On Profiling the Energy Consumption of Distributed Simulations: A Case Study

Miguel A. Erazo, and Roberto Pereira

School of Computer and Information Sciences
Florida International University, Miami, FL 33199
Emails: {meraz001,rpere058}@cis.fiu.edu

***Abstract*—Reducing the energy consumption of computing systems has been the topic of many research studies. Specifically, optimizing the energy consumption of applications is important to reduce the carbon footprint and the associated costs. In this paper, we detail a method to profile the energy usage of distributed simulations in a grid environment and conduct several experiments towards characterizing the power behavior of our PRIME network simulator. We conclude that although PRIME shows a high level of parallelization in some scenarios, the energy consumption increases significantly as more compute nodes are used to run a network model. Also, we show that cross-traffic has a tremendous impact in the energy usage.**

***Index Terms*—Network Simulation, Energy consumption**

I. Introduction

Some recent reports claim that the global IT sector is responsible for about 2% of human carbon dioxide emissions each year, very similar to the global airline industry. Many different research fronts are open to tackle this problem. One of these focuses on building instruments to help research community understand how to measure and minimize energy consumption, e.g., the GreenLight [1] project. Others focus on optimizing the energy consumption at three different levels: 1) architectural, i.e., reduce the processor energy usage, 2) system, and 3) application. Regarding the latter, growing concern about the carbon footprint of computing systems [2], [3] is a major drive for energy optimization of applications. Not less important are the increases in the energy utility bills, IT power provisioning, backup infrastructure, and cooling equipment.

In this paper we profile the energy consumption of our PRIME [4] network simulator as the first step towards its future energy optimization. Many different approaches can be used to profile the energy consumption of a application, including: simulation, measurement and estimation. In [5], the authors created an architectural simulator with the goal of estimating the power consumption of a CPU. These estimates are based on hardware power models and resource usage obtained from cycle-level simulation. Similarly, in [6] a complete system simulator for power profiling is proposed. This approach relies entirely in simulation and hardware power models to generate power profiles that can be used to help improve designs and power optimization techniques.

Other research efforts focus on measuring the energy usage using hardware equipment. For instance, JouleSort [7], a energy-efficiency benchmark with fully specified workload, metric, and rules, used a digital power meter between the system and the outlet. In [8] the authors used direct measurements to calculate the power consumption of a computer system, a pocket computer in this case. Their approach consists of measuring the current across one of the resistors located in the power circuit of the computer. With the current and the voltage supplied known, they obtain the instantaneous power rating of the circuit, which is integrated over a given period of time yielding the total energy consumed.

Different to aforementioned approaches, other studies trust on hardware resource measurements to estimate energy usage. In [9], estimates for CPU power consumption are obtained based on hardware measurements. This method relies on the utilization of performance counters to obtain data such as cache misses, bus transactions, branch mispredictions and instructions retirement in order to estimate activity. A similar approach is taken in [10]. There, the energy consumption is estimated to provide software developers visibility into the application's energy usage at design time.

Our ultimate goal is to enable our discrete-event network simulator, PRIME [4], with power-awareness, i.e., the simulator will provide some insight to the experimenter regarding the expected energy-consumption of a network model. In this way, the experimenter can make "green" decisions by choosing a particular execution environment of a network model. To that end, in this document we detail a method to profile the energy consumption of distributed simulators and perform several experiments to characterize the power consumption of PRIME in order to identify the main factors that lead to an increase of it. Our approach is similar to [9] and [10], which rely on the measurement of the usage of hardware resources. Specifically, we conduct our studies considering the CPU, Memory, Disk, and NIC. We use the NCSA's Lincoln [11] cluster; which provides many useful tools for application's profiling. Several experiments are performed within two main scenarios that we detail in section IV.

The rest of the paper is organized as follows. In Section II, we briefly describe the salient features of our PRIME network simulator. Section III is devoted to detail the approach used to estimate the energy consumption. Section IV describes our experiments and the obtained results. Finally, we conclude this paper and outline our future work in Section V.

II. The simulator

Key to our case study is the PRIME network simulator; whose energy behavior is profiled in this document. In the following paragraphs we describe the most salient features of PRIME relevant to this paper.

