SimBA: a Discrete Event Simulator for Performance Prediction of Volunteer Computing Projects

Michela Taufer, Andre Kerstens, Trilce Estrada, David Flores, and Patricia J. Teller
Computer Science Department
College of Engineering
University of Texas at El Paso
El Paso, TX 79968
{*mtaufer, akerstens, tpestrada, daflores, pteller*}*@utep.edu*

Abstract

SimBA (Simulator of BOINC Applications) is a discrete event simulator that models the main functions of BOINC, which is a well-known framework used in Volunteer Computing (VC) projects. SimBA simulates the generation and distribution of tasks that are executed in a highly volatile, heterogeneous, and distributed environment as well as the collection and validation of completed tasks.

To understand the strengths and weaknesses of BOINC under distinct scenarios, project designers want to study and quantify project performance without negatively affecting people in the VC community. Although this is not possible on production systems, it is possible using SimBA, which is capable of testing a wide range of hypotheses in a short period of time. Our experience to date indicates that SimBA is a reliable tool for performance prediction of VC projects. Preliminary results show that SimBA's predictions of Predictor@Home performance are within approximately 5% of the performance reported by this BOINC project.

1. Introduction

The Volunteer Computing (VC) community consists of people, called volunteers, that donate computer and storage resources to scientific projects, and scientists who utilize the PetaFLOP-scale aggregation of these resources for large-scale simulations. Volunteers expect that their computer resources are used for meaningful computations and that their processor cycles are not wasted, while scientists expect to receive reliable results. In order to use the strengths and contain the weaknesses of VC environments for different scientific projects, project designers need to study the efficiency of different project parameters and scheduling policies without affecting the VC community. However, several challenges arise in doing this within VC environments:

- The time required to measure project throughput, total number of results delivered to the scientists in a given amount of time, under different parameter settings can be significant.
- Problems due to testing might upset volunteers, even to the degree that they leave a project.
- Every experiment is unique and unrepeatable, making the generalization of conclusions difficult.

Consequently, it is preferable to conduct performance studies in simulation environments, where it is possible to test a wide range of hypotheses in a short period of time without affecting the VC community.

The need for mechanisms to help efficiently design VC projects and the challenge of testing new strategies on in-production VC environments are the main forces that have driven us to design and implement SimBA (Simulator of BOINC Applications), a discrete event simulator that accurately models the main functions of VC environments. SimBA, the main focus of this paper, has been designed to mimic BOINC [1], a well-known, master-worker, runtime framework used in Volunteer Computing (VC) projects. SimBA simulates the generation and distribution of tasks executed in highly volatile, heterogeneous, and distributed environments, as well as the collection and validation of completed tasks. It allows project designers and administrators to extensively study improvements in VC projects, such as new scheduling strategies and validation techniques, without affecting the VC community. SimBA is built upon a modular framework that can easily be extended to include new features and to model a wide range of VC projects: distinct SimBA modules implement various flavors of events. This makes SimBA a general-purpose, discrete event simulator for BOINC environments.

The remainder of this paper is organized as follows: Section 2 presents background and related work on volunteer computing systems and simulators of these systems. SimBA's modular framework and usage are described in Section 3. In Section 4 we validate SimBA by comparing its functionalities with those of BOINC and comparing the simulated results of SimBA with those of a real VC project, Predictor@Home (P@H). Section 5 concludes the paper and presents some future work.

2. Background and Related Work

2.1. Volunteer Computing and BOINC

Volunteer Computing environments employ computing resources (e.g., desktops, notebooks, and servers) owned by the general public and connected through the Internet. Traditionally, VC projects have targeted large search problems in science and, therefore, usually generate large sets of work-units that are distributed across volunteer computing resources. Replication of work 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 work-units (also called work-unit instances) are distributed to different volunteer computing resources (also called workers) that execute them. When finished, the workers 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 when results associated with the same group of work-unit instances agree, and identifying them as invalid when they have been negatively affected by e.g., malicious attacks, hardware malfunctions, or non-authorized software modifications.

BOINC (Berkeley Open Infrastructure for Network Computing) [1] is a well-known representative of VC environments. It is an open-source system that harnesses the computing power and storage capacity of thousands or millions of PCs owned by the general public for large-scale scientific simulations. The computing resources available to a BOINC project are highly diverse: the workers differ by orders of magnitude in their processor speed, available RAM, disk space, and network connection speed. Some workers connect to the Internet by modem only every few days, while others are permanently connected. Computations may have varying bounds on completion time, and in some cases computations are aborted by the project server or by the volunteers. Nevertheless, BOINC projects are highly attractive for large-scale simulations because they can potentially sustain a very high processing rate (tens or hundreds of TeraFLOPS). For example, SETI@Home now runs on approximately one million computers, providing a sustained processing rate of about 250 TeraFLOPS [2]. Currently BOINC supports several VC projects in Biology and Medicine, Mathematics and Strategy Games, Astronomy/Physics/Chemistry, and Earth Sciences.

