Web Service of Access to Computing Resources of BOINC Based Desktop Grid

Evgeny Ivashko and Natalia Nikitina

Karelian Research Centre of the Russian Academy of Sciences
Institute of Applied Mathematical Research
{ivashko,nikitina}@krc.karelia.ru

Abstract. In the paper we describe a web service of access to computing resources of a desktop grid built in the High-performance Data Center of Karelian Research Centre of the RAS. The grid is based on lightweight, highly scalable BOINC platform. We present the architecture of the system, the current results of implementation and plans for the future. We also present the overview of a scientific research that was carried out with use of the system.

Keywords: distributed computing, volunteer computing, grid, BOINC.

1 Introduction

Recently, supercomputers have been supporting a lot of scientific applications demanding significant computing power. The most common examples include mathematical modeling, numerical computation, physical and biological simulation etc. But not only are supercomputers powerful in solving diverse scientific problems, they are also extremely expensive in use, maintenance and scaling. For applications that process huge amounts of data and at the same time feature a high degree of data parallelism, grid technology has become a priority. Grid computing has been applied by the National Science Foundation's [1] National Technology Grid, NASA's [2] Information Power Grid, European DataGrid [3], Russian Data Intensive Grid [4] etc.

By definition, a computational grid is "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" concerned with "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" [7,8]. Grids allow to integrate computing power of a set of distributed computers connected by the network.

Grid systems can be divided into two groups: the first one includes grids that combine specially dedicated computers (usually supercomputers), the second one includes those utilizing resources of desktops while they are idle. A grid system combining personal computers connected to the Internet and voluntarily provided by their owners is called a system of volunteer computing [10]. A desktop grid [11] is a system that integrates resources of computers in a local network of an organization.

In February 2009, a computing cluster with peak performance of 850 GFlops was launched at the Institution of the Russian Academy of Sciences, Karelian Research

V. Malyshkin (Ed.): PaCT 2011, LNCS 6873, pp. 437–443, 2011.

Centre [12]. In order to provide access to computing resources for users and organize regular work of the cluster, the Center for collective use, "High-performance Data Center", was established on basis of Institute of Applied Mathematical Research (IAMR). Computing resources of the cluster are being used for solving computationally intensive tasks within scientific and applied research as well as for training of parallel programming specialists.

However, recently computing resources of the cluster have ceased to suffice for convenient work of users as the workload increased to 83%. As a rule, applications have to wait in queue until the necessary number of computational nodes is available. At the same time, it has been found that a whole number of users solve computational tasks that are well data-parallelized. Consequently, there was made a decision to create a grid within the framework of the Center for collective use.

Our objective is to implement a service of distributed execution of applications which require significant computational resources, are data parallel and can relatively easily be separated into a number of tasks. We intend to do this on basis of the desktop grid built on the BOINC software. The nodes of the grid are the cluster nodes, servers and desktop PCs of IAMR that will devote to the project their idle time.

2 Software Platform

There are a number of software systems, or middlewares, for grid computing. The most popular ones are Globus Toolkit [13], Unicore [14], GLite [15], Condor [16], ARC [17]. However, for the implementation of the web service we have chosen the BOINC platform [9], which was designed specially for volunteer computing and desktop grids. BOINC (Berkeley Open Infrastructure for Network Computing) is a free, distributed under the GNU LGPL license software platform for distributed computing, developed at the University of California, Berkeley. BOINC platform has been a framework for many independent volunteer computing projects [18-20]. In comparison with other popular middlewares that also support volunteer computing, such as Condor, BOINC is much more lightweight, less complicated in deployment and expansion, and available for a larger variety of platforms.

There have been several other systems for distributed computing with use of idle computers in a network**,** e.g. QADPZ [6], Bayanihan [23] and Entropia [24]. However, the BOINC software is the most actively developed at the moment and supports the widest range of applications [25]. Being cheap in deployment and maintenance, BOINC server software has high performance and good opportunities in scalability. The client part is available for many computing platforms and is easy to deploy. BOINC is an open source software, has a lot of features and provides documented interfaces to many of its key components.

One of the basic concepts of BOINC software is a *project*. A project is an autonomous entity that does distributed computing. Projects are independent; each one has its own applications, database, web site and servers, and is not affected by the status of other projects. Each project is identified by the URL of its web site.

A BOINC project can include multiple *applications*, each of which consists of several programs for different computing platforms and a set of *workunits* and *results*.

A workunit is an independent computational task. A result, each associated with a workunit, describes an instance of a computational task, either *unstarted*, *in progress*, or *completed*. In some cases there may be several instances, or "replicas", of a given workunit. This is used to ensure that clients don't send back to server intentionally or unintentionally wrong results. The BOINC server creates one or more instances of a computational task according to project preferences, distributes them to client hosts, collects the output files, handles the results and after all deletes the I/O files.

Typically, creation and support of a BOINC project involve the work of an administrator whose main duties are to manually create applications, fill in the preferences for computational tasks etc. Hence, to enable wider access to the computing resources of the desktop grid we have decided to implement a web service that is described further in this paper.

