Distributed Systems Meet Economics: Pricing in the Cloud

Hongyi Wang[†] Qingfeng Jing[‡] Rishan Chen[◦] Bingsheng He[†] Zhengping Qian[†] Lidong Zhou[†]
[†]*Microsoft Research Asia* [‡]*Shanghai Jiao Tong University* [◦]*Peking University*

Abstract

Cloud computing allows users to perform computation in a public cloud with a pricing scheme typically based on incurred resource consumption. While cloud computing is often considered as merely a new application for classic distributed systems, we argue that, by decoupling users from cloud providers with a pricing scheme as the bridge, cloud computing has fundamentally changed the landscape of system design and optimization. Our preliminary studies on Amazon EC2 cloud service and on a local cloud computing testbed, have revealed an interesting interplay between distributed systems and economics related to pricing. We believe that this new angle of looking at distributed systems potentially fosters new insights into cloud computing.

1 Introduction

Recent cloud providers (e.g., Amazon Web Services, Google App Engine, and Windows Azure) have enabled users to perform their computation tasks in a public cloud. These providers use a pricing scheme according to incurred resource consumption. For example, Amazon EC2 provides a virtual machine with a single CPU core at the price of $0.095 per hour. This pay-as-you-go model lets users utilize a public cloud at a fraction of the cost of owning a dedicated private one, while allowing providers to profit by serving a large number of users. Case studies from these cloud providers [2, 10, 27] indicate that a variety of applications have been deployed in the cloud, such as storage backup, e-commerce and high-performance computing. For a provider, it is a non-trivial task to define a uniform pricing scheme for such a diverse set of applications.

This cloud-computing paradigm has transformed a traditional distributed system into a "two-party" computation with pricing as the bridge. A provider designs its infrastructure to maximize profit with respect to the pricing scheme, while a user designs her application according to the incurred cost. This is in contrast to a traditional distributed system, where the goal is to optimize for throughput, latency, or other system metrics as a *single* and *whole* system.

Pricing in cloud computing has two intertwined aspects. On the one hand, pricing has its root in system design and optimization. Resource-consumption based pricing is particularly sensitive to how a system is designed, configured, optimized, monitored, and measured (as shown in Section 4). On the other hand, pricing also has its root in economics, where key concepts such as *fairness* [17] and *competitive pricing* in a multi-provider marketplace affect the actual pricing. The pricing-induced interplay between systems and economics has fundamental implications on cloud computing, an important angle that should be explored by researchers. In this paper, we explore several dimensions of such an interplay.

Cost as an explicit and measurable system metric. With pricing, the dollar cost of computation becomes an explicit and measurable metric for system optimizations. This begs the questions: how do we optimize a system based on this new metric? How is this metric related to traditional system metrics such as throughput? If both providers and users optimize based on their dollar cost and profit, does this lead to a globally optimal system that is most effective in getting the work done at the lowest cost? Clearly, pricing impacts the answers to those questions.

Pricing fairness. The key concepts of pricing such as competition and fairness affect choices in the design of user applications and system infrastructures. As a start, we investigate the pricing fairness in the cloud. Since users and providers have different and often conflicting incentives, pricing fairness balances user cost and provider profit. Even within the scope of pricing based on resource utilizations, we have choices on what to charge (e.g., virtual machine time vs. actual CPU time)

and different approaches achieve different levels of fairness.

Evolving system dynamics. The underlying systems for cloud computing evolve over time: individual machines might grow more powerful, have more CPU cores [6], accelerator [13] and/or adopt new storage medium such as flash in place of disks [16]. Will a provider gain a competitive advantage by adopting those new technologies for selected applications? Should pricing evolve along with innovations in systems?

Cost of failures. Failures are bound to happen in cloud computing, due to both hardware failures and software issues [4]. Traditional system design cares mostly about tolerating failures and about recovery from failures. In a pay-as-you-go cloud, we have to worry also about the expenses incurred by failures and more importantly face the question of who is to be responsible for those expenses. Under the current pricing scheme used by Amazon, those expenses fall entirely on the shoulder of users, regardless of root causes.