2.2. Grid and VC Simulators

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

While there are several grid simulators (e.g., SimGrid [4] and GridSim [12]) that have been used to implement and evaluate performance on grid computing environments, to our knowledge little has been done in terms of simulating VC environments. The performance of grid applications also has been studied in the past using grid emulators such as MicroGrid [15]. In contrast to SimBA, these simulators and emulators do not capture the characteristics of VC since they have not been specifically tailored for this kind of environment.

To our knowledge the closest work to that presented in this paper is the work of Kondo [9]. Kondo's SimBOINC is based on the SimGrid toolkit and uses traces of synthetic applications with short task turnaround times (on the order of minutes or seconds). In contrast to SimBOINC, SimBA uses real traces from existing BOINC projects such as Predictor@Home, of which the tasks are characterized by short, medium, and long turnaround times. The differences in the traces and the models of the simulated computing environments led Kondo to conclusions different from ours. In [9] he claims that the history of nodes does not affect the turnaround time if machines are prioritized and used according to their processor speeds. In contrast, in a dynamic environment such as the Internet, it has been shown that an a priori determination of worker priority is not possible and the history of the workers/machines plays a key role in determining turnaround time [5].

2.3. Predictor@Home

Predictor@Home, or P@H, [13] is a BOINC project for large-scale protein structure prediction. The protein structure prediction algorithm in P@H is a multi-step pipeline that includes: (a) conformational search using a Monte Carlo simulated-annealing approach using MFold [8] and (b) protein refinement, scoring, and clustering using the CHARMM Molecular Dynamics simulation package [3]. In

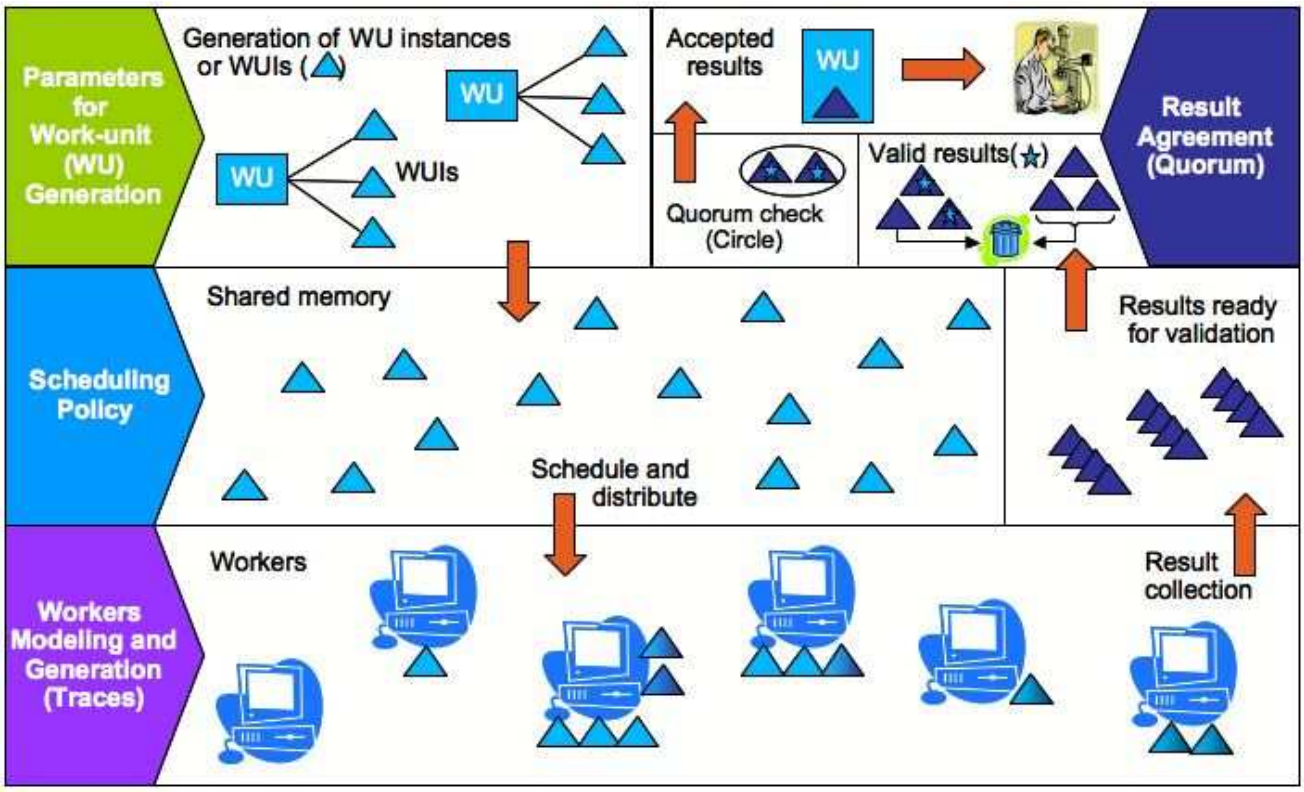

Figure 1. Overview of SimBA

this paper, these two applications serve as our reference applications.

