2011 IEEE International Conference on Services Computing

Towards *Pay-As-You-Consume* Cloud Computing

Shadi Ibrahim, Bingsheng He*, Hai Jin

Cluster and Grid Computing Lab
Services Computing Technology and System Lab
Huazhong University of Science and Technology
Wuhan, 430074, China
{shadi, hjin}@hust.edu.cn

*School of Computer Engineering
Nanyang Technological University
Singapore, 639798
bshe@ntu.edu.sg

***Abstract*—Cloud computing enables users to perform their computation tasks in the public virtualized cloud using a pay-as-you-go style. Current pay-as-you-go pricing schemes typically charge on the incurred virtual machine hours. Our case studies demonstrate significant variations in the user costs, indicating significant unfairness among different users from the micro-economic perspective. Further studies reveal the reason for such variations is interference among concurrent virtual machines. The amount of interference cost depends on various factors, including workload characteristics, the number of concurrent VMs, and scheduling in the cloud. In this paper, we adopt the concept of pricing fairness from micro economics, and quantitatively analyze the impact of interference on the pricing fairness. To solve the unfairness caused by interference, we propose a pay-as-you-consume pricing scheme, which charges users according to their effective resource consumption excluding interference. The key idea behind the pay-as-you-consume pricing scheme is a machine learning based prediction model of the relative cost of interference. Our preliminary results with Xen demonstrate the accuracy of the prediction model, and the fairness of the pay-as-you-consume pricing scheme.**

Keywords- Cloud Computing; Virtualization; Pay-As-You-Go; Pay-As-You-Consume; Machine Learning.

I. INTRODUCTION

Cloud computing has recently emerged as a popular paradigm for harnessing a large number of commodity machines in the cloud. In such a paradigm, users are allowed to use the computation resources with respect to a pricing scheme similar to the economic exchanges in the utility market place. However, unlike the utility market, which typically has standard fine-grained charging units (such as *kilowatt per hour* (kwh) in electricity market), there are no standard pricing units in a cloud environment[1], particularly for computation as a services cloud. One of the common schemes used by recent cloud providers is primarily based on *virtual machine* (VM) hours on the virtualized cloud environment (e.g. Amazon charges per small instance hour at $0.085 [1]). Many existing studies [2, 3, 4] have focused on reducing the virtual machine hour usage. In contrast, we investigate the variance on the virtual machine hour usage in the cloud.

A cloud is a multi-tenant infrastructure - users may share the same physical infrastructure. Hence, there are interferences between VMs, such as interferences in disk and network I/O, and CPU (since the L2 data cache is shared on multi-core processors). For example, a user may have multi VM instances running within shared physical servers, on which the resource consumption of each VM varies due to the interference. This sharing leads to several fundamental observations: interferences may increase the running time of a certain task on a VM, resulting in unfairness between users (not only do interferences result in lower performance and higher cost for user, but the cost of interferences can also vary with users). Nevertheless, the amount of interference cost depends on various factors, such as an application's type, the number of VMs, and VM scheduling algorithms in the hypervisor [5, 6, 7, 8].

Since cloud computing is an economically-driven distributed system paradigm, pricing fairness is an important economic feature for a good pricing scheme [9, 10]. In economics, pricing fairness includes personal and social fairness. Personal fairness is subjective, and typically means that the pricing should be low enough, while social fairness mainly investigates whether users have the same financial cost for the same task. An unfair pricing scheme could foster dissatisfaction from users, and eventually the provider could lose customers. A previous study [11] has demonstrated the cost variance between different runs on Amazon EC2. In this study, we provide a quantitative study of the pricing fairness with respect to the virtualization internals, and investigate how we can remedy the pricing unfairness.

As a start to understanding these two aspects in the pricing fairness, we define two metrics to quantify these two kinds of fairness. For the same task, we use the difference in the cost for the same user when running the VM instance alone and in the presence of other VM instances to measure the personal fairness, and we use the difference in the cost among different users to measure the social fairness. Since in the cloud users run different tasks, we extend the social fairness to be the difference in the ratio of the extra cost caused by interference to the total cost. Accordingly, we study the performance interference caused by the abovementioned interference factors in our local Xen

[1] Infrastructure as a Service cloud can be classified into Storage as a Service and Computation as a Service. Our focus is on the pricing scheme in the CaaS, as in the SaaS we already have well defined fine-grained charging unite, data size/transfer per Gigabyte.

978-0-7695-4462-5/11 $26.00 © 2011 IEEE
DOI 10.1109/SCC.2011.38

virtualized cloud with a focus on both the personal and social fairness of users cost. We quantify the cost interference to be the difference between the concurrent execution and the execution without interference. We define the *Effective Virtual Machine Time* (EVMTime) to finish a task - the amount of time when the VM is only running on the physical machine - as the charging unit. With the two aforementioned fairness metrics, we observe that the current pricing scheme based on virtual machine time is neither personally or socially fair. To remedy the unfairness caused by interference, we propose a *pay-as-you-consume* pricing scheme, which charges users according to their effective resource consumption excluding interference. Thus, the *pay-as-you-consume* scheme reflects the real cost of executing the task and provides a fair cost to users, by means of the *Effective Virtual Machine Time*. Accordingly, our pricing scheme embraces an intelligent prediction model on the relative cost of interference.