It is *not* our intention to answer all these questions in this paper. Rather, we focus on making the case that those problems are worth exploring in the context of the current state of affairs in cloud computing. We do so through our preliminary studies on Amazon EC2, one of the major cloud providers, and with *Spring*, our home-grown testbed. Amazon EC2 provides a real cloud service, which we can study as a black box, while Spring offers us the opportunity to investigate the underlying distributed system. We find that (i) optimizing for cost does not necessarily lead to an optimal system; optimization is hard for users due to their limited knowledge of the underlying mechanisms that a provider has in place; (ii) pricing unfairness is evident with the current pricing scheme used by Amazon; (iii) Different system configurations have a significant impact on the cost and profit; (iv) failures do appear and incur non-negligible expenses on users.

Organization. The rest of the paper is organized as follows. We introduce the background on pricing in Section 2. Section 3 provides an overview of our experimental methodologies, followed by the experimental results in Section 4. We conclude and discuss future work in Section 5.

2 Background on Pricing

This section introduces the background on the pricing scheme, and related studies on pay-as-you-go charging.

2.1 Pricing

Pricing plays a key role in the marketplace, which has been well studied in economics [17]. Among various factors that impact pricing, fairness [19] and competition [17] are the most relevant to our current study.

Pricing fairness consists of two aspects: *personal* and *social* fairness [19]. Personal fairness is subjective, meaning that price meets users' personal expectation. Social fairness is objective, meaning that price is the same for all users, does not give a provider unreasonably high profits and so on. For example, the differentiated economy class air ticket prices are socially unfair because some passengers have to pay more than others. Maxwell summarized the fairness in economic exchange: what is priced, who gets price exceptions, what is included in the price, and so on. For example, a fair price should be charged to everyone, but adjusted for the needy. We refer readers to Maxwell [19] for more details. In this paper, we focus on the social fairness of the pricing scheme, and examine whether price is the same for all users.

Competition unleashes the power of markets in economics [17]. With competition, providers cannot set their prices in a way most favorable to them. Instead, they gain a competitive advantage through adopting new technology and lowering their cost.

2.2 Pay-as-you-go Model

Pricing is not only important for economics, but also helps to shape how systems are used; for example, pricing can control congestion on Internet resources [18], and adjust the computation resource demand and supply in the grid [7].

In the pay-as-you-go model, the pricing scheme becomes an important bridge between users and providers. The current practice among major cloud providers is to price computing based on virtual-machine hours; for example, Amazon charges $0.095 per virtual-machine hour. Moreover, pricing schemes are evolving and more pricing schemes are introduced. For example, Amazon now has a set of different pricing schemes including auction pricing. Additionally, several alternative pricing schemes have been proposed for better system behavior in the cloud. Singh et al. [23] suggested dynamic pricing when reserving computation resources. Jimenez et al. [20] developed a bilateral accounting model between users and providers to avoid malicious overcharge. This paper focuses on the common one among major cloud providers, charging based on incurred virtual-machine hours.

Evaluating and optimizing user expenses in a pay-as-you-go cloud has recently attracted research interest [4, 8]. Napper et al. [21] and Walker [26] compared Amazon EC2 with a private cloud for high-performance computing. The cost, availability, and performance of Amazon services was studied with simple operations [22, 9]. In

contrast, this study focuses on both costs and profits with respect to pricing, and the resulted interplay between systems and economics.

3 Methodology Overview

We have assembled a set of workloads to approximate a typical workload in current cloud computing. With these workloads, we use two complementary approaches for evaluations. One is a black-box approach with Amazon EC2. As other cloud providers such as Google and Microsoft use similar pricing schemes, we expect that our pricing-related findings to be applicable to those as well. Our second approach is to set up a cloud-computing testbed, called *Spring*, so that we can perform fully-controlled experiments with the full knowledge of how the underlying system works.

3.1 Workloads

We have identified several popular applications to simulate different applications domains listed in the case studies in cloud providers [2, 10, 27].

Postmark. We use Postmark [15] as an I/O-intensive benchmark, representing the file transactions for various web-based applications. The default setting in our experiment is as follows: the total file size is around 5 GB (1000 files, 5000 KB each); the number of transactions is 1000.

