This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2016.2597818, IEEE Computer Architecture Letters

Cloud Server Benchmark Suite for Evaluating New Hardware Architectures

Hao Wu, Fangfei Liu, and Ruby B. Lee
Princeton University, Princeton, NJ, 08544 USA
E-mail: haow.princeton@gmail.com, {fangfeil, rblee}@princeton.edu

Abstract—Adding new hardware features to a cloud computing server requires testing both the functionality and the performance of the new hardware mechanisms. However, commonly used cloud computing server workloads are not well-represented by the SPEC integer and floating-point benchmark and Parsec suites typically used by the computer architecture community. Existing cloud benchmark suites for scale-out or scale-up computing are not representative of the most common cloud usage, and are very difficult to run on a cycle-accurate simulator that can accurately model new hardware, like gem5. In this paper, we present PALMScloud, a suite of cloud computing benchmarks for performance evaluation of cloud servers, that is ready to run on the gem5 cycle-accurate simulator. We conduct a behavior characterization and analysis of the benchmarks. We hope that these cloud benchmarks, ready to run on a dual-machine gem5 simulator or on real machines, can be useful to other researchers interested in improving hardware micro-architecture and cloud server performance.

Index Terms—Cloud Computing, Benchmarks, Behavior Characterization, Performance Evaluation, Gem5, Simulation.

1 Introduction

Simulation of new hardware architecture is a necessary stage in computer hardware design. Ideally, we would like to evaluate the functionalities and performance on a detailed simulation platform at the design stage of the new systems before committing to expensive chip fabrication and prototype systems. There are open-source cycle-accurate simulators that have been worked on for a long time (e.g., gem5 [18], PTLsim [28], MARSSx86 [27], etc), that simulate hardware at a detailed level, and provide good performance data. They can also be used to add new hardware features into the detailed hardware models.

However, demonstrating performance of different hardware configurations requires representative benchmarks for the performance evaluation of today's computing environments. Frequently used benchmarks for performance may not be representative of current or future computing paradigms. Currently, the computer architecture community uses SPEC Integer and Floating-point Benchmarks [16] for general-purpose computing and some PARSEC benchmarks [12] for parallel computing workloads. These are mostly compute-intensive benchmarks and do not represent today's Cloud Computing scenarios, e.g., Amazon EC2 [1]. Cloud Computing provides different IT resources (e.g. computing, storage, software development, system testing, etc.) as services on demand to customers. So the most critical attribute of Cloud Computing is resource virtualization. For example a company can buy real hardware resources (e.g. servers, etc.), or instead, lease Virtual Machines (VMs) from cloud infrastructure providers. The company may deploy web servers and application servers in several of these VMs, and deploy data servers (e.g. MySQL) in other VMs. In this paper, we select a suite of representative cloud computing benchmarks from the most common cloud computing services and workloads, and we perform a detailed benchmark behavior characterization.

The contributions of this paper are:

- PALMScloud: a representative suite of cloud computing benchmarks, ready to run on gem5, with benchmark characterization and analysis.
- Open availability of the simulation framework and cloud computing benchmarks, for researchers [11].

Section 2 discusses representative cloud computing server benchmarks, and a behavior characterization and analysis of the benchmarks. Section 3 describes how we set up the dual system to run PALMScloud in the gem5 simulator. Section 4 concludes the paper.

2 Cloud Computing Server Benchmarks

Cloud computing services can be categorized into several fundamental models [21]: Infrastructure as a service (IaaS), Platform as a service (PaaS), Software as a service (SaaS), etc. Rackspace, the leader in hybrid cloud and founder of OpenStack, put together a Top-10 list [19] of the most common cloud computing use cases: File Storage and Sharing, Cloud Database, Email, PaaS for Web Applications, Web Site Hosting, etc. Some obsolete benchmarks (e.g., SPECweb [16], SPECmail [16], etc.) and currently in-use benchmarks (e.g., SPECjbb [16], TPC-C [20], TPC-W [20], etc.) only cover some of these. Also, since they target specific commercial purposes and include too many features, they run way too slow under a cycle-accurate simulator like gem5. CloudSuite [23] is a recent benchmark suite for scale-out workloads, covering many of the Top-10 list. However, strictly speaking, most of them are big data applications (e.g. Hadoop Mapreduce for Data Analytics, Memcached

