HolDCSim: A Holistic Simulator for Data Centers

Fan Yao[†], Kathy Ngyugen[‡], Sai Santosh Dayapule[‡], Jingxin Wu[‡], Bingqian Lu[§], Suresh Subramaniam[‡], and Guru Venkataramani[‡]

[‡]Department of ECE, The George Washington University
[†]Department of ECE, University of Central Florida
[§]Department of CSE, University of California Riverside

***Abstract*—A comprehensive *data center simulation infrastructure*, which models major hardware and system components and offers interfaces to manage the interplay between computation and communication resources, is critical in advancing future research for more effective performance and energy optimization in these environments. In this paper, we present *HolDCSim*, a light-weight, holistic, extensible, event-driven data center simulation platform that effectively models both server and network architectures. HolDCSim can be used in a variety of data center system studies including job/task scheduling, resource provisioning, global and local server farm power management, and performance analysis. We demonstrate the design of our simulation infrastructure, and illustrate the usefulness of our framework with a case study that analyzes server-network performance and energy efficiency. We also perform validation studies for our simulator on a physical testbed with Intel Xeon-based processors and Cisco network switches.**

I. Introduction

Data center infrastructure typically involves thousands of servers, hundreds of network devices (e.g., switches, routers), and complex network topology configurations. Oftentimes, physical access to large-scale testbeds is very limited, restricting researchers' ability to understand the landscape for optimizing these systems on performance and energy fronts. Therefore, a comprehensive simulation platform that models both computation and communication hardware is a necessary tool for effective and rapid modeling of such systems, and would enable *end-to-end, holistic* large-scale data center workload studies including workload characterization, performance engineering, and energy optimization.

Existing simulation tools generally fall into two classes: 1) simulators that model details of the machine hardware (e.g., [3]). These tools are able to provide cycle-accurate simulation with fine-grained performance and power models; 2) tools that simulate a cluster of servers in a distributed setting where each individual server is modeled as a pool of resource slots for task scheduling [6], [10]. While these simulators have been demonstrated to be useful in several use cases, they have potential shortcomings that make them *less effective* for *comprehensive* studies of data center systems: (1) architecture simulators typically take many hours to simulate program execution spanning just a few seconds in real systems. This makes them unsuitable to capture macro-level operations for distributed applications running in data center systems. (2) existing cloud simulators either model server-only [10] or network-only data center components [11], or they model only at a very high level largely abstracting away hardware characteristics (e.g., low power states for line cards) that make them less effective in simulating fine-grained hardware activities for certain applications such as latency-critical workloads.

As data center systems continue to evolve, it is crucial to understand application performance and power characteristics by considering both server and networking hardware components, along with their interactions as a whole. Firstly, there needs to be an appropriate level of abstraction for servers that sufficiently exposes hardware-level knobs (e.g., core- and package-level power management features) that could be leveraged by high-level system components. Secondly, the simulator will need to jointly incorporate computation and communication in a systematic manner in order to enable full control of the system across the hardware and software stack.

In this paper, we present *HolDCSim*, a lightweight, highly-extensible event-driven data center simulator that jointly simulates data center servers and networks. *HolDCSim* efficiently models the components of server and network devices with necessary hardware details (queuing, dynamic power management and idle power management) to capture end-to-end application performance and power characteristics. We systematically present the various modules in *HolDCSim* along with our overall design. HolDCSim considers queuing effects from both network and servers, and accounts for performance based on various sources of delays in them. Finally, it provides interfaces to access many state-of-the-art power management features that are available in modern server and network switch systems, which enable future studies on joint server and network power optimization.

II. Need for Holistic Data Center Simulation

Today's data center systems offer high request throughputs using a considerable amount of networking and computing resources. When a job request is sent to a data center, it typically involves spawning of multiple tasks that are inter-dependent. The computation and inter-task communication exercise both server and network resources in data centers. Therefore, considering data center server and networking hardware components holistically is becoming necessary for optimizing performance and energy efficiency for many scale-out applications (such as latency-critical workloads).