PARSEC. PARSEC [5] is a benchmark suite composed of real-world applications. We choose Dedup and BlackScholes to represent storage archival and high-performance computing in the cloud, respectively. Dedup compresses a file with de-duplication. BlackScholes calculates the prices for European options. The default setting is as follows: Dedup has around 184 MB input data for deduplication; the number of options for BlackScholes is 10 million.

Hadoop. We use Hadoop 0.20.0 for large-scale data processing. We choose WordCount and StreamSort from the GridMix benchmark [11]. The default input data set is 16GB for both applications.

3.2 Methodology on Amazon EC2

In the experiments with Amazon EC2, our execution is charged according to the pricing scheme of Amazon. We consider the amortized cost in a long running scenario, and calculate user expenses as $Cost_{user} = Price \times t$, where t is the total running time of the task in hours, and $Price$ is the price per virtual machine hour. We exclude the costs on storage and on data transfer between the client and the cloud, since they are negligible in our experiments (less than 1% of the total cost).

3.3 Methodology on the Spring System

Spring virtualizes the underlying physical data center and provides virtual machines to users. Spring consists of two major modules, namely VMM (Virtual Machine Monitor) and an auditor. VMM is responsible for VM (Virtual Machine) allocation, consolidation and migration among different physical machines. The auditor calculates user expenses and estimates profits for the provider. The estimation helps to understand the impact of pricing in the cloud.

We estimate the provider profit by subtracting the total provider cost from the expected payment from users. While the total user payment is directly measured from incurred virtual machine hours, it is challenging to have an accurate estimation on the provider profit in a real data center. As a starting point, we adopt Hamilton's estimation of the total cost of a large-scale data center [12]. Hamilton calculates the amortized cost of running a data center as the sum of server costs, power consumption, power and cooling infrastructures and other costs. Hamilton further defines *full burdened power consumption* as the total power consumption of IT equipments, and power and cooling infrastructures [12].

Following Hamilton's estimations, we calculate the total cost of the full burdened power consumption to be $Cost_{full} = (p \times P_{raw} \times PUE)$, where p is the electricity price (dollars per kWh), P_{raw} is the total energy consumption of IT equipments including all the servers and routers (kWh), PUE is the PUE value of the data center [25]. *PUE* [25] is a metric to measure the efficiency of a data center's power distribution and mechanical equipment.

The total provider cost is estimated with $Cost_{provider} = (Cost_{full} + Cost_{amortized}) \times Scale$, where $Cost_{amortized}$ is the total amortized server cost, and $Scale$ is the ratio of the estimated total cost to the sum of the cost of full burdened power consumption and $Cost_{amortized}$ in Hamilton's estimation.

To estimate $Cost_{amortized}$, we estimate the amortized cost per sever to be $(C_{amortizedUnit} \times t_{server})$, where $C_{amortizedUnit}$ is the amortized cost per hour per sever and t_{server} is the elapsed time on the server (hours).

We further estimate P_{raw} as the total power consumption of servers and routers. For a server, we use a simple linear regression model [14] estimating the energy consumption based on resource utilization, i.e., $P_{server} = P_{idle} + u_{cpu} \times c_0 + u_{io} \times c_1$, given CPU utilization and I/O bandwidth u_{cpu}% and u_{io} MB/sec, respectively, and c_0 and c_1 are the coefficients in the model. We adopt the power consumption model of the network router from a previous study [3].

Instance Type	CPU (#virtual core)	RAM (GB)	Storage (GB)	Price ($/h)
Small	1	1.7	160	0.095
Medium	2	1.7	350	0.19

Table 1: The configurations and prices on different VM types on Amazon (Linux, California, America, Jan-2010)

	Eight-core machine	Four-core machine
CPU	Intel Xeon E5335 8-way 2.00GHz	Intel Xeon X3360 Quad 2.83GHz
RAM (GB)	32	8
Disk	RAID 5 (SCSI disks)	RAID 0 (SATA disks)
Network	1 Gigabit	1 Gigabit
Power model	$P_{idle} = 299$, $c_0 = 0.46$, $c_1 = 0.16$	$P_{idle} = 250$, $c_0 = 0.4$, $c_1 = 0.14$