There are a few things that the term "web service" may refer to. We use a broad definition of the web service given by IBM [22]:

> *"A self-contained, modular application that can be described, published, located, and invoked over the Web. Platform-neutral and based on open standards, Web Services can be combined with each other in different ways to create business processes that enable you to interact with customers, employees, and suppliers."*

Each BOINC application includes the main executable (probably with other files necessary to run it) and descriptions of I/O files that are specific for this application and common for all its workunits. The administrator of the BOINC server must provide application-specific programs for validation and assimilation of the computational results. Given the main executable, descriptions of I/O files, validation and assimilation programs, a new application can be created and run automatically.

The Leiden Classical project [5] (which is originally aimed at supporting test simulations of molecules and atoms in a classical mechanics environment) has implemented a web-based interface allowing users to submit BOINC jobs, including uploading their own input files. However, the system allows to create workunits for registered BOINC applications only. Moreover, it lacks the feature of uploading many input files at a time and some other important functions such as custom number and format of I/O files. In our work the aim was to develop a web-based interface that would assist in creating users' applications that require different I/O parameters.

Any existing executable can be run under the BOINC software with use of a *wrapper program* supplied by BOINC [21], thereby without need of modification of the source code. The wrapper runs the executables, or "real" applications, as subprocesses and handles all communication with the core client. There may be more than one executable under the wrapper, e.g. a long task may be separated into multiple tasks for the purpose of enabling checkpointing. Both wrapper program and executable files must be built for the computing platforms that grid working nodes have. BOINC allows applications to have various versions for a wide range of computing platforms.

The client part in the desktop grid includes files that are necessary to run the client program, bind it to an account and allow remote control and monitoring of client. Grid server is directly connected to the web server.

3 The System Architecture

The web service of access to computing resources of the desktop grid is based on the following components:

- *website* that serves as an access point to the service and provides a user interface;
- *grid server* (based on BOINC platform) that controls creation and distribution of computational tasks and handles the results;
- *grid working nodes* that perform computational tasks.

The system architecture is shown in Fig. 1. The workflow in the system is described as follows:

- the user uploads files and specifies parameters required to create computational tasks via the website;
- the uploaded data is processed by a server-side program that creates a new BOINC application and computational tasks;
- instances of the new application together with one or more computational tasks are sent to working nodes that have enough resources to execute them;
- the user monitors execution progress via the website;
- the server collects the results from the working nodes, processes them and sends to the user.

4 Implementation

In Fig. 2 there is a screenshot of the web-based interface of the service. The form provides the fields to upload the executable, additional files (if any) and a zip archive of input files; the fields for physical names of I/O files that will the executable require; the fields for estimates of resources to complete the tasks and the field for email address which are the results to be sent to. The field for upload of the main executable and the field for email address are compulsory, others are optional.

A user enters the website, uploads the main executable and probably additional files, specifies the names for I/O files that the executable will require, and uploads a couple of input files, each one for a separate computational task. All uploaded data is processed by a server-side program that creates a new application and makes templates for input and output files. In our case every application contains the wrapper program in addition to user's executable. Computational tasks, or workunits in terms of BOINC, are generated with use of input files together with their templates and templates of output files that are to be received.

The grid working nodes periodically send to the server requests for more work. The BOINC scheduling program responds to work requests by distributing unsent results between the working nodes that have asked for work and meet certain criteria, e.g. have enough disk space and memory to handle the tasks.

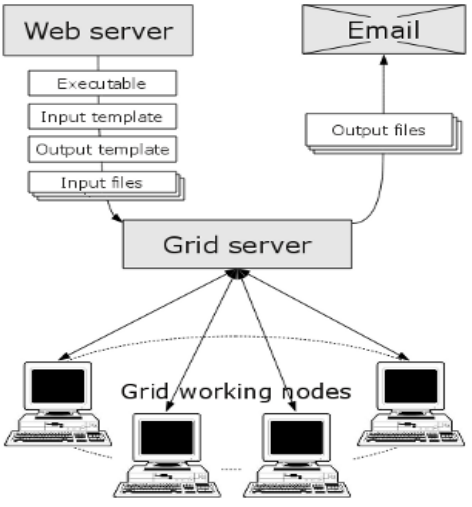

Fig. 1. The system architecture

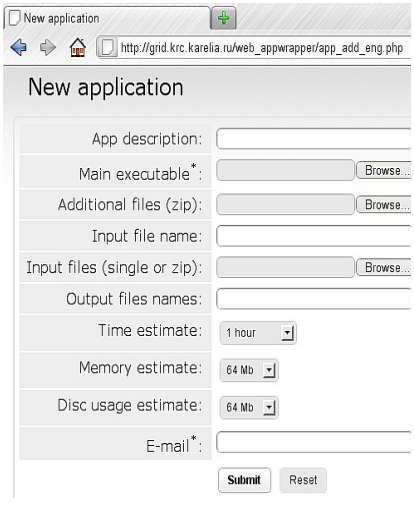

Fig. 2. Web interface of the service

The overall execution progress is displayed on the web site. At the moment the user may view the whole number of workunits and their status, i.e. in progress or completed. Once the completed result of a workunit is reported by a grid working node, a validation program checks the output files and marks the result either as valid or invalid. Valid results are then handled by an assimilation program that stores them in a zip archive and sends it to user via email as soon as all results have been handled.

