

# EmBOINC: An Emulator for Performance Analysis of BOINC Projects

Trilce Estrada, Michela Taufer  
University of Delaware  
{estrada, taufer}@udel.edu

Kevin Reed  
IBM  
knreed@ibm.us.com

David P. Anderson  
University of Berkeley  
davea@ssl.berkeley.edu

## Abstract

*BOINC is a platform for volunteer computing. The server component of BOINC embodies a number of scheduling policies and parameters that have a large impact on the projects throughput and other performance metrics. We have developed a system, EmBOINC, for studying these policies and parameters. EmBOINC uses a hybrid approach: it simulates a population of volunteered clients (including heterogeneity, churn, availability, reliability) and it emulates the server component; that is, it uses the actual server software and its associated database. This paper describes the design of EmBOINC and validates its results based on trace data from an existing BOINC project.*

## 1 Introduction

Volunteer Computing (VC) is a form of distributed computing in which ordinary people volunteer processing and storage resources to computing projects. BOINC is a software platform for volunteer computing [1] supporting computing projects in e.g., physics, molecular biology, medicine, chemistry, astronomy, climate dynamics, mathematics. In the past five years, the VC community powered by BOINC has significantly grown and currently there are approximately 50 projects and 580 000 volunteer computers supplying an average of 1.2 PetaFLOPs to these projects. A main strength of BOINC systems is its capability to provide the scientists with PetaFLOPs computers at a low cost.

The server component of BOINC embodies a number of scheduling policies and parameters that have a large impact on the project throughput and other performance metrics. Unfortunately, it is difficult (if not impossible) to do controlled performance experiments in the context of a large volunteer computing project because there are many factors that cannot be controlled and because poorly-performing mechanisms can waste lots of resources driving away volunteers. On the other hand, exploring new policies and tuning setting parameters for high performance can be done in simulated environments, where it is possible to test a

wider range of hypotheses in a shorter period of time without affecting the BOINC community. A simulated environment can be based on: a simulator implementing an abstract model of a system, an emulator using all or part of a real system, or a combination of both (hybrid approaches). Simulating policies accurately is a lost cause because an abstract model of BOINC is limited by the system complexity. Fortunately, this complexity can be kept under control in emulated environments, making emulators the ideal approach to study performance.

In this paper we present EmBOINC, a trace-driven emulator of BOINC projects. EmBOINC allows project designers to tune existing policies and to predict performance of new policies rigorously. EmBOINC is an emulator because it uses part of the BOINC platform (i.e., its server) to emulate the interaction of the BOINC server with the heterogeneous, volatile population of volunteers' hosts. EmBOINC interacts directly with the real BOINC daemons triggering the generation, distribution, collection, and validation of jobs. By using the BOINC server, EmBOINC minimizes the maintenance burden caused by the rapid development of BOINC. At the same time, by plugging directly into a BOINC server, EmBOINC allows testers to directly and accurately tune BOINC policies for different host populations. Results in this paper show the high accuracy of EmBOINC in predicting throughput and latency in BOINC projects compared with real projects.

This paper is organized as follows: Section 2 presents a short overview of VC and BOINC. The EmBOINC framework and its software components are described in Section 3. In Section 4 we validate EmBOINC by comparing its predictions with those of a real BOINC projects. Section 5 compares EmBOINC with other work. Section 6 concludes the paper and presents some future work.

## 2 Volunteer Computing and BOINC

### 2.1 Volunteer Computing

Volunteer Computing (VC) projects employ computing resources (e.g., desktops, notebooks, and servers) owned by

ordinary people and connected to the Internet. Traditionally, VC projects target large search problems in science and, therefore, generate large sets of jobs that are distributed across volunteer computing resources. Replication of jobs is used to address the volatility of these systems as well as other issues like malicious attacks, hardware malfunctions, or software modifications that ultimately affect the reliability of results. Replicas of jobs (also called job instances) are distributed to different volunteer computing resources (also called hosts) that execute them. When finished, the hosts send their results to the project server, which collects the results and distinguishes between successful and unsuccessful results. Unsuccessful results are those that either are erroneous or are returned too late, i.e., timed-out. The project server validates successful results, identifying them as valid or invalid. Jobs are valid when they agree across instances of the same job and invalid when negatively affected by malicious attacks, hardware malfunctions, or non-authorized software modifications.

