The Performance Study of a Virtualized Multicore Web System

Chien-Te Lu[1], C.-S. Eugene Yeh[2*], Yung-Chung Wang[1] and Chu-Sing Yang[3]
[1] Dept. of Electrical Engineering, National Taipei University of Technology
[e-mail: ctlu@ntut.edu.tw, ycwang@ntut.edu.tw]
[2] Dept. of Information Management, Kainan University
[e-mail: eyeh@ieee.org]
[3] Dept. of Electrical Engineering, National Cheng Kung University
[e-mail: csyang@mail.ee.ncku.edu.tw]
*Corresponding author: C.-S. Eugene Yeh

Received October 12, 2015; revised March 1, 2016; revised April 19, 2016; accepted September 25, 2016; published November 30, 2016

Abstract

Enhancing the performance of computing systems has been an important topic since the invention of computers. The leading-edge technologies of multicore and virtualization dramatically influence the development of current IT systems. We study performance attributes of response time (RT), throughput, efficiency, and scalability of a virtualized Web system running on a multicore server. We build virtual machines (VMs) for a Web application, and use distributed stress tests to measure RTs and throughputs under varied combinations of virtual cores (VCs) and VM instances. Their gains, efficiencies and scalabilities are also computed and compared. Our experimental and analytic results indicate: 1) A system can perform and scale much better by adopting multiple single-VC VMs than by single multiple-VC VM. 2) The system capacity gain is proportional to the number of VM instances run, but not proportional to the number of VCs allocated in a VM. 3) A system with more VMs or VCs has higher physical CPU utilization, but lower vCPU utilization. 4) The maximum throughput gain is less than VM or VC gain. 5) Per-core computing efficiency does not correlate to the quality of VCs or VMs employed. The outcomes can provide valuable guidelines for selecting instance types provided by public Cloud providers and load balancing planning for Web systems.

Keywords: Performance attribute, virtualization, multicore, scalability, Web systems, Cloud computing

1. Introduction

Information technology (IT) has been advancing very fast during the last three decades. Since the invention of World Wide Web in 1989, IT has become a vital and significant part of all business operations as well as daily lives. It provides business operations benefits of convenience, efficiency, flexibility, accuracy, productivity, and innovation. How to provide reliable, stable, fast, secure, and easy-to-use IT service systems is a big challenge to IT engineers. When designing an IT service system, many factors have to be considered, such as cost, functionality, performance, reliability, energy consumption [1]. Performance is one critical factor. Performance metrics of computing systems include response time (RT), availability, latency, throughput (TP), efficiency, and scalability [2-4]. Performance enhancement has been an important and attractive subject for researchers and IT professionals as well. Possible methods to enhance performance are parallel computing [5], distributed and Cloud systems [4], load balancing (LB), and special purpose processors.

Cloud computing is an emerging technology that sparks both research and industrial interests [3]. Many Cloud systems use modern virtualization technology [6]. Classical virtualization techniques were invented in 1960's [7]. They are parts of operating systems (OSes) nowadays. The modern virtualization goes down to hardware abstraction layer [4]. Hypervisors are added between the hardware and application software with guest OSes. For modern virtualization, an application can be packaged with the needed guest OS and other service software into a virtual machine (VM) image. Each VM basically is an independent entity. A VM can request more than one virtual core (or vCPU) for computing. VMs with different guest OSes are able to run on the same hypervisor. The benefits from modern virtualization include server consolidation, managed execution and isolation, portability, efficient use of resources, easier to deploy application software, reducing the complexity of system management, flexibility, security, and consuming less energy [8, 9]. Some popular hypervisors are VMWare, Citrix Xen, Microsoft Hyper-V, and KVM.

The Web-based technology is still popular and used by many software applications such as websites, portals, Web mails, social networking, search engines, e-commence, and blogs. A Web application can be deployed on a public Cloud, private Cloud, or traditional Web server. The workload characterization and service performance of websites are major concerns to their providers. Most current microprocessors contain multiple cores in a single chip, which definitely boost computing power. By utilizing these new technologies, the performance of Web systems can be enhanced tremendously.

In this paper, we study how the performance and scalability of Web systems are affected by different configurations of VMs and vCPUs. Data are collected, analyzed, and compared to evaluate the performance metrics of RT and TP, their gain and efficiency, and scalability. The outcomes provide a rule of thumb for engineers for the selection of instance types provided by public Cloud providers, capacity planning, design of LB mechanisms, and task scheduling / distribution for Cloud and Web services.

The rest of the paper is organized as follows. Section 2 briefly covers the background, related works and the motivation of the paper. In section 3, the test environment is depicted, and notations to be used are defined. Section 4 presents the results of three experiments on three performance metrics: response time, throughput, and CPU utilization. Then the performance gains, efficiencies, and scalability are evaluated and compared in sections 5. The conclusion and possible future researches are described in section 6.

2. Background and Motivation

Many software developers use Web technology to design their applications. A popular Web application may attract millions of users with a huge amount of connections a day. In addition, it may encounter a sudden surge of requests in a short time, a burst phenomenon. Such an example is the online broadcast of a popular special event (such as an Olympic Game). Millions of people or more may link to the website to watch the event simultaneously. The system needs to be capable of handling normal large traffic and burst phenomenon. The Web application provider needs to know the workload characterization and performance attributes of the production system, so that a high-quality service can be provided.

The workload characterization studies user request (type and volume), traffic pattern, response size, session length, hit rate, resource demand, resource usage, interval between requests, site popularity, and others. Some researches use level approach [10]; some running Web applications on physical computers [11, 12], VMs [13], and Cloud systems [14]. Understanding the workload characterization helps engineers on capacity planning, resource management, and performance tuning [11, 13]. In addition to workload characterization, the study of performance attributes of a system can also help enhance the quality of the system.

The multicore technology increases the computing power. The study of how a multicore chip enhances the performance of a Web system is a main research topic currently. Veal and Foong studied the performance scalability of a multicore Web server on a physical computer [15]. The throughput of the system scaled up 4.8x from single core to eight cores. They stated that the primary bottleneck was the address bus capacity. Hashemian et al. studied the scalability of a Web system with two quad-core chips for two different types of workloads, i.e. static and dynamic requests of Web pages. The system scaled up differently for different types of workloads [16, 17]. Cui et al. performed scalability study for 2 OLTP applications using an 8-core chip [18]. The speedups were 3.68 and 5.26, respectively. They found out that the main bottleneck was database buffer pool and synchronization. Harji et al. studied the performance improvement for two different Web server architectures: event-driven and pipelined [19]. They declared that good implementation and fine-tuning could improve the performance of Web servers better than Web architecture.

How virtualized Web systems perform attracts many researchers' interests. The study by Park et al. identified that the mapping of physical cores to virtual cores (vCPU) was one major source of performance variation [20]. It could cause performance drop up to 22%. They also studied the similarities and dissimilarities among three hypervisors. NasiriGerdeh et al. analyzed the performance of Xen-based Web systems [21]. Their experiments showed the response time of the tested system was 2.5 times longer on Xen than on a native server, due to the overhead of schduling domains and I/O handling by Xen. Chen et al. studied the scalability behavior for multicore network and CPU-intensive applications [22]. They identified load imbalance with respect to per-core workload and proposed a queueing model to predict expected performances and mitigate observed bottlenecks.

Load balancing (LB) is also an approach to enhance the performance of a Web system. LB can be used in many different areas such as parallel computing, distributed systems, Web-based applications, and Cloud computing [3]. It is a common and essential mechanism for handling large volumes of requests for Cloud and Web-based services. LB uses a load balancer or an algorithm to distribute the whole workloads to a set of servers and keep each server with a reasonable amount of workloads. Hence the system can respond every request in a reasonable duration of time. LB can serve for specific purpose (i.e. for one single application) or general purpose (i.e. for different applications). It can be static or dynamic type [23, 24]. For static LB, once a job is dispatched to a server, it will stay on it till the job is completed. For dynamic LB, the job may be migrated to another server if the hosting server has a heavy workload or for other purposes. In addition, LB may work in a homogeneous (i.e. using the same type of servers) or heterogeneous environment (i.e. using different types of servers). Many LB algorithms have been developed for various scenarios [3, 23-28].