3. SimBA Framework and Usage

SimBA (Simulator of BOINC Applications) is a sequential, discrete event simulator, written in Python, that models the behavior of a BOINC server while dealing with volatile, error-prone, and highly heterogeneous workers.

3.1. SimBA Overview

As shown in Figure 1, SimBA realistically simulates the creation, characterization, and termination of workers by using trace files obtained from real BOINC projects, e.g., P@H [13]. A trace file contains information that characterizes workers, e.g., creation time, processor architecture, operating system (OS), and life span. In general, SimBA:

- generates work-units and creates for each work-unit a number of instances or replicas according to the project replication policy;
- distributes instances according to worker requests and the selected scheduling policy;
- models worker volatility and heterogeneity using the worker's characterization obtained from the trace file;
- determines the status of a worker's returned results (successful or unsuccessful, i.e., erroneous) using the worker's error rate characterization obtained from the trace file;
- determines the validity of successfully completed results using a quorum, i.e., the required number of results in agreement as set by the project's validation policy; and
- computes the performance of the simulated VC project in terms of throughput.

3.2. SimBA Framework

Like many other discrete event simulators, SimBA consists of events, entities, and resources [7, 11, 10, 6]. Entities are units of traffic that generate events, which cause changes in the state of the simulation [7]. In SimBA the main entities are: *worker generator (WG)*, *worker*, *work-unit generator (WUG)*, *work-unit (WU)*, *work-unit instance (WUI)*, and *sampler*. Entities are characterized by attributes. Figure 2

presents two examples of the attributes of two entities of SimBA: a *worker* and a *work-unit*.

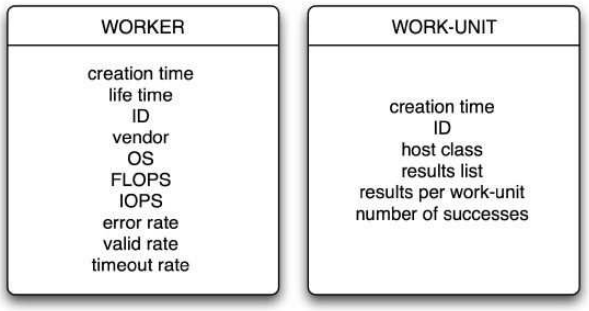

Figure 2. Examples of entities and their attributes in SimBA

Most worker attributes are defined using real BOINC project traces. Creation time is the time that the worker first attached to the project, while life time is the time between when the worker attached to the project and the time when it left the project. ID is a number that is generated by SimBA to internally keep track of a worker. Vendor determines the processor architecture of the worker (i.e., Intel, AMD, or PowerPC), while OS determines the operating system that runs on the worker (i.e., Linux, Windows, or Mac OS X). FLOPS provides the number of floating-point operations per second and IOPS provides the number of integer operations per second. The error rate represents the number of erroneous results returned by a worker divided by the number of *WUI*s sent to it. The valid rate is computed by dividing the number of valid results returned by a worker by the number of *WUI*s successfully returned by that worker. The timeout rate is given by the number of results that timed-out (were not returned by a worker within the specified deadline) divided by the number of *WUI*s unsuccessfully returned by the worker. A trace file characterizes, and SimBA uses during the simulation, the exact number of workers that were active in the original BOINC project for the given time interval.

SimBA generates all work-unit attributes. Creation time is the time that the work-unit is generated by the simulator. ID is a number that is generated by SimBA to internally keep track of a work-unit. Host class is a field used by the homogeneous redundancy scheduler [14] to store the type of worker that is assigned the first instance of the work-unit. Results list shows the instances that are generated for the work-unit. Results per work-unit is the number of valid results that are needed to generate a canonical result for the work-unit (quorum) and number of successes is the number of successfully completed (not necessarily valid) results returned for the work-unit.

An event is a condition that occurs at a certain point in time and that causes changes in the simulated state. Events are triggered by entities stored in an entity list; the processing of an event may cause one or more entities to be added to the list. The main events in SimBA are:

1. *generate worker*: triggered by the *worker-generator* entity; generates a new *worker* entity that is added to the entity list. The worker's attributes are garnered from traces of real VC projects.

2. *generate work-unit*: triggered by the *work-unit generator* entity; creates a new *work-unit* entity that is added to the entity list.

3. *generate work-unit instance*: triggered by the *work-unit* entity; generates one or more new *work-unit instance(s)* that is/are stored in the shared memory of the project server. The shared memory is a resource with limited capacity that holds unassigned instances of a work-unit until they are distributed to workers.

4. *assign work-unit instance*: triggered by the *worker* entity; retrieves a *work-unit instance* from the shared memory and assigns it to a worker based on the selected scheduling policy.