This work was supported in part by DHS/AFRL FA8750-12-2-0295 and NSF CNS-1218817.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2016.2597818, IEEE Computer Architecture Letters

TABLE 1: PALMScloud Server Benchmarks and Client-side Driving Tools and Commands

	Server	Client	Client-side Stressing Commands and Benchmark Descriptions
Web Server	Apache httpd	Apache ab	*ab -c 10 -t 120 http://server-ip:8080/* send HTTP requests to the server with a concurrency of 10 requests at the same time, lasting 120 seconds
Database Server	MySQL	SysBench	*sysbench –test=oltp –mysql-host=server-ip –mysql-user=... –mysql-password=... –mysql-db=test –oltp-table-name=sbtest –db-driver=mysql –mysql-engine-trx=yes –num-threads=4 –max-time=120 –oltp-table-size=10000* preparation: create test tables with 10000 records running: do advanced transactions with 4 concurrent threads, lasting 120 seconds
Mail Server	Postfix	Postal	*postal -t 1 -m 1 -c 3 send-list server-ip rcpt-list* 1 connection, maximum message size is 1 KB, 3 messages per connection
File Server	Samba smbd	DBench	smbd.write: *dbench -B smb -smb-share//server-ip/share -smb-user=% -loadfile=smb-writefiles.txt 2* smbd.read: *dbench -B smb -smb-share//server-ip/share -smb-user=% -loadfile=smb-readfiles.txt 2*
Streaming Server	ffserver	openRTSP	*openRTSP -r -p [local-port] rtsp://server-ip:7654/*.mp3* 3 workloads: X remote requests (X=1, 3 and 30)
Application Server	Tomcat	Apache ab	*ab -c 2 -t 120 http://server-ip:8080/(X URLS)* Send HTTP requests to the server with a concurrency of 2 requests for each URL, lasting 120 seconds 3 workloads: X=1, 3 and 11 (different URLs contain different jsp and servlet examples)
Compute Server	libsvm	a1a.t	*svm-predict a1a.t train.model test.output*

for Data Caching, Cassandra NoSQL for Data Serving, etc.), many of which are specifically designed for physical machines and require direct disk IO access. Thus CloudSuite benchmarks use big data infrastructures, which are hard to run on a cycle-accurate simulator. Virt-LM [24] is a suite of benchmarks used to evaluate the performance of live VM migration strategies among different software and hardware environments in a data center scenario, and includes 5 of Rackspace's top 10 benchmarks.

2.1 PALMScloud Workloads

Based on Virt-LM and Rackspace's Top-10 list, we carefully selected an initial set of 6 representative cloud computing workloads, including the workloads for a Web Server, Database Server, Mail Server, File Server, and Application Server, and also a Streaming Server, which is not in the Virt-LM or Rackspace's list. Below, we describe the programs we selected, and the parameters we use for each program to do the benchmark behavior characterization. We also select a popular compute-intensive (machine learning) benchmark for comparison. Table 1 summarizes our server benchmarks and the client-side driving tools and commands in our PALMScloud benchmark suite.

Web Server and Client: Apache HTTP server (httpd) [3] has been the most popular web server on the Internet since 1996. The project aims to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. For the client side, we choose **Apache Benchmark Tool (ab)** [2], which is a single-threaded command line benchmarking tool well suited for a non-GUI testing environment under gem5. This tool allows picking the stressing time limit (in seconds) and the number of concurrent requests.