In a production data center, engineers typically adopt commercial, mature products to deploy a Web system either on a private Cloud or a public Cloud, instead of implementing an “experimental” load distribution or scheduling algorithm. When employing the above-mentioned cutting-edge technologies under such a circumstance, a systematic method to study the performance attributes of a deployed Web system helps them understand the characteristics of their system and workloads. The outcomes of performance research can provide a guideline for them to plan and arrange computing resources, properly invoke a VM instance when needed at the runtime, and efficiently utilize the available resources. Our research is to study how to achieve a good or acceptable performance for a Web system using a commercial hypervisor, multicore blade servers, and a load balancer. The performance metrics studied are RT and TP, their gain and efficiencies, scalability, and CPU utilization. Our research outcomes will provide IT professionals a good reference for planning and deploying their Web or Cloud systems.

3. Test Environment and Notations

Static load testing is used to collect test data and compute the statistics of the RT and TP of the system under test (SUT). **Fig. 1** shows the structure of the test environment. It consists of four functional parts: Test Generator, Load Balancer, Application, and Database. The test environment is homogeneous, i.e. all servers are identical, which are Cisco UCS blade B200 M3 with specification shown in **Table 1**. They all run Hyper-V version 3.0 in Windows Server 2012 as the virtualization platform. The guest OSes of all VMs are Windows Server 2008 R2.

The Test Generator resides on one physical server. It generates requests to feed into the SUT. Microsoft Visual Studio 2010 Load Test is used to perform the distributed stress test. It simulates multiple users visiting one same website simultanously. The Load Tester lets the user define the test flow and test parameters such as user size, test duration, workload model, and think time [11, 29]. It feeds the above test specification to the Test Controller, which manages Test Agents and collects the test data. A Test Agent simulates a group of users and, based on the test specification, generates and feeds test requests to the Load Balancer. Each block in the Tester Generator in **Fig. 1** is a VM with four virtual cores, 4,096 megabytes of memory, and one 10-Gbps network card.

The Load Balancer is a commercial product, i.e. A10 AX2500. It has the following specification: Intel Xeon chip with 2.27-GHz clock, 6 gigabytes of memory, one 8x1-Gbps cooper network interface card, 10 Gbps application throughput, 300,000/second (and 195,000/second) maximum layer-2 (and layer-7) connections. It provides static LB algorithms such as round-robin, weighted round-robin, least-connection, weighted least-connection. In our experiments, the least-connection is used. Other algorithms can be used too.

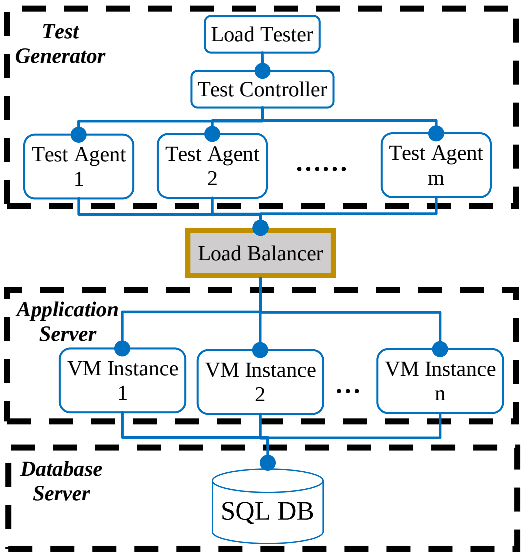

Fig. 1. The structure of the test environment.

Table 1. The specification of the test servers.

Feature	**Specification**
Brand	Cisco UCS blade servers
Processor type	Intel Xeon E5-2640
Clock frequency	2.5 GHz
Number of processor chips	2
Cores per chip	6
Total processor cores	12 = 6 * 2
Memory size	96 gigabytes
Network bandwidth	10 Gbps

In our test environment, there is only one single application as the SUT. A VM image is created to contain the application and the needed software with assigned virtual cores (vCPUs). A VM image can possess 1, 2, 3, 4, 6, or 12 vCPUs, 4096 megabytes of memory, and 10-Gbps network bandwidth with Windows Server 2008 R2 as the guest OS. Mulitple VM instances run simultaneously on one or several servers to serve user requests. The application is a Web-based course-election software which is being used by one of our schools to serve its students online. The application runs on a Cloud system. A student goes through several web pages to complete his/her selection of courses. To be closer to reality, the think time is added into test requests simulating possible users' activities (such as typing, thinking, waiting) during the process. If the VM instance of an application is unable to handle a request, an error message (e.g. "HTTP 500 Internal Server Error") will be returned to the requester. In that case, an error is recorded and the request has no response time.

The Database resides on its own physical server. The Database Management System (DBMS) is Microsoft SQL Server 2008 R2. A VM is created for the DBMS with 12 vCPUs, 40,960 megabytes of memory, and one 10-Gbps network card. The DBMS is capable of handling all database requests. So the DBMS is not a bottleneck in our test cases. Since it is a part of the system, its processing time is included in the total response time of a request.

Some notations and abbreviations are defined here for the convenience of reference.

- S(q, r) = a Web system running q VM instances with r virtual cores (vCPUs) in each VM instance. q is also referred as #VM, and r as #VC. #vCPU = the quantity of vCPUs. Total #vCPU in a system is denoted by Q_c . $Q_c = (q \times r)$ for S(q, r).
- RT = response time; TP = throughput, MaxTP = maximum TP. The performance, RT, and TP of S(q, r) are denoted by P(q, r), RT(q, r), and TP(q, r), respectively.
- PG(q, r) = the performance gain of S(q, r) with respect to S(1, 1). S(1, 1) is the reference system with simplest configuration of #VM=#VC=1. For RT, $PG_{RT}(q, r)$ = RT(1, 1) / RT(q, r). For TP, $PG_{TP}(q, r)$ = TP(q, r) / TP(1, 1).
- PE(q, r) = PG(q, r)/Q_c = the performance efficiency of S(q, r).
- TG = test group; E$x$TG$y$ = test group y in experiment x.

- CSes = concurrent sessions. A session (or user session) is a continuous time interval (and the activities within it) from login to logoff by a user requesting the service by the Web system. #CS = the number of current sessions.

4. Experiments and Results

Three experiments are performed with different combinations of #VM and #VC. For a test run, the Test Generator feeds the SUT with a constant #CS at any moment for a duration of 20 minutes. The same test run is performed three times to form a test case. Some sessions may fail. Let Wt and Ws be the total numbers of generated sessions (G sessions) and successfully processed sessions (S sessions) for a test case, respectively. The error rate=(Wt-Ws)/Wt. Due to limited space, diagrams of error rates are not presented in the paper.

When calculating RT, the think time is taken out. That is, the RT of a S session = (actual measured time – think time). The think time is not a key parameter influencing the characteristic of the RT of a system, because the RTs of test cases with think time and those without think time are strongly correlated [30]. When calculating the mean of RTs for a test case, only the RTs within ±3σ (standard deviation) from S sessions are included in the calculation. These sessions are referred to "valid sessions" (V sessions). The TP of a test case is defined as the average number of V sessions per minute per test run, i.e. TP = (total number of V sessions of the test case) / (3×20) sessions / minute. A session consists of 89 HTTP GET requests and 14 HTTP POST requests.

The configurations for the following three experiments are shown in **Tables 2 to 4** below, for ease of reference and comparison.

Table 2. One VM instance with variable vCPUs.

Test Group	Total #vCPU = (#VM) x (#VC)
1	1 = 1 x 1
2	2 = 1 x 2
3	3 = 1 x 3
4	4 = 1 x 4
5	6 = 1 x 6
6	12 = 1 x 12

Table 3. Variable VM instances with one vCPU per VM instance.