## 2.2 BOINC

BOINC (Berkeley Open Infrastructure for Network Computing) [1] is an open-source system that harnesses the computing power and storage capacity of thousands or millions of PCs owned by ordinary people for large-scale scientific simulations. The computing resources available to a BOINC project are highly diverse: the hosts differ by orders of magnitude in their processor speed, available RAM, disk space, and network connection speed. Some hosts connect to the Internet by modem only every few days, while others are permanently connected. Computations may have varying bounds on completion time. BOINC projects are highly attractive for large-scale simulations because they can potentially sustain a very high processing rate (tens or hundreds of TeraFLOPs).

The BOINC model involves projects and volunteers. Projects are organizations (typically academic research groups) that need computing power. Projects are independent; each operates its own BOINC server. Volunteers participate by running the BOINC client software on their computers (hosts). Volunteers can attach each host to any set of projects, and can specify the quota of bottleneck resources allocated to each project. A BOINC server is centered around a relational database, whose tables correspond to the abstractions of the BOINC's computing model, i.e., the platform or execution environment, the single or multiple applications or programs in a project, and the jobs and job instances associated with an application.

When a BOINC client is attached to a project, it periodically issues a scheduler *request* to the project's server. The request message includes a description of the host and its current workload (jobs queued and in progress), descrip-

tions of newly-completed jobs, and a request for new jobs. The *reply* message may contain a set of new jobs. Multiple jobs may be returned; this reduces the rate of scheduler requests and accommodates clients that are disconnected from the Internet for long periods.

BOINC daemons are in charge for generating new jobs (workgenerator); periodically replenishing the shared memory with job instances taken from the BOINC database (feeder); changing the status of jobs and job instances in the database, i.e., unsent, in progress, over, success (transitioner); validating results (validator); and removing the files associated with jobs that have been validated or that are not needed anymore (assimilator/file-deleter).

Sophisticated server-side scheduling policies have been implemented in several BOINC projects. Currently, World Community Grid has a number of criteria for job assignment [2], based on host and job diversity (e.g., size of the job and speed of the host relative to an estimated statistical distribution, disk and memory requirements for the job to be completed, homogeneous redundancy [12] and host error rate). A scoring-based scheduling policy uses a linear combination of these terms to select the best set of jobs that can be assigned to a given host. Projects can adjust the weights of these terms, or they can replace the scoring function entirely. The benefits of these terms and the different scoring functions can be quantified with an emulator such as EmBOINC.

## 3 EmBOINC

### 3.1 Emulator Framework

EmBOINC (Emulator of BOINC Projects) is a trace-driven emulator that models heterogeneous hosts and their interaction with a real BOINC server. By plugging into a BOINC server, EmBOINC triggers the server's daemons to generate and distribute jobs to the EmBOINC hosts. We use statistical information obtained from real BOINC traces to characterize volatile, heterogeneous, and error-prone hosts. A discrete event simulator is used in EmBOINC to accurately model the hosts' behavior. EmBOINC also interacts with the BOINC daemons and triggers the collection and validation of completed job instances.

As it occurs in real BOINC projects, EmBOINC projects can have different applications running together or at different times. Applications can share the simulated hosts partially or completely. For every application, the associated hosts can have different levels of heterogeneity, errors, availability, and reliability. Using EmBOINC, different patterns for job generations as well as different policies for job distribution and validation can be studied.

Contrary to what happens in BOINC projects, EmBOINC simulated hosts do not execute the job instances.

Since jobs are not actually executed, there is no real exchange of files between the simulated host and the BOINC server (download and upload are no longer needed). To maximize fidelity, the actual BOINC production code has been adapted to use simulated time instead of real time when running EmBOINC simulations. For these simulations, signals are used to synchronize the several daemons with the simulated time and maintain the correct flow of events at the server level.

Among its several features, BOINC has an administrative interface that project administrators use to access the BOINC database, add or remove platforms, check the status of the jobs (e.g., how many jobs are unsent or in progress, or successfully/unsuccessfully completed, are valid, or invalid). This interface has been extended to allow the EmBOINC user to monitor simulated project in a familiar environment as in real BOINC projects.

Figure 1 shows the traditional server and client interaction in a BOINC platform (left) and the changes to the platform due to EmBOINC (right). The figure on the right points out the two main differences that have been introduced with EmBOINC to the BOINC platform. First, we replaced the BOINC clients with a simulated population of hosts. Second, we extended the BOINC interface to access information of the simulated project at runtime.

### 3.2 Event Organization

In EmBOINC, the chain of events describing the behavior of the simulated hosts is managed by a discrete event simulator (DES). Like many other discrete event simulators, this simulator consists of events, entities, and resources [7, 10, 9, 6].