5. *determine instance output*: triggered by the *work-unit instance* entity; determines if an instance was returned successfully or not and, if returned successfully, determines if it is valid or not. Different validation methods, ranging from bit-to-bit to fuzzy comparisons, can be applied in VC projects.

6. *check if simulation is over*: regularly triggered by the *sampler* entity; determines whether the simulation has ended.

Figure 3 shows how events in SimBA are triggered by entities, and how events cause entities to be added to the entities list. Curly braces indicate that one or more entities can be generated from an event.

Figure 4 depicts the SimBA shared memory resource. This resource matches the shared memory segment of the BOINC environment and stores work-unit instances that have not been assigned to workers. When workers request computation, unassigned work-unit instances from the shared memory resource are assigned to these workers and removed from this resource. The space that is freed up by assigned instances is filled with new unassigned instances as soon as a work-unit entity causes the *generate work-unit* instances event.

The simulation time of SimBA is not associated with the actual execution time of the simulated VC project. It depends on the entities, i.e., workers, work-units, and associated work-unit instances, in the project, as well as the

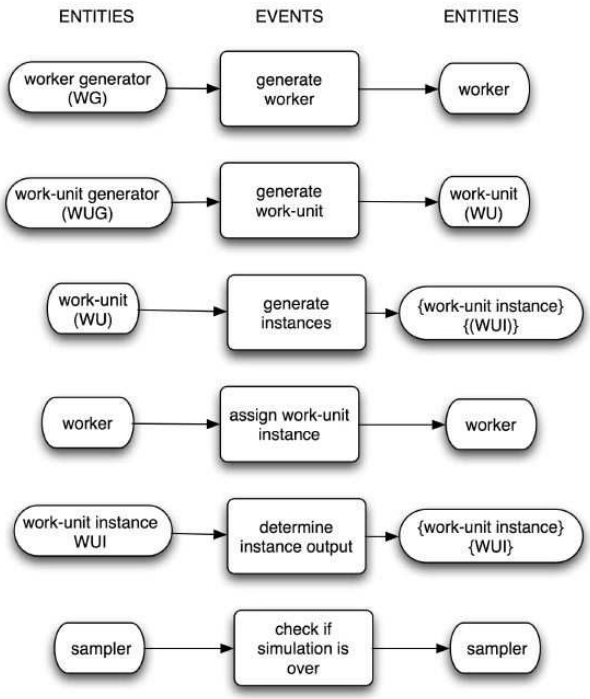

Figure 3. Sequences of SimBA events and entities

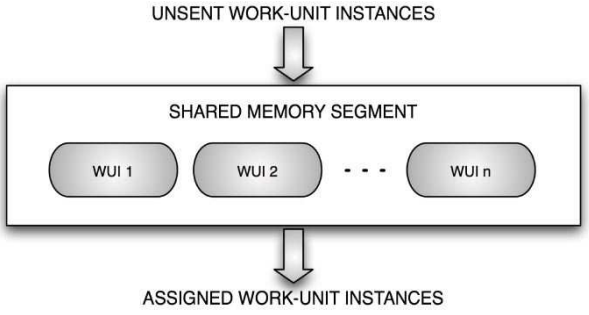

Figure 4. Shared memory resource in SimBA

execution times of the project's events. This information is obtained from the project trace. Entities are put in the entities list before the associated events actually are generated. For performance reasons this list is implemented as a heap. Scheduling polices are part of the *assign work-unit instance* event. Currently, SimBA includes several scheduling polices; some are already included in BOINC projects and others are under study for future deployment. Those included are homogeneous redundancy [14], which is used in projects such as P@H and the World Community Grid (http://www.worldcommunitygrid.org), the First-Come-First-Serve (FCFS) policy, which is used in most of the BOINC projects including SETI@HOME, and new fixed and adaptive threshold-based scheduling policies, which we suggested as effective scheduling polices for VC in [5]. Modules implement different flavors of the *assign work-unit instance* event: for example, based on the configuration of the simulator, the *assign work-unit instance* event assigns an instance following a strict homogenous redundancy policy or a First-Come-First-Serve scheduling policy.

SimBA reproduces the unpredictable behavior of VC environments through built-in randomness. In particular, SimBA uses random distributions for two purposes:

- *To determine the final status of an instance*: SimBA uses a standard uniform distribution to assign the final status of an instance, i.e., error, timed-out, valid, or invalid.
- *To simulate the non-dedicated nature of a worker in VC*: SimBA uses a Gaussian distribution to emulate delays in instance executions and worker network connections due to user interruptions.

These random mechanisms enable SimBA to behave in a non-deterministic way, mimicking a real VC project. Both distributions are calculated individually per worker using the characterization obtained from real BOINC projects in the traces.