Test Group	Total #vCPU = (#VM) x (#VC)
1	1 = 1 x 1
2	2 = 2 x 1
3	3 = 3 x 1
4	4 = 4 x 1
5	6 = 6 x 1
6	12 = 12 x 1

Table 4. Variable combinations of VM instances and vCPUs.

Test Group	Total #vCPU = (#VM) x (#VC)
1	12 = 1 x 12
2	12 = 2 x 6
3	12 = 3 x 4
4	12 = 4 x 3
5	12 = 6 x 2
6	12 = 12 x 1

4.1 Experiment 1: Variable Virtual Cores

In experiment 1, we study the performance of S(1, r). There is only one VM instance, which have 1, 2, 3, 4, 6, or 12 vCPUs, as shown in **Table 2**. This is equivalent to simulating a physical computer with multiple processors by virtualization. **Fig. 2** shows RTs and TPs of the TGs in the experiment. #CS = 10, 50, 100, 200, …, 1500. The TPs begin to saturate around #CS=250, then start to degrade around #CS=900 (over capacity). It indicates the system capacity gain is limited, even for #VC=12. The system S(1, r) may have the similar bus issues mentioned in [15]. The MaxTP is 378 sessions / minute with TG6 at #CS=400. As expected, RT(1, r_1) < RT(1, r_2) and TP(1, r_1) > TP(1, r_2), where $r_1 > r_2$. RT(1, r_1) increases slower than RT(1, r_2) for $r_1 > r_2$, because a system with more vCPUs has larger processing capacity, and hence lowers the queue times of the requests.

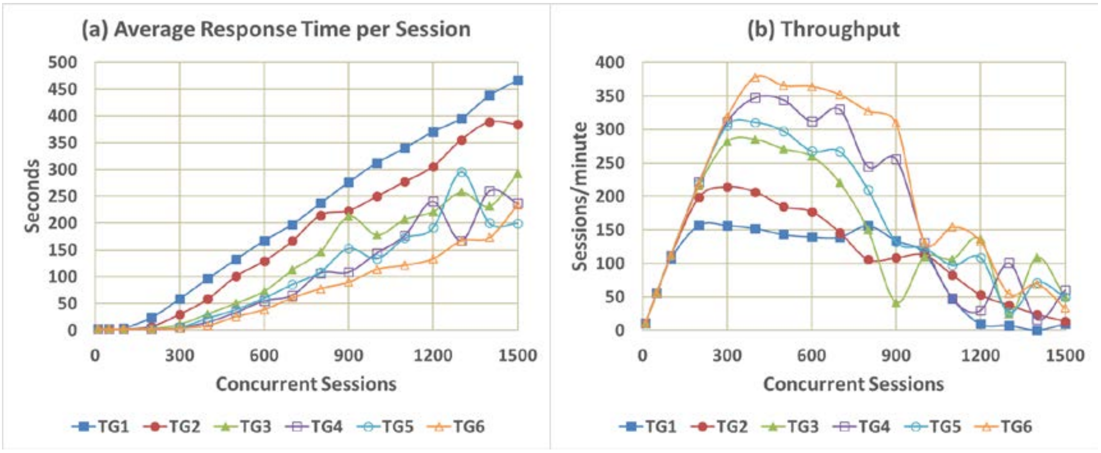

Fig. 2. The response times and throughputs of experiment 1.

4.2 Experiment 2: Variable Virtual Machine Instances

In this experiment, we study the performance of S(q, 1). The system runs 1, 2, 3, 4, 6, or 12 VM instances with #VC=1 for all TGs, as shown in **Table 3**. This is equivalent to simulating multiple physical computers with one processor each by virtualization. **Fig. 3** shows the RTs and TPs of the TGs in the experiment. #CS = 10, 50, 100, …, 3000. **Fig. 3(b)** shows that the TPs of TG1 and TG2 decrease to very low values after #CS=1100 and 2300, respectively, due to over capacity. The TPs start to saturate in the range of 250≤#CS≤1500. All RT curves seem to be linear. It is obvious that RT(q_1, 1) < RT(q_2, 1) and TP(q_1, 1) > TP(q_2, 1), where $q_1 > q_2$. The MaxTP is 1285 sessions / minute with TG6 at #CS = 1900. From both RT and TP curves, the system capacity gain is proportional to #VM gain. The system with #VM>2 is able to handle high workloads.

4.3 Experiment 3: Variable Combinations of VM Instances and Virtual Cores

In experiment 3, we study the performance of S(q, r) with $q \times r$ =12. The configuration is shown in **Table 4**. **Fig. 4** shows the RTs and TPs of the TGs in the experiment. #CS = 10, 50, 100, …, 3000. From RT and TP curves, all TGs behave similarly, but with different capacities. RT(q_1, r_1) < RT(q_2, r_2) and TP(q_1, r_1) > TP(q_2, r_2), where $q_1 \times r_1 = q_2 \times r_2$ and $q_1 > q_2$. It shows that the system performs much better with larger #VM than with larger #vCPU.

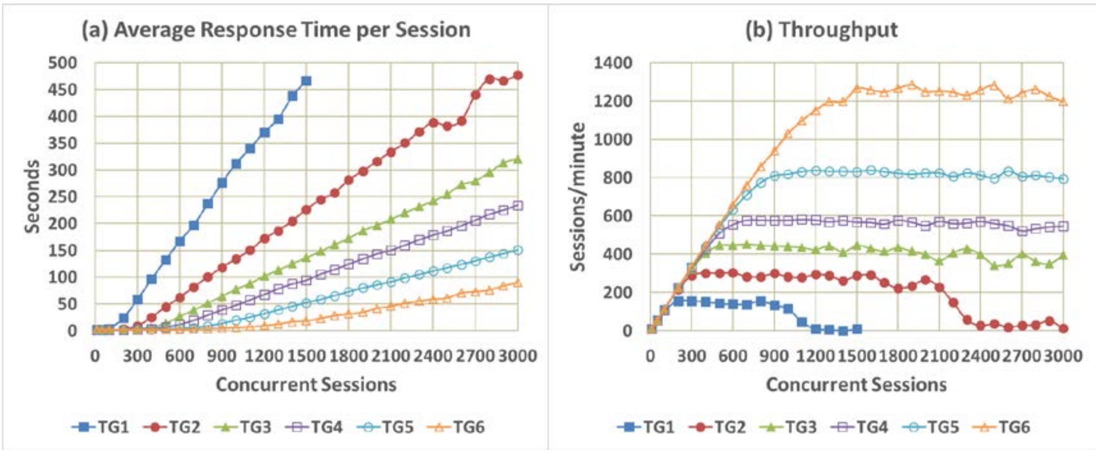

Fig. 3. The response times and throughputs of experiment 2.

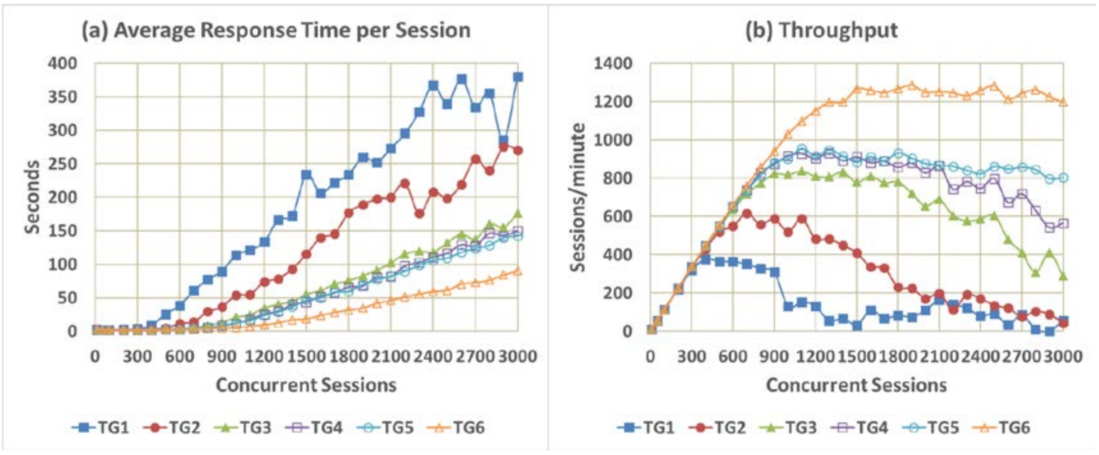

Fig. 4. The response times and throughputs of experiment 3.