Unfortunately, the cost estimation of the interference is a challenging issue, due to the following factors. The interference is caused by congestion in the shared resources: CPU, disk I/O and network I/O. Even worse, the fairness in individual resource does not guarantee global fairness, due to the misalignment among the scheduling in the individual resources. Another difficulty is that the hypervisor does not have full knowledge of the application running in the VMs. Intuitively; it seems to be very complicated task, even impossible, to model the interference in virtualized environments, but motivated by the huge success and accuracy level of using machine learning techniques for enhancing the system performance in storage systems [12], we propose to use machine learning techniques, particularly *Support Vector Machine* (SVM) algorithms, to automatically identify the key parameters affecting the interference. Our preliminary experimental results with Xen demonstrate the accuracy of the prediction model, and the fairness of the *pay-as-you-consume* pricing scheme. For example, our results of the data collected from the I/O benchmark, using libsvm [13] (a popular machine learning toolkit) demonstrate that our predication model can achieve around 90% accuracy in predicting the interference score for different I/O workloads. Thus, our *pay-as-you-consume* model can offer better fairness for users in terms of personal and social fairness.

The paper is organized as follows. Section 2 discusses the interference in Xen, followed by the motivating performance results in section 3. In section 4, we introduce the pay-as-you-consume model. We discuss related work in section 5 and conclude in section 6.

II. INTERFERENCE IN XEN

A. Architecture Overview

The Xen hypervisor is a para-virtualizing virtual machine monitor (VMM) [14, 15], in which the machine architecture presented to an operating system is not identical to the underlying hardware. The Xen hypervisor is responsible for resource (CPU, memory and I/O device, etc.) allocation for the various virtual machines running on the same hardware device. There is an initial domain, called Domain 0, which is a modified Linux kernel. Dom0 is a unique virtual machine running on the Xen VMM that has privilege to access physical I/O devices as well as interact with the other VMs. Other VMs sharing the same host with Dom0 are called DomainUs or Guest Os.

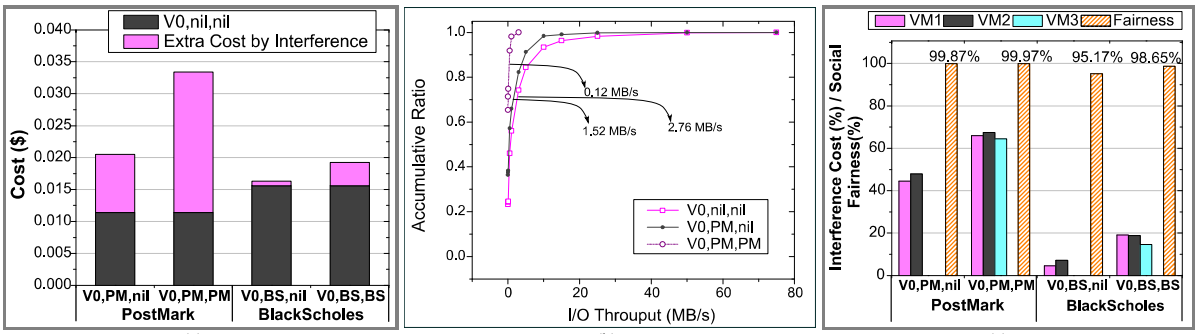
Xen Schedulers. Xen is unique among VMM software because it allows users to choose among different CPU schedulers and I/O schedulers. From version 3.1.0, Xen has two different CPU schedulers available, *Credit* and *Simple Earliest Deadline First* (SEDF), both allowing users to specify CPU allocation via CPU weights. Moreover, there are currently four available I/O schedulers in the 2.6 Linux kernels: Noop, Anticipatory, Deadline, and *Complete Fair Queuing scheduler* (CFQ). Furthermore, users can select the I/O schedulers on the fly in both Dom0 and DomUs. For more details about CPU and I/O schedulers used in Xen, readers can refer to [5, 6, 7].

B. Intra-machine Interference in Xen

The Xen hypervisor is responsible for providing isolation among the virtual machines and managing their access to hardware resources, that is, the Xen hypervisor performs functions such as scheduling processes and allocating memory among different guest operating systems. As the hardware resources are shared by multiple VMs, the current virtualized system experiences unpredictable and unstable performance, not to mention the performance degradation in some scenarios [8, 16, 17]. Different software solutions [18, 19] can be adopted to reduce the interference in the shared environment. Previous studies have revealed that the reason of such behavior is due to VMs interference. Interference in a virtualized environment is caused by two conflated reasons as explained below:

- **Inherited Interference**. This is the interference caused by the underlying technology, hardware and software, which is still a key problem in traditional (non-virtualized) systems, such as the shared L2 data cache on multi-core processors [20] (hardware). There is also the interference by selecting the right scheduler, CPU and I/O schedulers, within the operating system (software) when diverse applications are introduced [5, 7].
- **VMM interference**. The interference caused by the architecture of the Xen hypervisor, which is the tradeoff between risk isolation and fairness. In particular, we have driver domain interference as it provides access to the actual hardware I/O devices. Thus I/O resources will be shared by multiple VMs. In addition to the interference introduced by the resources contention between Dom0 and other domains that are running CPU applications, when I/O applications are performed [6, 21], all the traffic must pass through the driver domain.

Due to the various shared resources, interference does occur, especially when diverse applications are introduced on different VMs. This situation is even worse in the cloud, as a provider who is responsible to maintain and configure the VMM schedulers has no knowledge about the applications being run by users.