PRIME stands for Parallel Real-Time Immersive network Modeling Environment [4]. The parallel discrete-event simulation engine used by PRIME is based on the Scalable Simulation Framework (SSF), which is a standard API for parallel large-scale simulation [12]. PRIME can run on most parallel platforms, including shared-memory multiprocessors (as a multi-threaded program), distributed-memory machines (via the message-passing interface), or a combination of both. Supporting parallel and distributed simulation allows PRIME to run extremely large network simulations.

A rich set of network elements is provided by PRIME: routers, links, network queues, and protocol implementations, including TCP, ICMP, UDP, and various application-layer protocols (such as FTP and HTTP). These network elements and protocols can be used to construct various network experiment scenarios. In [13], PRIME is used to show the potential of real-time simulation for studying complex behaviors of distributed applications under large-scale network conditions. In order to increase realism in simulation studies, we ported thirteen Linux TCP variants to PRIME [14]. Furthermore, we enhanced our previous implementation to include real TCP message exchanges in such a way that real TCP instances can communicate seamlessly with PRIME simulated hosts.

PRIME has been shown to be capable of simulating and emulating large scale scenarios [13]. In there, 25 machines with OpenVZ images where loaded using Emulab [15]. Twelve physical machines were used to run PRIME simulator, 12 others ran 1008 CoralCDN [16] instances (inside OpenVZ containers), and 1 machine was devoted to run an Apache web server. During those experiments, 3 metrics were collected: *cache hit rate*, *web server load*, and *response time*.

In this document, we make use of PRIME's capabilities to simulate large-scale scenarios in our experiments.

III. Energy Estimation

In a grid environment, multiple jobs are running concurrently and it is not known exactly when a job is going to be scheduled for execution. Furthermore, we have limited access to cluster machines on Teragrid [17]. In consequence, directly measuring the energy consumption of our simulations is not feasible. Instead, we estimate the energy consumption through resource usage. We use a simple and feasible energy model similar to that used in [10] and [18]. In our model, the energy consumption of a particular resource is a function of its states(e.g, read, write, idle, etc). For our calculations, we take into account four resources: CPU, memory, Disk, and NIC.

Thus, the expressions we use to estimate the energy consumption are the following:

$$E_{total} = E_{CPU} + E_{Mem} + E_{Disk} + E_{Net} \quad (1)$$

$$= \sum_{i \in R} \left(\sum_{j \in S_i} (P_{ij} * f_{ij}) \right) * T \quad (2)$$

In the above equation, R is the set set of resources we are considering. S_i is the set of states of each resource i. P_{ij} is the power of resource i in state j. f_{ij} is the percentage of time that the resource i is in state j during the experimentation time. Finally, T is the time that each resource is used, which in our setup is the same for every one.

The approach presented above is general and can be applied to estimate the enegy consumption of any application. Throughout the whole study we consider two states for each resource: active and idle. Then, our estimation focuses on computing the time that our application uses a specific resource (T_{active}). Using the specific power consumption corresponding to each resource in active and idle states, P_{active} and P_{idle} respectively, we use the following expression to calculate the energy consumed by each resource:

$$E_{resource} = P_{active} T_{active} + P_{idle}(T - T_{active}) \quad (3)$$

In the next subsections we briefly describe the specific considerations applied for each resource.

1) CPU: We use the expression shown below to estimate the CPU time, i.e., the time that our application uses the CPU.

$$\text{Time}_{CPU} = \frac{Cycles_{app}}{clockSpeed} \quad (4)$$

In the expression shown above, Time_{CPU} is time that our application uses the CPU, Cycles_{app} is the number of CPU cycles our application has occupied the CPU, and clockSpeed is the number of clock cyles per second.

2) Memory: If Data/Instructions are not found in L1 and L2 caches, the main memory is accessed and a *line* (which ranges in size from 8 up to 512 bytes across different processor architectures) is fetched. Also, when this happens, not only a line is fetched from the memory but also more data from it is loaded by the pre-fetcher (e.g. a Pentium-4 processor loads 256 bytes). For the estimation of the time our application uses the memory, we take into acccount the memory accesses due to L2 cache misses and not the data prefetched because of the reasons we detail in section IV.

