2017 IEEE 10th International Conference on Cloud Computing

Cloud Bursting Scheduler for Cost Efficiency

Young Choon Lee* and Bing Lian†
*(이영춘) Department of Computing
Macquarie University, Sydney, Australia 2109
Email: young.lee@mq.edu.au
†School of Information Technologies
The University of Sydney
NSW 2006 Australia
Email: blian@uni.sydney.edu.au

***Abstract*—Clouds have been increasingly adopted due to primarily their elasticity and pay-as-you-go (PAYG) pricing. While many organizations outsource the entire ICT solution to public clouds like Amazon Web Services Elastic Compute Cloud (EC2), others consider occasional workload offloading (cloud bursting) due to various reasons including governance and security. In this paper, we present Cloud Bursting Scheduler (CBS), a new cloud bursting algorithm. CBS explicitly takes into account cost factors of private in-house system (or private cloud) and public cloud. In particular, CBS attempts to optimize the cost to performance ratio by offloading jobs to public cloud explicitly taking into account time-varying electricity rates with private clouds and the time-invariant rental rate of many public clouds. Based on simulation results obtained using real workload traces, CBS saves costs of running workloads by 55% and 12% compared with costs of cloud sourcing and private cloud, respectively. It also improves resource utilization (to 89%) by judiciously (de)activating in-house resources and dynamically provisioning cloud resources.**

***Keywords*—Cloud computing; cloud bursting; cost efficiency; scheduling; energy efficiency**

I. Introduction

The elasticity and cost effectiveness (pay-as-you-go pricing) are two main driving forces of the cloud. Although cloud service providers (e.g., Amazon Web Services and Microsoft Azure) claim to offer users access to abundant resources[1] in an on-demand, open manner, the performance and cost of a cloud deployment are heavily dependent on how effectively the resource abundance of the cloud is exploited [1], [2], [3]. In other words, resource provisioning level in *size* and *shape* should be optimally determined.

The use of a public cloud for the complete computing solutions, so-called cloud sourcing, might be a cost-effective alternative. Cloud sourcing is particularly beneficial for small and medium sized organizations who cannot easily afford all ICT costs including staffing, maintenance and energy costs. Here, the elasticity in resource provisioning with clouds is a main advantage over private in-house systems. There are many case studies including Neflix (https://www.netflix.com) and Pinterest (https://www.pinterest.com).[2]

However, as many organizations already operate their own computer systems, the dynamic use of public clouds in addition to their own resources (cloud bursting) is a more practical option. We advocate cloud bursting with three advantages: (1) scalability (scaling out: sporadic workload surges can be effectively handled, (2) resource utilization: resource provisioning level can be set to the average/normal workload, and (3) security: a multi-cloud security policy can be enforced. Security is a major concern with the adoption of public cloud. With cloud bursting, sensitive data can be kept and processed in the private system, and less security sensitive workloads may be offloaded to the public cloud.

In this paper, we present a new cloud bursting algorithm, (Cloud Bursting Scheduler, or CBS), which dynamically dispatches workloads/jobs across private system and public cloud aiming to minimize the cost of running those jobs. We primarily consider energy costs for the private system as they continue to increase, account for a major portion of operating costs of private system, and in many cases, surpass overall hardware costs [4], [5]. In the meantime, hourly rental costs are considered for public cloud usage. CBS attempts to optimize the cost to performance ratio by offloading jobs to public cloud explicitly taking into account different electricity rates with the private system and the fixed rental rate of the public cloud.

To evaluate CBS, we have run an extensive set of simulations using workload traces of real systems [6], [7]. Based on simulation results, CBS saves costs of running workloads by 55% and 12% compared with costs of cloud sourcing and private cloud, respectively. It also improves resource utilization (89%) by judiciously activating/leasing resources.

The rest of this paper is organized as follows. Section II discusses related work. Section III gives the description of target system and workloads. Section IV presents the cloud bursting scheduler (CBS) preceded by its cost model. Results are then presented in Section V followed by our conclusion in Section VI.

II. Related Work

The adoption of cloud computing has fueled by ever increasing electricity prices and excessive power consumption of computer systems. Traditionally, server providers and users tend to improve service performance without (seriously) considering energy costs. However, in recent years, as cost of hardware constantly decreases while energy prices and server

[1]In this paper, since we are particularly interested in Infrastructure as a Service (IaaS) clouds, resources refer to virtualized servers or virtual machines (VMs) unless stated otherwise.

[2]A list of case studies with Amazon EC2 in particular can be found in http://aws.amazon.com/solutions/case-studies/.

2159-6190/17 $31.00 © 2017 IEEE
DOI 10.1109/CLOUD.2017.112

power draw increase dramatically, cloud bursting is getting much attention. Besides, resource utilization in most (private) computer systems (data centers) is 10-20% and idle servers still consume 50% of peak power [8], [9].

In the recent past, there have been a number of studies conducted on data center energy efficiency and cost efficiency of cloud computing including efforts on developing cloud bursting solutions, more precisely, scheduling and resource allocation algorithms in hybrid clouds [10], [11], [12], [13]. Scheduling is inherently hard, and yet it becomes even harder in clouds due to resource heterogeneity and dynamism coupled with the diversity of workloads. In principle, scheduling and resource allocation in computer systems can be described as the process of identifying the best task-resource matches in time and space based on a given objective function without violating constraints; this is an important and but computationally intractable problem (i.e., NP-hard) [14].