(a) (b) (c)

Figure 1. VMs consolidation impact on interference for both PostMark (PM) and BlackScholes (BS) benchmarks: (a) the cost of the same VM with different VM consolidation and application type. (b) CDFs of the I/O size per second in the virtual machine for PostMark benchmark with different VM consolidation (c) Interference cost for each VM and social fairness among all VMs within the physical host.

III. EMPIRICAL STUDY ON XEN

A. Micro Benchmarks

Looking at case studies about cloud providers [22, 23], we identify several popular applications, such as web-related tasks, storage backup, and high performance computing. As a start, we use the following micro benchmarks to mimic the workload in these popular applications. These include Postmark (I/O-intensive benchmark) [24] and PARSEC (we chose BlackScholes as an example to study the pricing fairness for CPU-intensive applications) [25] running on a single machine. We use the same settings as [11].

B. Experimental Setup

Our experiment is conducted on a physical node, equipped with two 2-core 2.33GHz Xeon processors, 4GB of memory and 500GB of disk, running CentOS. All results described in this section are obtained using Xen version 3.4.2. All the virtual machines used in our experiments are configured with 1 VCPU pinned to its own core and 768MB of memory and 60GB of virtual disk. We adopt the pricing scheme from Amazon: $0.085 for a small virtual machine instance [1]. We conduct our experiments on our local test bed so we have full control of the environment to get detail results of how the system internals affect the cost. We study the price fairness through evaluating different consolidation strategies as well as the Xen scheduler.

Metrics. We use the following metrics to evaluate the price fairness. The personal fairness is based on the extra cost caused by interference. We use the following formula:

$$Interference\ Cost = \frac{Extra\ Cost\ By\ Interference}{Total\ Cost} \quad (1)$$

In order to find the interference cost fairness (social fairness), we use Jain's fairness measure [26] to quantify the fairness among the VMs when running different applications within the same physical node:

$$Fairness = \frac{\left(\sum_{i=1}^{m} x_i\right)^2}{m \times \sum_{i=1}^{m} x_i^2} \quad (2)$$

where x_i denotes the interference cost of VM_i, and m is the number of the VMs within the same physical node.

1) The Impacts of the VM Consolidation on Fairness

In order to elaborate the impacts of VM consolidation on the fairness of users' cost, we vary the number of VMs which are deployed within the host to two and three VMs, while we run similar applications in the background[2].

Personal Fairness. We observe that, in the presences of VM consolidation, the current pricing scheme is far from being fair. Moreover, the interference cost is increasing with the number of VMs that are deployed within the same host and varies according to the running application.

For instance, the interference cost of VM when running the I/O intensive application *Postmark* has dramatically increased with different consolidation levels. As shown in Fig.1-a, the interference costs of $VM_{Postmark}$ when it shares the resources with two VMs and three VMs are nearly 45% and 66%, which indicates that the total VM' cost is two and three times higher than when it uniquely runs within the physical machine. The interference cost can be explained due to I/O congestion which adversely affects the I/O throughput of each VM as shown in Fig.1-b. Fig.1-b explains the previous result as it shows the *cumulative distribution function* (CDF) of I/O throughput in the virtual machine for the three aforementioned scenarios.

Fig.1-a shows that the interference cost of the VM when running CPU intensive applications has slightly increased with different consolidations. The interference cost is 5% and 19% when it shares the resources with two VMs and three VMs, respectively. This can be explained due to the cache interference between the different CPUs, which is relatively small. In our study all the VCPUs are pinned to a specific CPU core, and obviously the interference is increasing as the number of VMs increases.

Social Fairness. As shown in Fig.1-c, the social fairness is nearly optimal for the same applications regardless the VM consolidation. For example, the proportion of social fairness is nearly 99.5% when all VMs are running I/O intensive applications, because the default I/O VMM scheduler (CFQ) guarantees fairness in sharing the I/O resources amongst different VMs [7]. Moreover, the social

[2] We refer to the applications running on other VMs as background applications

fairness is nearly 96% when all the VMs are running CPU intensive applications, which is due to the fairness of the default CPU scheduler, the credit scheduler [5, 27].

In summary, we observe that for the same application, due to the resource contention in the presence of VM consolidation, the current pricing scheme based on a virtual machine time is not personally fair, while it is socially fair because the default VMM scheduler (CPU and I/O schedulers) tends to fairly share the resources among different VM instances.

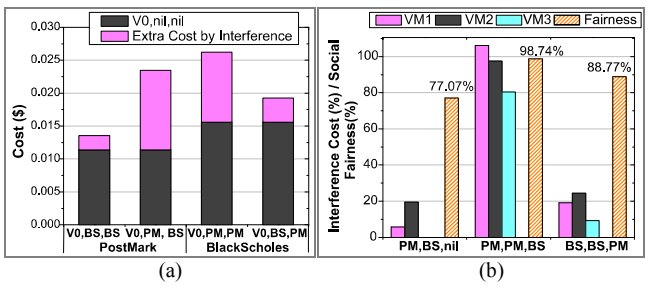
2) The Impacts of the Application Types on Fairness

In the cloud users may run different types of applications simultaneously, where the key difference is that they consume different types of resources (e.g., CPU, memory, network or disk). Thus we fix the number of VMs which are deployed within the physical host to three VMs while varying the applications which are running on each between the Postmark and BlackScholes benchmarks.