4.4 The Comparison of CPU Utilizations

Fig. 5 shows the average physical CPU utilizations for experiments 1 and 2. The adopted server has 12 physical cores. It shows that the system with more vCPUs (or VMs) has higher physical CPU utilization. The maximum average utilizations for E1TG6 and E2TG6 are 75.57% and 96%, respectively, for both $Q_c = 12$. For E1TG1 and E2TG1 (i.e. #vCPU=1), the average physical CPU utilizations only reach 20%. It indicates that VM method (experiment 2) can reach higher maximum CPU utilization than VC method (experiment 1).

Fig. 6 shows average vCPU utilizations for experiments 1 and 2. The system with more vCPUs (or VMs) has lower vCPU utilization. This is opposite to the physical CPU utilization. It implies a VM does not need so many vCPUs. Several test groups almost reach to 100% of vCPU utilization for large #CS. **Fig. 6** also indicates that VM method (experiment 2) can reach higher maximum vCPU utilization than VC method (experiment 1) for large #CS.

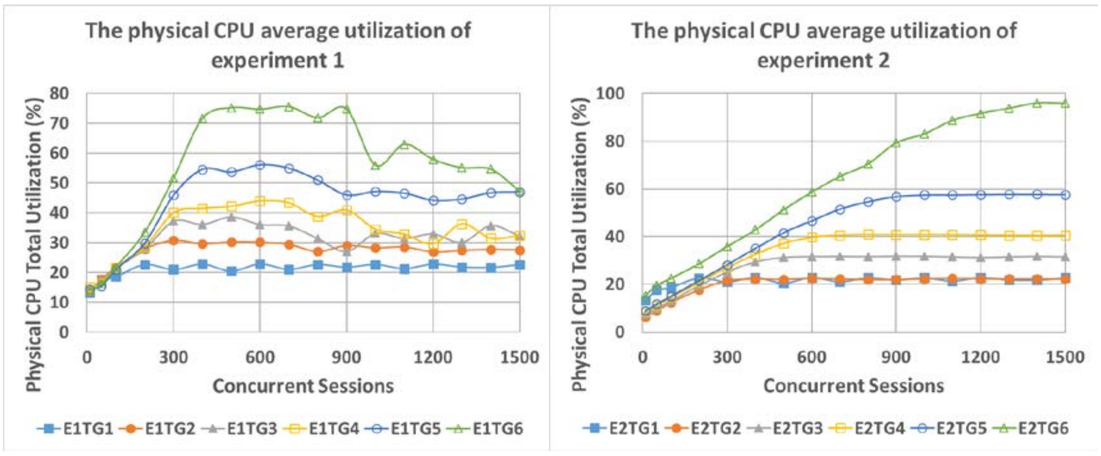

Fig. 5. The average physical CPU utilizations of experiments 1 and 2.

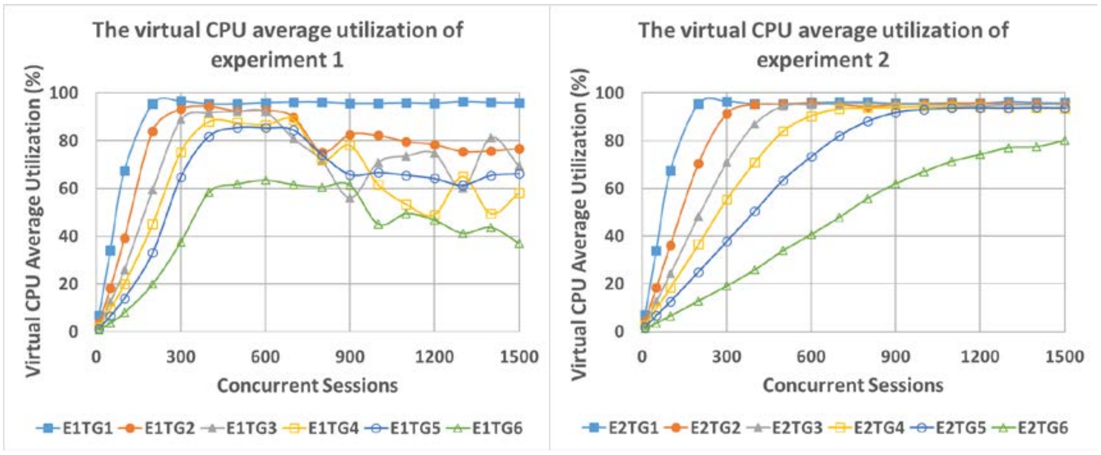

Fig. 6. The average vCPU utilizations of experiments 1 and 2.

The experimental results from this section are summarized in the following.

1. RT(1, r_1) < RT(1, r_2) for r_1>r_2. RT(q_1, 1) < RT(q_2, 1) for q_1>q_2. TP(1, r_1) > TP(1, r_2) for r_1>r_2 before systems degrade. TP(q_1, 1) > TP(q_2, 1) for q_1>q_2. More virtual cores or VM instances perform better.
2. RT(q_1, r_1) < RT(q_2, r_2), TP(q_1, r_1) > TP(q_2, r_2), and the higher the ratio q_1/q_2, the higher the ratio RT(q_2, r_2)/RT(q_1, r_1), for $q_1 \times r_1 = q_2 \times r_2$ and q_1>q_2. A system performs better by adopting multiple single-core VM instances than by adopting single multiple–core VM.
3. The system capacity of S(q_1, r_1) is larger than that of S(q_2, r_2), where $q_1 \times r_1 = q_2 \times r_2$ and q_1>q_2. The system capacity gain is proportional to the number of VM instances adopted, but not to the number of virtual cores assigned in a VM.
4. A system with more #VM or #VC has higher physical CPU utilizations and lower vCPU utilization.

Similar small-scale experiments are also performed with VMWare vSphere 5.1. Our preliminary results show that the performance curves of the SUT on VMWare and on Hyper-V behave similarly. That indicates different virtualization platforms may cause a minor performance difference for a Web system running on them. In addition, all VMs in our experiments are given 4 gigabytes of memory. Memory usage data are collected and studied. In all test cases, the SUT consumes less than 4 gigabytes of memory. So, in our experiments, memory does not affect our performance study. Due to the limited space, we are unable to include those results and diagrams in the paper.

5. Performance Gain, Efficiency and Scalability

In this section, we will look into the performance gains, efficiencies, and scalability from the data we collect in section 4. We will do comparison across experiments. E1TG1 or S(1, 1) is used as the reference for comparison. An individual gain is calculated from a pair of RT or TP values. An efficiency is the corresponding gain divided by Q_c. Note that E1TG1 = E2TG1, E1TG6 = E3TG1, and E2TG6 = E3TG6 (both test groups are S(12, 1)).

5.1 Response Time Gains and Efficiencies of Experiment 1

Fig. 7 shows the RT gains and efficiencies of experiment 1 with respect to (w.r.t.) E1TG1. Only in a small range of #CS, RT gains and efficiencies are reasonably high. Generally, $PG_{RT}(1, r_1) > PG_{RT}(1, r_2)$ for $r_1 > r_2$, but not the case for PE. PE < 1 for small or large #CS. The cause of the bottleneck mentioned in [15] may apply to this case. All PG and PE curves are bell-shaped with long small-valued asymptotic tails. It indicates that given extra vCPUs does not enhance the performance much for large amount of user sessions.