In general, entities are units of traffic that trigger events [7]. In EmBOINC the main entities are: *Init Entity*, *EmBOINC DES Controller*, *Host Generator*, *Simulated Host*, *Job Generator*, and *Simulated User*. Simulated host and simulated user are also referred as host and user respectively. Entities are characterized by attributes and are stored in a priority queue. The priority queue orders entities based on their next activation time. Each entity is associated with one or more events. Events are actions triggered by the associated entity at a certain point in time that cause changes in the simulated states, consuming one or more resources and changing the priority queue. The processing of an event may cause the triggering entity to be removed from the queue and one or more new entities to be added to the queue at certain points in time.

A queue of events associated with EmBOINC entities is shown in Table 1. The table also shows the sequence in which events are triggered when an entity is at the top of the priority queue. In the table, entities are in bold; an event is denoted by a number; a group of events is enclosed in

**Table 1. Table of events triggered by entities**

<b>EmBOINC Init</b>	<b>Job Generator</b>
1 Set next time	5 Read traces
2 Get entity's time	6 Read job
3 Enqueue	7 Make job
4 Terminate	
<b>EmBOINC DES Controller</b>	<b>Host Generator</b>
5 Read configuration	5 Read traces
6 Read traces of users	6 Read host
7 Read user	7 Make host
8 Make user	
9 Update simulated time	<b>Simulated Host</b>
10 Pop entity from queue	5 Write request
11 Execute entity	6 Send request
12 Call transitioner	7 Parse reply
13 Call feeder	8 Get job
14 Call validator	9 Get job instance
15 Call assimilator	10 Execute job instance
16 Wait for signal	11 List current job load
17 Continue simulation	12 Calculate requested job
<b>Sequence of Events</b>	
EmBOINC DES Controller:	
5, 6, (7, 8) <sup>+</sup> , 9, ((10, 9, 11, [12, 13]) <sup>+</sup> , 12, 13, 14, 15, 16, 17) <sup>+</sup> , 4	
Work Generator: (2, 5, (6, 7) <sup>+</sup> , 1, 3) <sup>+</sup> , 4	
Host Generator: 2, 5, (6, 7) <sup>+</sup> , 4	
Simulated Host: ((2, (10, 12)*, 11, 5, 6, 7, (8, 9)*)*, 1, 3) <sup>+</sup> , 4	
Notation: * 0 or more, <sup>+</sup> 1 or more, () group, [] optional group	

parenthesis; a group of events that may or may not occur depending on the entity state is enclosed by brackets; a star denotes an event or group of events occurring zero or more times; and a plus sign denotes an event or group of events occurring one or more times. The entity *Simulated User* is a passive entity and does not trigger any event therefore is excluded from the table.

Resources in EmBOINC include jobs and job instances. These resources are generated in response to a scheduler reply assigning a specific set of job instances to a host. Simulated resources match the properties of jobs and job instances in the BOINC server database. Once the host has “completed” the execution of a given job instance, its result is returned to the server on a request message. The server responds with an acknowledgment in the reply message. Once the acknowledgment is received by the server, the resources and the memory associated with them are freed from the simulator.

### 3.3 Host Modeling

An important component of EmBOINC is the modeling of the simulated host including their non-dedicated nature. To simulate delays in job execution and host connection, EmBOINC uses Gaussian and Weibull statistical distributions respectively. To determine the final status of a job