Personal Fairness. In Fig.2-a we see that the interference cost in a VM when running I/O intensive applications in two scenarios varies with the applications' diversity. When the two background applications are both CPU applications, the interference cost of VM has increased slightly which can be explained mainly because of the priority boost impacts in the Xen credit scheduler [6], that is, when an I/O event is incurred the credit scheduler will be invoked and boost the priority of an idle domain receiving an I/O event. As a result the I/O application will perform very close to the case where it does not share the physical host with any VM. However, the I/O application will suffer a slight degradation due to the CPU interference, that is, the cache interference between the VCPU in Dom0 and the VCPUs in others domains running CPU applications, knowing that CPU overhead in Dom0 is caused by the memory page exchange by the I/O application [21]. The interference cost in the $VM_{Postmark}$ is increasing as the number of the VMs with similar applications is increasing.

On the other hand and for the same reasons, partially due to L2 cache interference and mainly because of the priority boost impacts in the Xen credit scheduler [6], the VM with CPU intensive application suffers from relatively higher interference when both background applications are I/O applications. In contrast to the previous results, the interference cost in the $VM_{BlachSholes}$ is decreasing as the number of the VMs with same applications is increasing.

The previous discussion leads to a very important observation: when different applications are running within the same host, the interference cost is dominated by the resource contention between these applications and the VMM (i.e. the interference cost of an application is contributed to by direct interference with the VMM, for example, a CPU intensive application's performance degrades due to the L2 cache interference with VMM when a background application outperforms it in I/O operations) and indirect interference is caused by the VMM, for example an I/O application's performance degrades due to interference between Dom0 and other domains running CPU intensive applications.



(a) (b)

Figure 2. Applications types impacts on interference: (a) Extra cost by interference, (b) Interference cost for each VM and social fairness.

Social Fairness. Fig.2-b shows that when two different applications are sharing the host resources, the interference cost of a CPU-heavy application is relatively higher than an I/O application as explained earlier, and the proportion of social fairness is 77%. Moreover, when three VMs are deployed - two of them are running similar applications - the social fairness is relatively high. For example, it is 88% for the case of (CPU, CPU, I/O) and 98% for the case (CPU, I/O, I/O).

In summary, similar to the previous observation, we observe that social fairness varies according to the resource contention among the diverse applications running in the VMs along with the resource contention between VMs and VMM. In addition, the social interference is inversely proportional to the diversity of the applications that are sharing the same physical host, and it increases as more similar applications are sharing the host.

3) The Impacts of the VMM Schedulers on Fairness

The following experiments evaluate the impacts of the different VMM schedulers used in Xen, for instance, the I/O schedulers and the CPU schedulers, for interference when running CPU and/or I/O intensive applications on two VMs, and Table I presents the result.

Previous studies have reported on the importance of selecting the right I/O scheduler or CPU scheduler and tuning them according to the running applications [6, 7 , 16, 21], however our study tackles a different problem, and use a different approach. We study the impacts of both VMM schedulers and CPU schedulers side by side with the disk I/O schedulers, when diverse applications are running in the VMs, in terms of personal and social fairness, thus, we are not trying to detail the reason of the performance as it is well explained in the aforementioned research papers. Furthermore, as this paper is intending to identify unfairness in the cloud and the consequences of a provider's user-unaware administration, for the I/O schedulers we study the impacts of VMM scheduler regardless of the I/O scheduler running on the VM. As shown in Table I, the performance of different applications, *EVMTime*, vary slightly according to the selected CPU and I/O schedulers in the VMM layer when one VM is uniquely deployed in the physical host. However, when consolidation is introduced, the variation of the applications' performance and the system throughput (referred to as $Cost_{1,2}$ in Table I, where less cost indicates better system throughput) is increasing according to the resources' alignment policy decided by both CPU and I/O scheduler in the VMM layer.

TABLE I. PERSONAL AND SOCIAL FAIRNESS WITH DIFFERENT PAIR SCHEDULER (CPU: CREDIT AND SEDF, DISK I/O: CFQ (CF), ANTICIPATORY (AS), NOOP (NP), AND DEADLINE (DL)

	BS,BS								**PM,PM**								**BS,PM**							
	Credit				***SEDF***				***Credit***				***SEDF***				***Credit***				***SEDF***			
	CF	***AS***	***NP***	***DL***	***CF***	***AS***	***NP***	***DL***	***CF***	***AS***	***NP***	***DL***	***CF***	***AS***	***NP***	***DL***	***CF***	***AS***	***NP***	***DL***	***CF***	***AS***	***NP***	***DL***
EVMTime	559	**531**	532	536	543	557	550	562	405	366	450	405	377	400	460	387	559	**531**	532	536	543	557	550	562
TimeVM$_1$	587	559	570	567	575	583	558	579	738	724	1112	739	734	646	1061	898	687	586	533	555	585	550	589	558
TimeVM$_2$	584	594	590	584	549	566	542	577	786	787	1422	909	804	780	1424	825	454	418	525	476	458	514	600	437
Cost1,2($10^{-2}$$)	3.25	3.2	3.22	3.2	3.12	3.19	**3.06**	3.21	4.23	4.2	7.04	4.58	4.27	**3.96**	6.9	4.79	3.17	2.79	2.94	2.86	2.9	2.95	3.3	**2.76**
I$_1$	0.05	0.05	0.07	0.05	0.06	0.04	**0.02**	0.03	0.45	0.49	0.6	0.45	0.49	**0.38**	0.57	0.57	0.19	0.09	0.00	0.03	0.07	**-0.01**	0.07	**-0.01**
I$_2$	0.08	0.11	0.1	0.08	0.01	0.02	**-0.01**	0.03	**0.48**	0.53	0.68	0.55	0.53	**0.49**	0.68	0.53	**0.06**	0.12	0.14	0.15	0.18	0.22	0.23	**0.11**
Social Fair	95.3	88.6	96.5	96	67.2	83	78.9	99.6	99.9	99.8	99.5	99	99.8	98.5	99.2	99.9	79.5	98.1	50.3	71.3	84.7	55.4	76.6	55.2

Personal Fairness. The interference score varies by 35%, 14%, and 124% with different CPU and I/O schedulers' combination in the three scenarios: two CPU applications, two I/O applications and one CPU application concurrently running with one I/O application in the same host, respectively.