Assuming that one line is fetched per access (we do not consider prefetched data) we calculate the time per memory access as follows:

$$\text{Time}_{MemAcess} = T_{RCD} + ((CL - 1) * Cycle_Time) + T_{AC} \quad (5)$$

In the expression shown above, T_{RCD} is the time required between the computer defining the row and column of the given memory block and the actual read or write to that location. CL or column Address Strobe(CAS) latency is the time between the moment the memory controller tells a memory module to access a particular memory column, and the moment the data from the given array location is available on the module's output pins. T_{AC} is the time to fetch the actual data.

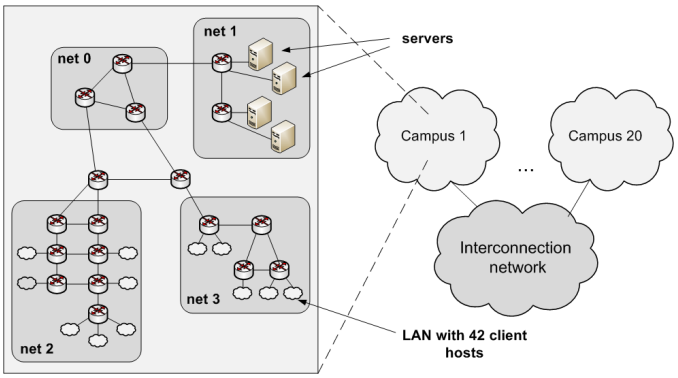

Fig. 1. Campus network.

The total time our application uses the memory is then computed as follows:

$$\text{Time}_{Mem} = L2misses * Time_{MemAcess} \quad (6)$$

3) Disk: In order to perform precise measurements, tools like blktrace can be used. *blktrace* is a block layer IO tracing mechanism which provides detailed information about request queue operations up to user space. However, a patch to the Linux kernel which includes the kernel event logging interfaces, is needed and we do not have it available in our experimentation environment, i.e. Teragrid [17]. Instead we use *strace* as the tool to record all *read* and *write* system calls to approximate the time that our application accesses the disk.

In order to estimate the time that our application uses the disk, denoted by Time_{Disk}, we make two asumptions: first, we assume that the files being transferred from the disk are not fragmented, this is a reasonable assumption as most of these files are newly created; second, we use the *internal sustained transfer* rate obtained by averaging the innermost track transfer rate with the outermost track transfer rate, both of these rates are provided in the manufacturer's specifications. In this way, we obtain the Time_{Disk} using the following expression:

$$\text{Time}_{Disk} = \frac{Bytes_{Disk}}{InternalSustainedTransferRate} \quad (7)$$

In the above expression, Bytes_{Disk} is the total number of bytes written and read to and from the disk.

4) NIC: We did not compute the energy consumed in the NIC because of lack of data regarding the hardware devices but we did count the number of packets that are sent between physical machines; which are the packets that actually go through the network.

IV. Experiments

A. The Network Model

The network model we used for our study is an interconnected network composed of 20 *campus networks*. A campus network consists of more than 500 end-hosts connected by 30 routers, as shown in Fig. 1. This simulated topology contains four subnets; within net2 and net3, there are 12 local area networks (LANs), each configured with a router switch and 42 end-hosts. The LANs are 10 Mbps networks. The links connecting routers within net1 and net2 are set with a bandwidth of 100 Mbps and a link delay of 10 ms.

B. Environment Setup

We conducted our experiments on Teragrid [17], a grid environment which features high-performance computers, data resources and tools, and high-end experimental facilities. PRIME was configured therein to run as a distributed simulator, i.e., multiple instances can run on multiple compute nodes to take advantage of multi-processors or multi-cores, and installed on the Lincoln [11] cluster. We chose the Lincoln cluster because it provides a number of installed utilities for profiling applications, e.g., Perfsuite. Another tool that we used is *strace*, a debugging utility in Linux to monitor the system calls used by a program and all the signals it receives.