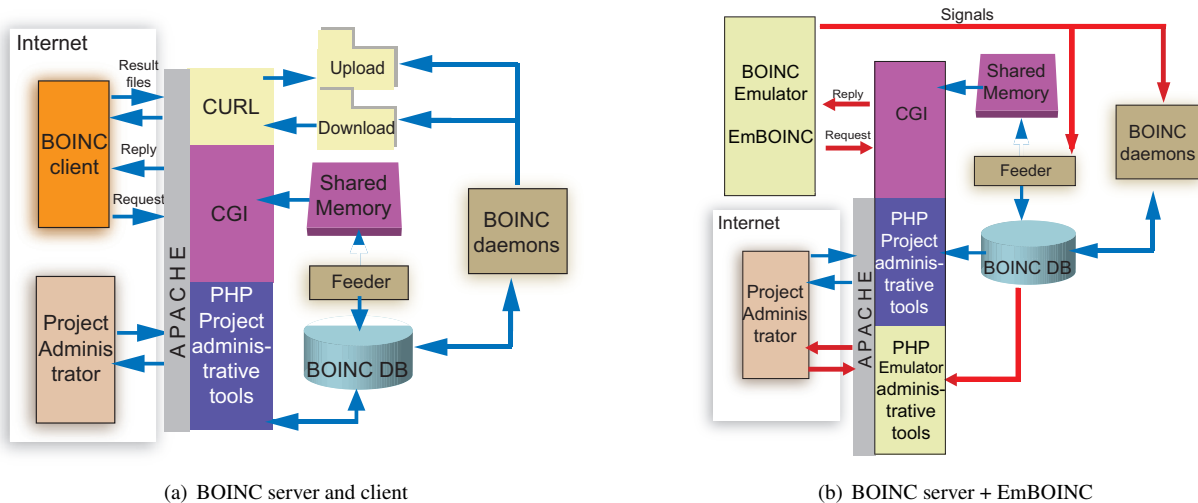


Figure 1. BOINC platform with and without EmBOINC

(i.e., to decide if the host produces an error, or a valid or invalid result), EmBOINC uses discrete uniform distributions. These distributions are calculated individually per host. They can be obtained from real BOINC traces or generated using synthetic statistical information. The statistical behavior of BOINC hosts has been studied and validated in [5]. Figure 2 shows the four main statistical distributions used to model a simulated host.

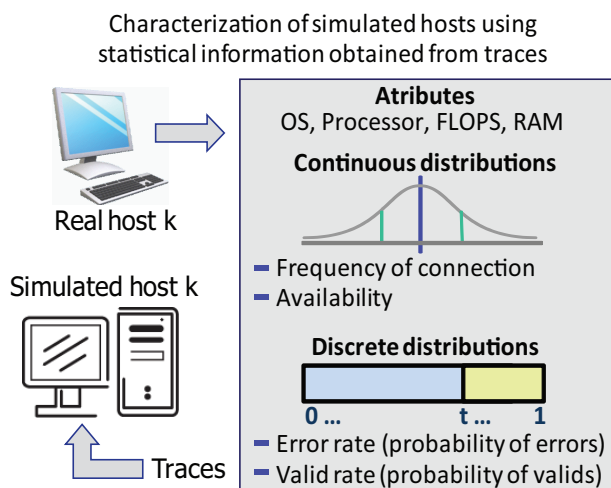


Figure 2. Statistical models used in simulated hosts

### 3.4 Performance Metrics

Performance metrics in BOINC should not be generalized across projects. Different projects have different re-

source needs and performance expectations. With EmBOINC it is possible to conduct extensive experiments targeting different performance metrics. BOINC metrics that can be captured by EmBOINC can be group in three classes: throughput-based, latency-based, and starvation-based metrics. In particular, EmBOINC measures the following metrics:

**Throughput-based metrics** include *project throughput* (or number of valid jobs per project) and *application throughput* (or number of valid jobs per application).

**Latency-based metrics** include distribution latency (or time from the job generation to the job instance distribution), in-progress latency (or time from the job distribution to the collection of job results), execution latency (or time the job is executed on the host), and validation latency (or time from the result collection to its validation).

**Starvation-based metrics** measures the capability of the BOINC server to generate and distribute suitable jobs for the inquiring hosts. Host starvation can occur when: (1) no work is available; (2) resource mismatching (e.g., the host does not have a sufficiently large disk space); and (3) policy constraints (e.g., homogeneous redundancy conditions are not fulfilled [12]).

Simulations can run either over a fixed amount of simulated time or until a selected number of metrics converge to defined thresholds. In both cases we can collect the performance metrics during and at the end of the simulated interval. Metrics are stored in log files and can be accessed through the EmBOINC interface.

## 4 Validation

To validate the performance predictions of EmBOINC, we used traces from three World Community Grid applications: Human proteome folding, Genome comparison, and FightAIDS@Home (<http://www.worldcommunitygrid.org>). Traces were used to build the simulated hosts and their statistical behavior, as well as the workload specifications of the project. For each application, we simulated 32 days: during this period of time, the three applications were running in parallel with 46,000 active hosts, 155,00 users, and 1,500,000 results. The main EmBOINC requirements were one PC with at least 2GByte of RAM running the BOINC server. The 32-day EmBOINC simulations took in average only 20 hours.

