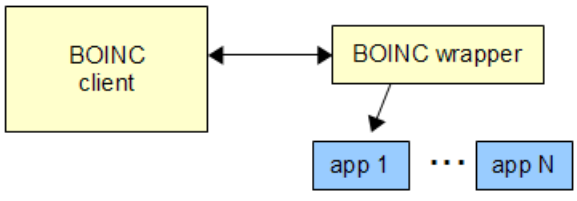

Figure 3. BOINC wrapper

You need only stand-alone application, for example TestApp.exe which loads data from *in* file, processes data and writes output to file *out*.. In this example, we assume only a support windows platform (for other it is similar). Communication with BOINC client handle the wrapper.

Now we should add a new application to the project. Adding an application to the project just puts the information into database. There are two options for adding an application. The first option is via admin web interface. Second option is via *xadd*. We use the first, more convenient way. In *control panel* follow the link *manage applications*. At the bottom of page you can add the name and

description (also called *user friendly name*) for a new application. Same form can be used to mark an application as *deprecated*.

To release a new application you need to put the executable (and other) file(s) in a particular directory where BOINC's *update_version* script can find them. Now it is time to run *update_version* script.

At this time we need to create the work. First step is to copy input files into the flat directory structure.

For one file type the command

cp test_files/in_001 `bin/dir_hier_path in_001` # copy input files *in_001* to download *download dicertory* structure.

If you show location of *input file* use command

./bin/dir_hier_path in_001 # show where is *in_001*

And now you can create work unit

./bin/create_work -appname testApplication -wu_name wu_testApplication_001 -wu_template templates/testApp_in -result_template templates/testApp_out in_001

Create one work unit for application *testApplication*, named *wu_testApplication_001*, use input files *in_001* and use input and output templates files *testApp_in* and *testApp_out*.

4. Simple work generator is shown in appendix 1.Conclusions

Using a volunteer computing is cheap and easy way how to get great computational power. Another advantage of it to raise awareness of the scientific tasks between other colleagues and students. Volunteer computing is also form of social network. You have to constantly attract new volunteers and retain existing. The only thing you can offer is credit and raise awareness of current scientific problems solved.

When I set-up my first BOINC project, during few day I was able to run first app. Is desirable to have knowledge of server administration, network administration, web coder and programmer of course. After the first application is a long way to improving and getting to know all the processes. I have already spent more than six weeks and is still learning new things and debug many errors.

.

Acknowledgement (EM Chapter 1 style)
This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS OHK1-042/13

References

Anderson, & David, P. (2006). Designing a Runtime System for Volunteer Computing. *SC 2006 Conference, Proceedings of the ACM/IEEE.*

Anderson, U., & David, P. (2006). The Computational and Storage Potential of Volunteer Computing. *CCGRID '06 Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid.*

BOINC. (2013). *Server directory structure*. Retrieved from BOINC: http://boinc.berkeley.edu/trac/wiki/ServerDirs

BOINC. (2013). *The BOINC Wrappe*. Retrieved from BOINC wiki: http://boinc.berkeley.edu/trac/wiki/WrapperApp

BOINCSTAT. (2013). Retrieved from http://boincstats.com/en/stats/-1/host/list/0/0/0

paddysinspace. (2013). Retrieved from paddysinspace: http://www.paddysinspace.com/forum/viewtopic.php?f=1&t=320

Sarmenta, L. F., & Satoshi, H. (1999). Bayanihan: building and studying web-based volunteer computing systems using Java. *Future Generation Computer Systems*, 675-686.

Upton, Z. (2013). *BOINC Statistics for the WORLD!* Retrieved from BOINC Synergy: http://www.boincsynergy.com/stats/index.php

wikipedia. (2012). *List of distributed computing projects*. Retrieved from WIKIPEDIA: http://en.wikipedia.org/wiki/List_of_distributed_computing_projects

Appendix 1. – simple flow generator

Simple work generator

```
#!/bin/bash
for i in {11..100}
 do
   a=`printf "%03d" $i`
   cp test_files/in-$a `bin/dir_hier_path in-$a`   # infile  in-001, in-002..in-$i
   ./bin/create_work -appname testapp2 -wu_name wu_testapp2_$a -wu_template
   templates/testapp1_in -result_template templates/testapp1_out in-$a
   echo $a
 done
```