Database Server and Client: MySQL [9] is a well-known open-source relational Database Management System (DBMS). For the client side, we choose **SysBench** [17]. It is a modular, cross-platform and multi-threaded benchmark tool for evaluating OS parameters that are important for a system running a database under intensive load. In SysBench, we use the OLTP (on-line transaction processing) test mode, which benchmarks a real database's performance. In the preparation phase, we create test tables with 10000 records. For the running phase, we use 4 client threads to do advanced transactions continuously.

Mail Server and Client: We mainly focus on Simple Mail Transfer Protocol (SMTP), and we choose **postfix** [14] to act as the SMTP server. We set the server side IP address to link with domain: domain1.com. Postfix is on the server side, and is responsible for delivering all the emails. The receivers of email are local users in domain1.com. During the server initialization phase, we add 20 local users: user0 ~ user19 for our testing purposes. For the client, we use **postal** [13] to send messages to the server. **Postal** aims at benchmarking mail server performance. It shows how fast the system can process incoming email.

File Server and Client: SMB provides file sharing and printing services to Windows clients as well as Linux clients. We use the SMB protocol and choose **Samba smbd** [15] as the file server. On the client side, we choose **Dbench** [6] as the workload generator. It can generate different I/O workloads to stress either a file system or a networked server. We need to specify **dbench**'s stressing backend (smb in this case), the shared file server folder and the user-password pair in the command line configuration. Moreover, **Dbench** has a key concept of a "loadfile", which is a sequence of operations to be performed on the file server's shared folder. The operations could be "Open file 1.txt", "Read X bytes from offset Y in file 2.txt", "Close the file", etc. In our experiments, we generate two different "loadfiles", one is a write-intensive load (smbd-writefiles.txt), another is a read-intensive load (smb-readfiles.txt). Finally, we can add a number **n** at the end of the **dbench** command to specify the total clients simultaneouly performing the load.

Streaming Server and Client: We use **ffserver** [7] as our streaming server. It is a streaming server for both audio and video, supporting mp3, mpg, wav, etc. ffserver is part of the ffmpeg package. It is small and robust. Before starting the server, we need to register server side media-files at ffserver.conf file. On the client side, we choose **openRTSP** [10]. RTSP protocol can control the streaming. Clients issue VCR-like commands, such as play and pause, to facilitate real-time control of playback of media files from the server. In the experiment, ffserver registered several mp3 audio files on the server side. In section 2.2, ffserver.sX means using **openRTSP** to send X remote client connection requests to the ffserver for mp3 files streaming.

Application Server and Client: For web applications, the application server components' main job is to support the construction of dynamic pages [5]. Application servers differ from web servers by dynamically generating html pages each time a request is received, while most http

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2016.2597818, IEEE Computer Architecture Letters