Over the past decade, several simulators have been developed specifically for data centers. Bighouse [10] models a cluster of servers with support for multi-core processing and

low-power state management. However, it does not consider network aspects including switching devices and network topologies. Network simulation tools such as *ns-3* [11] simulate detailed network protocols with modeling of topology, links and ports, but they are not able to provide server performance analysis. Moreover, they do not scale well due to the excessively detailed implementation of a single system component (i.e., network protocols). Finally, cloud simulators such as CloudSim [6], DCSim [12] consider resources at a much coarser level (i.e., VM instances), and do not render systematic interfaces for accessing hardware-supported power management in server and network devices, making them less effective for energy optimization-related studies. We note that it is critical to have a comprehensive simulator that allows users to have a more holistic view of data centers by modeling both data center server and network resources with sufficient integration of low-level hardware characteristics. Specifically we envision that in order to formulate effective system power policies and performance tuning, it is important for the simulation platform to provide the following modeling capabilities:

- *Multi-core/multi-processor server simulation with support for hardware-level parallelism.* Modeling of resource sharing in multi-core and multi-processor servers is critical in understanding performance of today's distributed applications.
- *Modeling of global and local job scheduling.* Data centers commonly feature a global-level job scheduler that dispatches job requests to individual servers, and a local scheduler within each server which maps tasks to each execution unit. Scheduling policies can have significant performance and energy efficiency impact.
- *Inclusion of comprehensive power state control mechanism.* Particularly, the simulator should offer existing and emerging power management mechanisms that can be leveraged for enhancing idle power and dynamic power consumption.
- *Integrated modeling of servers, network devices and topology.* Existing research have shown that packet forwarding and network topology configuration play a crucial role in determining the overall application performance. Information on how the network and server contribute the performance and power can greatly guide server/network management techniques.

III. HolDCSim System Design

To satisfy the aforementioned needs for holistic data center simulations, we design a new data center simulation infrastructure called HolDCSim. At a high level, HolDCSim has three major components: *workload generator module*, *server module*, and *network topology/switch module*. The *workload generator module* generates load to the simulated data center by injecting job requests. The server module instantiates a cluster of servers based on a configurable user script. Servers schedule local hardware resources including core and memory to each task. Core performance is determined by its hardware configuration and task settings.

HolDCSim simulates a complete data center infrastructure by modeling network devices and interconnection among various nodes in the system. The network module creates a complete topology by connecting the switches and servers with network links. Each network switch contains several key hardware components including *ports*, *line cards*, and *chassis*. HolDCSim models packet queuing and packet-forwarding for each switch. Similar to servers, we also build a power model considering various switch power management features. HolDCSim takes a workload model, and server and switch profiles as inputs to run experiments. During simulation, HolDCSim keeps track of several types of runtime statistics including power and energy consumption, network delays, job latency, and power state transitions.

Server Architecture. Our simulator can model servers with multi-core multi-socket processors. Cores have the capability to enter multiple levels of power states. Each server maintains a local queue where all incoming task requests are buffered. Meanwhile, each core can also have its local task queue, and serves one task at a time. Queuing delays are taken into account for task processing latencies. The task processing time is determined by the service time of the task and the operating frequency of the core on which it executes.

HolDCSim models server power based on the widely deployed Advanced Configuration and Power Interface (ACPI standard) [4]. ACPI uses global states (G_x) to represent the state of the entire system, system sleep states (S_x) to define power status for various server components, and C states to enable fine-grained core and uncore components' low-power modes to achieve various levels of power savings. Modern processors generally provide high parallelism by integrating multiple cores within a processor package. Low-power C states are supported at both core and package levels that are modeled in HolDCSim. Finally, performance states can be configured to determine the speed of instruction execution at runtime (i.e., dynamic frequency and voltage scaling). Our simulator supports per-core DVFS management in order to mimic the capability of recent server-class processors.

Switch and Network Architecture. The profile of network switches is modeled from real systems and online power modeling tools. Network switches can have multiple line cards, and each line card contains many ports. Configurable parameters of line cards include number of ports and power consumption in different power states. The link rate, buffer size, power state, and transition delay can be configured for each port.

In our simulator, we assume three power states for each switch port: active, LPI (Low Power Idle [5]), and off state. Currently, we are also assuming three power states for each switch line card: active, sleep, and off. HolDCSim provides the interfaces to model switch-based, server-based, and hybrid network architectures [15].

Job and Task Modeling. In HolDCSim, each job consists of

multiple tasks that may have both spatial (e.g., multi-tiered applications) and temporal dependence (e.g., input/output dependence). Servers in the simulated environment can be configured to perform different tasks. Tasks within a job have two types of inter-dependence: 1) spatial inter-dependence, which specifies a server a specific task could be dispatched to. For example, a data center query task can only be served by the database servers in the data center; and 2) temporal inter-dependence, wherein a task cannot start executing until all of its parent tasks have finished their execution, and until after their results have been communicated to the server assigned to the task. To account for these dependencies, we model each job j as a directed acyclic graph (DAG) $G^j(V^j, E^j)$, where V^j is the set of tasks of job j. In the DAG, if there is a link from task i to task r, then task i^j must finish and communicate its results to task r^j before r^j can start processing. Each task $v^j \in V^j$ has a workload requirement, namely task size or execution time requirement w_v^j for the core. For each link in E^j, there is a data transfer size D_l^j associated with it, which denotes the bandwidth requirement to transfer the result over link l (from the task at the head of DAG link to the task at the tail) when assigned a network flow.