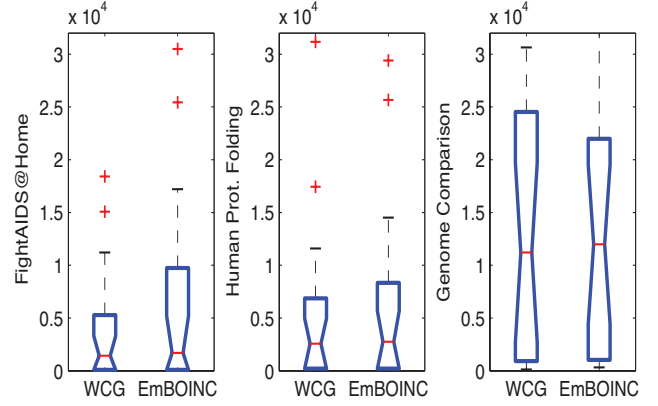
We validated EmBOINC in terms of throughput-based metrics (i.e., jobs distributed and collected) and latency-based metrics (i.e., execution latency). With reference to the throughput-based metrics, we first compared the statistical behavior of jobs distributed per day and jobs collected per day in EmBOINC simulations versus the same data from the traces of WCG. Figures 3 and 4 show an estimation of the medians with a 95% of confidence. In all the three cases the notches of EmBOINC overlap with the notches of the real BOINC project. The overlapping indicates that EmBOINC results are statistically equivalent to the trace data. Then we compared the number of jobs distributed and collected on a day-per-day basis. Figures 5 and 6 show this comparison. In these two figures EmBOINC can properly track and closely follow the changes in number of jobs distributed per day and in number of results collected per day.

With reference to the latency-based metrics, Figure 7 shows the comparison of the execution latency for EmBOINC simulations versus World Community Grid traces. More in particular, the figures present the number of jobs with a given execution latency (in hours) for the three World Community Grid applications when executed on a BOINC platform (left) and with EmBOINC (right). The figure shows how both the latency of EmBOINC jobs and their number closely match the latency and number in the three real BOINC projects considered.

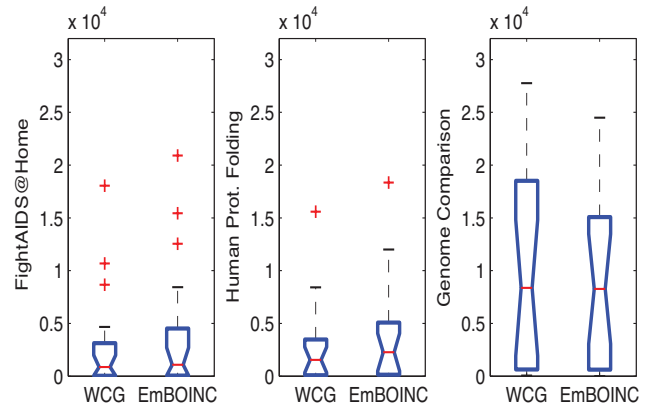
## 5 Related Work

### 5.1 Grid Simulators

Grid systems are different from VC environments in general and BOINC systems in particular. Generally, grid systems share computing resources within and between organizations such as universities, research labs, and companies. In contrast, BOINC systems are based on volunteer resources and, therefore, are not predictable or controllable by the designers and administrators of BOINC projects.



**Figure 3. Statistics of job distributed per day with EmBOINC and real BOINC projects**



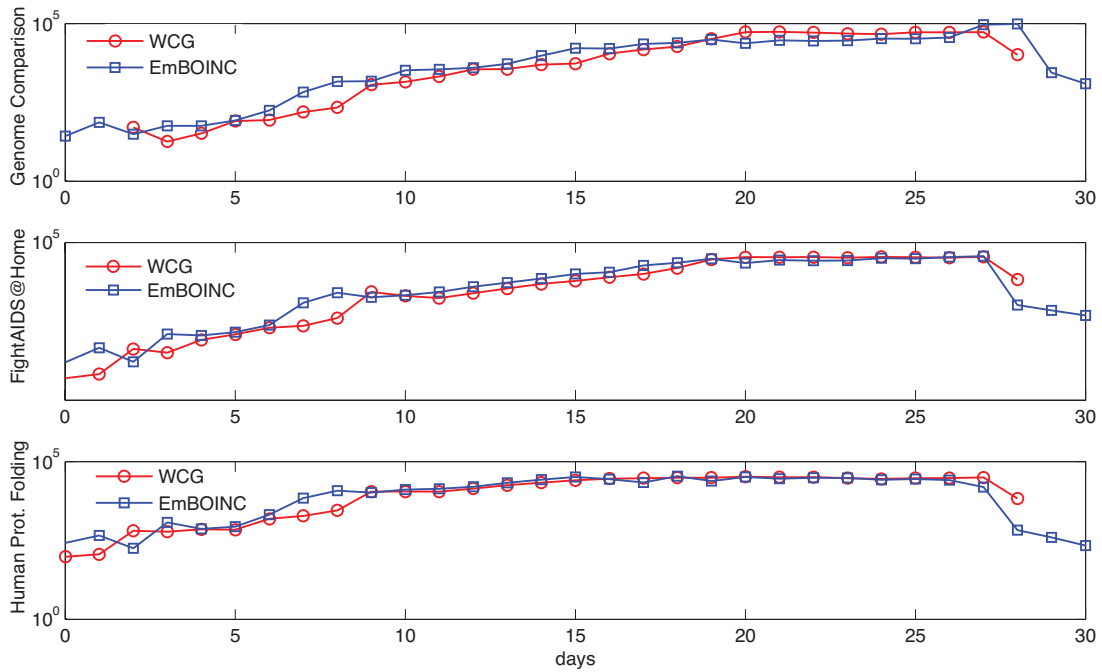
**Figure 4. Statistics of job collected per day with EmBOINC and real BOINC projects**