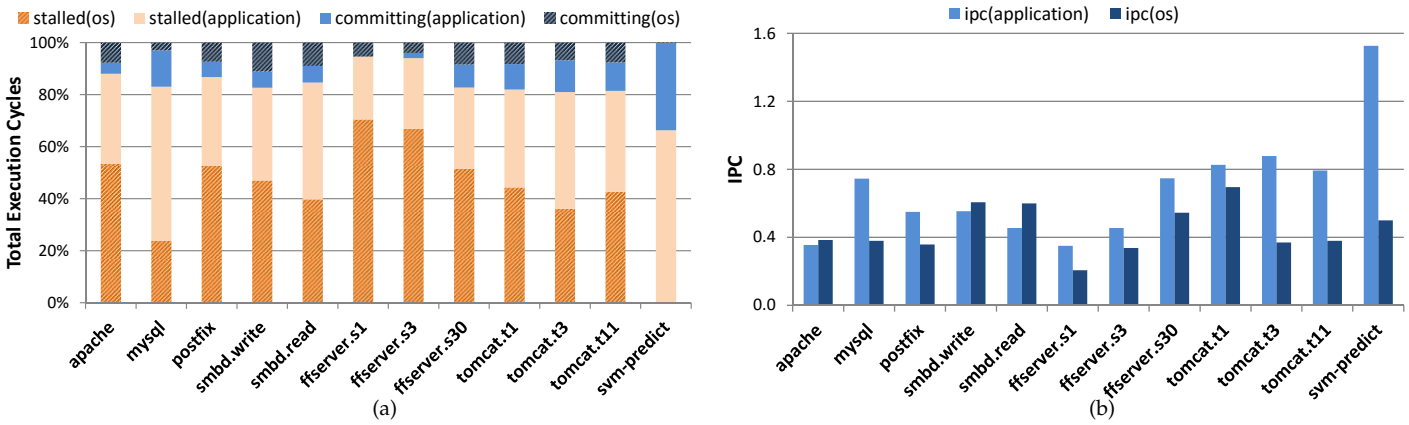

Fig. 1: PALMScloud: (a) Execution Cycles Breakdown, and (b) IPC

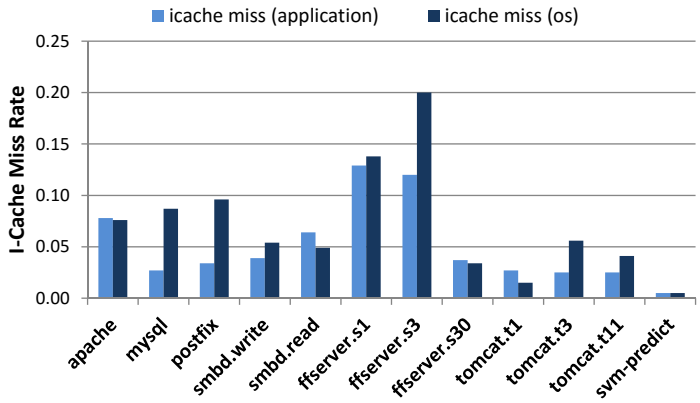

Fig. 2: I-Cache Miss Rate

servers just fetch static web pages. Application servers can utilize server-side scripting languages (PHP, ASP, JSP, etc.) and Servlets to generate dynamic content. We use **Tomcat** [4] on the server side. It is a small, robust application server that also provides many useful small jsp and servlet examples. For the testing, we use **Apache ab** to send HTTP requests to Tomcat (Apache ab is described previously as a web server client).

Compute Server (Machine Learning): We also want to use some computing benchmarks to represent compute servers. We choose LIBSVM [22] to do Support Vector Machine (SVM) classification. On the server side, we have already trained an SVM model using data set a1a's training data from UCI's machine learning repository [25]. a1a stands for Adult Data Set, which can be used to predict whether a person's income exceeds $50K/yr based on his census data. We use this model to do SVM classification on a1a's testing data: a1a.t.

TABLE 2: Server's Architecture Parameters

Processor	32nm Intel Xeon E5-1410, operating at 2.80GHz, SMT enabled
Microarchitecture	Sandy Bridge
# CPU Cores/ # HW Threads	4/8
Core Width	4-wide issue and retire
Reorder Buffer	168 entries
Reservation Stations	54 entries
Load/Store Buffers	64/36 entries
L1 Cache	32KB I/D, 4-cycle latency
L2 Cache	256KB per core, 12-cycle latency
LLC (L3 Cache)	10MB, 28-cycle latency
Memory	32GB

2.2 Benchmark Characterization

We conduct our characterization study on a dual-machine setting. All the server-side benchmarks are installed and configured in a PowerEdge R320 rackmount chasis, while all the workload-driving tools reside in another PowerEdge R720. PowerEdge R320 server contains an Intel Xeon E5-1410 processor with 4-OoO cores (8 HW Threads with SMT enabled in our study), and the processor has a three-level cache hierarchy, with private L1/L2 cache and shared LLC. Table 2 summarizes the processor and hardware parameters. The server runs Ubuntu 14.04 LTS, with Linux kernel 3.5.0.