However, as shown in Table I, for the same applications, selecting the pair scheduler (SEDF, Anticipatory) leads to the lowest interference score in both VMs, hence it achieves the best system throughput. The interference scores are 0.02 and negative 0.01 for CPU applications and 0.38 and 0.49 for I/O applications. When different applications are running concurrently, the choice of the pair of schedulers is only sub-optimal for one application. Here the pair (SEDF, Deadline or Anticipatory) is the best for CPU application while (Credit, CFQ) is the best for the I/O applications.

Social Fairness. As shown in Table I, when both VMs are running CPU applications the social fairness varies by 12% with different pairs of schedulers, while the best social fairness is achieved when the pair (SEDF, Deadline) is selected in the VMM layer. For I/O applications the social fairness is nearly the same for all pairs schedulers with a slight advantage to the pairs of schedulers (credit, CFQ) and (SEDF, Deadline). However, the worst social fairness scenario occurred when different applications are sharing the same host, where the pair scheduler (SEDF, Deadline) had a social fairness score of 55.2%, although this pair scheduler achieves the best system throughput. These results are consistent with those in [6] (i.e. SEDF guarantees better system throughput and worse fairness than Credit in the case of I/O and CPU applications that are sharing the same host).

In summary, we observe that the choice of an appropriate pairs of schedulers (CPU, disk) at the VMM layer has a significant impact on the application performance inside each VM (personal fairness) and intra-application isolation among different VMs (social fairness). More importantly, in the cloud different workloads may be performed on the same host, and as a result there is no optimal pair of schedulers when diverse applications are introduced, although some of them are sub-optimal for different workloads and can achieve better overall system throughput. However, as this paper is a call to action, we encourage further study in the area of VMM schedulers to consider both personal and social fairness.

IV. PAY-AS-YOU-CONSUME PRICING STYLE

We argue that because the cloud is an economy-driven distributed system, we should consider the fairness in monetary costs. Therefore, we propose the new *pay-as-you-consume* pricing scheme, which charges users according to their effective resource consumption. However, due to the existence of interference, it is very hard to accurately determine the effective use of resources among different users. Accordingly, we define the *Effective Virtual Machine Time* to finish a task: the amount of time required when the VM is the only VM running on the physical machine. As for a VM, we consider the effective virtual machine time to be: virtual machine time less the interference time. Based on the effective virtual machine time, we define the pricing fairness: any user using the same amount of effective virtual machine time is charged at the same price.

$$Cost_{New\ Model} = Instance\ Per\ Hour \times EVMTime \tag{3}$$

Unlike current pricing schemes, our *pay-as-you-consume* pricing scheme solves the unfairness by charging the users according to their effective resources consumption. Since there are various factors affecting the interference, we propose to use a machine learning model to predict the interference based on the resource usage during the running time and charge users for their effective virtual machine time as shown in Equation 3. Thus, the same task tends to have the same cost, resulting in better personal and social fairness.

A. Interference Predication Model

If only one VM runs on the physical machine, the interference of the VM is zero. If the time for executing a task on a VM without any concurrent VM is t, and the time for executing the same task of the VM with other concurrent VMs is $t`$, we call the overhead of interference is $t`$-t.

We define I_i, the interference factor to VM_i:

$$I_i = \frac{t_i`}{t_i} - 1 \tag{4}$$

The intra-machine fairness means that: given all VMs $\{VM_1, VM_2, ..., VM_n\}$, running on the same physical machine, we should satisfy the two conditions:

$$I_i \to 0 \quad and \quad I_1 = I_2 = \ldots = I_i \tag{5}$$

To predict the interference factor when concurrent VMs are running within the same host, a VM is represented by a vector. The problem is transformed to: given multiple vectors, we estimate the overhead of interference on each vector. The vector includes the following items:

- **CPU**: the CPU time, CPI, the number of L1 data cache misses per instruction, the number of L2 data cache misses per instruction, the number of DTLB misses per instruction.
- **RAM**: the average amount of occupied main memory, the working set size.
- **Disk**: the number of I/O operations per second, the amount of data accessed per second, and the average length of the I/O queue per second.

We want to develop a prediction model:

$$(V_0, V_1, \ldots, V_n) \to I_0 \tag{6}$$

where V_0, V_1,…., and V_n are vectors for VM_0, VM_1, …, VM_n, respectively. I_0 is the interference factor to VM_0. The idea is to estimate the interference factor for V_0, with the super vector composed of $(V_0, V_1 \ldots V_n)$. Naturally, we can see: V_0 is different from $V_1 \ldots V_n$; (V_0, any permutation of $V_1 \ldots V_n$) will get the same results I_0. To train our model, we consider the following definitions:

- $(V_0, nil, nil \ldots nil)$ means we schedule the VM alone, and we can get the measurement t.
- $(V_0, V_1, nil \ldots nil)$ means we schedule two VMs, and we can get the measurement t_0' and t_1'.
- $(V_0, V_1, \ldots V_n)$ means we schedule n VMs, and we can get the measurement t_0', t_1', .. , and t_n'.