Workload Modeling. We use two types of workload arrival models: synthetic workloads based on stochastic process, and actual system trace-based workloads. Synthetic workloads based on stochastic process can be modeled in two ways: 1. *Poisson-based job arrivals*, where the job inter-arrival and service times are follow an exponential distribution with means $1/\lambda$ and $1/\mu$, where λ is the job arrival rate and μ is the server service rate. This job arrival pattern could be used to model non-bursty data center workload traffic. 2. *MMPP-based job arrivals* uses a continuous-time Markov chain to model different stages or states of the workload. By orchestrating the transitions among various states with high and low λs, *MMPP* is able to model workload *burstiness* at a finer-grain level. Finally, HolDCSim supports use of data center traces collected from real systems.

Job/Task Scheduling. The simulated data center has a global scheduler which receives job requests from the front end. It is responsible for constructing a set of inter-dependent tasks corresponding to the request. The global scheduler then assigns tasks to servers based on a programmable static/dynamic scheduling policy. After a task has been assigned to a server, it will be passed on to the server's local task scheduler. The local task scheduler performs task assignment based on the availability of processor cores.

Power Model. Users can derive system power characteristics either by configuring the system in various activity states and making power measurements or by using other power estimation tools specifically for this purpose. In the first case, the power measurement can be done by reading performance counter in the processor (e.g., Intel RAPL interface [7]) or by using external power meters. The second option is to build power profiles through power modeling tools such as CACTI [8]. Similar to server power, the user can model various network components in network switches such as ports and line cards. Line cards can be considered to have their own C-state and P-state to conserve power similar to servers. The default C-state controller controls idle to sleep state transitions using the packet queue size as a threshold. The user has the capability to implement custom transition logic which can look at other observable behaviors such as traffic patterns.

IV. Case Study

Apart from providing the capability to characterize various power-saving schemes for servers [14], [16], [17], HolDCSim can also be used to study techniques that co-optimize server and network energy. In this paper, we present a case study to demonstrate such capability in HolDCSim[1].

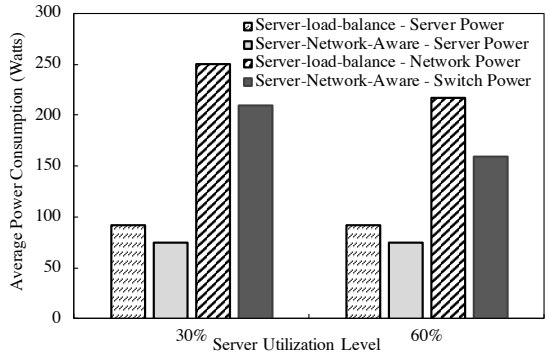
Specifically, we modeled a data center using the widely deployed fat-tree topology [2]. To simulate network traffic, each job is simulated as a set of inter-dependent tasks. The dependence among tasks is modeled as a DAG where traffic patterns among these tasks are known. We model our power management strategy in HolDCSim as follows: whenever there is a need for an additional server to transit to active state, it would first identify the server with the *least network cost*—the amount of additional switches to be woken up in order to allow communications to that server. For baseline comparison, we consider a policy where jobs are strictly load balanced among servers. In this experiment, the network module models flow-based communication. With HolDCSim, we study the latency and power consumption for a set of jobs with randomly assigned job execution time. Figure 1a presents the results for average power consumption for web search application, along with the CDF of job execution latency in Figure 1b for a simulation of 2000 jobs using Poisson arrivals. We can see that we can obtain about 20% server and 18% network power savings with negligible increase in job latency.

V. Validation of HolDCSim Components

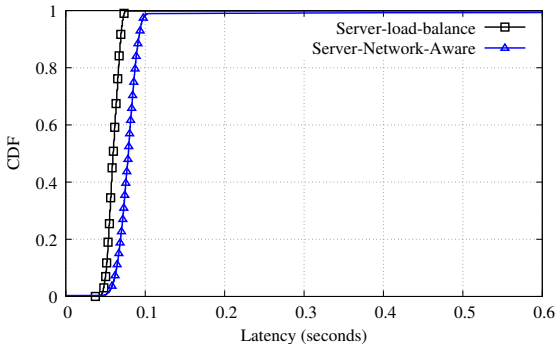
Server Power Validation. In order to validate the power model of HolDCSim, we set up experiments based on a 10-core Intel Xeon E5-2680 processor server. We utilize the publicly-accessible NLANR trace [1], which includes job arrival times for web service requests. We set up apache web service on the Xeon-based machine to serve users' requests.

