WPress: An Application-Driven Performance Benchmark For Cloud Virtual Machines

Amir Hossein Borhani*, Philipp Leitner†, Bu-Sung Lee‡, Xiaorong Li*, Terence Hung*

*Institute of High-Performance Computing (IHPC), 16-16 Connexis, Singapore 13863
Email: {borhaniah, lixr, terenceg, }@ihpc.a-star.edu.sg

†s.e.a.l. software evolution & architecture lab, University of Zurich, Switzerland
Email: leitner@ifi.uzh.ch

‡Parallel and Distributed Computing Center (PDCC), School of Computer Engineering,
Nanyang Technological University, Singapore 639798
Email: ebslee@ntu.edu.sg

***Abstract*—Approaching a comprehensive performance benchmark for on-line transaction processing (OLTP) applications in a cloud environment is a challenging task. Fundamental features of clouds, such as the pay-as-you-go pricing model and unknown underlying configuration of the system, are contrary to the basic assumptions of available benchmarks such as TPC-W or RUBiS. In this paper, we introduce a systematic performance benchmark approach for OLTP applications on public clouds that use virtual machines (VMs). We propose *WPress* benchmark, which is based on the widespread blogging software, WordPress, as a representative OLTP application and implement an open source workload generator. Furthermore, we utilize a CPU micro-benchmark to investigate CPU performance of cloud-based VMs in greater detail. Average response time and total VM cost are the performance metrics measured by *WPress*. We evaluate small and large instance types of three real-life cloud providers, Amazon EC2, Microsoft Azure and Rackspace cloud. Results imply that Rackspace cloud has better average response times and total VM cost on small instances. However, Microsoft Azure is preferable for large instance type.**

Keywords—Benchmarking, Cloud Computing, Virtual Machine, Amazon EC2, Microsoft Azure, Rackspace Cloud, OLTP, CPU Micro-Benchmark.

I. Introduction

Cloud computing [1] has seen significant adoption in recent years. Clouds are employed in preference to specialized clusters and supercomputers due to their reliability, scalability and cost effectiveness. The Infrastructure-as-a-service (IaaS) model [2] enables customers to rent computing and storage resources from different cloud providers in the form of VMs.

Recently, OLTP applications have benefited from public cloud platform for its large scale computing and storage capacities, effortless accessibility for customers, regular maintenance, high availability and pay-as-you-go cost model. TPC-W and RUBiS benchmarks have been widely used to evaluate the performance of OLTP applications in clouds [3]–[6]. Nevertheless, they are not originally designed for cloud computing platforms with unknown hardware configuration and pay-as-you-go pricing model [7]–[9].

To address this research gap, this paper will focus on achieving a performance benchmark for OLTP applications deployed on public cloud VMs. However, design and implementation of a comprehensive performance benchmark in a highly distributed computing environment raise considerable challenges. First, it is a time consuming task which involves several repetitions to present statistically reliable results. In addition, choosing a representative application and generating enough load need to be carried out proficiently. Furthermore, considerable amount of data generated from long-run experiments requires a suitable data collection approach. Moreover, basic requirements of benchmarks like fairness, relevancy, verifiability and cost effectiveness as already discussed by Folkerts and Huppler [3], [10] should be taken into consideration. We make the following contributions to address the challenges mentioned above:

- We propose *WPress* benchmark, which uses WordPress as a widely used open-source blogging and publishing platform in today's market [11]. We also describe various transactional use-cases for WordPress, which we argue make it a good representative for OLTP applications in general.
- We implement an open-source Benchmark Client Application, called *WPressClient*, to generate the most frequent tasks for WordPress and collect the results.
- We use a distributed infrastructure to test *WPress* on small and large instance types of Amazon EC2 [12], Microsoft Azure [13] and Rackspace Cloud [14].
- We run *WPress* on each cloud provider to evaluate and compare average response times and total cost of its VMs.
- We implement a CPU micro-benchmark as proposed in [15], to detect periods during which the VM is not assigned CPU time by the hypervisor. Results of the CPU micro-benchmark are used to investigate the effect of CPU performance on observed average response times and total cost of VMs.

The rest of this paper is organized as follows: Section II reviews previous works of benchmarking OLTP applications in the cloud. Section III describes design and implementation of *WPress*. Section IV illustrates and discusses the results of our proposed benchmark. Section V concludes the paper with a summary of our findings. Finally, Section VI provides an outlook over future research.

II. Related works

The most relevant works are reviewed in this section. Schad *et al.* [16] carried out a comprehensive study on performance variation of small and large instances on Amazon EC2. MapReduce application as well as micro-benchmarks including, *Unix benchmark utility* (Ubench) for CPU and Memory speed, Bonnie++ for Disk I/O and Iperf for network bandwidth, were used. Significant variations in performance of both small and large instances are illustrated. Lenk *et al.* [17] measured actual performance of virtual machines running a specific IaaS service. Phoronix test suite is used to evaluate Amazon EC2, Flexiscale and Rackspace Cloud. They pointed out the strong relation between CPU type and achieved performance on each provider. Liang *et al.* [18], proposed the CARE framework for evaluating different cloud application hosting servers and cloud databases and ran it on Amazon Web Services (AWS), Google App Engine and Microsoft Azure. Small compute instances on each provider were compared. It is shown that Amazon EC2 and Azure has larger response time than that of Google App when concurrent requests increase. In addition, different cloud databases including Amazon S3, Amazon SimpleDB, Amazon LocalDB, Azure table storage, Azure blob storage and App Engine datastore were tested in their work. Klems *et al* [19] proposed a quality measurement and analysis framework for runtime management of cloud database service systems. Cloud database service system is defined as a system which uses cloud database software on top of a distributed compute cloud. They showed that EC2-based cassandra cluster outperforms Amazon database services: SimpleDB and DynamoDB. Cloud storage services often sacrifice consistency for availability. Bermbach *et al.* [20] proposed an approach to benchmark consistency of Amazon's simple storage service (S3). Moreover, a benchmark for cloud-hosted storage systems to quantify the consistency guarantees is introduced in [21]. Curino *et al.* [22] proposed OLTP-Bench, a new benchmark to investigate the performance and resource consumption of relational database servers (RDS) used in the cloud environment. They used multiple workloads, including YCSB [23] and Wikipedia and ran the experiment on five EC2 RDS instance sizes: Small, Large, HighMem XLarge(XL-HM), HighMem 2XLarge(2XL-HM) and HighMem 4XLarge(4XL-HM). Although 4XL-HM has the maximum throughput and minimum latency, XL-HM has the best price/performance ratio for the majority of customers. Accurate estimation of the true cost users are charged for in Database-as-a-Service (DaaS) providers is another challenge discussed in [9]. The authors proposed the concept of BaaS (Benchmark-as-a-service), which aimed to provide transparent price estimation for customers based on their workload patterns.