Table 2: Hardware configuration of machines in Spring

	On a small instance		On a medium instance	
	Elapsed time (sec)	Cost ($)	Elapsed time (sec)	Cost ($)
Postmark	204.0	0.0054	203.2	0.0106
Dedup	45	0.0012	14	0.0008
BlackScholes	934	0.0246	215	0.0113

Table 3: Elapsed time and costs of single-machine benchmarks on small and medium instances on EC2

4 Preliminary Evaluations

In order to identify various dimensions of the pricing-induced interplay between systems and economics, we have performed a series of experiments on both Amazon EC2 and on Spring.

4.1 Experimental Setup

The same set of experiments is run on both Amazon EC2 and Spring.

Setup in Amazon EC2. We use the two default on-demand virtual-machine types provided by EC2, namely *small* and *medium* instances. These virtual machines run Fedora Linux and are located in California, USA. Table 1 shows their configurations and prices.

Setup in Spring. We use VirtualBox [24] to implement a virtual machine in Spring. VirtualBox allows us to specify the allocation of computation resources so that we can approximate the configurations and prices of different kinds of instances on Amazon. The host OS is Windows Server 2003; the guest OS is Fedora 10.

The physical machine configuration is shown in Table 2. We use an eight-core machine to evaluate the single-machine benchmarks, and a cluster consisting of 32 four-core machines to evaluate Hadoop. We assign a CPU core exclusively to a VM to approximate a virtual core in Amazon EC2. For example, we consolidate at most four medium instances on the eight-core machine.

We also list the parameter values in our power consumption model for both types of machines. We use a power meter [1] to measure the actual power consumption of a server, and construct the power consumption model based on these measurements. We have validated the power consumption model, and the estimation is close to the measured value given by a real power meter.

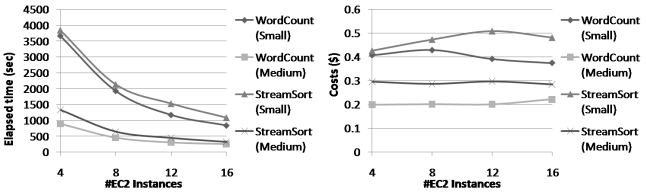
For estimating the total dollar cost, we follow Hamilton's parameter settings on a data center of 50 thousand servers [12]: $PUE = 1.7$, $Scale = 2.24$, energy price $p = \$0.07$ per kWh and amortized server unit cost $C_{amortizedUnit} = \$0.08$ per hour.

As an example of evaluating the effect of hardware changes, we evaluate the case of flash in place of disks. In that evaluation, we use an Intel 80 GB X25-M SSD (Solid State Drives) to replace a SATA hard drive. The current price of the SSD is around $350, and the price of a SATA hard drive (500GB) is around $50. We adjust the amortized cost in the machine with an SSD to $\$0.09$ per hour. Compared with hard disks, SSDs also offer a power efficiency advantage, and we adjust power consumption accordingly.

We study the system throughput of Spring in numbers of tasks finished per hour, as well as user costs and provider profits. Additionally, we calculate the efficiency of a provider's investment using ROI (Return on Investment), i.e., $ROI = \frac{Profit}{Cost_{provider}} \times 100\%$.

4.2 Optimizing for Cost

We study the difference between optimizations for cost and optimizations for performance on users and providers separately. We first present the results of user optimizations on EC2, since the results on Spring are similar to those on EC2. Next, we present the results of provider optimizations, including consolidation and different workload scheduling algorithms on Spring.



