Estimation of MPI Application Performance on Volunteer Environments

Girish Nandagudi[1], Jaspal Subhlok[1], Edgar Gabriel[1], and Judit Gimenez[2]

[1] Department of Computer Science, University of Houston
{jaspal,egabriel}@uh.edu
[2] Barcelona Supercomputing Center
judit@bsc.es

Abstract. Emerging MPI libraries, such as VolpexMPI and P2P MPI, allow message passing parallel programs to execute effectively in heterogeneous volunteer environments despite frequent failures. However, the performance of message passing codes varies widely in a volunteer environment, depending on the application characteristics and the computation and communication characteristics of the nodes and the interconnection network. This paper has the dual goal of developing and validating a tool chain to estimate performance of MPI codes in a volunteer environment and analyzing the suitability of the class of computations represented by NAS benchmarks for volunteer computing. The framework is deployed to estimate performance in a variety of possible volunteer configurations, including some based on the measured parameters of a campus volunteer pool. The results show slowdowns by factors between 2 and 10 for different NAS benchmark codes for execution on a realistic volunteer campus pool as compared to dedicated clusters.

1 Introduction

Most desktop computers and workstations are virtually idle as much as 90% of the time, representing what the volunteer computing community sees as an attractive "free" platform for parallel computations. Idle desktops have been successfully used to run sequential and master-slave task parallel codes, most notably under Condor [1] and BOINC [2]. Extending the classes of application that can be executed in a volunteer environment is challenging, since the compute resources are heterogeneous, have varying compute, memory and network capacity, and become unavailable for computation frequently and without warning. The Volpex project team, that includes the authors of this paper, has developed middleware for executing communicating parallel programs on volunteer nodes with orchestrated use of redundancy and communication logging. Volpex Dataspace API [3] provides a mechanism for applications to communicate through anonymous, asynchronous Put/Get operations. VolpexMPI [4] and P2P MPI [5] are MPI implemetations customized for volunteer environments.

Porting a large scale application to a volunteer environment presents numerous challenges even for MPI, a portable Message Passing Interface, examples of which are as follows:

- A modest effort and some expertise is necessary to deploy applications in a BOINC or CONDOR execution management environment.
- As volunteer nodes may have limited memory, a memory intensive application may have to be recast, e.g., on a higher number of parallel processes, to run correctly and efficiently.
- Communication intensive applications may require some customization for scheduling, such as limiting execution to nodes on a campus network.
- For applications employing licensed software, arrangements have to be made that the applications can run correctly and legally on volunteer nodes.

The reasons that make parallel computing in a volunteer environment challenging - primarily heterogeneity, variable compute and communication capacity, and high rates of failure - also make performance prediction on such environments challenging. Hence it is often unclear if a parallel message passing code will run effectively in a volunteer environment. Given that porting to a volunteer environment represents a significant investment, it is highly desirable that an estimate of the expected performance in a volunteer environment be available *before* the actual porting. The key objective of this research is to build a framework for estimating the performance of message passing MPI codes on volunteer environments. The paper also specifically characterizes the performance of NAS benchmarks in volunteer environments.

Performance on a volunteer environment depends on the computation and communication characteristics of the volunteer environment. This research employs the Dimemas framework to predict application performance in different types of volunteer environments. The high failure rates in volunteer environments can add an additional overhead, if, e.g., checkpoint restart is used for recovery. This overhead is dependent on a large set of parameters such as failure characteristics, degree of redundancy, and checkpointing frequency and overheads. These are not the subject of this paper and are partially addressed in related work. The main subject of this work is to estimate the performance of a given MPI application under a variety of scenarios including scenarios representing measured characteristics of a campus LAN.

2 Related Work

This work is in the context of fault tolerant MPI libraries. Several MPI libraries incorporate checkpoint-restart for fault-tolerance, with MPICH-V [6] being probably the best known example. This library is based on uncoordinated checkpointing and pessimistic message logging. MPI/FT [7] and P2P-MPI [5] are based on process replication techniques. Volpex MPI [4], which is the context of this work, employs checkpointing as well as replication. The focus of this work is estimating the performance of an MPI library in a volunteer computing environment in general.

Dimemas [8] is a performance analysis tool that allows the user to predict the performance of a parallel application on a simulated target architecture. Dimemas is the basis of of the simulations employed in this work and is discussed in more detail in the following sections. Dimemas has been used in heterogeneous environments to predict the performance of applications for resource selection [9].