For this experiment, we enabled two sleep states for the processor: C0 and C6. We measured the power consumption for each core as well as the entire package when they are in these sleep states. Based on the power profile, the power model for a 10-core processor server is configured in HolDCSim. We then replay the same NLANR trace in the simulator by enabling trace-based simulation. The power consumption statistics are generated and collected in HolDCSim. Our validation experiment shows that the power profile generated using HolDCSim matches with the power consumption curve of our physical machine with negligible error. We observe that the average power difference between the physical server and the

[1] A full paper of this work with detailed case studies can be accessed at [18].



(a) Server and network power consumption



(b) Job response time

Fig. 1: Server and network power consumption (top figure) and job response time CDF (bottom figure).

simulated server is only 0.22W, which indicates a minimal error (around 1.3%). Additionally, the standard deviation on simulator power is only about 1.5 W. Overall, the patterns and values in the simulated and physical servers' power traces match closely, validating that our simulator can faithfully model servers' power consumption.

Switch Power Validation. The swtich power validation was performed on a `Cisco WS-C2960-24-S` network switch. In the simulator, we set up 24 servers connected to one switch using the star topology. We configure the switch power model using the power profile of the physical switch. The simulated switch has 24 ports, a base power of 14.7W, and a per port power of 0.23W. The cluster is configured to simulate a Wikipedia web service using load balanced scheduling policy. Our load generator generates user requests to the simulated cluster using the Wikipedia trace [13]. We implement a script for the physical switch which controls the status of the switch (including the line card and each of the ports). We use a physical power meter to monitor the switch power. The power consumption of the physical switch is sampled every 1 second. We observe that the average power difference is less than 0.12W with a standard deviation of 0.04W for a 2-hour trace simulation. In summary, we can see that HolDCSim can capture the power profiles for switches fairly accurately under realistic data center workloads.

VI. Conclusion and Future Work

In this work, we demonstrated *HolDCSim*, a light-weight, holistic, extensible event-driven data center simulation platform that effectively models both server and network architectures. HolDCSim is able to guide users for a variety of data center system studies including understanding workload patterns in job execution, data center resource provisioning, global and local power management, as well as detailed combined network and server performance analysis.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant Number CNS-178133. Kathy Nguyen was supported through a REU supplement under the NSF award. All of this work was performed when Fan Yao and Bingqian Lu were PhD students at GWU.

References

[1] (2012) The NLANR Projects. http://www.nlanr.net.
[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, 2008.
[3] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, 2011.
[4] L. Brown, "ACPI in linux," in *Linux Symposium*, 2005.
[5] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro, "IEEE 802.3 az: the road to energy efficient ethernet," *IEEE Communications Magazine*, 2010.
[6] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations," in *International Conference on Utility and Cloud Computing*. IEEE, 2011.
[7] Intel, "Intel R 64 and IA-32 Architectures Software Developer Manual," 2013.
[8] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-P: Architecture-level modeling for sram-based structures with advanced leakage reduction techniques," in *ICCAD*. IEEE, 2011.
[9] B. Lu, S. S. Dayapule, F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "PopCorns: Power Optimization Using a Cooperative Network-Server Approach for Data Centers," in *ICCCN*. IEEE, 2018.
[10] D. Meisner, J. Wu, and T. F. Wenisch, "Bighouse: A simulation infrastructure for data center systems," in *ISPASS*. IEEE, 2012, pp. 35–45.
[11] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Springer Berlin Heidelberg, 2010.
[12] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *CNSM*. IEEE, 2012.
[13] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, 2009.
[14] F. Yao, J. Wu, S. Subramaniam, and G. Venkataramani, "WASP: Workload Adaptive Energy-Latency Optimization in Server Farms Using Server Low-Power States," in *CLOUD*. IEEE, 2017.
[15] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "A comparative analysis of data center network architectures," in *ICC*. IEEE, 2014.
[16] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "A Dual Delay Timer Strategy for Optimizing Server Farm Energy," in *CloudCom*. IEEE, 2015.
[17] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "TS-BatPro: Improving Energy Efficiency in Data Centers by Leveraging Temporal-spatial Batching," *IEEE Translations on Green Communications and Networking*, 2018.
[18] F. Yao, K. Ngyugen, S. S. Dayapule, J. Wu, B. Liu, S. Subramaniam, and G. Venkataramani, "HolDCSim: A holistic simulator for data centers," *arXiv preprint arXiv:1909.13548*, 2019. [Online]. Available: https://arxiv.org/abs/1909.13548