Figure 5 shows a didactic example of a short simulation performed by SimBA. The content of the shared memory resource also is shown in the figure. The unit of time is one hour and the simulation ends after 10 hours of simulated time. The sampler entity adds itself to the list so that it is activated every five hours (the sample period) to check whether the simulation is over or not. At *t=0*, an empty shared memory resource is created and the three entities required to start a simulation are added to the list: a sampler, a work-unit generator (*WUG*), and a worker generator (*WG*). All of these are inserted into the entity list with an event timestamp, t, of 0, i.e., *t=0*. When the simulation starts at *t=0*, the first entity, the sampler, is taken from the list to check whether the simulation has completed. Since this is not the case, the sampler inserts itself back into the list with *t=5*. Still at *t=0*, the *WUG* generates a work-unit, *WU* 1, which in this example is inserted into the list with *t=2*. The *WUG* inserts itself back into the list with a timestamp that depends on the rate of *WU* creation; the SimBA user defines this rate. For this example we insert it back with *t=8*. This is followed by the *WG* that generates worker 1 with *t=3*. The *WG* is then inserted back into the list with a timestamp that depends on the rate of worker generation; in this example it is inserted with *t=12*. The worker generation rate is computed using the associated trace. In the next time step (*t=2*), *WU* 1 is processed. It generates two work-unit instances: *WUI* 1 and *WUI* 2, which, as can be seen in Figure 5, are both put into the shared memory. In