PerfSuite [19] is a collection of tools, utilities, and libraries for software performance analysis. Perfsuite uses the Performance Application Programming Interface [20] (PAPI), a consistent interface which enables users to profile software using processor events. Compute nodes on the Lincoln cluster, log these events, i.e., occurrences of specific signals related to the processor's function. We use two specific command line utilities from Perfsuite: psrun and psprocess. We use *psrun* to gather hardware information from unmodified software. Running this command using an executable (PRIME in our case) as a parameter, creates an xml file with performance information related specifically to the passed application. We use *psprocess* to post-process the results of a performance analysis experiment; namely, the xml file created after running psrun. The execution of this command causes a number of derived metrics to be generated based on the ones measured when running psrun. It is important to remark that, depending in the hardware architecture, only some events captured at processor-level are available. In *PAPI Standard Events by Architecture* [20] it is detailed the set of events captured for each architecture.

We use Perfsuite, strace, and output information provided by PRIME to gather information about the usage of each resource we consider in our study. For the CPU we got the Time_{CPU} directly from the output of psprocess. Regarding the memory, we did not take into consideration the amount of data prefetched from the memory because the PAPI_PRF_DM (data prefetch cache misses) PAPI event is not available in the infrastructure provided by Lincoln in Teragrid. Thus, using expression 6 and the number of L2 cache misses obtained from psprocess we approximate Time_{Mem}. Time_{Disk} is estimated using the strace Linux command. We parse the output of strace in order to get the number of bytes associated with each read and write system call. From there, we sum all bytes and use the average internal sustained transfer rate to get an approximation of Time_{Disk}. Finally, for the NICs we count the number of packets transmitted between compute nodes. The energy consumed by the NICs will be proportional to the number of packets received from and transmitted to it. Thus, in

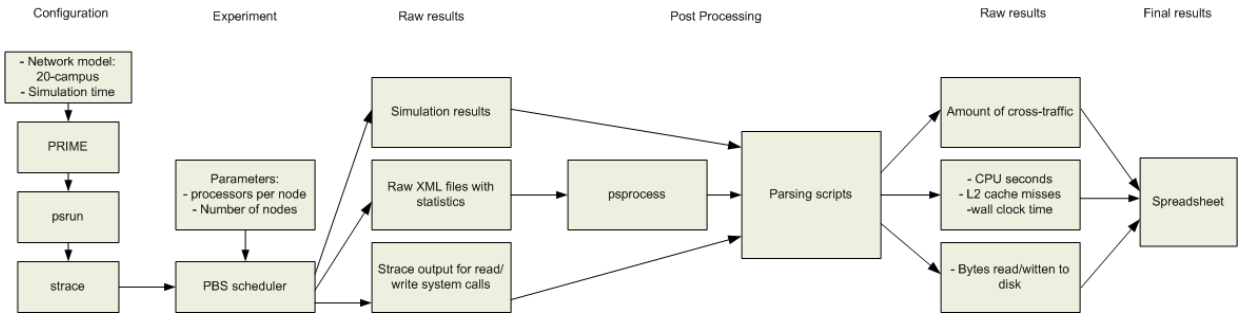

Fig. 2. Process followed to get final results

counting the packets we do not attempt to compute the energy consumption, but to provide a way to compare the profile of different network models.

C. Experiments and Results

The performance of distributed simulations, i.e., the wall-clock time it takes to complete, and the energy consumption depend on a number of factors, including: network model being run, and number of compute nodes used. Here, we profile the energy consumption of distributed simulations by exploring these two parameters. The network model can be changed in various ways; in this document we vary the amount of cross-traffic, i.e, the traffic that goes between two compute nodes. To explore the effect of changing the number of compute nodes, we run PRIME using a different amount of nodes for each set of experiments.

We profiled the energy consumption of PRIME using two different scenarios. The first scenario aims to study the energy signature of a network model where there is no cross-traffic between compute nodes running a subnetwork. To that end, at every particular run we use the same network model (traffic, topology, and applications) as the number of compute nodes increases; but assuring that no packet crosses a compute node boundary. For the second scenario, the objective is to study the energy signature of a network model as we increase the amount of cross-traffic while using a fixed number of compute nodes for the simulation. Thus, we fixed the number of compute nodes to 20 and increased the percentage of cross traffic by increments of 10 percent for each set of experiments.