To analyze the micro-architectural behavior of the benchmarks, we use Intel Vtune [8] to monitor the server-side benchmarks on the PowerEdge R320. Vtune is a software tool that has access to the processor performance counters. For all the workloads, we start by launching the client-side stressing tools first, then we perform a 60-second measurement on the server-side benchmarks after they finish the ramp up phase. Below, we present the characterization results of the workloads.

Figure 1a shows the breakdown of the total execution cycles of the workloads. Nowadays, many CPUs have out-of-order (OoO) multi-width superscalar cores. Our server-side CPU uses Sandy Bridge-EN cores, which have 4-wide issue and retire pipeline slots. In each cycle, the processor front-end generates up to 4 micro-ops (μops) placed into the pipeline slots, which means the ideal IPC for one core would be 4. However, usually not all of the slots are filled with useful μops (squashed due to branch misprediction or stalled due to back-end resource starvation or data unavailability). In Figure 1a, committing cycles reflect the amount of μops that actually get committed into the architectural state. For example, if during the commit cycle, 3 slots have useful μops that get committed, we count 75% of that cycle as committing, and 25% of that cycle as stalled. The total commit cycles and the total stalled cycles add up to 100%.

Figure 1a shows that the execution cycles of the PALMScloud workloads are dominated by stalls in both application and OS codes. Unlike the first 6 cloud workloads, the last workload, svm, hardly traps into the OS. This compute-intensive benchmark behaves like the compute-intensive desktop (SPECint) and parallel (PARSEC) benchmarks analyzed in [23]. Also, from Figure 1b, our benchmarks (not including svm) have an average IPC of 0.69 for application

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2016.2597818, IEEE Computer Architecture Letters

codes, and 0.45 for OS codes, in comparison with svm's application IPC 1.53, SPECint's average application IPC 1.4 [23], and PARSEC's average application IPC 2.0 [23]. Many software modules in our benchmarks require trapping into the OS kernel a lot, whose intrinsic behavior, along with the poor instruction-level parallelism in both application and OS codes, could be the main reasons that cause the low IPCs.

Figure 2 shows the I-Cache miss rates for both application and OS codes. For our cloud benchmarks (minus svm), the average application code I-Cache miss (5.5%) and OS code I-Cache miss (7.7%) are much larger than those of the svm benchmark ($<$ 1%), the SPECint benchmarks ($<$ 1%) and the PARSEC benchmarks ($<$ 1%) [23]. This indicates that the cloud benchmarks' instruction working sets tend to exceed the I-Cache capacity compared with desktop benchmarks.

In summary, our Cloud benchmarks differ from SPEC and PARSEC benchmarks in exhibiting relatively low IPCs, high I-Cache miss rates and a significant amount of stalls in both application and OS codes. The low IPC can be explained by the large amount of stalls, which are likely due to the significant number of networking and disk accesses, and handling of user connections and requests, which all require trapping into the OS kernel, incurring a lot of context switches. Also, OS code tends to have a lot of dependencies, which reduces parallelism and lowers the IPC.

3 Running PALMScloud on Gem5 Simulator

We have prepared our cloud benchmark suite to run easily on the gem5 simulator using a flexible client-server dual system configuration, as shown in Figure 3. In particular, the server, which is the test system, is run in cycle-accurate CPU mode, while the client, which is the drive system, can use a simple CPU mode with only functional modeling in gem5. The server and the client can communicate via the Ethernet link and each gets an IP address. By taking advantage of the dual system, we can adjust the driving workload from the client side and measure the performance on the server side, instead of using a server-side fixed input script. Also, the dual system simulates a real two-node network environment. The server applications provide services and monitor input requests on their specified TCP ports, while the client programs request for services (HTTP requests, streaming requests, etc.) through these server-side ports.