Moreover, TPC [24] and RUBiS [25] benchmarks were extensively used to evaluate performance of cloud environment for OLTP applications [3]–[6], [10], [26]–[28]. Key aspects of a good benchmark are classified in [3]. TPC-Benchmark, including a series of benchmarks for performance evaluation of transactional database systems and web-based applications, was considered as a well structured benchmark. Folkerts *et al.* [10] extended Huppler's work to present a more comprehensive classification of benchmark requirements in cloud environments. They listed sample use-cases and proposed appropriate benchmarks for each use-case including TPC-C. They also discussed meaningful metrics, variable workloads, scalability, fairness and repeatability as the main challenges for designing a good benchmark in cloud environments. Kossmann *et al.* [4], [5] used TPC-W benchmark to evaluate alternative database architectures offered by AWS, Google App and Microsoft Azure. Significant differences were revealed among providers in terms of price plan and database architectures. In [4], it is shown that increasing workload leads to price reduction, which means that there is a fixed charge for AWS and Azure database offerings independent of workloads. Moreover, Azure and AWS S3 have the best throughput in these tests. Hill *et al.* [6] investigated the performance of computing, storage and database services offered by Microsoft Azure. For Azure database, they compared TPC-E benchmark results on three scenarios: 1) SQL-Server and clients are installed on the same machine, 2) SQL-Server and clients are located on the same local LAN and 3) SQL-Server and clients are installed on separate Azure VMs. Not surprisingly, for single threaded applications, Local LAN performs better than Azure cloud. However the latter is superior when the number of threads increased. However, inefficiencies of using TPC and RUBiS Benchmarks on cloud environments are discussed [7], [8]. Binning et al [7], disclosed inefficiencies of using TPC-Benchmark in public cloud environments. Major drawbacks for TPC-W are prerequisite of having fix configuration for the *system under test (SUT)*, forcing *SUT* to provide ACID properties (storage consistency) and unfitness of its measured metrics: (1) (*web interaction per second (WIPS)* and (2) *ratio of cost and performance ($/WIPS)*), in cloud environments. Deficiencies of using TPC benchmark in cloud environments were further discussed in [8] and [9]. Additionally, latest versions of TPC-W and RUBiS date back to 2003 and 2004, respectively. Therefore, they cannot properly represent Web 2.0 applications [29]. It is also worth mentioning that TPC annually charges $15000 for its full membership which is quite expensive. Therefore, It cannot meet extensibility and cost effectiveness as two requirements of a good benchmark already mentioned in [10].

So far, we have mostly focused on related work in the area of OLTP-related benchmarks. However, there exists a significant body of research related to performance evaluation and prediction for High-Performance Computing (HPC) applications on top of IaaS clouds, for instance [30]–[33]. These papers are related to our work, but not directly comparable, as the assumed application model is quite drastically different from the one that we have for *WPress*. Finally, we also need to mention the recent work on CloudBench [34], a generic framework for automating different types of benchmarks. *WPress* could conceptually be integrated into CloudBench as a separate benchmark module.

WPress is different from existing benchmarks, as it uses the existing, real-life WordPress blogging and publishing platform. We argue that this approach has several advantages over the simulated E-commerce applications used in TPC and RUBiS.

First, it is the first ranked blogging and publishing platform on the Internet [35]. Second, it is based on the latest technologies used for web applications, with its current version at the time of writing being released in January 2014. Third, it is open-source and has a remarkably user friendly interface, which meets benchmark's availability requirement discussed in [10]. Fourth, more than 29000 open-source plugins provide several transactional use-cases for WordPress and make it an extensible OLTP benchmark application suitable for cloud computing environments. Last but not the least, *WPress* measures total VM cost, which is a meaningful price metric to assess public cloud providers with pay-as-you-go cost models.

III. Methodology

In this section we introduce our proposed benchmark, called *WPress*, to study the performance of cloud providers when they host OLTP applications. First, we describe WordPress as the benchmark application and explain its advantages over E-commerce applications used in TPC and RUBiS. After that, we describe the functionality and implementation details of *WPress* client application, called *WPressClient*. This section ends with explanation of the experimental environment we developed to run *WPress* on Amazon EC2, Microsoft Azure and RackSpace cloud.