In both scenatios, each client in the 20-campus network requests packets from a server (see Fig. 1). The servers were configured to send packets upon request, using TCP, according to a deterministic function. In order to change the amount of cross-traffic, we set a client to request packets from either a server in a campus network running on the same compute node (no cross-traffic) or a server running on another compute node. Regarding the number of compute nodes used, we partition the 20-campus network model using a service provided by PRIME which uses METIS [21].

The steps followed from the configuration of an experiment up to the generation of the final results are shown in Fig. 2. First, the network model is customized for a particular run. Then, the PBS scheduler executes strace which operates over

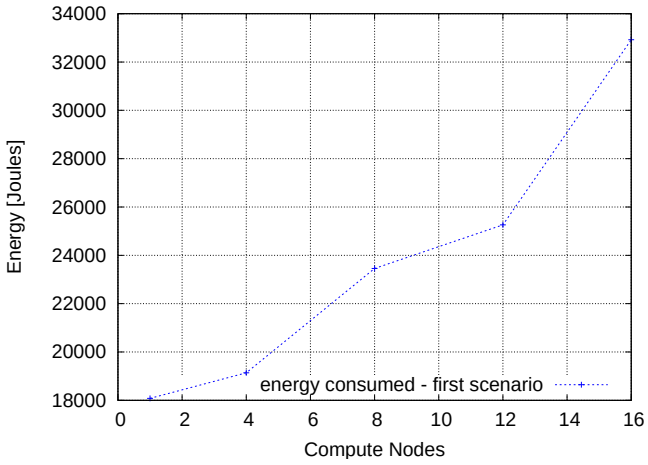

Fig. 3. Energy consumption for first scenario.

the output of running PRIME in multiple compute nodes,using psrun, for 1000 simulated seconds. After the job has been executed, the following files are generated: one file with the PRIME-generated simulation results, one XML file per physical machine used (generated by psrun), and one file with the output of strace. Next, we process the XML files using psprocess to obtain many derived metrics, including Time_{CPU} and number of L2 cache misses. Then, all this files are inputted to parsing scripts which generate raw results. Lastly, we use an spreadsheet to obtain the final results.

The energy consumption corresponding to the first scenario is shown in Fig. 3. It was computed as the summation of the energy consumed in each compute node. We performed 5 runs per number of compute nodes and averaged the results. According to our results, the total energy consumed increases linearly as more compute nodes are used; where the CPU accounts for close to 90% of the total energy consumed (see Fig. 6 for the energy distribution). To explain these results, we plot the total CPU time used to compute the energy consumption and the wall-cock time needed to run the simulation, see Fig. 4. In there, it is shown that the total CPU time increases linearly with the number of compute nodes used; which explains the behavior of the total energy consumed. In the ligth of Amdahl's law, the wall-clock time depicted

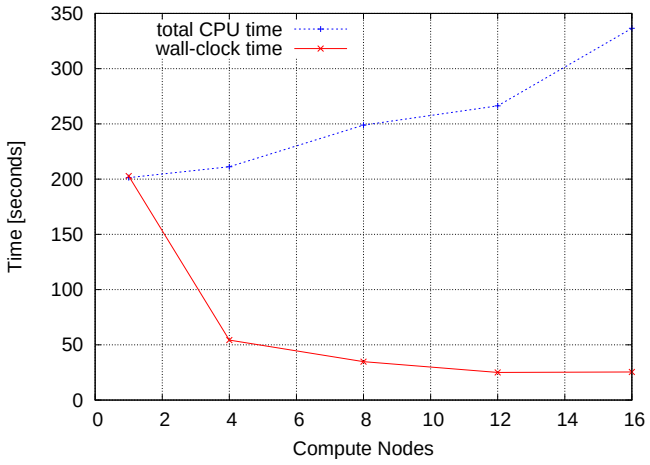

Fig. 4. CPU time for first scenario.

