Integration of Everest Platform with BOINC-based Desktop Grids

Oleg Sukhoroslov
Institute for Information Transmission Problems of the Russian Academy of Sciences
Bolshoy Karetny per. 19, build.1, Moscow 127051 Russia
sukhoroslov@iitp.ru

Abstract

Desktop grids is an important class of distributed computing infrastructures (DCIs) used for solving complex scientific problems. The inherent complexity of DCIs and used technologies limit the wide adoption of distributed computing in practice. Everest is a web-based distributed computing platform that uses service-oriented approach and cloud computing models to solve this problem. This paper discusses the possible approaches for integration of Everest and BOINC-based desktop grids and presents the current prototype implementation. The proposed integration enables Everest users to seamlessly access computing resources of desktop grids and build generic or domain-specific web services for submission and automation of computations in desktop grids.

1 Introduction

Computational methods are now widely used for solving complex scientific and engineering problems. These methods often require a large amount of computing resources. Nowadays, there is a wide range of such resources including servers and personal computers, clusters and supercomputers, grids and clouds. Distributed computing technologies enable integration of independent resources into *distributed computing infrastructures (DCIs)* that can provide significant computing power and increase the efficiency of use of individual resources.

Desktop grids [KTB$^+$04, Fed12] represent an important class of DCIs that aggregate computing power of idle personal computers. The most widely used technologies for building desktop grids are HTCondor [TTL05] and BOINC [ACA06]. While HTCondor is most suited for integration of resources within an organization, i.e. enterprise desktop grids [Iva14], BOINC is originally designed to support volunteer computing projects [And04, ACA06] with globally distributed resources donated by their owners. Modern desktop grids are capable of integrating resources of hundreds of thousands of personal computers with an aggregate computing power comparable to the top supercomputers. The distinctive advantage of desktop grids is the low costs of building and operating such infrastructures, which makes it affordable for small research projects.

Despite the abundance of computing resources, the inherent complexity of distributed computing technologies and infrastructures, along with the lack of required IT expertise among the researchers, limit the wide adoption of these technologies in practice. This problem can be solved by providing high-level services with domain-specific interfaces that hide the mentioned complexity from the user. These services should automate all common actions

In: E. Ivashko, A. Rumyantsev (eds.): Proceedings of the Third International Conference BOINC:FAST 2017, Petrozavodsk, Russia, August 28 - September 01, 2017, published at http://ceur-ws.org

needed to perform computations on remote resources or DCIs. The use of service-oriented approach can also improve the research productivity by enabling publication, sharing and composition of computing applications as services.

Everest [SVA15, Eve] is a web-based distributed computing platform that implements the described approach. The platform supports publication, execution and composition of computing applications in a distributed environment. Unlike other solutions, Everest is based on the Platform as a Service (PaaS) cloud computing model by providing its functionality via remote web and programming interfaces. A distinctive feature of Everest is the support for running applications on arbitrary combinations of computing resources attached by users. Currently the platform supports integration with standalone servers, computing clusters and European Grid Infrastructure [SSV16].

This papers presents integration of Everest platform with BOINC-based desktop grids. The proposed integration has the following benefits. First, it enables Everest users to seamlessly access computing resources of desktop grids and combine them with other types of resources already supported by the platform. Second, it makes it possible to use Everest for building generic or domain-specific web services for submission and automation of computations in BOINC-based desktop grids. Hopefully, the proposed approach will make such DCIs accessible to a wider group of researchers.

The paper is structured as follows. Section 2 provides a brief overview and relevant technical details of both used technologies. Section 3 discusses the possible approaches for integration of Everest and BOINC and presents the current prototype implementation. Section 4 concludes and discusses future work.

2 Everest and BOINC Overview

2.1 Everest

Everest [SVA15] is a web-based distributed computing platform. It provides users with tools to quickly publish and share computing applications as web services. The platform also manages execution of applications on external computing resources attached by users. In contrast to traditional distributed computing platforms, Everest implements the PaaS model by providing its functionality via remote web and programming interfaces. A single instance of the platform can be accessed by many users in order to create, run and share applications with each other. The platform is available online to all interested users [Eve].

Everest supports development and execution of computing applications following a common model. An application has a number of *inputs* that constitute a valid request to the application and a number of *outputs* that constitute a result of computation corresponding to some request. Upon each request Everest creates a new *job* consisting of one or more computational *tasks* generated by the application according to the job inputs. The tasks are executed by the platform on computing resources specified by a user. The dependencies between tasks are currently managed internally by applications. The results of completed tasks are passed back to the application and are used to produce job outputs or new tasks if needed. The job is completed when there are no incomplete tasks are left. The described application model is generic enough to support a wide range of computing applications, including many-task applications.