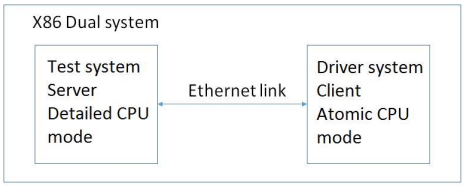

Fig. 3: Running PALMScloud on Gem5 Simulator with Dual-System Configuration

It's not easy to install server benchmarks under the gem5 simulator system. So we install them on a real machine, and copy the installed package and all the required dynamic libraries to gem5's image disk. Both server side and client side use the basic Linux system installed on gem5's image disk. The server has different server-side applications installed. When booting the dual system, both server and client configure their network interface. The server then initiates the server-side services and sends a ready signal to the client, so that the client can drive the server side with requests. Some server-side benchmarks actually require some time to start up before their services are available to the client under gem5. We use a daemon program to test periodically if the service ports are opened up by the benchmark. Only at that time will the server send a 'ready' signal to the client.

4 Conclusions

We have defined a new Cloud benchmark suite that represents the common workloads of today's public cloud computing usage. We did a behavior characterization and performance analysis of the benchmarks, and compared them to compute-intensive benchmarks - cloud workloads are often network and I/O intensive benchmarks. A detailed case study of the use of our PALMScloud benchmark suite can be found in [26]; this evaluates the performance of a cloud server implementing a new secure cache architecture, Newcache, on gem5. With this paper, we are also providing opensource access to our cloud server benchmarks, already compiled, configured and tested for gem5 simulations. They can be downloaded from [11]. We hope that this paper and our PALMScloud benchmark suite can expedite cloud performance evaluation for other researchers when simulating new hardware features.

References

[1] "Amazon EC2," http://aws.amazon.com/ec2/.
[2] "Apache HTTP Server Benchmarking Tool," http://httpd.apache.org/docs/2.2/programs/ab.html.
[3] "Apache HTTP Server Project," http://httpd.apache.org/.
[4] "Apache Tomcat," http://tomcat.apache.org/.
[5] "Application Server Definition," http://en.wikipedia.org/wiki/Application_server.
[6] "Dbench Workloads Generator," http://dbench.samba.org/.
[7] "ffserver," http://www.ffmpeg.org/ffserver.html.
[8] "Intel Vtune Amplifier," https://software.intel.com/en-us/intel-vtune-amplifier-xe/.
[9] "MySQL," http://www.mysql.com/.
[10] "openRTSP," http://www.live555.com/openRTSP/.
[11] "PALMScloud," http://palms.ee.princeton.edu/PALMScloud.
[12] "PARSEC," http://parsec.cs.princeton.edu/.
[13] "Postal," http://doc.coker.com.au/projects/postal/.
[14] "Postfix," http://www.postfix.org/.
[15] "SMBD File Server," http://www.samba.org/.
[16] "SPEC Benchmarks," http://www.spec.org/.
[17] "SysBench," http://sysbench.sourceforge.net/.
[18] "The gem5 Simulator System," http://www.gem5.org.
[19] "Top 10 Uses for the Cloud for 2012," http://www.rackspace.com/blog/top-10-common-uses-for-the-cloud-for-2012/.
[20] "Transactions Processing Council," http://www.tpc.org/.
[21] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing: Principles and Paradigms*. Wiley Press, 2011.
[22] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM TIST*, vol. 2, 2011.
[23] M. Ferdman *et al.*, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," ser. ASPLOS '12, pp. 37–48.
[24] D. Huang, D. Ye, Q. He, J. Chen, and K. Ye, "Virt-lm: A benchmark for live migration of virtual machine," in *ICPE*, 2011, pp. 307–316.
[25] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[26] F. Liu, H. Wu, K. Mai, and R. B. Lee, "Newcache: secure cache architecture thwarting cache side channel attacks," *IEEE Micro*, vol. 36, no. 5, September 2016.
[27] A. Patel, F. Afram, S. Chen, and K. Ghose, "Marss: A full system simulator for multicore x86 cpus," in *DAC*, 2011, pp. 1050–1055.
[28] M. Yourst, "PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator," in *IEEE ISPASS*, 2007, pp. 23–34.