A. WordPress-Based Benchmark Application

WPress is designed based on WordPress blogging and publishing platform identified as the best blogging and publishing platform on the Internet [35]. WordPress can be used for several transactional use-cases such as E-commerce, on-line booking systems and blogging platforms. In *WPress*, we use blogging platform because: (1) we do believe that popularity of blogging systems is increasingly growing, and (2) WordPress has more than 25000 plugins and 10000 themes which make it the first solution for most blog owners [11]. There are a plethora of free, or at least cheap, plugins available for WordPress that can be incorporated with *WPress* in the future. For instance, WordPress provides a full fledged E-commerce website with integrated shopping card flow and PayPal payment gateway. Available plugins include *WP E-commerce* [1], *eShop* [2], *WooCommerce* [3] and *Quick Shop* [4]. Moreover, it can also be employed as an on-line booking system with *Rezgo Online Booking* [5] and *EzyOnlineBooking* [6] plugins. Other use-cases for WordPress include Job board or classified website, private social network, portfolio as well as news websites [36].

Extensibility, representativeness, availability and cost effectiveness are the main requirements for a good benchmark application [10]. The first two requirements are guaranteed because WordPress is implemented by the latest web technologies and various plugins can be easily associated with its main core. Therefore, it can be extended to a benchmark suite comprising several OLTP use-cases. In TPC and RUBiS, such flexibility does not exist. Availability and cost-effectiveness are also guaranteed since WordPress (1) is an open-source project with many plugins being available, and (2) has a user friendly interface which makes it easy to be used even by unskilled customers.

TABLE I. List of tasks generated by *WPressClient*

Task	Description	HTTP Requests
1	Searching a keyword	3
2	Publishing a post	5
3	Browsing pages	8
4	Replying a post	4
5	Loading a page	5
6	Uploading photos	5
7	Deleting posts	7
8	Drafting new post	5
9	Adding new user	6
10	Approving comments	6

B. Client Implementation

WPressClient is the integrated client application implemented for *WPress*. It is an open-source multi-threaded Java application developed using Selenium web-driver library [37]. Selenium web-driver uses Firefox plugin to steer an actual browser window, so that execution of each thread in the *WPressClient*, is as close to a real user who works with WordPress as possible. Essentially, *WPressClient* has two functionalities in this paper: (1) to generate representative workloads for WordPress and (2) to calculate average response times and total cost of VMs on the clouds under test.

Workloads are generated based on ten predefined WordPress common tasks named task 1 to task 10 (Table I). Each task is associated with realistic delay and scrolling actions, to mimic the real behavior of a human user of the system. Task 1, comprising three HTTP requests, simply searches for a random keyword. The first request opens WordPress landing page, which is then scrolled down for 1000 pixels. Afterwards, *WPressClient* waits for 6 seconds before scrolling up again. Within 12 seconds (which corresponds to the elapsed time for a real user to think for a keyword) a sample keyword (ten-characters long) is typed into the search box. In six seconds (estimated time for typing the keyword) a search request is sent. Resulting page is scrolled down for 500 pixels and returned back to its original place 6 seconds later. Next, one of the links in the home page is clicked. This task ends by scrolling down the newly loaded page for 500 pixels. For reasons of brevity, details of scrolling and delays are not mentioned in the description of task 2 to task 10.

Task 2 is implemented to publish random text-only posts. Post title and body have 10 and 1000 characters, respectively. Some users in WordPress just want to read other's posts. We apply task 3 to browse through the WordPress contents. In addition to browsing, unsuccessful login is also covered in this task to simulate a user who forgets his credential. Task 4 browses a specific post for which it leaves a comment. Replier name, email, website and the comment left, have 33, 18, 34 and 300 characters, respectively. In Task 5, a so called *core-task* is executed five times. Essentially, *core-task* sends a request to open the WordPress landing page and then closes the Firefox

[1] https://wordpress.org/plugins/wp-e-commerce/

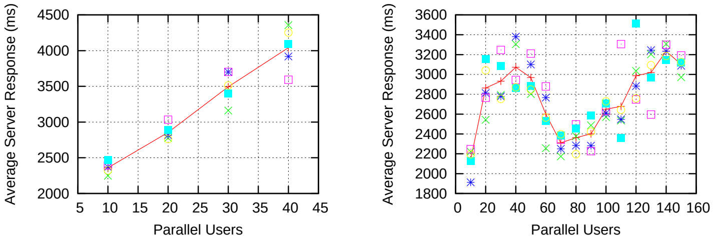
[2] https://wordpress.org/plugins/eshop/

[3] http://wordpress.org/plugins/woocommerce/

[4] https://wordpress.org/plugins/quick-shop/

[5] https://wordpress.org/plugins/rezgo-online-booking/

[6] https://wordpress.org/plugins/ezyonlinebookings-online-booking-system/installation/



(a) (b)

Fig. 1. Average response times for every parallel users in Microsoft Azure for Read Workload (a) Small instances, (b) Large instances

window after being successfully loaded. In Task 6, in addition to text, a single photo for each post is uploaded. Post title and text have 33 and 700 characters, respectively with a 640*480-pixel photo whose size is about 100KB. Moreover, each post is published on five different categories. In task 7, published posts are deleted from WordPress. By default, posts are sorted by publishing date and shown in multiple windows with 20 posts each. In this task, the first 20 published posts (which are actually the latest ones) are deleted. Further capabilities of this task are deleting posts published on a particular date, changing the number of posts to be deleted and deleting drafted posts. Another common task for all WordPress users is to create a new post and draft it for further editing. Task 8 is implemented to create and draft a new post whose title and text are 32 and 1800 characters, respectively. Then, the newly created post is saved into 5 categories. In task 9, a new user is added with user-name, e-mail address, first name, last name and password of 29, 59, 9, 10 and 9 characters, respectively. Moreover, possible roles, which are randomly assigned to each generated user, are: subscriber, administrator, editor, author and contributer. Although task 4 has left comments for a post, those comments remain in the pending state until they get approved. Task 10 is designed to approve the 20 latest pending comments. The implemented tasks are summarized in Table I. Each entry of the Table includes a short description of what each task does and the maximum number of HTTP requests sent to WordPress server.