Users can publish applications via provided generic application template that makes it possible to avoid programming. The template supports running arbitrary applications with command-line interface and produces a single task corresponding to a single command run. There are two approaches for implementing many-task applications on Everest. First, it is possible to dynamically add new tasks or invoke other applications from a running application via the Everest API. This enabled users to create and publish complex many-task applications with dependencies between tasks, such as workflows. Everest also provides a ready-to-use generic application [VS15] for running a large number of independent parametrized tasks. i.e. parameter sweep experiments.

An application is automatically published as a RESTful web service with a unified interface. This enables programmatic access to applications, integration with third-party tools and composition of applications into workflows. The platform's web user interface also generates a web form for running the application via web browser. The application owner can manage the list of users that are allowed to run the application.

Instead of using a dedicated computing infrastructure, Everest performs execution of application tasks on external resources attached by users. The platform implements integration with standalone machines and clusters through a developed program called *agent* [SSV16]. The agent runs on the resource and acts as a mediator between it and Everest enabling the platform to submit and manage computations on the resource. The platform also supports integration with resources of the European Grid Infrastructure. Everest manages execution of tasks on remote resources and performs routine actions related to staging of input files, submitting a task, monitoring a task state and downloading task results. The platform also monitors the state of resources and uses this information during scheduling.

Everest users can flexibly bind the attached resources to applications. In particular, a user can specify multiple resources, possibly of different type, for running an application [SSV16]. In this case the platform performs dynamic scheduling of application tasks across the specified resource pool.

2.2 BOINC

BOINC (Berkeley Open Infrastructure for Network Computing) [And04, ACA06] is an open-source middleware platform for volunteer computing projects. Originally developed to support the SETI@home project, it became a de facto standard for running volunteer computing projects supporting research in different areas of science and technology. Volunteers participate by running the BOINC client software on their computers. They can attach each computer to any set of projects, and can control the allocation of resources among the projects. Currently BOINC brings together about 200-300 thousand active participants and 700-800 thousand active computers worldwide contributing around 16-17 PetaFLOPS of processing power in total.

Each BOINC-based project provides its own server to manage execution of project-specific applications on volunteers' computers and distribute data files. BOINC clients download application executables and data files from the project server, carry out tasks or workunits by running applications against the specific data files, and upload the output files to the server. BOINC software includes server-side components, such as scheduler and daemon programs that manage job distribution and collection, and web-based interfaces for volunteers and project administrators. All information about project applications and jobs is stored in a relational database on the server.

2.2.1 Application Development

The inherent feature of desktop grids is the heterogeneity of computing hosts resulting in different CPU architectures, OS types, software versions, etc. The original approach of porting an application to BOINC is to build a separate application binary for each supported platform. This approach requires significant efforts, especially in cases when the application was originally developed to run in a particular environment.

A more recent approach is running applications in virtual machines, the so-called "VM apps" [Boid]. In this case the developer creates a VM image with all required software that will be downloaded and used by BOINC client for running the application. This approach avoids tedious porting of application to different platforms, but requires installation of VirtualBox on hosts, introduces a small runtime overhead and doesn't support GPU applications.

Recently, a new approach for BOINC application deployment based on Docker containers has been introduced [Boia]. In comparison to virtual machines, Docker images are generally much smaller, can be instantiated much faster, and have less runtime overhead. The developer creates or reuses a Docker image for running the application. Docker-based applications are run under a universal BOINC application that uses a generic VirtualBox VM to host Docker containers. The developer can use arbitrary Docker images when submitting jobs. The Docker image files are passed inside a BOINC workunit and are cached on the hosts. This approach is implemented in *boinc2docker* tool [boib].

In this work the Docker-based approach was used for packaging and running applications in BOINC, because it requires a minimal effort from the developer.

2.2.2 Job Submission

The original approach for submitting jobs to a BOINC application uses the *create_work* command-line tool. This tool is intended for the use by project administrators and requires the submitter to have a login access to the BOINC server. Recently, the remote job submission mechanism was introduced to support submitting jobs by programs running outside the BOINC server [Boic]. This mechanism is based on Web RPCs and does not require login access to the server.

3 Integration of Everest with BOINC

3.1 Prototype Implementation

The most straightforward approach for integration of Everest with BOINC is to reuse the same mechanism that is used for integration with computing clusters.