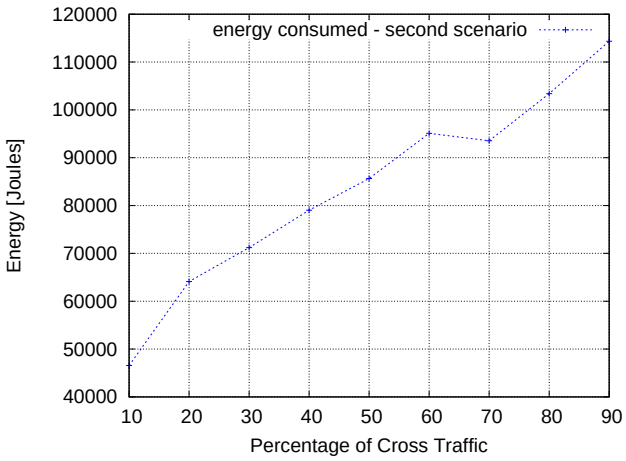

Fig. 5. Energy consumption for second scenario.

in Fig. 4 roughly corresponds to a 95% optimized code for parallel execution. Although this level of parallelization may be considered high, it has a tremendous impact in the wall-clock time and CPU time achievable by a model and thus in the energy consumed. It can be seen that for 16 compute nodes, the energy consumed increases in close to 80%. According to our results, for scenario one, the optimal setup corresponds to 4 compute nodes, where performance increases in 72.61% and energy only increases in 5.85%.

Fig. 5 depicts the energy as the cross-traffic is increased. Its corresponding total CPU time and wall-clock time are shown in Fig. 7, and the number of packets transmitted between computed nodes is shown if Fig. 8 . Again, it is observed a linear behavior for the energy consumption. When 90% of the traffic crosses compute node's boundaries, the energy increases in more than 120%. Here it can be easily noticed the huge impact that traffic between compute nodes has in the energy consumption of distributed simulations.

We think that the energy consumption can be decreased by carefully partitioning the network model and mapping it to compute nodes. Experiments are needed to asses the impact of partitioning on the energy behavior. Also, it remains to investigate how a model with a high-level of cross-traffic behaves as the compute nodes are changed. Furthermore, we want to explore how the energy consumption behaves in an scenario where a the state of an application is logged to files at regular intervals, which would increase significantly the energy consumed in the disk.

V. Conclusions and Future Work

In absence of cross-traffic PRIME achieves close to 95% code optimization. Nevertheless, this has a tremendous impact in the total CPU time and therefore in the energy consumption of a network model. Although the performance of the simulator increases, in terms of wall-clock time achieved when more compute nodes are used, the energy consumed raises in 80% for 16 nodes. Also we show that the energy consumption increases significantly when cross-traffic increases, leading to

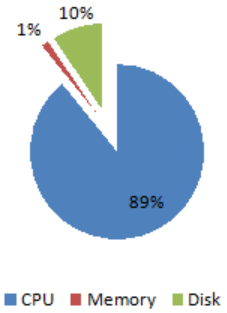

Fig. 6. Energy consumption distribution among resources in both scenarios.

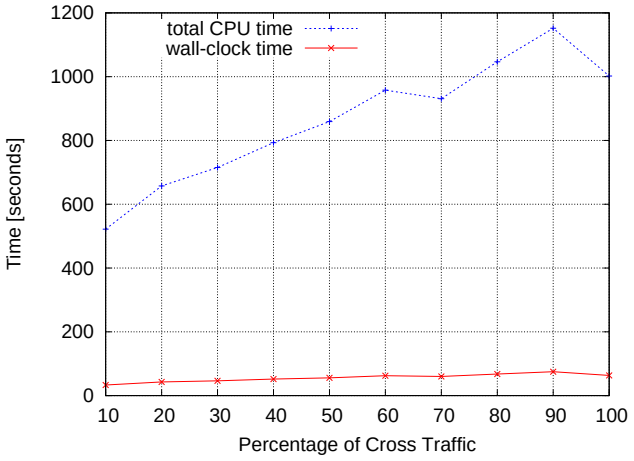

Fig. 7. CPU time for first scenario.