The performance of grid applications has been studied in the past using grid simulators such as SimGrid [3] and Grid-Sim [11] or emulators such as MicroGrid [13]. Contrary to EmBOINC, these simulators and emulators do not capture the characteristics of BOINC projects since they have not been specifically tailored for highly volatile, heterogeneous, and error-prone environments as BOINC projects are.

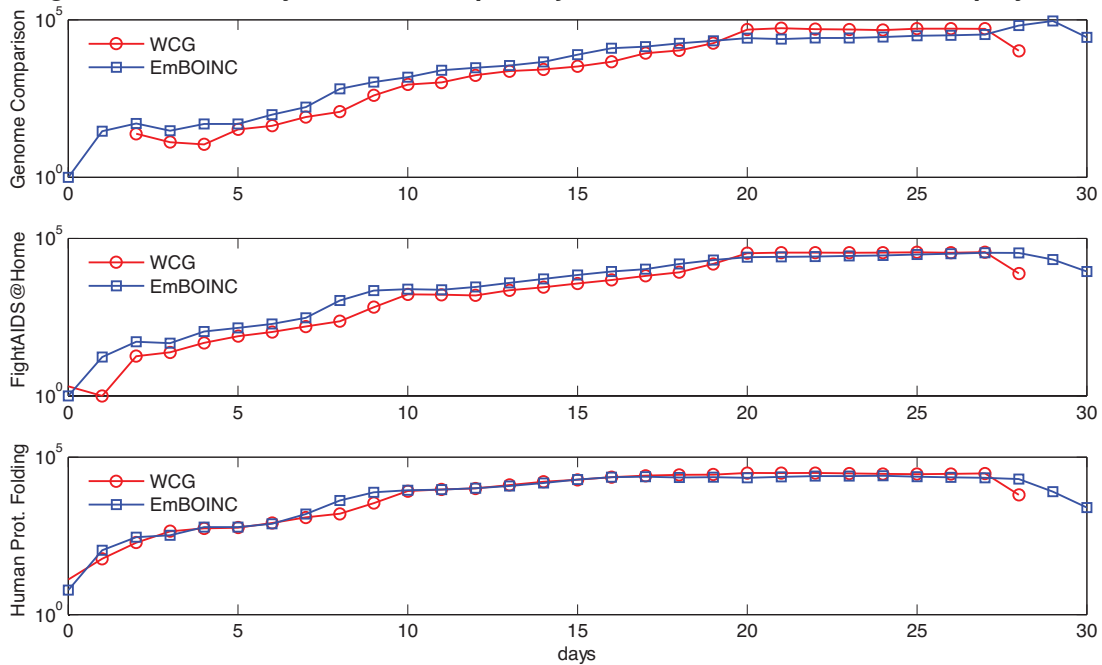
### 5.2 BOINC Simulators

While there are several grid simulators and emulators that have been used to implement and evaluate performance on grid computing environments, to our knowledge little has been done in terms of simulating BOINC systems. There are two main BOINC simulators: SimBOINC and SimBA.

SimBOINC [8] simulates the *BOINC client scheduler*. This simulator is based on the SimGrid toolkit and uses