(a) Time for Hadoop (b) Costs for Hadoop

Figure 1: Performance and costs for Hadoop vs. the number of same-type instances on EC2

#VM per physical machine	One VM	Two VMs	Four VMs
Average elapsed time (sec)	127	125.5	425
Average cost per task ($)	0.004	0.004	0.012
Total cost of users ($)	0.014	0.014	0.047
P_{raw} (kWh)	0.046	0.024	0.038
$Cost_{provider}$ ($)	0.024	0.012	0.020
Profit ($)	-0.009	0.002	0.028
ROI (%)	-40.0%	17.2%	142.0%
Throughput (tasks/h)	28.3	56.4	33.9

Table 4: Effects of virtual-machine consolidation in Spring (every four Postmark, small VM type)

#VM per physical machine	One VM	Two VMs	Four VMs
Elapsed time (sec)	231.2	239.5	334.5
Average cost per task ($)	0.013	0.013	0.019
Total cost of users ($)	0.051	0.053	0.074
P_{raw} (kWh)	0.080	0.043	0.032
$Cost_{provider}$ ($)	0.042	0.022	0.016
Profit ($)	0.010	0.031	0.058
ROI (%)	22.8%	140.8%	365.2%
Throughput (tasks/h)	15.6	30.1	40.0

Table 5: Effects of virtual-machine consolidation in Spring (every four BlackScholes, medium VM type)

4.2.1 User Optimizations on EC2

User optimizations on EC2 include application-level optimizations for a fixed instance type, choosing the suitable instance type, and tuning the number of instances.

For a fixed instance type, we tune the number of threads for the elapsed time and cost on Postmark and PARSEC. Such tuning improves both performance and user cost. On current consumption-based pricing, there is no gap between optimizations for performance and optimizations for cost.

Choosing the suitable instance type is important for both performance and cost. Table 3 shows the elapsed times and costs of optimized single-machine benchmarks. Postmark has slightly larger elapsed time on a small instance, but achieves almost 50% smaller cost on a small instance than on a medium instance. This indicates a conflict in choosing the suitable instance type between for performance and for cost. In contrast, Dedup and BlackScholes on a medium instance have a smaller execution time, and a smaller cost than those on a small instance. This indicates that choosing the suitable instance type for cost may not result in the best performance, and vice versa.

Figure 1 shows the performance and cost of Hadoop as we vary the number of instances from four to sixteen. There is no clear patten in terms of cost when we vary the number of instances and the instance types. Again, the setting that achieves the minimum cost differs from that of the best performance.

4.2.2 Provider Optimizations on Spring

We mainly focus on VM consolidation optimization on Spring, which tune the number of concurrent VMs running on the same physical machine.

We first study the effect of virtual-machine consolidation when running the same set of tasks on Spring. We vary the number of VMs on the eight-core machine from one to four. Tables 4 and 5 show the results for running Postmark and BlackScholes tasks continuously, respectively. We report the number when the execution is steady. We choose Postmark and BlackScholes to run on small and medium instances respectively, as they have a smaller cost on those VM types, similar to the results observed on EC2. We do not present the results for Dedup, since they are similar to those of BlackScholes.

We make the following three observations.

First, consolidation greatly reduces power consumption. In this experiment, consolidation reduces around 150% and 21% on P_{raw} for BlackScholes and Postmark,respectively.

Second, since consolidation increases the total cost for users and decreases the cost of power consumption, a provider's profit increases significantly, so does its ROI. In our estimation, without consolidation, a provider actually loses money on Postmark. With consolidation, a provider enjoys a significant improvement in ROI, with an increase of 180% and 340% on Postmark and BlackScholes, respectively. To make profit, a provider should choose a suitable consolidation strategy.

Finally, as more tasks are consolidated to the same physical machine, the throughput reaches a peak at consolidating two VMs with Postmark, and then degrades. Although a provider can make higher profit on consolidating more VMs, the system throughput can degrade up to over 64% compared with the peak. This points to a potential flaw in pricing: as a provider adopts a strategy that maximizes its profit, the strategy could lead to sub-optimal system throughput.

We further study multi-machine benchmarks on Hadoop. Table 6 shows the results of two cases for running Hadoop on eight VMs. Consolidation also significantly increases a provider's profit, with an increase of 135% and 118% on ROI for WordCount and StreamSort, respectively. However, all these increases in the profit come at the cost of degrading system throughput with a reduction of 12% and 350% on WordCount and StreamSort, respectively. This confirms the problem we have seen in single-machine benchmarks.