WPressClient generates three workload types:

1) Read (called *R*), in which, executed tasks only read from database. For this type, workload is generated based on task 1, task 3 and task 5.
2) Write (named *W*), in which, database is updated. *W*-type workload is generated based on task 2, task 4, task 6, task 7, task 8, task 9 and task 10.
3) Read/Write (labeled *R/W*) in which all 10 tasks are executed.

For each type of workload, *WPressClient* selects tasks using round-robin technique from available task pool.

WPressClient is also responsible to measure the average response times and total cost of VMs on the clouds under test. In this paper, *WPressClient* is executed on small and large instance sizes of Amazon EC2, Microsoft Azure and Rackspace cloud (experimental environment will be discussed in more details in section III-C). *WPressClient* is configured to generate workload, based on available resources on the

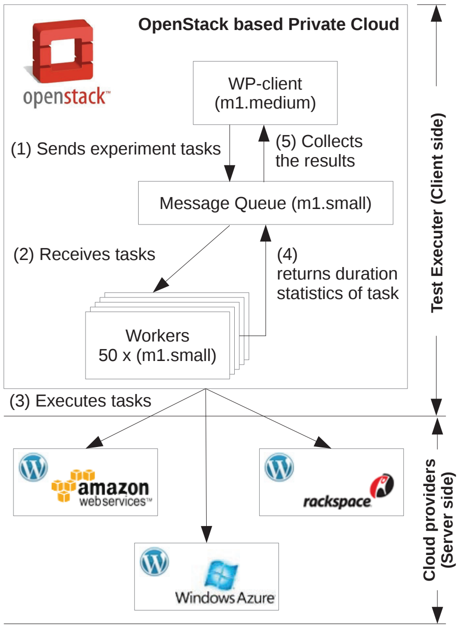


Fig. 2. Experimental environment

instance under test. The number of parallel users are obtained experimentally. Increasing number of parallel users leads to relatively large average response times particularly for *W* workloads in Amazon EC2. As we are not testing a scalable framework for this benchmark, we decided to go with 40 (for small size) and 150 (for large size) parallel users as the performance results up to these figures is sufficient to give us a good sense of the behavior of different cloud VMs. Each user is simulated by a single Java thread which runs a specific task (see Table I). For example, the first 10 tasks for R-type workload would be: task 1, task 3, task 5, task 1, task 3, task 5, task 1, task 3, task 5, task 1 which are executed through 10 parallel Java threads. Task execution is repeated 5 times for any number of parallel users and the average response time is calculated. Moreover, total execution time of the whole experiment, i.e. completion time of 40 parallel users for small instances or completion time of 150 parallel users for large instances, is recorded by *WPressClient* for computing the total cost of VMs. Figure 1 depicts average response times observed for Microsoft Azure with R-type workload.

The systematic structure of *WPressClient* along with the distributed infrastructure discussed in section III-C is an efficient and cost effective approach to make representative workloads for WordPress. Furthermore, it allows researchers and businesses to add new tasks or modify existing ones according to their requirements. *WPressClient* source code is available at [38].

C. Experimental Environment

The experimental environment used in this paper is illustrated in Figure 2. It includes two parts: (1) server side that involves small and large instances on public cloud providers, and (2) client side which is a distributed infrastructure to run the *WPressClient* application.

On the server side, three well-known IaaS public cloud

TABLE II. DETAILED SETUP OF INSTANCES DEPLOYED IN OUR EXPERIMENT

Name	Cores (ECUs)	RAM [GB]	Disk [GB]	Cost [$ / h]
Amazon EC2				
m1.small	1 (1)	1.7	160	$0.060
m1.xLarge	4 (8)	15	1680	$0.480
RackSpace Cloud				
1 GB Standard	1	1	40	$0.060
8 GB Standard	4	8	320	$0.480
Microsoft Azure				
(A1) Small	1	1.75	197	$0.060
(A4) Large	4	7.0	412	$0.240

providers in the current market have been selected as representative samples for our benchmarking approach: (1) Amazon EC2, is the de facto standard and industry leader for public IaaS cloud providers. Its data-centers are distributed all over the world and various types of services are provided to handle different computing and storage demands. (2) Microsoft Azure also delivers a wide range of computing and storage services. It is making a huge effort toward being a prominent cloud provider by offering cost-effective services. (3) RackSpace Cloud is one of the biggest cloud providers in the United States of America (USA) with an easy to use control panel and strong customer service [39].

Two instance types of each provider are studied in this paper (Table II). For the sake of simplicity, they are shown by small and large labels in this paper. Small implies m1.small in EC2, 1 GB standard in Rackspace, and (A1) Small in Azure. Large indicates m1.Xlarge in EC2, 8 GB standard in Rackspace, and (A3) Large in Azure. To achieve experiment fairness, corresponding instances are selected in a way to have comparable computing power, memory space, disk capacity and hourly cost. We use Ubuntu server (version 12.04 LTS 64-bit), WordPress (version 3.5.2) and complete LAMP platform (Linux Ubuntu 64-bit, version 12.04; Apache HTTP Server, version 2.2.14; MySQL database, version 5.1.70 and PHP, version 2.3.2) on all six instances. Web-server and database are located on the same instance in all our tests. Therefore, WordPress uses the local database associated with the LAMP server.

The client side of our benchmark is a distributed infrastructure, named *Test executer*, located in an OpenStack based private cloud. *Test executer* consists of 52 instances: (1) A m1.medium instance, *WP-Client*, which is the front-end used to trigger tests and collect the results. (2) A cluster of 50 m1.small instances, named *Workers*, responsible to execute *WPressClient* application in the private cloud. (3) A m1.small instance, referred as *MQ*, which runs an Apache ActiveMQ message queuing system. *MQ* enhances the extensibility of our experiment by decoupling *WP-Clients* from the *Workers* so that we can easily add or remove *Workers*. To increase number of *Workers*, we only need to start more instances which can hot-plug into the system running our experiment. Similarly, if one of the *Workers* crashes or turns off, it will neither get any further tasks nor break the experiment. Table III depicts detailed setup of each instance used in *Test executer*.