**Figure 5. Number of job distributed per day with EmBOINC and real BOINC projects**

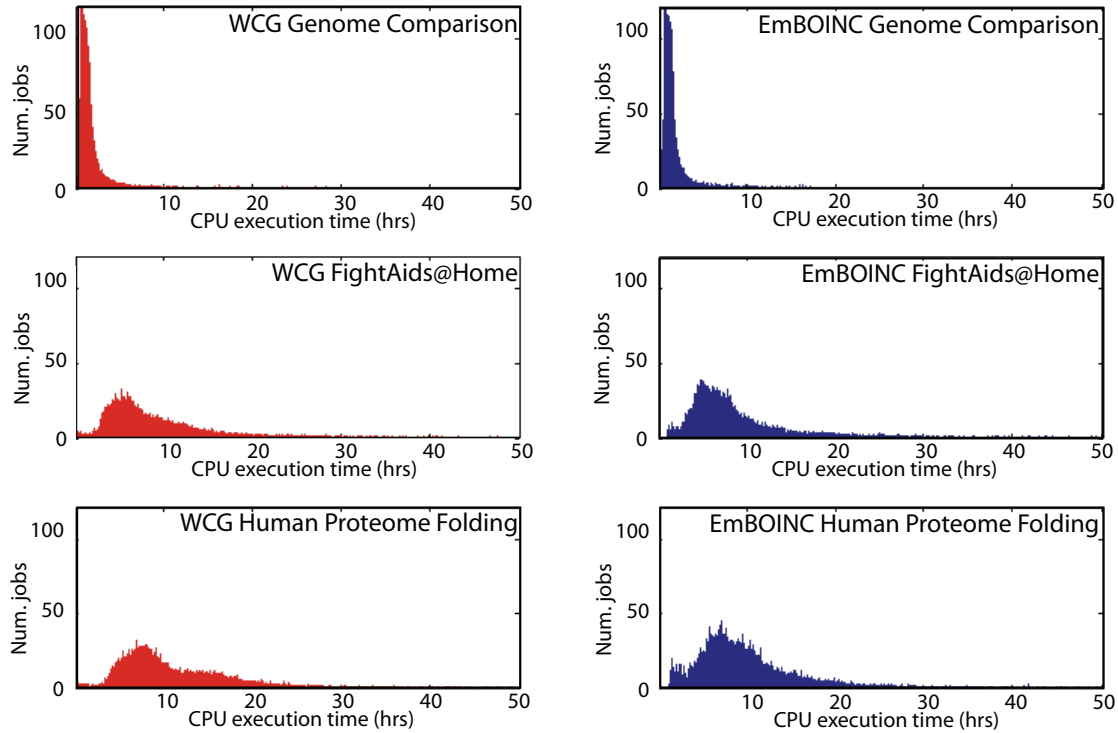


**Figure 6. Number of job collected per day with EmBOINC and real BOINC projects**

traces obtained from real BOINC clients. Its goal is to study and replicate the way in which job instances are scheduled and executed inside a single BOINC client. Since SimBOINC simulates BOINC from the client perspective, aspects like work generation, replication, and validation of

results are not feasible.

In contrast to SimBOINC, SimBA [4] simulates the *BOINC server scheduler* and its interaction with a large number of simulated hosts. As EmBOINC does, SimBA simulates the generation and distribution of jobs that are ex-



**Figure 7. Execution latency with EmBOINC (right) and a real BOINC project (left)**

ecuted in a highly volatile, heterogeneous, and distributed VC environment. It also simulates the collection and validation of completed jobs. SimBA uses real traces from existing BOINC projects. Contrary to EmBOINC, the simulated BOINC environment is completely embedded into the SimBA’s discrete event simulator. While this may be an advantage, because SimBA does not have dependencies with BOINC or any other system, it is also a disadvantage because BOINC changes constantly, and those changes are not immediate in SimBA: changes in the simulator are required and can demand time. In addition, policies in BOINC are implemented as abstract models in SimBA. The accuracy of those policies depends greatly on the developer’s interpretation. Considerable testing is required before accepting those policies as reliable simulations of BOINC policies. We used the knowledge and experience of building SimBA to design and implement EmBOINC. The EmBOINC structure makes it more robust, accurate, easy to use, and easy to maintain. The core discrete event simulator of SimBA has been integrated in EmBOINC, but this time for emulating the interaction of the simulated hosts with a real BOINC server. The emulation ensures that any change made in the BOINC server are immediately available to EmBOINC and that BOINC policies are mapped one-to-one in EmBOINC.

Table 2 compares the features available in EmBOINC, SimBA, and SimBOINC. If a feature is marked with a dash,

then that feature is not simulated or is not available. The table also shows when a feature is simulated or emulated, and when a feature is characterized by values from traces or by static input values

## 6 Conclusions and Future Work

In this paper we presented EmBOINC, an emulator of BOINC applications. EmBOINC is easy to install: as easy as creating a new BOINC project. It is also easy to use: the selection of host population and parameters can be done through the BOINC interface. The analysis of results can also be done through the BOINC Web interface. Last but not least EmBOINC is easy to maintain: EmBOINC is integrated directly into BOINC code and can be used with the very last version of the BOINC server (BOINC scheduler and daemons). Its integration in BOINC makes EmBOINC convenient for testing and tuning: every policy tested or tuned with EmBOINC works unchanged with BOINC. The comparison of EmBOINC generated BOINC project with real BOINC traces shows its accuracy in capturing the behavior of the real projects.