Currently the number of users of the service is not restricted. There may be as many workunits as required and the time of their execution will depend only of the number of available nodes. However, BOINC allows to assign priorities to applications of different users.

5 Conclusion, Results and Further Work

In this paper we describe a web service of access to computing resources of a desktop grid based on the BOINC software and present the current results of its implementation. The service is intended mainly for applications that are data parallel and can relatively easily be separated into a number of tasks. The final version of the service will be available for users of the Center for collective use as well as users of other institutions within the agreements.

The web service has been used in solving the computational problem of finding the optimal parameters for the mathematical model of hydride decomposition within the scientific research devoted to studying the White Sea. This is a non-classical boundary-value problem with a free boundary and nonlinear Neumann boundary conditions. The finite-difference numerical method was implemented in Fortran-90/95. The problem is to find the set of kinetic parameters that provide the best (in the least squares sense) approximation of the desorption curve obtained experimentally. A difficulty is that the solution can be not unique.

Currently we have obtained the following results:

- the software platform has been selected;
- the system architecture has been designed;
- the BOINC based desktop grid has been deployed;
- sixteen working nodes have been added to the grid, among them ten cluster compute nodes, a cluster frontend, four servers and a desktop computer;
- the website has been created;
- a series of users' projects have been implemented.

Three main directions for the further work are:

- improvement of the web interface:
  - provide access to BOINC features such as automatic generation of workunits, validation of results, monitoring the whole progress of an application etc.;
  - implement personalization including management of personal accounts, viewing the statistics of use, access to the cluster, organization of the projects that may include several applications.
- grid development:
  - add more computers to serve as grid working nodes;
  - adjust parameters of server-side programs and application workunits so as to minimize waiting time;
  - implement a feature of uploading source code for further cross-compilation on the server side.
- scientific research:
  - develop a mathematical model of the job queueing process in the system and select the optimal queue discipline (e.g., the one that would guarantee the minimal average waiting time, or the minimal queue length, etc.);
  - investigate the statistical distribution of completion times of the workunits and develop a procedure of their adaptive assignment to those clients who will have enough resources with high probability;
  - investigate the dependence between size of the workunit and time for it to be completed in the grid system and develop a method to automatically separate the large problem into smaller workunits most effectively.

References

1. The National Science Foundation, http://www.nsf.gov
2. The National Aeronautics and Space Administration, http://www.nasa.gov
3. European Data Grid Project, http://grid.web.cern.ch/grid
4. Russian Data Intensive Grid, http://www.egee-rdig.ru
5. The Leiden Classical project, http://boinc.berkeley.edu/trac/wiki/UserJobs
6. QADPZ - Quite Advanced Distributed Parallel Zystem, http://qadpz.sourceforge.net/

7. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
8. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications 15(3) (2001)
9. BOINC: open-source software for volunteer and grid computing, http://boinc.berkeley.edu
10. Volunteer computing, http://boinc.berkeley.edu/trac/wiki/VolunteerComputing
11. Grid computing with BOINC, http://boinc.berkeley.edu/trac/wiki/DesktopGrid
12. High-performance Data Center, KarRC, RAS, http://cluster.krc.karelia.ru
13. The Globus Toolkit, http://www.globus.org/toolkit
14. UNICORE: Distributed Computing and Data Resources, http://www.unicore.eu
15. Glite: Lightweight Middleware for Grid Computing, http://glite.web.cern.ch/glite
16. Condor: High Throughput Computing, http://www.cs.wisc.edu/condor
17. ARC: The Advanced Resource Connector, http://www.knowarc.eu
18. SETI@home, http://setiathome.berkeley.edu
19. Einstein@home, http://einstein.phys.uwm.edu
20. Rosetta@home, http://boinc.bakerlab.org/rosetta
21. The BOINC Wrapper, http://boinc.berkeley.edu/trac/wiki/WrapperApp
22. IBM(R) WebSphere(R) Host Publisher; Administrator's and User's Guide. Glossary, http://www.ibm.com/software/webservers/hostpublisher/library/publications/guide40/guide16.htm
23. Sarmenta, L.F.G.: Bayanihan: Web-Based Volunteer Computing Using Java (1998)
24. Chien, A., Calden, B., Elbert, S., Bhatia, K.: Entropia: Architecture and Performance of an Enterprise Desktop Grid System (2001)
25. Anderson, D., Christensen, C., Allen, B.: Designing a Runtime System for Volunteer Computing (2008)