TABLE III. DETAILED SETUP OF INSTANCES DEPLOYED IN TEST EXECUTER

Instance Type	Processor Architecture	Virtual CPU	Memory (GB)	Disk Space (GB)
m1.small	AMD 64-bit	1	1	40
m1.medium	AMD 64-bit	2	3	40

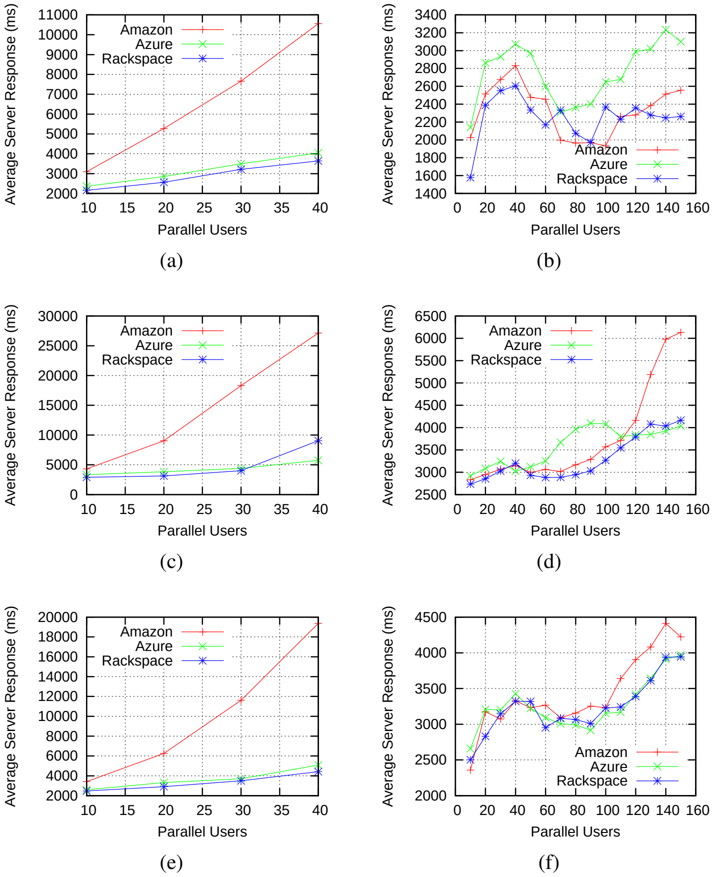
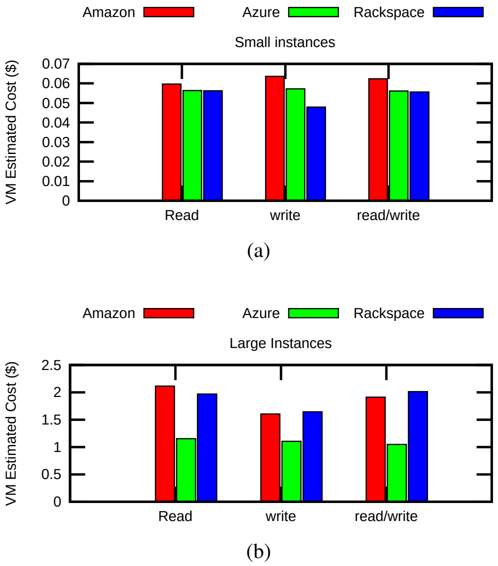


Fig. 3. (a) Average response times for small instances (Read Workload), (b) Average response times for large instances (Read Workload), (c) Average response times for small instances (Write Workload), (d) Average response times for large instances (Write Workload), (e) Average response times for small instances (Read/Write Workloads), (f) Average response times for large instances (Read/Write Workload)

As illustrated in Figure 2, *Test executer* essentially involves five steps. First, *WP-Client* generates tasks and sends them to *MQ*. Each task is actually an execution of a single task in our *WPressClient* application. Then, *Workers* receive tasks from *MQ* based on a first come first serve model. After that, *Workers* run each task five times against each WordPress instance on Amazon EC2, Rackspace Cloud and Microsoft Azure. Next, each active *Worker* returns duration statistics of its running task to *MQ* and finally, *WP-Client* collects the results from *MQ*.

IV. RESULTS AND DISCUSSION

In this section, we illustrate and discuss the results of applying *WPress* to Amazon EC2, Rackspace cloud, and Microsoft Azure. In order to investigate our results further, we reuse the CPU micro-benchmark introduced in [15] to demonstrate periods during which the VM is not allotted CPU time by the hypervisor. All the experimental results



(a)