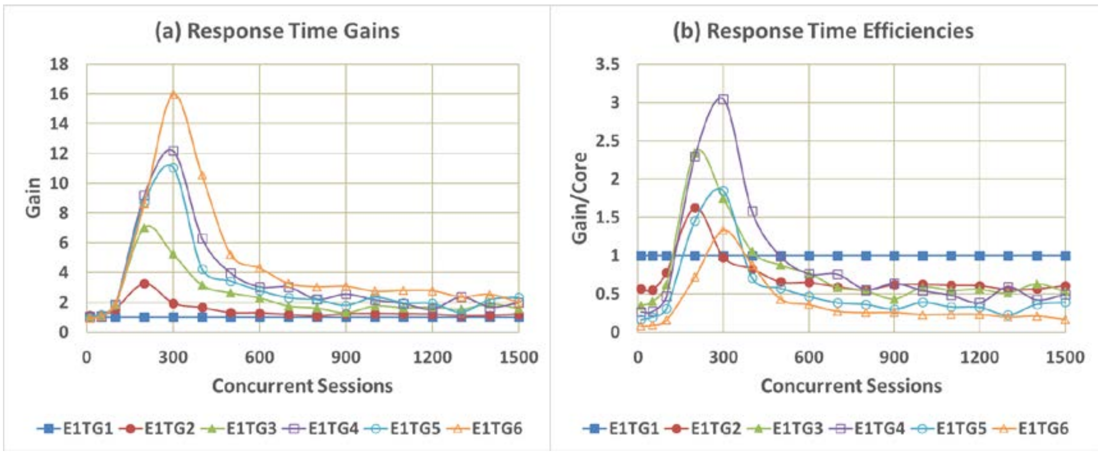

Fig. 7. The response time gains and efficiencies of experiment 1.

5.2 Response Time Gains and Efficiencies of Experiment 2

Fig. 8 shows the RT gains and efficiencies of experiment 2 w.r.t. E1TG1. All curves have similar shapes. For small #CS, E2TG5 has higher PE, but for large #CS, E2TG6 has higher PE. $PG_{RT}(q_1, 1) > PG_{RT}(q_2, 1)$, where $q_1 > q_2$. All PG and PE curves are bell-shaped too, but last longer at the top and degrade slower, compared with those in **Fig. 7**.

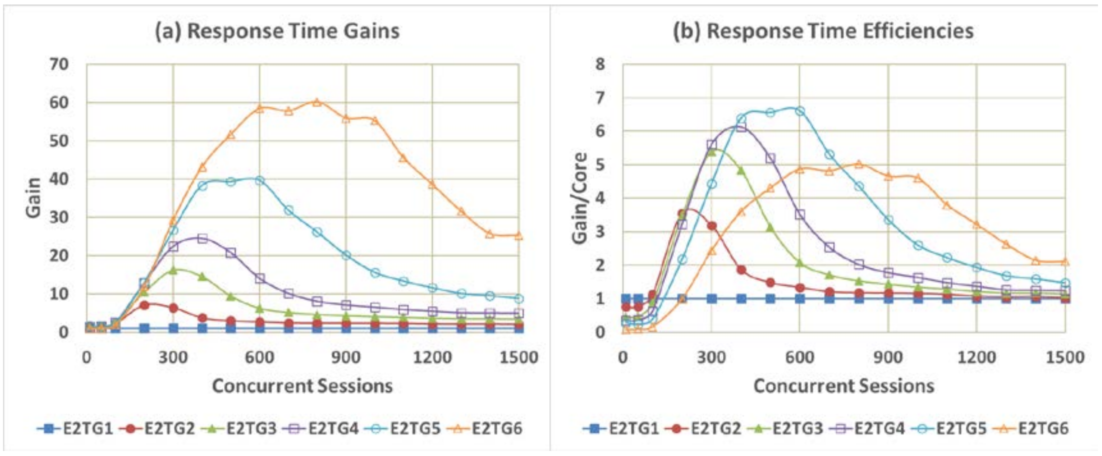

Fig. 8. The response time gains and efficiencies of experiment 2.

5.3 Response Time Gains and Efficiencies of Experiment 3

All TGs in experiment 3 have Q_c=12. So we calculate RT gains and efficiencies w.r.t. E3TG1 first, then w.r.t. E1TG1, so that comparison can be studied from different viewpoints. We also compare $PG_{RT}(q_1, r_1)$ and $PG_{RT}(q_2, r_2)$, where $q_1 \times r_1 = q_2 \times r_2$, see **Fig. 11** and **Fig. 12**.

Fig. 9 and **Fig. 10** show the RT gains and efficiencies of experiment 3 w.r.t. E3TG1 and w.r.t. E1TG1, respectively. It shows $PG_{RT}(q_1, r_1) > PG_{RT}(q_2, r_2)$, where $q_1 \times r_1 = q_2 \times r_2$ and $q_1 > q_2$. That is, the RT gain is larger by adopting more VMs than by more vCPUs. The #VM of a system affects its RT gain. RT efficiency is not proportional to #VM. In **Fig. 9**, asymptotic curves of PG and PE are close.

Fig. 11 shows the RTs of the systems with the same Q_c. Four pairs of test groups presented in **Fig. 11** are (E1TG2, E2TG2), (E1TG4, E2TG4), (E1TG5, E2TG5), and (E1TG6, E2TG6) with Q_c=2, 4, 6, and 12, respectively. Again it confirms that a system performs better by adopting more VMs than by adopting more vCPUs. **Fig. 12** shows the relative RT gains of these four pairs of systems, each pair having the same Q_c.

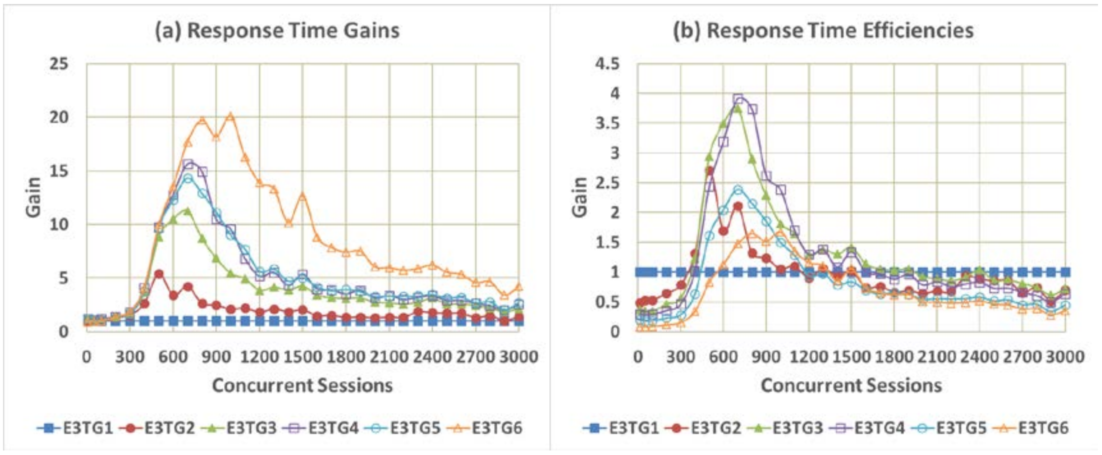

Fig. 9. The response time gains and efficiencies of experiment 3 with respect to E3TG1

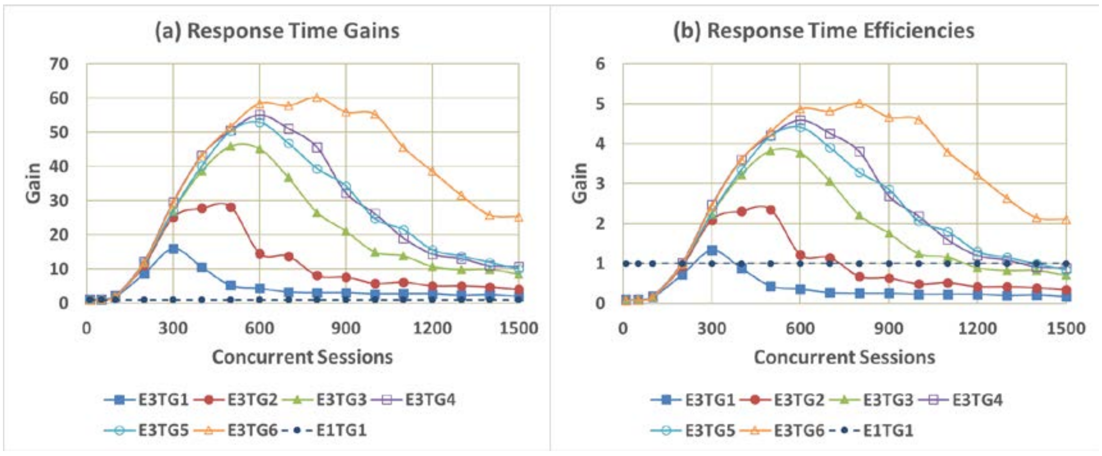

Fig. 10. The response time gains and efficiencies of experiment 3 with respect to E1TG1.