In machine learning, the accuracy of our model strongly depends on the training data set, and the correct execution of the different benchmarks which represent different workloads with different behaviors. Therefore, we need to expose the characteristics of interferences in the multiple shared layers in virtualization.

B. A Case Study on I/O Applications

Since the interference between I/O applications is higher than the one between CPU applications, in this paper we illustrate the effectiveness and accuracy of our model with I/O applications. Testing our prediction model, when diverse applications are running, is ongoing work in our group.

1) Experimental setup

Our experiment is conducted on a physical node, equipped with two 4-core 2.33GHz Xeon processors, 8GB of memory and 1TB of disk, running RHEL5 with kernel 2.6.22, and is connected with 1Gbps Ethernet. All results described in this section are obtained using Xen version 3.4.2. All the virtual machines used in our experiments are configured with 1 VCPU pinned to its own core and 1GB of memory and 60GB of virtual disk. VMM schedulers are set to the defaults: Credit for the CPU scheduler and CFQ for the I/O disk scheduler. We perform our experiments by repeatedly executing the benchmarks. According to previous studies on analyzing and predicting the performance of different workloads of I/O intensive applications [12, 28], we use a set of training workloads, reflecting various types of real-world I/O workloads including sequential and random read and write applications as shown in Table II. The workloads, shown in Table II, are generated using *Sysbench* [29]. We gather information about four different resource metrics related to CPU and disk I/O, which represent the items of the vector of our prediction system. These statistics are all gathered using *vmstat* [30], therefore, minimizing the effects of the monitoring system on our resource measurements.

TABLE II. TRAINING AND TESTING WORKLOADS

	Workload	Description
Training	Sequential write/Read	Writing/Reading sequentially different number of files varies from 1-512 files with differen size: 128KB-1GB
	Random write/Read	Writing/Reading randomly differen number of files varies from 1-5 12 files with differen size: 128KB-1GB with different threads number 6-512 threads
Testing	Sequential write (Seq Wr)	Writing sequentially 2 GB of data
	PostMark (PM)	Write 5 GB (1000 files with 5000 KB each)
	FFSB[31]	Multi threaded benchmarks that provide I/O operations, we configured FFSB with 128 threads and write operation of 5GB of data.

2) Predication Validation

To evaluate the accuracy of our model, we choose three different widely-used I/O applications in the cloud: sequential write, Postmark, and *Flexible File System Benchmark* (FFSB) [31] (shown in Table II). We use *libsvm* [13] to evaluate the feasibility of our model in predicting the interference. Table III shows the prediction accuracy of our model, for the three aforementioned test workloads, with other VMs that are running I/O workloads chosen from our trained workload.

We observe that, for the case of seq write which is one of the training model workloads, it is expected to get 97%-100% accuracy, but surprisingly, the accuracy varies between 94% and 97%, which can be explained due to the instability of the VMs performance under Xen [11]. For the rest of applications our model can achieve a prediction accuracy of 87% on average, where the best is 93% and the worst is 82%. As shown in Table III, our prediction model can achieve better accuracy when the interference score is relatively high, which can be explained due to the less noise caused by the interference amongst the different VCPUs of domains U and domain 0.

TABLE III. PREDICATION VALIDATION

	Real I_0	Predicted I_0	Accuracy
(Seq Wr, Seq Wr)	1.35	1.27	94%
(FFSB, Seq Wr)	0.24	0.28	83%
(Seq Wr, Seq Wr, Seq Wr)	2.9	2.81	97%
(PM, , Seq Wr, Seq Wr)	0.72	0.67	93%
(FFSB, Ran Wr, Ran Wr)	2.48	2.2	88.7%

3) Discussion on Users Cost Fairness

To quantify the effectiveness of our prediction model under our *pay-as-you-consume* pricing scheme using effective resource consumption charging methods, we study the fairness using an example of two VMs running within the same host, each of them representing one user and running two different I/O applications, Postmark and sequential write. As shown in Fig. 3, our *pay-as-you-consume* pricing scheme can achieve a personal fairness of 3% in the case of the Postmark application and 2% in the case of the sequential write application which is very small compared to the pay-as-you-go scheme (28% and 50% for both aforementioned applications). Moreover, with the social fairness given by the fairness of the interference cost of the different VMs within the same physical node, as shown in Fig. 3, we can achieve 98% while the pay-as-you-go model achieves 86%. We observe that both the personal and social fairness of our model is strongly proportional to the prediction accuracy. For example, the best personal fairness (the extra cost caused by interference) is achieved when the accuracy is 100% and thus the personal interference is zero.

One may expect that the new proposed model brings fairness to the cloud users, while it could lead to a loss in profit for the providers (i.e. the provider may not be able to recover even the cost of operating). However, previous work demonstrated that virtualization, featured with server consolidation, can significantly benefit the system providers, by achieving reduced server power consumption and close to optimal system throughput [11]. In summary, cloud providers, using our *pay-as-you-consume* model, can provide users with stable and fair cost, in particularly personal fairness and social fairness, while gaining considerable positive profit because they have increased trust from users by having more consistent charging. In addition, provider can gain a competitive advantage through *pay-as-you-consume* pricing scheme in the market of multiple providers.