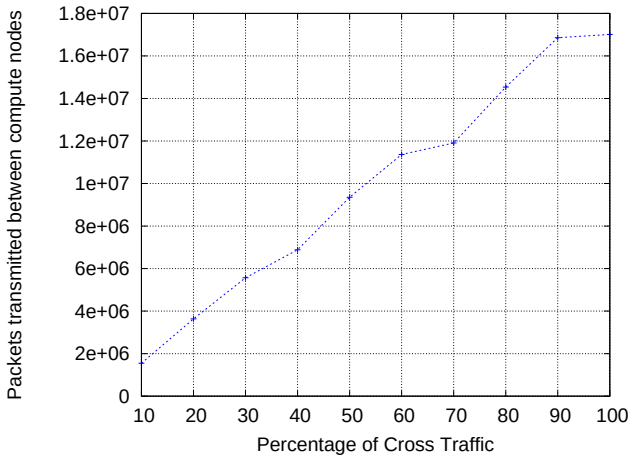

Fig. 8. Packets transmitted between compute nodes.

120% increase when 90% of the traffic crosses compute node's boundaries.

As a future work, we plan to extend our studies an explore the partitions conducted on a network model to map it to a distributed environment. After sufficient data is collected regarding's PRIME behavior, we plan to implement power-awareness mechanisms in it.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (grants OCI-0636031, OISE-0730065). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the NSF.

We want to make a special acknowledgment to all the people who contributed to this work: Reng Zeng, Yu Huang, Jingxuan Li, and Su Liu from Florida International University; Song Jiewei, Wang Kongmin, and Li Ji from CNIC in China; and Rogerio Rondini from Polytechnic School from Sao Paulo University.

REFERENCES

[1] "The greenlight project," http://greenlight.calit2.net//.

[2] J. G. Koomey, "Estimating total power consumption by servers in the u.s. and the world," in *Technical report, Lawrence Berkeley National Laboratory*, 2007.

[3] E.P.A., "Report on server and data center energy efficiency," in *Technical report, US Environmental Protection Agency, Energy Star Program*, 2007.

[4] J. Liu, "The PRIME research," http://www.cis.fiu.edu/prime/.

[5] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *SIGARCH Comput. Archit. News*, vol. 28, no. 2, 2000.

[6] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John, "Using complete machine simulation for software power estimation: The softwatt approach," in *HPCA '02: Proceedings of the 8th International Symposium on High-Performance Computer Architecture*. IEEE Computer Society, 2002, p. 141.

[7] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis, "Joulesort: a balanced energy-efficiency benchmark," in *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007.

[8] K. I. Farkas, J. Flinn, G. Back, D. Grunwald, and J. M. Anderson, "Quantifying the energy consumption of a pocket computer and a java virtual machine," *SIGMETRICS Perform. Eval. Rev.*, no. 1, 2000.

[9] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," in *ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design*. ACM, 2001.

[10] A. Kansal and F. Zhao, "Fine-grained energy profiling for power-aware application design," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, 2008.

[11] "Intel 64 tesla linux cluster lincoln," http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/Intel64TeslaCluster/.

[12] J. Cowie, D. Nicol, and A. Ogielski, "Modeling the global Internet," *Computing in Science and Engineering*, vol. 1, no. 1, pp. 42–50, January 1999.

[13] J. Liu, Y. Li, and Y. He, "A large-scale real-time network simulation study using prime," in *Proceedings of the 2009 Winter Simulation Conference (WSC)*, 2009.

[14] M. A. Erazo, Y. Li, and J. Liu, "Sveet! a scalable virtualized evaluation environment for tcp," in *In Proceedings of 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops(TridentCom 2009)*, 2009.

[15] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[16] E. F. Freedman, M. J. and D. Mazieres, "Democratizing content publication with coral," in *In Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI04)*, 2004.

[17] "Teragrid," https://www.teragrid.org/.

[18] A. Kansal and F. Zhao, "Fine-grained energy profiling for power-aware application design," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, 2008.

[19] "Perfsuite," http://perfsuite.ncsa.uiuc.edu/.

[20] "Performance application programming interface," http://icl.cs.utk.edu/papi/.

[21] "METIS - Family of Multilevel Partitioning Algorithms," http://glaros.dtc.umn.edu/gkhome/views/metis.