SimBOINC [10] is a simulator designed to test new scheduling strategies in BOINC and other desktop and volunteer systems. SimBOINC simulates a client-server platform where multiple clients request work from a central server. The characteristics of the client, the workload and the network connecting the client and server can be specified as simulation inputs. The results provide scheduler performance metrics, such as effective resource shares, and task deadline misses. EmBOINC [11] (Emulator of BOINC Projects) is a trace-driven emulator that models heterogeneous hosts and their interaction with a real BOINC server. EmBOINC uses statistical information obtained from real BOINC traces to characterize volatile, heterogeneous, and error-prone hosts. Both the above tools (SimBOINC and EmBOINC) are focused on scheduling and node selection and do not provide a way to simulate the performance of communicating parallel applications on desktop grids.

3 Simulation Framework

The main goal of this research is to estimate the performance of an MPI application under volunteer environments with different execution characteristics. The simulation framework is based on the *Dimemas* tool chain developed at Barcelona Supercomputing Center. Dimemas [8] simulates the performance of an application on a user-defined virtual architecture. Dimemas has an extensive suite of parameters that can be used to customize a virtual environment to mimic a real-life environment. The following important properties of an execution environment can be defined with Dimemas:

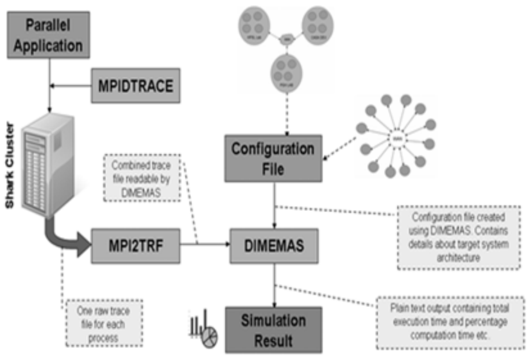

Parallel Application
MPIDTRACE
Shark Cluster
Combined trace file readable by DIMEMAS
Configuration File
MPI2TRF
DIMEMAS
Configuration file created using DIMEMAS. Contains details about target system architecture
One raw trace file for each process
Simulation Result
Plain text output containing total execution time and percentage computation time etc.

Fig. 1. Dimemas based simulation framework

– *Simulated Node Architecture* including the number of CPUs per node, relative processor speed, memory bandwidth, file system parameters, and number of parallel network links.

– *Simulated Network Architecture* including LAN/WAN configuration, maximum bandwidth on network links, latency on network links, external network traffic model, and network sharing model.

In addition to the the Dimemas simulator, the framework employs tools to record and merge execution traces. *MPIDTRACE* program traces the application execution and generates a trace file (one per process) containing the run time actions of the application including communication calls and computation blocks, along with details such as elapsed times and the source and destination processes for communication calls. *MPI2TRF* program is used to merge the per-process trace files generated by the MPIDTRACE program into a single file that can be read by Dimemas. Estimating the performance of a given application on a volunteer environment consists of the following steps: i) Execute the application linked with MPIDTRACE on a reliable execution platform like a cluster and collect the trace, ii) combine the individual process traces into a single trace with MPI2TRF, iii) configure Dimemas to reflect the desired volunteer computing environment, and iv) perform the simulation with Dimemas to obtain the predicted execution time. The simulation steps are also summarized in Figure 1.

4 Experiments and Results

4.1 Simulation Configuration

The simulation is performed on two types of network configurations. We refer to the first configuration as the *Desktops over Internet (DOI)* configuration, and to the second as the *Collection of Networked Computers (CNC)* configuration. These configurations are illustrated in Figure 2.

The DOI configuration consists of desktop PCs connected with an institutional LAN or Wide Area Network/Internet. This environment is characterized by the latency and bandwidth between individual nodes and the network. There are no shared network resources that would represent a fundamental bottleneck. The assumption is that the switch that connects the PCs in a LAN environment offers full bisection bandwidth. For nodes connected by the Internet, it is implicitly assumed that the PCs in different regions do not share the same route in the Internet, and if they do, the traffic generated by the nodes is negligible as compared to the rest of the Internet traffic on shared routes.

The CNC configuration on the other hand, is a two stage hierarchy of networks of computers. Individual groups of computers, representing separate labs or computation centers physically, are connected to each other over a LAN within the lab. Then these groups/labs communicate with others through WAN/Internet. In contrast to the DOI configuration discussed above, the connections between PCs in one lab communicating to PCs in another lab have to share a single link.