V. Related Work

A. User's Cost in the Cloud

A number of studies [32, 33] have been dedicated to measure the cost of adopting the pay-as-you-go cloud in terms of monetary cost, performance, and availability. Some studies [11, 34] have reported on the cost variations in the cloud. Our work quantifies the variations with price fairness and investigates the reason of this variance. Recent studies [35] have demonstrated a case study of a consumer-centric resource accounting model to verify any discrepancies in consumer's bills. Yao et al. [36] introduced an accountability service model to unambiguously identify the reason and the responsible party in case of faulty service. In contrast, this paper investigates the interplay between micro-economic issues and the system design and implementation.

B. Interference in Shared Environment

There have been a lot of studies on performance interference in virtualized or shared servers. Boutcher et al. [7] and Kesavan et al. [16] have examined the impacts of the choice of disk I/O scheduler in both VMM and VMs on application performance. Despite our work being focused on the monetary cost, a key difference between our work and their work is that we are studying the impact of a provider's administration by selecting the VMM schedulers for I/O and CPU. Mei et al. [37] have measured the performance interference among two VMs running network I/O workloads that are either CPU bound or network bound and elaborated the impacts of co-locating applications in a virtualized cloud in terms of throughput and resource sharing effectiveness. Our work focuses on fairness in the monetary cost. Koh et al. [8] have elaborated the performance interference effects between two virtual machines by looking at the system-level workload characteristics. They identified clusters of applications that generate certain types of performance interference and developed mathematical models to predict the performance of a new application from its workload characteristics.

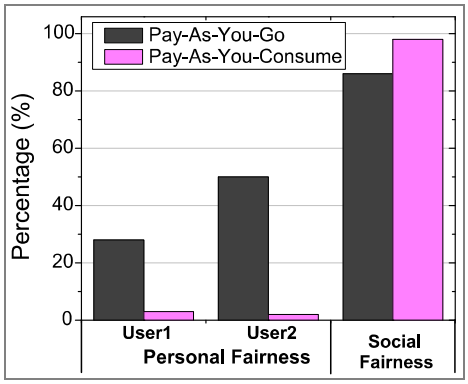


Figure 3. An example of the effectiveness of our prediction model associated with the Pay-As-You-Consume pricing scheme, when running two VMs, each represents different users and runs different I/O applications (Postmark and Sequential write). In this example we achieved accuracy of 86% and 97% respectively.

VI. Conclusion and Future Work

With the pay-as-you-go charging, the public cloud has become an economic market for both cloud users and providers. However, virtualization with server consolidation can cause performance interference, leading to non-guaranteed quality of service. In this study, we investigate the pricing fairness on the pay-as-you-go charging, and introduce the *pay-as-you-consume* model to resolve the unfairness in the current pay-as-you-go pricing scheme. While the *pay-as-you-consume* model seemingly reduces the cloud providers' profit, it urges providers to improve their system design and optimization to provide good services and to gain competitive advantages. We have demonstrated a case study on I/O applications for validating the accuracy of our model; interestingly our predication model can achieve up to 90% accuracy regardless the VM consolidations or the I/O workloads. This paper is intended as a call for action, and its goal is to motivate further research on economic concepts in the cloud. We hope the findings in this paper will foster new techniques as well as new pricing schemes in the cloud to really reflect the spirit of economic markets.

ACKNOWLEDGMENT

This work is supported by National 973 Key Basic Research Program under grant No.2007CB310900, the International Cooperation Program funded by Technology Bureau of Wuhan under grant No.201171034311, and a start-up grant No.M58020024 from Nanyang Technological University. We would also like to thank Lidong Zhou from MSR Asia for his insightful discussions.

REFERENCES

[1] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/, 2011.

[2] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for QoS-aware clouds," *Proc. of the 5th European conference on Computer systems (EuroSys'10)*, Paris, France, April 13-16, 2010, pp.237-250.

[3] B. S. He, M. Yang, Z. Y. Guo, R. S. Chen, W. Lin, B. Su, and L. D. Zhou, "Comet: Batched Stream Processing for Data Intensive Distributed Computing," *Proc. ACM Symposium on Cloud Computing (SOCC'10)*, Indiana, USA, June 10-11, 2010, pp.63-74.

[4] S. Ibrahim, H. Jin, L. Lu, B.S. He, L. Qi, and S. Wu, "LEEN: Locality/Fairness- aware Key Partitioning for MapReduce in the Cloud," *Proc. of the 2010 IEEE 2nd International Conference on Cloud Computing Technology and Science (CloudCom'10)*, Indiana, USA, Nov. 30-Dec. 03, 2010, pp.17-24.

[5] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the Three CPU Schedulers in Xen," *SIGMETRICS Performance Evaluation Review*, Vol.35, Sep. 2007, pp.42-51.

[6] D. Ongaro, A. L. Cox, and S. Rixner, "Scheduling I/O in Virtual Machine Monitors," *Proc. of the 4th International Conference on Virtual Execution Environments (VEE'08)*, Washington, USA, Mar. 5-7, 2008, pp.1-10.

[7] D. Boutcher and A. Ch, "Does Virtualization Make Disk Scheduling Passe?," *ACM SIGOPS Operating Systems Review*, Vol.44, Jan. 2010, pp.20-24.

[8] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An Analysis of Performance Interference Effects in Virtual Environments," *Proc. of the IEEE International Symposium on In Performance Analysis of Systems & Software (ISPASS'07)*, California, USA, Apr. 25-27, 2007, pp.200-209.