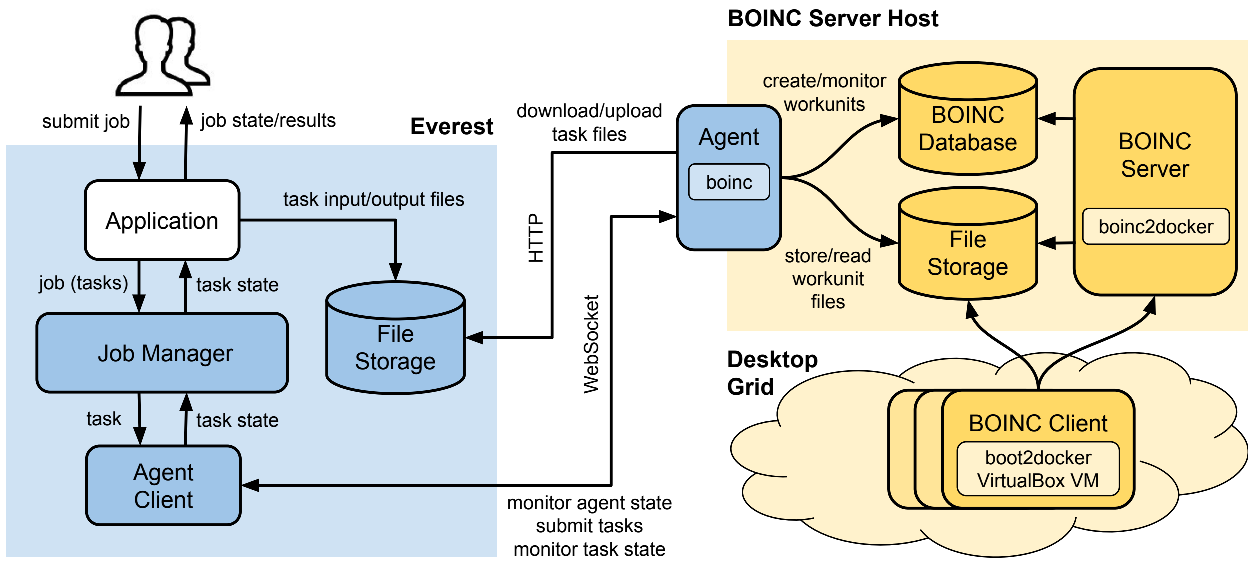

Figure 1: Integration of Everest with BOINC via Everest agent

In this mechanism, the Everest agent runs on the cluster submission node, receives tasks from the platform and translates them to jobs submitted to the cluster. The agent supports interaction with various resource managers through the pluggable adapters. The adapter receives generic resource requests (get resource state, submit task, get task state, cancel task, etc.) from the agent and translates these requests into commands specific to a particular resource manager. The developed adapters for computing clusters support integration with commonly used batch systems such as TORQUE, Slurm and Sun Grid Engine.

In case of BOINC, the similar approach has been implemented as follows (see Figure 1). The Everest agent is deployed on a BOINC server and connected to the platform. To support the interaction of the agent with BOINC, a special *boinc* adapter has been developed. This adapter converts a task received from Everest to a BOINC job workunit by placing the task input files in the downloads folder and registering the workunit in the BOINC server database. The adapter monitors the state of submitted workunits by querying the database. When the workunit result is assimilated, the adapter passes the output files to the agent which uploads them to Everest.

To support the submission of arbitrary tasks from Everest to BOINC, the current implementation relies on the universal *boinc2docker* BOINC application [boib]. This application supports running BOINC jobs in arbitrary Docker images, thereby implementing the Docker-based approach described in Section 2. The required Docker image is specified during the job submission. Currently, the Docker image used for running Everest tasks is fixed in the *boinc* adapter configuration. Also, to be able to locate the output files of each workunit, the current implementation relies on a custom assimilator that copies output files to a directory checked by the *boinc* adapter.

The implementations of the *boinc* adapter and the custom assimilator are available in the git repository of the Everest agent[1].

3.2 Experimental Evaluation

The end-to-end testing of the described implementation has been performed by running a real-world parameter sweep application. The application represented the virtual screening of 100 ligand molecules using the molecular docking program Autodock Vina.

The application, including the *vina* Linux binary, auxiliary Python script and input files, was submitted to Everest via the generic Parameter Sweep service [VS15]. During the submission, a resource has been selected that represented the BOINC project attached to Everest using the described implementation. The agent was configured to submit workunits using the *frolvlad/alpine-python2* Docker image, a compact (∼50MB) image with Alpine Linux and Python 2.7. The project had four active desktop hosts during the testing.

[1]`https://gitlab.com/everest/agent/tree/boinc`

The application was successfully executed via Everest by dispatching all application tasks to BOINC and collecting the task results. This demonstrated that the described implementation enables using BOINC-based desktop grids as computing resources for running existing Everest applications.