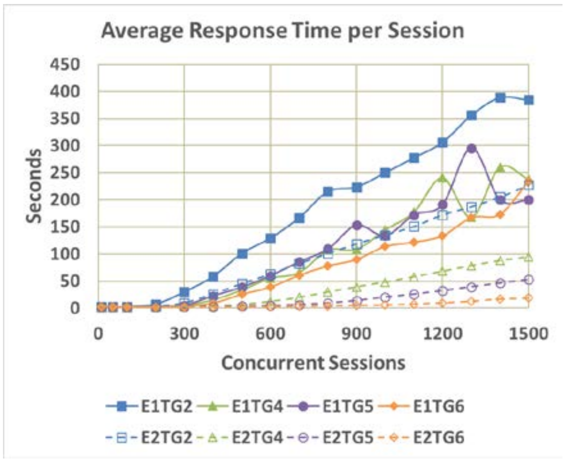

Fig. 11. The response times of 4 pairs of systems with the same #vCPU.

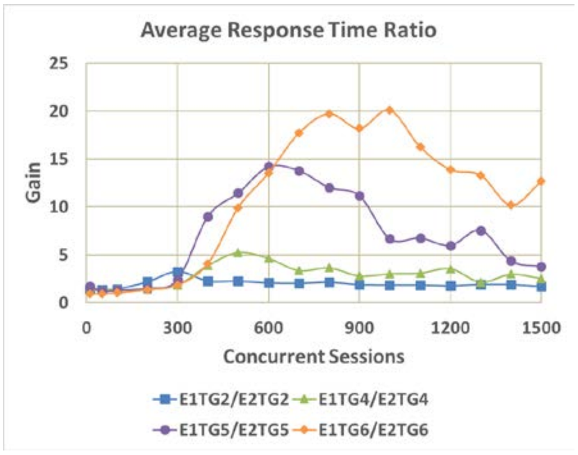

Fig. 12. The RT ratios of four pairs of systems, each pair having the same total number of vCPUs.

5.4 Peak Response Time Gains and Efficiencies

Table 5 and **Fig. 13** show the peak RT gains and efficiencies of all test groups. The peak RT gains are obtained from **Figs. 7(a)**, **8(a)**, and **10(a)**, and listed at column 4 of **Table 5**. All peak RT gains are compared w.r.t. E1TG1. For a particular row, the value of column 5 = (the value of column 4) / (the value of column 3). The peak RT gain of S(12, 1) is roughly 3.7 times larger than that of S(1, 12). The response time improvement of a system by adding more vCPUs is limited. For peak RT efficiency, VM method has higher values than VC method.

Table 5. The peak response time gains and efficiencies of all test groups.

VM Instances	#vCPU / VM	Total #vCPU	Peak RT Gain	Peak RT Efficiency
1	1	1	1.00	1.00
1	2	2	3.26	1.63
1	3	3	7.03	2.34
1	4	4	12.17	3.04
1	6	6	11.07	1.85
1	12	12	15.96	1.33
1	1	1	1.00	1.00
2	1	2	6.37	3.54
3	1	3	16.13	5.38
4	1	4	24.51	6.13
6	1	6	39.66	6.61
12	1	12	60.18	5.02
1	12	12	15.96	1.33
2	6	12	28.17	2.35
3	4	12	45.92	3.83
4	3	12	55.10	4.59
6	2	12	52.97	4.41
12	1	12	60.18	5.02

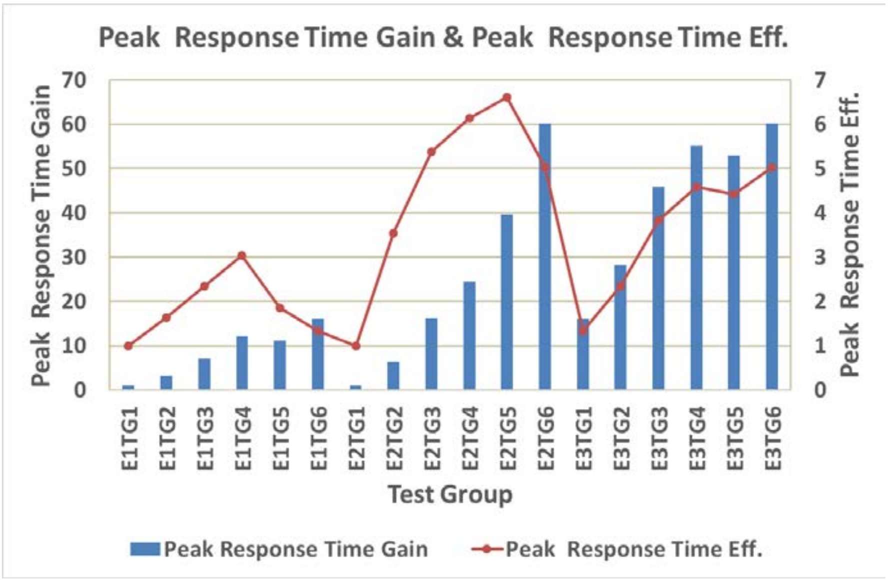

Fig. 13. The peak response time gains and efficiencies of all test groups.

5.5 Maximum Throughputs, Gains and Efficiencies

Table 6 and **Fig. 14** show the MaxTPs and their corresponding gains and efficiencies of all TGs. From **Fig. 2(b)**, **3(b)**, and **4(b)**, the MaxTPs are listed at column 4 of **Table 6**. The value of column 5 at row r is equal to (the value of column 4 at row r) / (the value of column 4 at row 1). For a particular row, the value of column 6 = (the value of column 5) / (the value of column 3). The MaxTP relates to the capacity of the system; the high, the better. Obviously, a system with more #VM or #VC can have larger MaxTP and gain. In terms of efficiency, the result is opposite. Also the MaxTP and efficiency of S(q_1, r_1) are larger than those of S(q_2, r_2) where $q_1 \times r_1 = q_2 \times r_2$ and $q_1 > q_2$.

Table 6. The maximum throughputs, gains and efficiencies of all test groups.

VM Instances	#vCPU / VM	Total #vCPU	MaxTP	MaxTP Gain	MaxTP Efficiency
1	1	1	157	1.00	1.00
1	2	2	214	1.36	0.68
1	3	3	285	1.82	0.61
1	4	4	347	2.21	0.55
1	6	6	310	1.97	0.33
1	12	12	378	2.41	0.20
1	1	1	157	1.00	1.00
2	1	2	303	1.93	0.96
3	1	3	453	2.89	0.96
4	1	4	582	3.71	0.93
6	1	6	838	5.34	0.89
12	1	12	1285	8.18	0.68
1	12	12	378	2.41	0.20
2	6	12	617	3.93	0.33
3	4	12	836	5.32	0.44
4	3	12	932	5.94	0.49
6	2	12	955	6.08	0.51
12	1	12	1285	8.18	0.68