Both configurations apply a *linear* model for access to a shared network implying that only one node accesses a shared LAN/WAN at a given time. In general, DOI has the relative advantage over CNC that no links to the WAN are shared, while CNC has the relative advantage over DOI that there is a reduced possibility of delays at the central WAN as some communication can be completed within the local LANs. The latter is most significant for all-all type communication patterns.

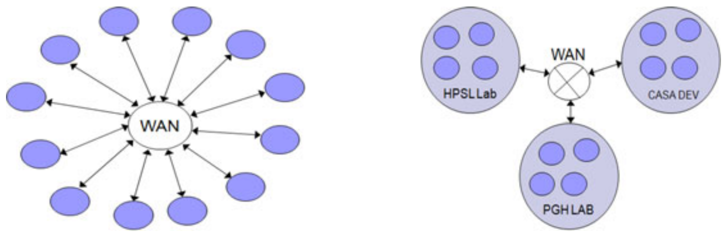


Fig. 2. Graphical representations of the Desktops over Internet configuration (left) and Collection of Networked Computers configuration (right)

All traces for the simulations were generated on the "Shark" cluster at the University of Houston. The Shark is a heterogeneous cluster consisting of 24 Sun X2100 nodes, each with a 2.2 GHz dual core AMD Opteron processor and 2 GB main memory, and 5 Sun X2200 nodes, each with two 2.2 GHz quad core AMD Opteron processors (8 cores total) and 8 GB main memory. The network interconnect of Shark cluster comprises of a 4xInfiniBand network interconnect and a Gigabit Ethernet network. For results presented, the Gigabit Ethernet network has been used. This cluster also serves as a reference cluster for estimating relative performance on volunteer nodes.

Four NAS benchmarks are used for evaluation: The CG, IS, SP, and FT represent diversity in terms of communication volume and communication pattern. The CG and the IS benchmarks are considered communication intensive for different reasons. CG sends a large number of point-to-point messages between processes. IS is dominated by an all-to-all communication pattern with relatively few large messages. The FT benchmark also uses an all-to-all communication pattern, but with smaller messages and more frequent operations. Due to the complexity of the compute operations involved in the FT benchmark the ratio of computation to communication in FT is lower than in IS. The dominant communication pattern of the SP benchmark is a 5-point stencil communication, which leads to a 'localized' communication behavior of the code.

4.2 Desktop over Internet Configuration Results

We report on experiments carried out on the DOI configuration. Execution of NAS Class B benchmarks CG, FT, IS and SP on 16 and 32/36 processes was simulated with a range of (synthetic) bandwidth and latency values for the network. The *reference* execution simulated a cluster with 128MBps bandwidth and 0.05 seconds latency. Simulations were performed for bandwidths {12.5 MBps, 1.25MBps, 0.125 MBps} and latencies {0.1ms, 1 ms and 10 ms} spanning the range from clusters to very slow Internet connected nodes. The goal is to estimate the impact on execution time of reduced bandwidth and increased latency that represent various flavors of volunteer computing environments. The results are plotted in Figure 3.

Figure 3 (a) shows expected increase in execution time for a comprehensive set of scenarios as compared to cluster execution. We omit detailed comments for brevity, and instead focus on specific scenarios in Figure 3 (b) and (c). Figure 3 (b) simulates the expected slowdown when the available bandwidth is reduced from the reference cluster level value of 128MBps while the latency is fixed at .1 msecs. We limit our discussion to the case of 16 nodes as the patterns for 32/36 nodes are qualitatively similar.

We notice that the slowdown for reduction of bandwidth to 12.5 MBps is relatively modest, ranging from a low of 17% for FT to a high of 265% for IS. Reduction of bandwidth to 1.25MBps has a far more dramatic effect: from a roughly 2fold slowdown (190%) for FT and up to a 30fold slowdown (2913%) for IS. The conclusion is that for NAS benchmarks, the impact of reduction in bandwidth is specific to the computation/benchmark in question, and relatively modest (say around or below a factor of 2) in many interesting scenarios.