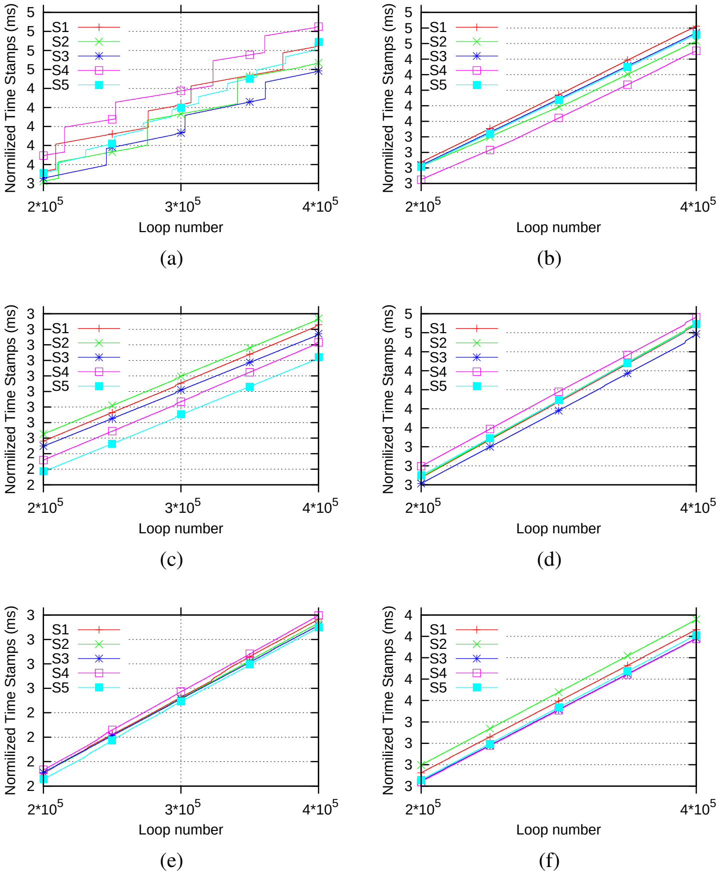
(b)

Fig. 4. Comparing total VM cost for Read, Write and Read/Write workloads on different cloud providers. (a) Small instances, (b) Large instances.

illustrated in this section were obtained from October 2013 to February 2014. In Figure 3, we see smooth curves of average response times in small instances for all workload types. However, several fluctuations are observed for large instances. We believe that undesirable effect of having no CPU affinity in *WPress* leads to inconsistency among large instances. Each VM in small instances has only one virtual core, hence the only scheduling overhead is at the hypervisor level. However, for large instances, in which each VM possesses 4 virtual cores, we have an extra overhead in the application level. Therefore, even for the first 40 parallel users in large instances (Figures 3b, 3d and 3f) we see that the curves are not as smooth as those of small instances (Figures 3a,3c and 3e).

Figure 4 illustrates total VM cost for each cloud provider based on instance and workload types. Among small instances (Figure 4a) the highest and lowest total VM cost are observed for Amazon EC2 and Rackspace, respectively, nevertheless, they have comparable results for large instances (Figure 4b). Moreover, the lowest total VM cost is noticed for Azure large instances (Figure 4b), which is due to the lowest hourly VM price for Azure large instances at the time of running this experiment (see Table II).

For further analysis, we employ the CPU micro-benchmark proposed in [15] to figure out any possible relationships between *WPress* results and CPU performance. The micro-benchmark runs on small and large instances of Amazon, Azure and Rackspace and continuously records CPU time for 1,000,000 times. We run the application five times on each instance and use labels *S1* to *S5* (*S* stands for *Sample*) to identify them. Moreover, the micro-benchmark is executed at different times of a day (8am, 4pm and 11pm) to investigate any performance variations on the clouds under test. The micro-benchmark is the only running application on the instances. Therefore, we expect to see recorded values in a linear trend with positive slope, provided that allocated CPU to the instance is not shared with any other active instances. Figure 5 depicts sample outputs of small and large instances



(a)

(b)

(c)

(d)

(e)

(f)

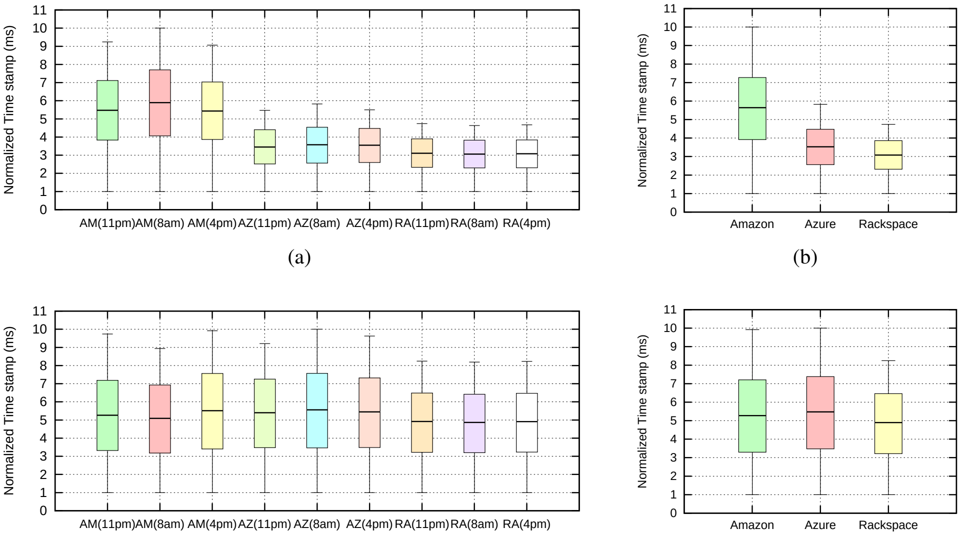
Fig. 5. CPU micro-benchmark results observed at 4pm : (a) Amazon EC2 (Small), (b) Amazon EC2 (Large), (c) Microsoft Azure (Small), (d) Microsoft Azure (Large), (e) Rackspace Cloud (Small), (f) Rackspace Cloud (Large).

on Amazon, Azure and Rackspace recorded at 4pm time-slot. Loop number is fixed between 200,000 to 400,000 to magnify small changes for readers. Recorded Time stamps are measured in millisecond and are rather long (up to 13 digits). Therefore, for the sake of simplicity values are normalized in the rage of 1 to 10.