3.3 Alternative Approaches

In addition to the presented implementation, there are other possible approaches for integration of Everest and BOINC that are briefly discussed below.

3.3.1 Remote Job Submission

This approach relies on the remote job submission mechanism introduced recently in BOINC server [Boic]. Instead of running the Everest agent on the server, the platform could use this mechanism to directly submit jobs to BOINC, query their status and download results.

In comparison to the current implementation, this approach avoids running of additional software (agent) on the BOINC server and relies only on passing to Everest the authenticator token of submission account. However, this approach requires a more complex implementation of the integration logic on the server side of Everest. Besides that, the remote job submission mechanism is relatively new and not stable that further complicates the implementation of this approach.

3.3.2 Dynamic Agent Deployment

This approach leverages existing Everest agents by dynamically deploying them to hosts attached to a BOINC project. To do this, Everest submits to the BOINC server generic workunits containing the agent. The number of submitted agents can depend on the number of unprocessed tasks. When started on a host the agent connects to Everest and begins to execute tasks assigned by the platform on the host. The input and output files are transferred directly between the agent and the platform. A similar approach has been used previously for integration of Everest with European Grid Infrastructure [SSV16].

In this approach, there is no need to convert tasks of Everest applications to BOINC workunits. Also, the interaction between the agent and Everest does not differ from the case of a standalone resources. However, the agent is not deployed by a user and a single Everest resource representing the desktop grid can be backed by a dynamic pool of agents. Furthermore, when running as a BOINC workunit, the agent has a limited lifetime and can be suspended. The agent should also terminate its execution in the absence of new tasks from Everest.

Since this approach bypasses BOINC server for task scheduling and data transfer, its' potential advantages are related to more flexible scheduling by means of Everest, such as supporting small tasks or implementing dynamic load balancing. However, it is not clear how to implement validation of results or grant credits under this approach. Also, having thousands of agents connected to Everest could hit the current scalability limits of the platform.

4 Conclusion and Future Work

This paper discussed the possible approaches for integration of Everest and BOINC-based desktop grids and presented the current prototype implementation. The proposed integration enables Everest users to seamlessly access computing resources of desktop grids and build generic or domain-specific web services for submission and automation of computations in desktop grids. Future work will focus on improving the described implementation (e.g., adding support for integration with arbitrary BOINC applications) and conducting a large-scale experimental evaluation.

Acknowledgments

This work is supported by the Russian Science Foundation (project No. 16-11-10352).

References

[ACA06] David P Anderson, Carl Christensen, and Bruce Allen. Designing a runtime system for volunteer computing. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 33–33. IEEE, 2006.

[And04] David P Anderson. Boinc: A system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10. IEEE, 2004.

[Boia] BOINC and Docker. [online]. `https://boinc.berkeley.edu/trac/wiki/BoincDocker`.

[boib] boinc2docker. [online]. `https://github.com/marius311/boinc2docker`.

[Boic] RPCs for remote job submission. [online]. `https://boinc.berkeley.edu/trac/wiki/RemoteJobs`.

[Boid] Running apps in VirtualBox virtual machines. [online]. `http://boinc.berkeley.edu/trac/wiki/VboxApps`.

[Eve] Everest. [online]. `http://everest.distcomp.org/`.

[Fed12] Gilles Fedak. *Desktop grid computing.* Chapman and Hall/CRC, 2012.

[Iva14] Evgeny Evgen'evich Ivashko. Enterprise desktop grids. *Programmnye Sistemy: Teoriya i Prilozheniya [Program Systems: Theory and Applications]*, (1):19, 2014.

[KTB+04] Derrick Kondo, Michela Taufer, Charles L Brooks, Henri Casanova, and Andrew A Chien. Characterizing and evaluating desktop grids: An empirical study. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 26. IEEE, 2004.

[SSV16] Sergey Smirnov, Oleg Sukhoroslov, and Sergey Volkov. Integration and combined use of distributed computing resources with everest. *Procedia Computer Science*, 101:359–368, 2016.

[SVA15] O. Sukhoroslov, S. Volkov, and A. Afanasiev. A web-based platform for publication and distributed execution of computing applications. In *Parallel and Distributed Computing (ISPDC), 2015 14th International Symposium on*, pages 175–184, June 2015.

[TTL05] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the condor experience. *Concurrency and computation: practice and experience*, 17(2-4):323–356, 2005.

[VS15] Sergey Volkov and Oleg Sukhoroslov. A generic web service for running parameter sweep experiments in distributed computing environment. *Procedia Computer Science*, 66:477–486, 2015.