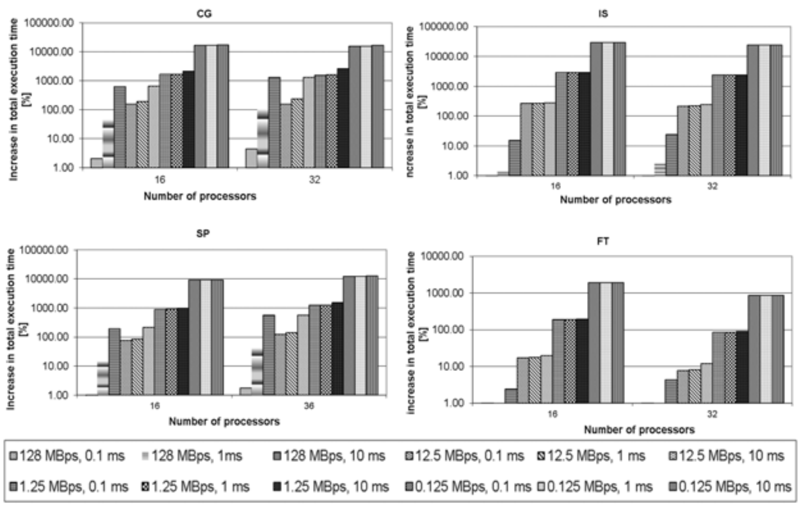
We now focus on Figure 3(c) that shows the sensitivity of the benchmarks to latency. Here, the bandwidth is fixed at 128 MBps and comparisons are made for various latency values against a reference latency value of 0.1 milliseconds. The graph shows that the impact is very low, below a 15% slowdown, for most scenarios. The notable exception is CG, where the slodown is 43% for a latency of 1 msecs and over 600% for a latency of 10 msecs. The slowdown of SP rises to 190% for a latency of 10 msecs.

4.3 Collection of Networked Computers Configuration Results

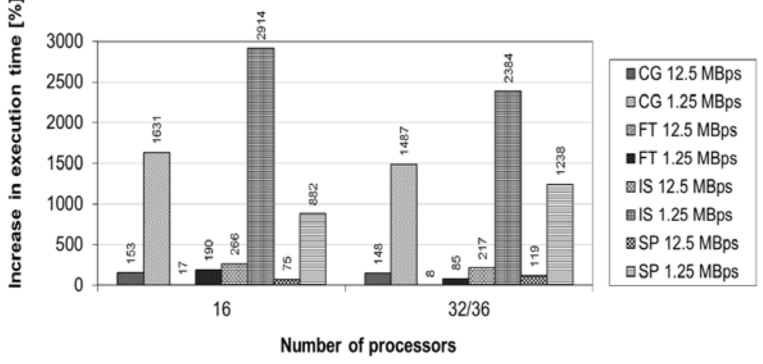
Simulations were conducted using the CNC configuration with the same set of benchmarks, problem sizes and number of processors. Selected results are presented in fig. 4. The key difference between DOI and CNC configuration is that in CNC configuration groups of nodes share a link to the backbone network. For this reason, CNC configuration is generally slower for a uniform setting of latency and bandwidth, but can be faster as fewer connections access the central WAN.

Figure 4 (a) simulates the expected slowdown when the available bandwidth is reduced with the same parameters as those for Figure 3 (b) for the DOI configuration. Similarly Figure 4 (b) shows the sensitivity of performance to latency for a fixed bandwidth with parameters same as Figure 3 (c) for the DOI configuration.

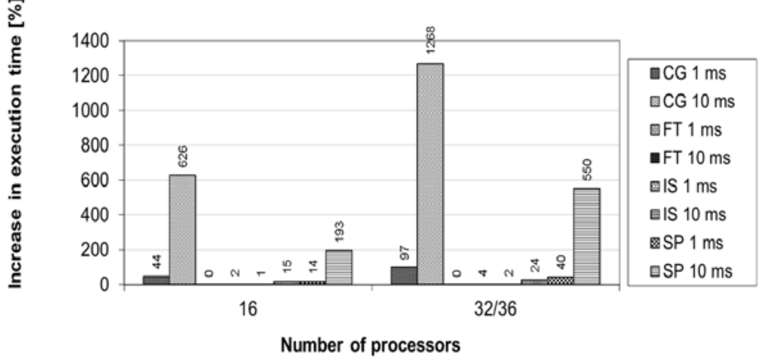
The general pattern of results is the same as for the DOI configuration but with generally higher slowdowns due to sharing. An interesting case is the FT benchmark which is far more sensitive to available bandwidth in CNC configuration as compared to its performance in DOI configuration. The probable reason is that the all-all communication in FT saturates the bandwidth of the links