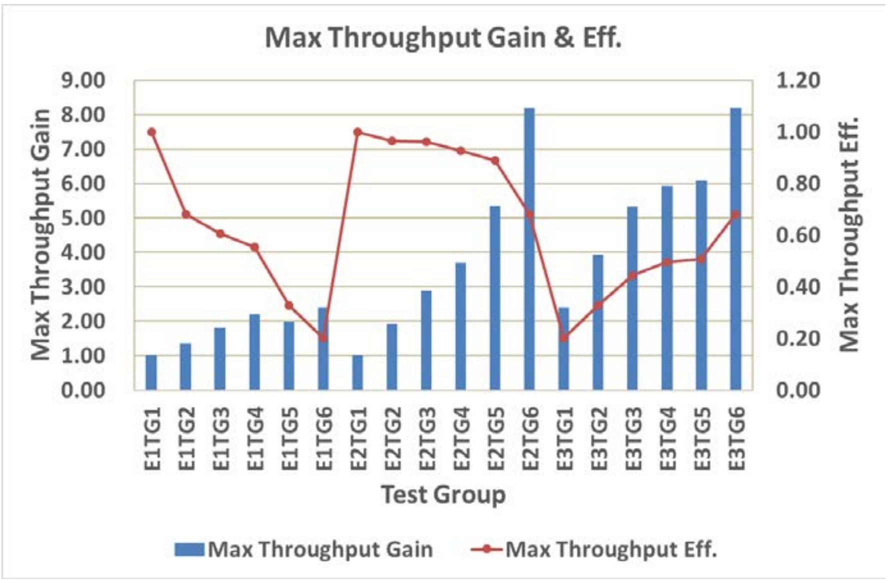

Fig. 14. The maximum throughput gains and efficiencies of all test groups.

5.6 Scalability

Fig. 15 shows the scalabilities of a system at VM and virtual core (VC) levels w.r.t. RT and TP. At VC level, the scalability starts to saturate at #vCPU=4. That means, a system scales up only to four vCPUs. This result is similar to the one in [15]. The scalability curves at VM level are close to linear. It indicates that the scalability of the system is much higher at VM level than at VC level.

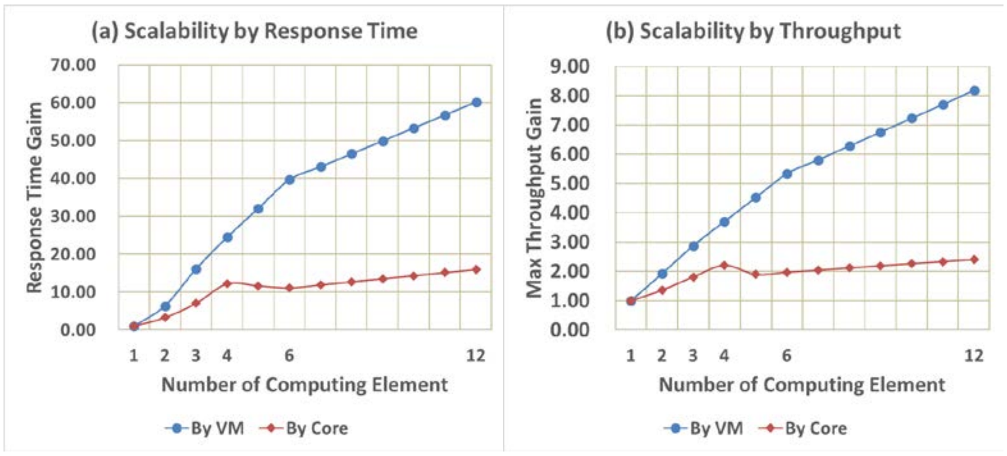

Fig. 15. The scalabilities at VM and virtual core levels

The analytic results in this section is the following.

1. For response time gain: PG(1, r_1) > PG(1, r_2) for r_1>r_2. PG(q_1, 1) > PG(q_2, 1) for q_1>q_2. In general, the larger #VM or #VC, the higher the response time gain. PG(q_1, r_1) > PG(q_2, r_2) for $q_1 \times r_1 = q_2 \times r_2$ and q_1>q_2.
2. The response time efficiency does not correlate to #VM or #VC. But PE(q_1, r_1) > PE(q_2, r_2) for $q_1 \times r_1 = q_2 \times r_2$ and q_1>q_2.
3. For peak response time gain: Peak PG(m, 1) > peak PG(1, m). Peak PG(q_1, r_1) > peak PG(q_2, r_2) for $q_1 \times r_1 = q_2 \times r_2$ and q_1>q_2. The peak response time efficiency has the same results as peak response time gain.
4. For maximum throughput: MaxTP(q_1, 1) > MaxTP(q_2, 1) for q_1>q_2. MaxTP (q_1, r_1) > MaxTP (q_2, r_2) for $q_1 \times r_1 = q_2 \times r_2$ and q_1>q_2. The maximum throughput gain has the same results as maximum throughput, and is less than #VM or #VC gain.
5. The maximum throughput efficiency decreases as #VM or #VC increases. But MaxTP efficiency of S(q_1, r_1) > MaxTP efficiency of S(q_2, r_2) for $q_1 \times r_1 = q_2 \times r_2$ and q_1>q_2.
6. The scalability of the system is much higher at VM level than at virtual core level. The latter saturates at low #vCPU.

6. Conclusions and Future Development

Modern virtualization and multicore technologies emerge around one and a-half decade ago. They remarkably change the approaches of designing and deploying computing and IT systems. To study how these technologies affect the performance of Cloud/Web systems, experiments are designed to cover test cases with variable number of virtual cores (VCs) and VMs. The experimental data of performance attributes provide valuable information of the capacity, performance, and scalability of the system. We show that the way of configuring VMs makes the system perform differently. Running n single-core VM instacnes performs much better than running single VM instance with n VCs. But the former may utilize more system resources (e.g. memory).

Our experimental approach allows engineers of a production data center to study the performance attributes of their Web system running either on a private Cloud or public Cloud. The experimental outcomes provide valuable guidelines to engineers for capacity planning, design of load balancing mechanisms, and task scheduling /distribution for Cloud and Web services. Public Cloud providers provide different instance types for users. An engineer can build a single-core VM with sufficient memory or select a suitable single-core VM instance type from a public Cloud for her application. Then based on the experimental performance curves, she can find out the number of VM instances she needs for the response time and throughput requirements of the application. The number of VM instances can be adjusted dynamically according to the actual workloads.

In the future, similar studies can be performed on public and hybrid Clouds. The same approach may be used to study how different load balancing algorithms affect the system performance, capacity, and scalability. Researches could be done on performance under general purpose heterogeneous or dynamic environments. Also one could study how other related elements (such as memory, disk I/O, network bandwidth) affect the system performance. Eventually methods or rules of thumb may be developed to design optimal configuration of systems to satisfy user requirements and specific workload characterization.

References

[1] H. Ghiasi and M.G. Arani, "Smart Virtual Machine Placement Using Learning Automata to Reduce Power Consumption in Cloud Data Centers," *Smart Computing Review*, Vol. 5, No. 6, pp. 553-562, 2015. Article (CrossRef Link)

[2] Wikipedia, "Computer performance," http://en.wikipedia.org/wiki/Computer_performance.

[3] N.J. Kansal and I. Chana, "Cloud Load Balancing Techniques: A Step Towards Green Computing," *IJCSI International Journal of Computer Science Issues*, Vol. 9, No. 1, pp. 238- 246, Jan. 2012.

[4] K. Hwang, G. Fox, and J. Dongarra, "Distributed and Cloud Computing: From Parallel Processing to Internet of Things," *Morgan Kaufman*, pp. 34-36, 130-133 and 569-572, 2012.

[5] G. Hager and G. Wellein, "Introduction to High Performance Computing for Scientists and Engineers," *CRC Press*, pp.95-104 and 115-120, 2011.

[6] K.L. Kroeker, "The Evolution of Virtualization," *Communications of the ACM*, Vol. 52, No. 3, pp. 18-20, March 2009. Article (CrossRef Link)

[7] B. Hayes, "Cloud Computing," *Communications of the ACM*, Vol. 51, No. 7, pp. 9-11, July 2008. Article (CrossRef Link)

[8] N.E. Jerger, D. Vantrease, and M. Lipasti, "An Evaluation of Server Consolidation Workloads for Multi-Core Designs," in *Proc. of IEEE 10th International Symposium on Workload Characterization (IISWC)*, Sep. 27-29, pp. 47-56, 2007. Article (CrossRef Link)

[9] R. Buyya, C. Vecchiola, and S. T. Selvi, *Mastering Cloud Computing*, Morgan Kaufmann, p. 93, 2013.