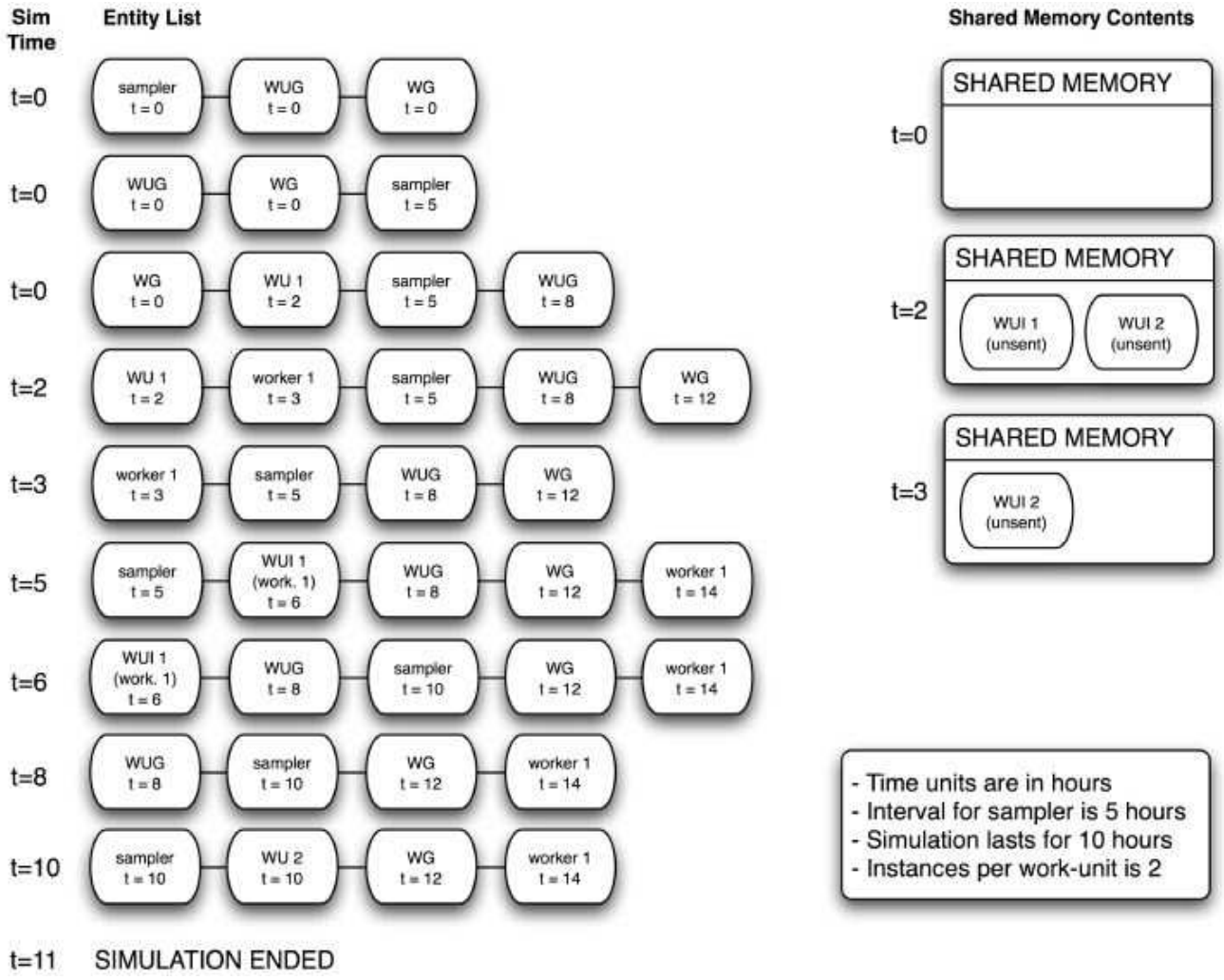

Figure 5. Example SimBA simulation

the next time step (*t=3*), worker 1 requests work and is assigned *WUI* 1. *WUI* 1 is removed from the shared memory resource after the assignment. An execution time for this *WUI* is calculated based on FLOPS information taken from the traces and the *WUI* is inserted accordingly in the list; for this example, execution is assumed to be complete at *t=6*. Then worker 1 inserts itself back into the list with a timestamp that is later than the sum of all the execution times of its pending results. In this example, it is inserted back with *t=14*, which is the time when it can request new *WUI*s. At *t=5*, the sampler checks whether the simulation has ended. Since we specified a simulation time of 10 hours, the sampler inserts itself back into the list with *t=10* (the current time plus the sample interval) to re-check for this condition. At *t=6 WUI* 1 has finished and the final status of this instance (i.e., success or error, valid or invalid) is determined using a statistical distribution. At *t=8*, the *WUG* generates *WU* 2, which is inserted at *t=10*. At the next timestep, i.e., *t=10*, the sampler checks whether the simulation has ended, which in this case is true. All entities that have a timestamp greater than *t=10* are not executed.

3.3. Inputs and Outputs

To run a simulation, SimBA requires the following input parameters:

- maximum number of work-unit instances simultaneously in the shared memory (MAX_SHM)
- total length of simulated time in hours (SIM_DURATION) and temporal resolution of the event queue (SAMPLE_INTERVAL)
- number of work-units generated before the first worker is generated (WU_INITIAL_POOL_SIZE)

- number of work-units generated per hour (WUS_PER_HOUR)
- upper limit for the number of work-unit instances that can be assigned when requesting computation (MAX_TASK_PER_WORKER)
- time that the BOINC server will wait for a result to return (TASK_DEADLINE)

The SimBA output includes the following data: total number of generated and distributed work-unit instances, total number of workers, total number of workers that disconnected from a project during the simulation, and the percentage of successful work-units. The output also shows how many of the distributed work-unit instances were valid, invalid, in error, timed-out, and successful. Among the successful ones, the output shows how many are valid and invalid, as well as how many were in progress when the simulation terminated.

4. SimBA Validation

4.1. Mapping Functionalities

Figure 6 compares the logic of SimBA and the logic of BOINC. At this high level of abstraction, both SimBA and BOINC perform exactly the same functions. Both create work-units (WUs), replicate them, and store these work-unit instances (WUIs) in the shared memory. Afterwards, the WUIs are distributed to workers requesting computation. After a certain amount of time, workers return results, at which point the validation process begins. Validation determines whether a quorum for a work-unit has been reached; this is based on the number of correct work-unit instance results returned successfully. If the quorum is reached (e.g., two out of three of the work-unit instances return the same result), the work-unit is marked as completed and credit is granted to those workers who returned correct results for the work-unit. A gray block in Figure 6 indicates that, for SimBA, the group of actions enclosed by the block is executed by one entity. For BOINC, each action is performed by a corresponding daemon. Note that in Figure 6, the blocks in BOINC corresponding to the handling of files are not present in SimBA because they are not needed in a simulation. Since SimBA is not simulating the actual computations done by the workers, there is no need to upload input files to workers from the server, upload output files from workers to the server, or to delete input files.

Since the simulator is meant to explore the design space, at the design level, there are major differences between SimBA and BOINC. First, SimBA is capable of simulating different scheduling policies and different intra- and inter-policy adaptations based on changes in the characterization of workers. In contrast, BOINC usually uses a First-Come-First-Serve scheduling policy with the possibility of enabling or disabling homogeneous redundancy and the standard BOINC worker punishment, respectively (for more information about this see http://boinc.berkeley.edu). Second, SimBA simulates workers joining and leaving a project based on traces from real-life projects. This allows us to model a real-life environment while producing repeatable experiments for testing purposes. The behavior of workers joining and leaving a BOINC project is completely unpredictable.

Different scheduling policies and different intra- and inter-policy adaptations in SimBA, like those defined in [5], can be implemented as plug-and-play modules that execute their own rules. A scheduling policy module defines the set of rules that determines, during the assign work-unit instances event, whether or not a particular worker is assigned work-unit instances. Intra- and inter-policy adaptations affect the distribution of work since an adaptation redefines the scheduling rules. Thus, the scheduling module affects the ”Schedule WUI” and ”Set WUI as in progress” blocks in the SimBA diagram.