(a) Slowdown for a suite of latency/bandwidth configurations

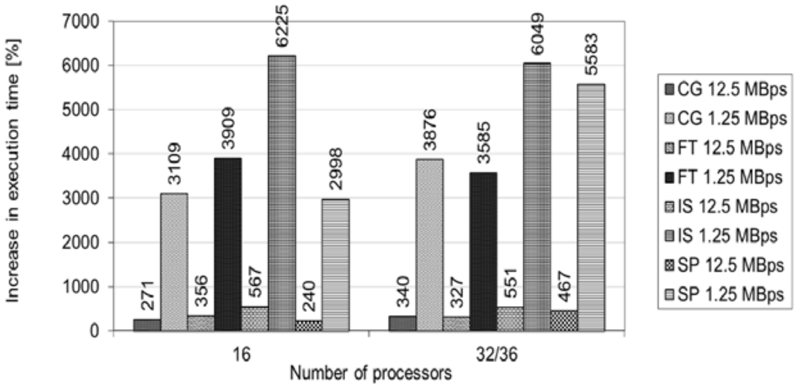


(b) Slowdown with fixed latency (0.1 msecs) and reduced bandwidth

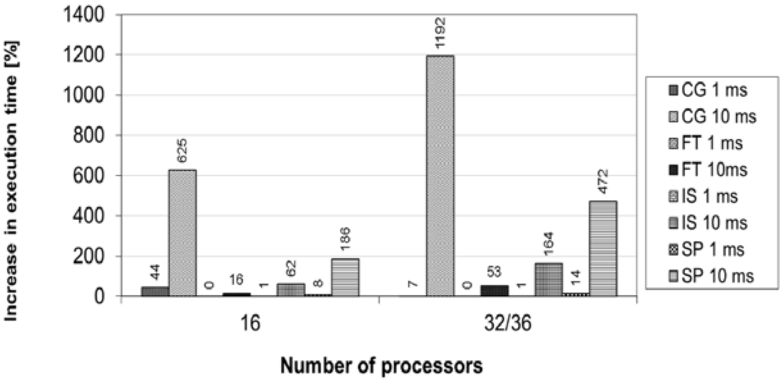


(c) Slowdown with fixed bandwidth (128 Mbps) and increased latency

Fig. 3. Percentage increase in execution time versus a cluster for the DOI configuration



(a) Slowdown with fixed latency (0.1 msecs) and reduced bandwidth



(b) Slowdown with fixed bandwidth (128 Mbps) and increased latency

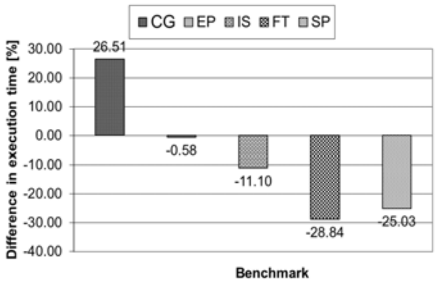
Fig. 4. Percentage increase in execution time versus a cluster for the CNC configuration

connecting LAN groups to WAN/Internet connection which cannot happen in the DOI configuration. The overall slowdown numbers underline the potential of LAN to WAN bottlenecks to impact performance: for bandwidth reduced to a worst case of 1.25Mbps, the slowdown approximately ranges between 30fold to60 fold for the CNC configuration. In comparison, the DOI configuration exhibited up to a 30fold slowdown.

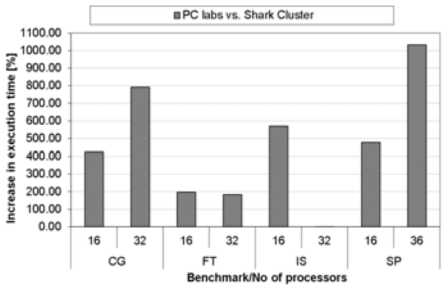
4.4 Simulation on Real-Life Environments

For this set of experiments, we measured the actual latency and bandwidth in the following scenarios in our computing environment. For the Shark cluster; the measured point to point bandwidth was 95MBps and latency 0.05 milliseconds. Between the desktops in a single lab in our building, the measured point to point bandwidth was 12MBps and latency .4 milliseconds. Between lab desktops connected across a LAN, the measured point to point bandwidth was 7MBps and latency .8 milliseconds.