In Figure 5a, Amazon EC2 significantly shares allotted CPU with other instances. Vertical lines in Amazon results demonstrate period of time that CPU is not running the micro-benchmark application. On the contrary, Amazon large instances (Figure 5b) do not show significant CPU sharing. Similarly, we do not observe considerable evidences of CPU sharing in small and large instances of Azure (Figures 5c and 5d) and Rackspace (Figures 5e and 5f).

We also depict the boxplot view of the observed micro-benchmark results. Figures 6a and 6c show variation of 5,000,000 numbers recorded on each provider per time slot, whereas, Figures 6b and 6d illustrate variation of 15,000,000 recorded values for each provider. Due to noticeable CPU sharing, Amazon small instances have considerably larger time stamps than those of Azure and Rackspace (Figures 6a and 6b). This accounts for drastically larger average response times of Amazon small instances (Figures 3a, 3c and 3e).

Moreover, we deploy pairwise t-test to investigate variation among three samples on each provider (Figures 6a and 6c)



(a) (b)

(c) (d)

Fig. 6. Performance variation of CPU on: (a) Small instances for different time-slots, (b) Small instances for three providers, (c) Large instances for different time-slots, (d) Large instances for three providers.
In (a) and (c) *AM*, *AZ* and *RA* represent Amazon EC2, Microsoft Azure and Rackspace Cloud, respectively.

TABLE IV. RECOMMENDED PROVIDERS FOR EACH WORKLOAD TYPE ON SMALL AND LARGE INSTANCES.

Workload Type	Small instance size		Large instance size	
	Average response time	Total VM cost	Average response time	Total VM cost
Read (*R*)	Rackspace	Rackspace	Rackspace	Microsoft Azure
Write (*W*)	Rackspace	Rackspace	Rackspace & Microsoft Azure	Microsoft Azure
Read/Write (*R/W*)	Rackspace	Rackspace	Rackspace & Microsoft Azure	Microsoft Azure

as well as variation among different providers (Figures 6b and 6d). Results imply that all samples are significantly different with 95% confidence interval, except *Rackspace(8am) vs Rackspace(4pm)* in small instances and *Rackspace(11pm) vs Rackspace(4pm)* in large instances. In addition to the impact of CPU affinity, significant performance variations of CPU can be another reason for observed fluctuations of average response times in large instances (Figures 3b, 3d and 3f). Obviously, CPU performance of large instances are better than those of small instances. In Figures 5 and 6 time stamps of small and large instances are normalized independently. Our findings are summarized as follow:

1) Amazon small instances show the largest average response times in our tests (Figure 3). This can be attributed to the impact of CPU sharing, which is predominantly observed on Amazon small instances. Therefore, for a predictive performance, Amazon small instances should be used with care.
2) We notice that among the three providers Rackspace has better average response times and total VM cost particularly for small instances (Figures 3a, 3c, 3e and 4a). Good performance of Rackspace in small instances has already been shown in our CPU micro-benchmark results (Figures 6a and 6b).
3) Despite lower total VM cost for *R*-type workload in Azure large instances (Figure 4b), their average response times are larger than those of Amazon and Rackspace (Figure 3b). For *R*-type workload, we observe better results in Rackspace, because of its smaller average response times and lower total VM cost than those of Amazon (Figures 3b and 4b).
4) For *W* and *R/W* workloads on large instances, Azure is a good choice. Although average response times of Azure are relatively larger than those of Rackspace (Figures 3d and 3f), we consider it as a favorable alternative for the majority of customers according to its lower total VM cost (Figure 4b).

Table IV depicts our recommended providers in terms of average response time and total VM cost based on the experimental results obtained from October 2013 to February 2014.

V. CONCLUSION

In this paper we propose *WPress* as a new open-source performance benchmark for OLTP applications on public clouds VMs. *WPress* is tested for small and large instance types of Amazon EC2, Microsoft Azure and Rackspace Cloud. Three different workload types (Read, Write and Read/Write) are considered. We also implement *WPressClient*, an open-source Client and load generator application. It generates representative workloads for WordPress and calculates the average response times and total cost of VMs for each provider based on the instance and workload types. Moreover, we employ the CPU micro-benchmark presented in [15] to analyze CPU performance on the clouds under test and identify its impact on *WPress* results.

Our experiments reveal better average response times and total VM cost in Rackspace small instances (for all workload types) and Rackspace large instances for Read workloads. Moreover, Azure large instances have better results for Write and Read/Write workloads. Furthermore, we notice that average response times have several fluctuations on large instances. We believe that undesirable effect of CPU affinity and significant performance variation of CPU are the main reasons for the observed fluctuations.

VI. Future Work

Designing a comprehensive OLTP benchmark, suitable for cloud environments with unknown hardware configurations and pay-as-you-go pricing model is a challenging task. In this paper we mainly addressed four fundamental challenges: 1) choosing representative, extensible and cost effective OLTP application, 2) generating representative workloads, 3) identifying cloud-specific performance metrics and 4) utilizing a distributed infrastructure to run *WPress* on different cloud providers. Possible improvements include enriching *WPress* with more transactional use-cases offered by WordPress, such as E-commerce and on-line reservation systems, and identifying more performance metrics compatible with cloud characteristics. Furthermore, executing *WPress* on clustered configurations with multiple web-servers and databases is considered as another valuable extension to our work.

Acknowledgment

This research was granted by Agency for Science, Technology and Research (A*Star) of Singapore, under A*STAR Thematic Programme: User and Domain Driven Data Analytics as a Service framework (SERC 1021580034) grant. Further, the second author of this paper has received funding from the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement 610802 (CloudWave).

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009.

[2] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, Oct 2009, pp. 17–25.

[3] K. Huppler, "Performance evaluation and benchmarking." Berlin, Heidelberg: Springer-Verlag, 2009, ch. The Art of Building a Good Benchmark, pp. 18–30.