	WordCount		StreamSort	
#VM per physical machine	One	Two	One	Two
Total cost of users ($)	0.412	0.461	0.592	2.815
Profit ($)	0.083	0.277	0.110	1.646
ROI (%)	25.0%	150.1%	22.8%	140.8%
Throughput (tasks/h)	3.8	3.4	2.7	0.6

Table 6: Effects of virtual-machine consolidation in Spring (with Hadoop, eight VMs)

Topic	#Threads	Ratio
Clarification on charging (e.g., the definition of instance hours)	67	38.7%
Complaints on charging too high for certain scenarios (e.g., charging on the idle instance)	34	19.7%
Cost comparison among different vendors (Amazon, Google and Microsoft etc)	19	11.0%
Different prices (Reserved mode, daily charge, debugging fee etc)	25	14.5%
Others (Security, economics, software licensing etc)	28	16.2%

Table 7: Categorized threads in official forums on Amazon EC2, Google App Engine and Windows Azure

4.3 Pricing Fairness

Personal fairness. To understand users' concerns with pricing fairness, we surveyed the official forums of recent cloud providers. We have obtained the threads related to pricing by searching with keywords "price" or "pricing". We have found 173 threads about pricing (up to Jan 10, 2010), each thread consisting of multiple replies. We further manually categorize them according to the topic, and the categorization is shown in Table 7. Users mostly need clarification on how they are charged. The second most popular topic is about complaints on charing too high for certain scenarios, which is an indication of personal unfairness in the pricing scheme.

Social Fairness. We investigate the social fairness of the pricing scheme by examining the variation of costs for the same task. We use two metrics to measure the cost variation: the coefficient of variation ($cv = \frac{stdev}{mean} \times 100\%$), and the maximum difference ($maxDiff = \frac{Hi-Lo}{Lo} \times 100\%$, where Hi and Lo are the highest and lowest values among different runs, respectively).

We first look at the variations of different runs on the same instances in Amazon EC2. We run each single-machine benchmark ten times. Table 8 shows the variations of execution time of the single-machine benchmarks. The variations are significant. For example, the worst case of Postmark incurs a cost that is 40% higher than its best case.

	Postmark	Dedup	BlackScholes
cv	9.1%	11.0%	3.9%
$maxDiff$	40.1%	38.8%	12.6%

Table 8: Variation of different runs on EC2

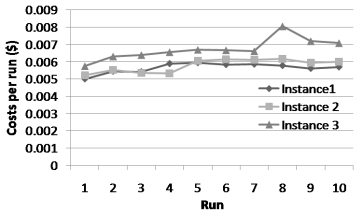


Figure 2: Variations among three instances (Postmark) on EC2

We also investigate the variations among different concurrent instances, which represents the scenarios of different users using a cloud simultaneously. Figure 2 shows the cost of running Postmark ten times on three different small instances. There is noticeable differences among different instances. The average cost on Instance 3 is over 20% higher than those on Instances 1 and 2.

We have observed similar variations in Spring to those in Amazon EC2 (as shown in Tables 4, 5 and 6 of Section 4.2.2). As more VMs are consolidated onto the same physical machine, users pay more money for the same task. For example, postmark with "Four VMs" costs around three times of that with "One VM".

The cost variations on both Amazon and Spring indicate social unfairness of the current pricing scheme.

4.4 Different Hardware Configurations

Figure 3 shows the costs and profits of running a Postmark task on a small instance with a hard disk and an SSD. The elapsed times of Postmark are around 180 and 400 seconds on the SSD and on the hard disk, respectively. Using the SSD greatly reduces the user's cost by 120%, and slightly decreases the provider's ROI from -40% to -44%. By supporting cost-efficient services to users, the provider has the power in adapting its price scheme to its advantage. Furthermore, it is likely that a provider needs to fine-tune its pricing structure, so that it can balance the benefit to users and the profit to itself.

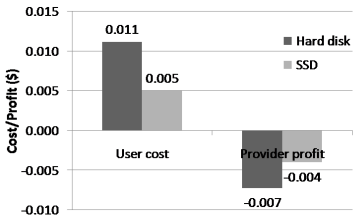


Figure 3: Costs and profits of running Postmark on a hard disk and an SSD without consolidation in Spring.