[9] S. Maxwell, *The Price is Wrong: Understanding What Makes a Price Seem Fair and the True Cost of Unfair Pricing*, Wiley, Jan. 2008.

[10] F. Black and M.S. Scholes, "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy*, Vol.81, June 1973, pp.637-654.

[11] H. Y. Wang, Q. F. Jing, R. S. Chen, B. S. He, Z. P. Qian, and L. D. Zhou, "Distributed Systems Meet Economics: Pricing in the Cloud," *Proc. of the 2nd USENIX Workshop on Hot Topics in Cloud computing (HotCloud'10)*, Boston, USA, June 22, 2010.

[12] Y. Zhang and B. K. Bhargava, "Self-learning Disk Scheduling," *IEEE Transactions on Knowledge and Data Engineering*, Vol.21, Jan. 2009, pp.50-65.

[13] C. C. Chang and C. J. Lin, "LIBSVM : A Library for Support Vector Machines," 2001, Software available at: http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[14] Xen Hypervisor Homepage, http://www.xen.org/

[15] I. Pratt, A. Warfield, P. Barham, and R. Neugebauer, "Xen and the Art of Virtualization," *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, New York, USA, ACM Press, Oct. 19-22, 2003, pp. 164-177.

[16] M. Kesavan, A. Gavrilovska, and K. Schwan, "On disk I/O Scheduling in Virtual Machines," *Proc. of the 2nd Workshop on I/O Virtualization*, Pennsylvania, USA, Mar. 13, 2010, pp.1-6.

[17] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforing Performance Isolation Across Virtual Machines in Xen," *Proc. of the ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'06)*, Melbourne, Australia, Nov. 27- Dec. 1, 2006, pp.342-362.

[18] A. Menon, S. Schubert, and W. Zwaenepoel, "TwinDrivers: semiautomatic derivation of fast and safe hypervisor network drivers from guest OS drivers," *Proc. Of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'09)*, Washington, USA, Mar. 7-11, 2009, pp.301-312.

[19] B. S. He and Q. Luo, Q, "Cache-oblivious query processing," *Proc. of the 3rd International Conference on Innovative Data Systems Research (CIDR'07)*, California, USA, Jan. 7-10, 2007, pp.44-55.

[20] P. Aparrao, R. Iyer, and D. Newell, "Towards Modeling and Analysis of Consolidated CMP Servers," *ACM SIGARCH Computer Architecture News*, Vol.36, May 2008, pp.38-45.

[21] L. Cherkasova, R.Gardner, "Measuring CPU overhead for I/O Processing in the Xen Virtual Machine Monitor," *Proc. of the USENIX Annual Technical Conference (USENIX'05)*, California, USA, April 10-15, 2005, pp.24-24.

[22] Windows Azure Case Studies, http://www.microsoft.com/azure/casestudies.mspx.

[23] Amazon Case Studies, http://aws.amazon.com/solutions/case-studies/.

[24] J. Katcher, "Postmark: a New File System Benchmark," *Technical Report*, Network Appliance, Aug. 1997.

[25] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The Parsec Benchmark Suite: Characterization and Architectural Implications," *Proc. of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT'08)*, Ontario, Canada, Oct. 25-29, 2008, pp.72-81.

[26] R. K. Jain, D. W. Chiu, and W. R. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," *Technical Report*, Digital Equipment Corporation, Sept. 1984.

[27] Credit Scheduler - Xen Wiki, http://wiki.xensource.com/xenwiki/CreditScheduler.

[28] S. L. Pratt and D. A. Heger, "Workload Dependent Performance Evaluation of the Linux2.6 I/O Schedulers," *Proc. of the Linux Symposium 2004*, Ottawa, Canada, July 21-24, 2004, pp.425-448.

[29] System performance benchmark, http://sysbench.sourceforge.net/

[30] Virtual memory statistic, http://linux.die.net/man/8/vmstat.

[31] Flexible File System Benchmark, http://sourceforge.net/projects/ffsb/.

[32] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for Science Grids: a Viable Solution?," *Proc. of the 2008 International Workshop on Data-aware Distributed Computing (DADC'08)*, Boston, USA, ACM Press, June 24, 2008, pp.55-64.

[33] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The Cost of Doing Science on the Cloud: the Montage Example," *Proc. of the 2008 ACM/IEEE conference on Supercomputing (SC'08)*, Texas, USA, Nov. 15-21, 2008, pp.1-12.

[34] S. L. Garfinkel, "An Evaluation of Amazon's Grid Computing Services: Ec2, S3 and SQS," *Technical Report TR-08-07*, Harvard University, July 2007.

[35] A. Mihoob, C. MolinaJimenez, and S. Shrivastava, "A Case for Consumer Centric Resource Accounting Models," *Proc. of the IEEE 2010 International Conference on Cloud Computing (Cloud'10)*, Florida, USA, IEEE Press, July 5-10, 2010, pp.506-512.

[36] J. H. Yao, S. P. Chen, C. Wang, D. Levy, and J. Zic, "Accountability as a Service for the Cloud," *Proc. of the IEEE International Conference on Services Computing (SCC'10)*, Florida, USA, July 5-10, 2010, pp.81-88.

[37] Y. D. Mei, L. Liu, X. Pu, S. Sivathanu, "Performance Measurements and Analysis of Network I/O Applications in Virtualized Cloud," *Proc. of the IEEE 2010 International Conference on Cloud Computing (Cloud'10)*, Florida, USA, IEEE Press, July 5-10, 2010, pp.59-66.