4.2. Comparing P@H and SimBA Results

To audit SimBA, we traced SimBA outputs to ensure that no events were lost, events were executed in the correct order, and the number of results was consistent with the number of generated workers and work-units. We also compared the results of six simulator runs (each with different random-number generator seeds) with real P@H results. In Table 1, we report the average percentage of valid, invalid, error, and timed-out results as well as those still in progress at the end of the simulations as compared to the number of distributed work-unit instances. We consider both applications used by P@H: the conformational search using MFold and the protein refinement using CHARMM (as described in Section 2.3). Overall, as we can see in Table 1, the maximum difference between SimBA predictions and the real results of P@H is approximately 5%. Please note that the differences between SimBA predictions and P@H results are normalized with respect to the total number of work-units distributed and expressed in a percentage accompanied by the confidence interval. Since VC projects target search problems in large conformational spaces, the collective running time for the real science simulations on VC computers might range from days to weeks, or sometimes even months. Since SimBA does not simulate the computation, it is able to simulate a real science simulation run on VC computers in a relatively short time, i.e., in the range of minutes. As shown in Table 1, a conformational sampling using MFold that took 15 days to complete on

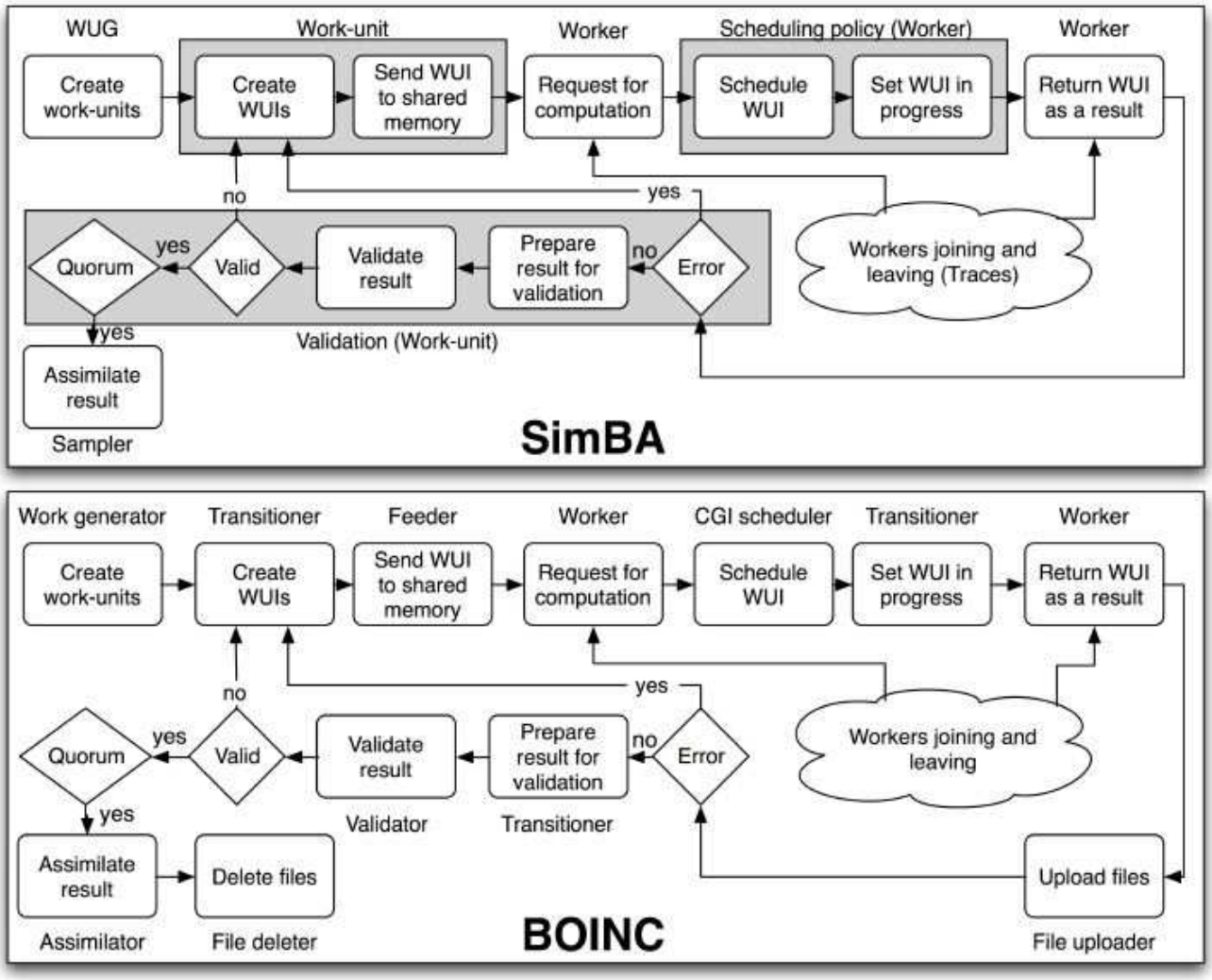

Figure 6. Mapping of the functions of SimBA with BOINC

P@H, a VC environment, is simulated in 107 minutes with SimBA; a protein refinement simulation using CHARMM that took 8 days on P@H takes only 44 minutes to simulate. SimBA simulations were performed on a 3.0 GHz Intel P4 with 1GByte of RAM. The traces of the real project contained 7810 active workers for MFold and 5093 workers for CHARMM. The data in Table 1 shows how the behavior of a VC project can be reliably reproduced in a short time.