4.5 Failures

During our study, we have met a few bugs in the benchmarks on Amazon EC2, even after testing the implementations on Spring. Here is one example: we have successfully executed Hadoop in Spring, but still met one Hadoop exception with a message "Address already in use"[1] on Amazon EC2. Since this kind of bug originates from the software stack of Hadoop interacting with Amazon EC2, users can hardly prevent such bugs without paying them.

Bugs are not the only cause of failures; transient failures in the cloud infrastructure also occur. For example, we observed a task failure when we ran StreamSort using Hadoop on eight VMs in Spring, causing a sharp eight-time increase in the total elapsed time. This shows that transient failures in the underlying infrastructure could be a significant factor for provisioning user costs.

5 Concluding Remarks

By embracing a pricing scheme that connects providers with users, cloud computing has managed to bridge between distributed systems and economics. In this paper, we focus on understanding the implications of this new paradigm and its impact on distributed system research. Our preliminary study has revealed interesting issues as a result of the tensions between users and providers and between distributed systems and economics. We hope that this will help launch an inter-disciplinary endeavor that eventually contributes to the emergence of cloud computing.

Acknowledgments

We are grateful to the anonymous reviewers, James Larus, as well as the System Research Group of Microsoft Research Asia for their insightful comments.

References

[1] http://wattsupmeters.com/.

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

[3] G. Ananthanarayanan and R. H. Katz. Greening the switch. In *HotPower*, 2008.

[4] M. Armbrust and A. Fox et al. Above the clouds: A Berkeley view of cloud computing. Technical Report EECS-2009-28, UC Berkeley, 2009.

[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *PACT*, 2008.

[1] http://issues.apache.org/jira/browse/HADOOP-5655.

[6] S. Boyd-Wickizer, H. Chen, R. Chen, Y. Mao, F. Kaashoek, R. Morris, A. Pesterev, L. Stein, M. Wu, Y. Dai, Y. Zhang, and Z. Zhang. Corey: An operating system for many cores. In *OSDI*, 2008.

[7] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience*, 14(13–15).

[8] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: the Montage example. In *SC*, 2008.

[9] S. L. Garfinkel. An evaluation of Amazon's grid computing services: EC2, S3 and SQS. Technical Report TR-08-07, Harvard Univ., 2007.

[10] Google App Engine Developer Profiles. *http://code.google.com/appengine/casestudies.html*.

[11] GridMix. *http://github.com/yahoo/hadoop/tree/54428cc8dd437b4de9efe070e777023ec171a498/src/benchmarks/gridmix*.

[12] J. Hamilton. *http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx*.

[13] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. Mars: a mapreduce framework on graphics processors. In *PACT*, 2008.

[14] A. Kansal and F. Zhao. Fine-grained energy profiling for power-aware application design. In *HotMetrics*, 2008.

[15] J. Katcher. Postmark: a new file system benchmark. Technical Report TR-3022, Network Appliance, 1997.

[16] Y. Li, B. He, Q. Luo, and Y. Ke. Tree indexing on solid state drives. In *Proceedings of VLDB Endowment*, 2010.

[17] N. Mankiw. *Principles of economics*. South-Western Pub, 2008.

[18] R. Mason. Simple competitive Internet pricing. 2000.

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

[20] C. Molina-Jimenez, N. Cook, and S. Shrivastava. On the feasibility of bilaterally agreed accounting of resource consumption. In *Service-Oriented Computing*, 2009.

[21] J. Napper and P. Bientinesi. Can cloud computing reach the top500? In *UnConventional high performance computing workshop*, 2009.

[22] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon S3 for science grids: a viable solution? In *DADC*, 2008.

[23] G. Singh, C. Kesselman, and E. Deelman. Adaptive pricing for resource reservations in shared environments. In *Grid*, 2007.

[24] Sun VirtualBox. *http://www.virtualbox.org/*.

[25] The Green Grid. The Green Grid data center power efficiency metrics: PUE and DCiE. Technical report, 2007.

[26] E. Walker. Benchmarking Amazon EC2 for high-performance scientific computing. *The USENIX Magazine*, 33(5), 2008.

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