[4] D. Kossmann, T. Kraska, and S. Loesing, "An evaluation of alternative architectures for transaction processing in the cloud," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10, 2010, pp. 579–590.

[5] D. Kossmann and T. Kraska, "Data management in the cloud: Promises, state-of-the-art, and open questions," *Datenbank-Spektrum*, vol. 10, no. 3, pp. 121–129, 2010.

[6] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early observations on the performance of Windows Azure," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10, 2010, pp. 367–376.

[7] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow?: Towards a benchmark for the cloud," in *Proceedings of the Second International Workshop on Testing Database Systems*, ser. DBTest '09, 2009, pp. 9:1–9:6.

[8] K. Huppler, "Benchmarking with your head in the cloud," in *Proceedings of the Third TPC Technology Conference on Topics in Performance Evaluation, Measurement and Characterization*, ser. TPCTC'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 97–110.

[9] A. Floratou, J. M. Patel, W. Lang, and A. Halverson, "When free is not really free: What does it cost to run a database workload in the cloud?" in *Proceedings of the Third TPC Technology Conference on Topics in Performance Evaluation, Measurement and Characterization*, ser. TPCTC'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 163–179.

[10] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun, "Benchmarking in the cloud: What it should, can, and cannot be," in *Selected Topics in Performance Evaluation and Benchmarking*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7755, pp. 173–188.

[11] WordPress - Blog Tool, Publishing Platform, and CMS. [Online]. Available: https://wordpress.org/

[12] AWS — Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. [Online]. Available: http://aws.amazon.com/ec2/

[13] Azure: Microsoft's Cloud Platform — Cloud Hosting — Cloud Services. [Online]. Available: http://azure.microsoft.com/en-us/

[14] Rackspace: The Leader in Hybrid Cloud. [Online]. Available: http://www.rackspace.com/

[15] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of Amazon EC2 data center," in *Proceedings of the 29th Conference on Information Communications*, ser. INFOCOM'10, 2010, pp. 1163–1171.

[16] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 460–471, Sep. 2010.

[17] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann, "What are you paying for? performance benchmarking for infrastructure-as-a-service offerings," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, July 2011, pp. 484–491.

[18] L. Zhao, A. Liu, and J. Keung, "Evaluating cloud platform architecture with the care framework," in *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*, Nov 2010, pp. 60–69.

[19] M. Klems, D. Bermbach, and R. Weinert, "A runtime quality measurement framework for cloud database service systems," in *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, Sept 2012, pp. 38–46.

[20] D. Bermbach and S. Tai, "Eventual consistency: How soon is eventual? an evaluation of Amazon S3's consistency behavior," in *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing*, ser. MW4SOC '11, 2011, pp. 1:1–1:6.

[21] D. Bermbach, L. Zhao, and S. Sakr, "Towards comprehensive measurement of consistency guarantees for cloud-hosted data storage services," in *Performance Characterization and Benchmarking*, ser. Lecture Notes in Computer Science, 2014, vol. 8391, pp. 32–47.

[22] C. A. Curino, D. E. Difallah, A. Pavlo, and P. Cudre-Mauroux, "Benchmarking OLTP/web databases in the cloud: The OLTP-bench framework," in *Proceedings of the Fourth International Workshop on Cloud Data Management*, ser. CloudDB '12, 2012, pp. 17–20.

[23] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10, 2010, pp. 143–154.

[24] TPC - Homepage. [Online]. Available: http://www.tpc.org/

[25] RUBiS - Home Page. [Online]. Available: http://rubis.ow2.org/

[26] T. Chen and R. Bahsoon, "Self-adaptive and sensitivity-aware QoS modeling for the cloud," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on*, May 2013, pp. 43–52.

[27] W. Dawoud, I. Takouna, and C. Meinel, "Dynamic scalability and contention prediction in public infrastructure using internet application profiling," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, Dec 2012, pp. 208–216.

[28] H. Wu, A. Tantawi, and T. Yu, "A self-optimizing workload management solution for cloud applications," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*, June 2013, pp. 483–490.

[29] L. Zhao, S. Sakr, and A. Liu, "A framework for consumer-centric SLA management of cloud-hosted databases," *IEEE Transactions on Services Computing*, vol. 99, no. PrePrints, p. 1, 2013.

[30] A. Iosup, S. Ostermann, M. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 931–945, June 2011.

[31] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, "Case study for running HPC applications in public clouds," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10, 2010, pp. 395–401.

[32] S. Akioka and Y. Muraoka, "HPC benchmarks on Amazon EC2," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, April 2010, pp. 1029–1034.

[33] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, and R. Biswas, "Performance evaluation of Amazon EC2 for NASA HPC applications," in *Proceedings of the 3rd Workshop on Scientific Cloud Computing Date*, ser. ScienceCloud '12, 2012, pp. 41–50.

[34] M. Silva, M. Hines, D. Gallo, Q. Liu, K. D. Ryu, and D. Da Silva, "CloudBench: Experiment automation for cloud environments," in *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, March 2013, pp. 302–311.

[35] The 15 Best Blogging Platforms on the Web Today. [Online]. Available: http://thenextweb.com/apps/2013/08/16/best-blogging-services/

[36] WordPress Use Cases. [Online]. Available: http://www.slideshare.net/teamphp/word-press-use-cases

[37] Selenium - Web Browser Automation. [Online]. Available: http://docs.seleniumhq.org/

[38] WPressClient Source Code. [Online]. Available: http://figshare.com/articles/WPressClient_source_code/978486

[39] IaaS Providers List: 2014 Comparison And Guide - Tom's IT Pro. [Online]. Available: http://www.tomsitpro.com/articles/iaas-providers,1-1560.html