[10] D. Menasce, "Workload Characterization," *IEEE Internet Computing*, pp. 89-92, Sep./Oct. 2003. Article (CrossRef Link)

[11] J. Dilley, "Web Server Workload Characterization," *HP Technical Report*, HPL-96-160, Hewlett-Packard Laboratories, 1996.

[12] J.A. Lacort, A. Pont, J.A Gil, and J.Shuquillo, "A Comprehensive Web Workload Characterization," in *Proc. of the Conference on Performance Modeling and Evaluation on the Heterogeneous Network*, pp. 67/1-67/10, July 2004.

[13] X. Wang, S. Huang, S. Fu, and K. Kavii, "Characterizing Workload of Web Applications on Virtualized Servers," *Big Data Benchmarks, Performance Optimization, and Emerging Hardware. Springer International Publishing*, pp. 98-108, 2014. Article (CrossRef Link)

[14] A.K. Misha, J.L. Hellerstein, W. Cirne, and C.R. Das, "Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 37, No. 4, pp. 34-41, March 2010. Article (CrossRef Link)

[15] B. Veal and A. Foong, "Performance scalability of a Multi-core Web Server," in *Proc. of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, ACM, pp. 57-66, 2007. Article (CrossRef Link)

[16] R. Hashemian, D. Krishnamurthy, M. Arlitt, and N. Carlsson, "Characterizing the Scalability of a Web Application on a Multi–core Server," *Concurrency and Computation: Practice and Experience*, 2014. Article (CrossRef Link)

[17] R. Hashemian, D. Krishnamurthy, M. Arlitt, and N. Carlsson, "Improving the Scalability of a Multi-core Web Server," in *Proc. of the ACM/SPEC International Conference on Performance Engineering*, ACM, pp. 161-172, 2013. Article (CrossRef Link)

[18] Y. Cui, Y. Chen, and Y. Shi, "Scaling OLTP Applications on Commodity Multi-core Platforms," *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, pp. 134-143, 2010. Article (CrossRef Link)

[19] A.S. Harji, P.A. Buhr, and T. Brecht, "Comparing High-performance Multi-core Web-server Architectures," in *Proc. of the 5th Annual International Systems and Storage Conference*, ACM, p. 1, 2012. Article (CrossRef Link)

[20] J. Park, Q. Wang, D. Jayasinghe, J. Li, Y. Kanemasa, M. Matsubara, D. Yokoyama, M. Kitsuregawa and C. Pu, "Variations in Performance Measurements of Multi-Core Processors: A Study of n-Tier Applications," in *Proc. of IEEE International Conference on Services Computing (SCC2013)*, IEEE, pp. 336-343, 2013. Article (CrossRef Link)

[21] R. NasiriGerdeh, N. Hosseini, K. RahimiZadeh, and M. AnaLoui, "Performance Analysis of Web Application in Xen-based Virtualized Environment," in *Proc. of 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, IEEE, pp. 256-261, 2015. Article (CrossRef Link)

[22] X. Chen, C.P. Ho, R. Osman, P. Harrison, and W.J. Knottenbelt, "Understanding, Modelling, and Improving the Performance of Web Applications in Multicore Virtualized Environments," in *Proc. of the 5th ACM/SPEC International Conference on Performance Engineering*, ACM, 2014. Article (CrossRef Link)

[23] H.Y. Sit, K.S. Ho, R.W. Luk, and L.K. Ho, "An Adaptive Clustering Approach to Dynamic Load Balancing," in *Proc. of the 7th International Symposium on Parallel Architecture, Algorithms and Networks (ISPAN'04)*, pp. 415-420, 2004. Article (CrossRef Link)

[24] S. Banawan and N.M. Zeidat, "A Comparative Study of Load Sharing in Heterogeneous Multicomputer System," in *Proc. of the 25th Annual Simulation Symposium*, pp. 22-31, April 1992. Article (CrossRef Link)

[25] T.D. Braun, H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Roberts, M.D. Theys, B. Yao, D. Hensgen, and R.F. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," *Journal of Parallel and Distributed Computing* , Vol. 61, No. 6, pp. 810-837, June 2001. Article (CrossRef Link)

[26] S.-C. Wang, K.-Q. Yan, W.-P. Liao, and S.-S. Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network," in *Proc. of 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, pp.108-113, 2010. Article (CrossRef Link)

[27] Wikipedia, "Load Balancing (computing)," http://en.wikipedia.org/wiki/Load_balancing_(computing).

[28] H.-J. Lu, C.-Z. Lai, M.-Y. Shih, Y.-H. Lee, C.-Y. Lu, C.-S. Yeh, and S.-A. Lee, "The Analysis of Load Balancing Performance for Virtualized Web Servers," (in Chinese) #F-417-1, Session F1, *TANET2012*, Oct. 23–25, Taoyuan, Taiwan, 2012.

[29] Editing Think Times to Simulate Web Site Human Interaction Delays in Load Tests Scenarios, http://msdn.microsoft.com/en-us/library/dd997697.aspx.

[30] C.-T. Lu, C.-S. Yeh, Y.-C. Wang, and F.-T. Tsai, “Research on Load Testing of Web Applications with Virtualization” (in Chinese), #1509, Session PDCC-2, *National Computing Symposium*, Taichung, Taiwan, Dec. 13-14, pp. PDCC 64-69, 2013.

Chien-Te Lu received M.S. degree in electrical engineering from National Taipei University of Technology, Taipei, Taiwan, in 2009. He is currently a Ph.D. student in the Department of Electrical Engineering of National Taipei University of Technology, Taipei, Taiwan. His research interests include cloud computing, web and mobile application architecture.

Chien-Te Lu received M.S. degree in electrical engineering from National Taipei University of Technology, Taipei, Taiwan, in 2009. He is currently a Ph.D. student in the Department of Electrical Engineering of National Taipei University of Technology, Taipei, Taiwan. His research interests include cloud computing, web and mobile application architecture.

Yung-Chung Wang received the M.S. and Ph.D. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1990 and 2000, respectively. From 1990 to 2001, he was a Research Engineer with the Chung-Hwa Telecommunication Laboratory, where he was engaged in research on the development of ATM switching systems and IP switch router systems. Since 2001, he has been with the Dept. of Electrical Engineering, National Taipei University of Technology (NTUT), Taipei, Taiwan, where he is a Full Professor. His research interests include cloud computing, wireless networks, optical networks, software defined network, queuing theory and performance evaluation of communication networks.

Chu-Sing Yang received the B.Sc. degree in Engineering Science from National Cheng Kung University (NCKU), Tainan, Taiwan, in 1976 and the M.Sc. and Ph.D. degrees in Electrical Engineering from NCKU in 1984 and 1987, respectively. He joined the faculty of the Dept. of Electrical Engineering, National Sun Yat-sen University (NSYSU), Kaohsiung, Taiwan, as an Associate Professor in 1988. Since 1993, he has been a Professor at the Dept. of Computer Science and Engineering, NSYSU. He was the chair of the Dept. of Computer Science and Engineering, NSYSU, from August 1995 to July 1999, and the director of the Computer Center from August 1998 to October 2002. He joined the faculty of the Dept. of Electrical Engineering, NCKU, as a Professor in 2006. He currently is a Professor of Electrical Engineering in the Institute of Computer and Communication Engineering at NCKU. He participated in the design and deployment of Taiwan Advanced Research and Education Network and served as the deputy director of National Center for High-performance Computing, Taiwan from January 2007 to December 2008. He was the Program Chair of ICS-96 and Program Co-Chair of ICPP-2003 and MTPP-2010. His research interests include network security, network virtualization, cloud computing, multi-core embedded system, smart grid, digital home, and intelligent computing.