5. Future Work and Conclusions

In this paper we present SimBA (Simulator of BOINC Applications), a discrete event simulator that accurately models the main functions of BOINC, which is a well-known master-worker framework used in Volunteer Computing (VC) projects. Given the dearth of work in the area of simulators of VC projects, SimBA represents a significant step in permitting designers and administrators of VC projects to identify and experiment with ways to enhance the performance of VC projects.

This paper describes SimBA's modular framework and its usage. Validation of SimBA, by comparing its functionalities with those of BOINC and comparing the simulated results of SimBA with those of a real VC project, the Predictor@Home (P@H) project, gives credence to its reliability for performance prediction of VC projects. For example, as shown in the paper, preliminary results show that SimBA's predictions of P@H performance are within approximately 5% of the performance reported by this BOINC project. Work in progress is evaluating the effectiveness of different scheduling policies, as well as intra- and inter-policy adaptations based on changes in the characterization

Table 1. Comparison of SimBA predictions with P@H results

	MFold P@H			CHARMM P@H		
	Real project (%)	SimBA simulation (%)	Difference and conf. interval (%)	Real project (%)	SimBA simulation (%)	Difference and conf. interval (%)
Distributed	100.00	100.00	0 ± 0.02	100.00	100.00	0 ± 0.01
Valid	71.89	67.94	3.9 ± 2.58	54.30	54.08	0.2 ± 0.32
Invalid	1.76	2.68	0.9 ± 1.42	5.78	8.52	2.7 ± 1.21
Error	1.88	6.53	4.6 ± 0.88	15.09	11.92	3.1 ± 0.54
Timed-out	13.83	14.81	0.9 ± 3.26	12.44	9.74	2.7 ± 2.16
In Progress	10.64	8.04	2.6 ± 1.72	12.39	15.74	3.3 ± 2.03
Execution Time	15 days	107 mins		8 days	44 mins	

of workers. Our experiments have shown that memory usage grows linearly with respect to the number of entities simulated and the running time grows polynomially; the assessment of SimBA's scalability is a work in progress.

Acknowledgement

This work was supported by the National Science Foundation, grant #SCI-0506429, DAPLDS - a Dynamically Adaptive Protein-Ligand Docking System based on multi-scale modeling. The authors thank David Anderson and 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 *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, 2004.

[2] D. P. Anderson and G. Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid*, 2006.

[3] B. R. Brooks and et al. CHARMM: a Program for Macromolecular Energy Minimization, and Dynamics Calculations. *J Comp Chem*, 4:187–217, 1983.

[4] H. Casanova. SimGrid: A Toolkit for the Simulation of Application Scheduling. In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, 2001.

[5] T. Estrada, D. Flores, M. Taufer, P. Teller, A. Kerstens, and D. Anderson. The Effectiveness of Threshold-based Scheduling Policies in BOINC Projects. In *Proceedings of the 2nd IEEE International Conference in e-Science and Grid Computing*, 2006.

[6] F. O. Gathmann. Python as a Discrete Event Simulation Environment. In *Proceedings of the 7th International Python Conference*, 1998.

[7] R. Ingalls. Introduction to Simulation. In *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[8] A. Kolinski and J. Skolnick. Assembly of Protein Structure from Sparse Experimental Data: an Efficient Monte Carlo Model. *Proteins: Structure, Function, and Genetics*, 32:475–494, 1998.

[9] D. Kondo. *Scheduling Task Parallel Applications for Rapid Turnaround on Enterprise Desktop Grids*. PhD thesis, UCSD, 2004.

[10] T. J. Schriber and D. T. Brunner. Inside Discrete-Event Simulation Software. In *Proceedings of the 2003 Winter Simulation Conference*, 2003.

[11] R. E. Shannon. Introduction to the Art and Science of Simulation. In *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[12] A. Sulistio, G. Poduvaly, R. Buyya, and C. Tham. Constructing a Grid Simulation with Differentiated Network Service using GridSim. In *Proceedings of the 6th International Conference on Internet Computing*, 2005.

[13] M. Taufer, C. An, A. Kerstens, and C.L. Brooks III. Predictor@Home: a "Protein Structure Prediction Supercomputer" Based on Public-Resource Computing. *IEEE Transactions on Parallel and Distributed Systems*, 17(8):786–796, 2006.

[14] 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 *Proceedings of the 14th Heterogeneous Computing Workshop*, 2005.

[15] H. Xia, H. Dail, H. Casanova, and A. Chien. The MicroGrid: Using Emulation to Predict Application Performance in Diverse Grid Network Environments. In *Proceedings of the Workshop on Challenges of Large Applications in Distributed Environments*, 2004.