First this data was used to validate and parameterize simulations as shown in Figure 5(a). While the absolute measured and predicted performance varies up to around 25%, the accuacy was judged sufficient for our purposes as the relative comparison of different configuratios is still meaningful. Next, the performance of the cluster environment was compared with our PC lab environment with 3 groups of nodes; the nodes within a group are connected at measured LAN specs of 12MBps and latency .4 milliseconds, while the WAN latency and bandwidth across the groups of nodes are .8 msecs and 7MBps respectively. The results are presented in Figure 5(b). The results show 2fold to 10fold slowdowns.



(a) Validation of simulation



(b) Lab PC environment vs. Shark cluster

Fig. 5. Validation and performance in realistic environments

5 Conclusions

This paper has presented a methodology based on Dimemas toolkit that allows the estimation of performance of MPI codes on heterogeneous volunteer nodes. The methodology was employed to estimate the performance of selected NAS benchmarks on a large number of scenarios including those that are typical of a volunteer environment built from idle desktops. The result also provide significant insight into likely performance of NAS benchmarks on realistic desktop PC environment. NAS benchmarks executing on volunteer nodes connected by commodity LAN network is a factor of 2 to 10 slower than a dedicated cluster. While a volunteer environment is not expected to compete with a dedicated cluster in performance for a fixed number of nodes, the results show that it offers a practical 'free' alternative that may be satisfactory for many application scenarios.

Acknowledgments. Partial support for this work was provided by the National Science Foundation's Computer Systems Research program under Award No. CNS-0834750. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. Concurrency - Practice and Experience 17(2-4), 323–356 (2005)
2. Anderson, D.: BOINC: a system for public-resource computing and storage. In: Fifth IEEE/ACM International Workshop on Grid Computing (November 2004)
3. Pedamallu, E., Nguyen, H., Kanna, N., Wang, Q., Subhlok, J., Gabriel, E., Cheung, M., Anderson, D.: A robust communication framework for parallel execution on volunteer PC grids. In: CCGrid 2011: The 11th IEEE/ACM International Symposium on Clusters, Cloud and Grid Computing, Newport Beach, CA (May 2011)
4. LeBlanc, T., Anand, R., Gabriel, E., Subhlok, J.: VolpexMPI: An MPI Library for Execution of Parallel Applications on Volatile Nodes. In: Ropo, M., Westerholm, J., Dongarra, J. (eds.) EuroPVM/MPI 2009. LNCS, vol. 5759, pp. 124–133. Springer, Heidelberg (2009)
5. Genaud, S., Rattanapoka, C.: Large-scale experiment of co-allocation strategies for peer-to-peer supercomputing in p2p-mpi. In: IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, pp. 1–8 (2008)
6. Bosilca, G., Bouteiller, A., Cappello, F., Djilali, S., Fédak, G., Germain, C., Hérault, T., Lemarinier, P., Lodygensky, O., Magniette, F., Néri, V., Selikhov, A.: MPICH-V: Toward a scalable fault tolerant MPI for volatile nodes. In: Proceedings of the SuperComputing 2002 Conference (November 2002)
7. Batchu, R., Neelamegam, J., Cui, Z., Beddhua, M., Skjellum, A., Dandass, Y., Apte, M.: MPI/FT: architecture and taxonomies for fault-tolerant, message-passing middleware for performance-portable parallel computing. In: Proceedings of the 1 IEEE International Symposium of Cluster Computing and the Grid (2001)
8. Badia, R., Labarta, J., Gimenez, J., Escale, F.: DIMEMAS: Predicting MPI applications behavior in Grid environments. In: Workshop on Grid Applications and Programming Tools, GGF8 (2003)
9. Lindner, P., Gabriel, E., Resch, M.M.: Performance Prediction Based Resource Selection in Grid Environments. In: Perrott, R., Chapman, B.M., Subhlok, J., de Mello, R.F., Yang, L.T. (eds.) HPCC 2007. LNCS, vol. 4782, pp. 228–238. Springer, Heidelberg (2007)
10. Taufer, M., Kerstens, A., Estrada, T., Flores, D., Teller, P.: Simba: a discrete event simulator for performance prediction of volunteer computing projects. In: International Workshop on Principles of Advanced and Distributed Simulation 2007 (March 2007)
11. Estrada, T., Taufer, M., Anderson, D.P.: Performance prediction and analysis of BOINC projects: An empirical study with EmBOINC. Journal of Grid Computing 7(4), 537–554 (2009)