Overall EmBOINC supports the project administrator in answer important questions such as: Given an application, what replication and scheduling policy are best? Can we detect malicious attackers? What if attackers team? Given

**Table 2. Comparison of BOINC simulators**

	<b>EmBOINC</b>	<b>SimBA</b>	<b>SimBOINC</b>
Server scheduler	emulation	simulation	-
Work generation	emulation/simulation	simulation	-
Work collection	emulation	simulation	-
Work validation	emulation	simulation	-
Number of hosts	> 100000	< 50000	1
Host scheduler	-	-	simulation
Host execution	traces+uniform dist	traces+uniform dist	simulation
Host connection	traces+weibull dist	traces+uniform dist	simulation
Host availability	traces+uniform dist	traces	simulation
Host reliability	traces+uniform dist	traces	-
Number of jobs	> 350000	< 200000	-
Number of job instances	> 2millions	< 1million	-
Workload specifications	traces	input	traces
Size of jobs	traces	input	traces
Replication factor	traces	traces	-
Number of users	> 15000	-	-
Reliability of users	traces	-	-
Number of applications	N	1	N

a set of jobs and hosts, how much time do scientists shall wait to get all their results?

Work in progress include (1) the design of more efficient policies for jobs generation, jobs distribution, and jobs validation in existing BOINC projects and (2) the testing of policies in extreme conditions such as extremely unreliable/unavailable community, extremely fast/slow turnaround jobs, and community with many malicious attackers is future work.

## Acknowledgment

This work was supported by the National Science Foundation, grant #OCI-0802650, "DAPLDS - a Dynamically Adaptive Protein-Ligand Docking System based on multi-scale modeling" and by the CONACyT fellowship #171595. The authors thank the BOINC community for their time, dedication, and computer resources.

## References

- [1] D. P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *Proc. of the 5th IEEE/ACM International Workshop on Grid Computing*, 2004.
- [2] D. P. Anderson and K. Reed. Celebrating Diversity in Volunteer Computing. In *Proc. of the Hawaii International Conference on System Sciences (HICSS)*, 2009.
- [3] H. Casanova. SimGrid: A Toolkit for the Simulation of Application Scheduling. In *Proc. of the 1st International Symposium on Cluster Computing and the Grid*, 2001.
- [4] T. Estrada, D. Flores, M. Taufer, P. Teller, A. Kerstens, and D. Anderson. The Effectiveness of Threshold-based Scheduling Policies in BOINC Projects. In *Proc. of the 2nd IEEE International Conference in e-Science and Grid Computing*, 2006.
- [5] T. Estrada, M. Taufer, and K. Reed. Modeling Job Lifespan Delays in Volunteer Computing Projects. In *Proc. of the 9th IEEE International Symposium on Cluster Computing and Grid (CCGrid)*, 2009.
- [6] F. O. Gathmann. Python as a Discrete Event Simulation Environment. In *Proc. of the 7th International Python Conference*, 1998.
- [7] R. Ingalls. Introduction to Simulation. In *Proc. of the 2002 Winter Simulation Conference*, 2002.
- [8] D. Kondo. *Scheduling Task Parallel Applications for Rapid Turnaround on Enterprise Desktop Grids*. PhD thesis, UCSD, 2004.
- [9] T. J. Schriber and D. T. Brunner. Inside Discrete-Event Simulation Software. In *Proc. of the 2003 Winter Simulation Conference*, 2003.
- [10] R. E. Shannon. Introduction to the Art and Science of Simulation. In *Proc. of the 1998 Winter Simulation Conference*, 1998.
- [11] A. Sulistio, G. Poduvaly, R. Buyya, and C. Tham. Constructing a Grid Simulation with Differentiated Network Service using GridSim. In *Proc. of the 6th International Conference on Internet Computing*, 2005.
- [12] M. Taufer, D. Anderson, P. Cicotti, and C.L. Brooks III. Homogeneous Redundancy: a Technique to Ensure Integrity of Molecular Simulation Results Using Public Computing. In *Proc. of the 14th Heterogeneous Computing Workshop*, 2005.
- [13] H. Xia, H. Dail, H. Casanova, and A. Chien. The MicroGrid: Using Emulation to Predict Application Performance in Diverse Grid Network Environments. In *Proc. of the Workshop on Challenges of Large Applications in Distributed Environments*, 2004.