The majority of previous attempts typically aim to minimize either makespan or cost, while the performance improvement and cost tradeoff is still not thoroughly studied, particularly when dealing with public clouds. Since the use of public cloud resources is associated with explicit costs and performance improvement might not always be proportional to those costs, the allocation of public cloud resources to workloads/jobs should explicitly take into account the cost to performance improvement ratio. The ExPERT framework in [12] is a solution for dynamic online selection of a Pareto-efficient scheduling strategy. Our recent work [13] further investigates the Pareto-optimality of cloud bursting for parallel scientific applications, so-called bag-of-tasks applications.

Calheiros et. al in [10] propose a resource management system for cloud bursting. This system adopts dynamic provisioning and scheduling of cloud resources to minimize cost while respecting job deadlines. The work in [11] proposes a comparison way to understand when outsourcing of applications to cloud is tenable from solely cost-centric perspective.

Our work in this paper is different from these previous studies in the following ways: (1) we explicitly consider different electricity rates with the private system in the cost model, (2) power consumption gradient with resource utilization is considered, and (3) traces of real-world workloads are used for the evaluation.

III. Preliminaries

A. Target System

We consider that there are exactly k different resource types in the public cloud and their corresponding speed and cost/rate are given by $\langle s_1, s_2, \cdots s_k \rangle$ and $\langle r_1, r_2, \cdots r_k \rangle$, respectively. The cost of renting a resource (irrespective of resource type) is typically charged based on usage, more precisely amount of unit time. We adopt the hour in this study as many public (IaaS) cloud providers including Amazon EC2 employ machine/instance hour. For the sake of simplicity, the private cloud consists of homogeneous resources (one type) with the speed of $\langle s_v \rangle$ and cost of $\langle r_v \rangle$. Each resource whether it is in private cloud or public cloud is associated with the number of processors (or processor cores), the size of memory (RAM) and the amount of disk storage.

Type	Duration	Price
Peak	2pm - 8pm	51.04627 c/kwh
Shoulder	7am - 2pm and 8pm - 10pm	24.44365 c/kwh
Off-peak	10pm - 7am	12.08196 c/kwh

TABLE I: Energy Australia: Energy Price Fact Sheet NSW Business (Electricity) [16].

Instance type	Rate ($)	Processing capacity (#vCPUs)	Memory (GiB)
m3.medium	0.067	1	3.75
m3.large	0.133	2	7.5
m3.xlarge	0.266	4	15
m3.2xlarge	0.532	8	30
c3.xlarge	0.21	4	7.5
c3.2xlarge	0.42	8	15
m4.2xlarge	0.431	8	32
m4.4xlarge	0.862	16	64
m4.10xlarge	2.155	40	160
c4.4xlarge	0.84	16	30
c4.8xlarge	1.68	32	60

TABLE II: Amazon EC2 prices for On-demand Linux instances based on US-East North Virginia Zone.

B. Workloads

Workloads/jobs considered in this study are compute-intensive and independent from each other, i.e., no inter-job communication or dependencies. As a result, inter-cloud data transfer cost is negligible. Job j_i is associated with a 4-tuple consisting of arrival time a_i, the number of processors required pc_i, the amount of memory required m_i and deadline d_i. We assume that the task's profile is available and can be provided by the user using job profiling, analytical models or historical information [15].

IV. CBS: Cloud Bursting Scheduler

In this section, we begin by describing the cost model used in this study and present a job scheduling algorithm, which is the core part of CBS solution.

A. Cost Model

The goal of cloud bursting in this paper is to minimize the total cost of running jobs across private and public clouds. As stated earlier, the dominant cost factor in the private cloud is energy cost explicitly considering "partly" used resources and that in the public cloud is the unit of charge, i.e., the hour in this study as in many public clouds including Amazon EC2. By partly used we mean that a resource/server with at least one processor core is active since resources in our study are all multi-core servers. The cost of a completely idle resource—that with none of its cores are active—is negligible since it can be put into a deeper sleep mode (e.g., C4 state in Intel processors) for which the power draw is near zero. Note that we do not consider per-core level dynamic voltage and frequency scaling in this study; hence, idle cores in partly used resources are assumed to consume 50% of peak power.

For a job j_i that requires pc_i processor cores[3], the cost of running a job j_i in the private cloud $c_i^{private}$ is the summation

[3]For the sake of simplicity, we assume that each resource is equipped with sufficient memory in the sense that the memory requirement of any job never exceeds the available memory as it is often the case in practice nowadays.

Algorithm 1: CBS job scheduling

Data: job j_i, used private resource set $P_{used}^{private}$, used public resource set P_{used}^{public}
Result: job schedule

```
1  begin
2      if #spare cores in private cloud ≤ pc_i then
3          calculate c_i^private
4      end
5      else
6          c_i^private ← ∞
7      end
8      calculate c_i^public
9      calcuate c_idle^private
10     c_actual^public ← c_i^public + c_idle^private
11     if c_i^private < c_actual^public then
12         dispatch j_i to public cloud
13         update P_used^public
14     end
15     else
16         dispatch j_i to private cloud
17         update P_used^private
18     end
19     P_idle^public ← idle resources in P_used^public
20     terminate P_idle^public
21 end
```

of cost of active resource (already being used) usage c_i^{used} and that of new resource usage c_i^{new}, each of which is a function of energy consumption and electricity rate. More formally,

$$c_i^{private} = \alpha + c_i^{used} + c_i^{new} \tag{1}$$

$$c_i^{used} = \sum_{k \in \Gamma} f(e_{core} * pc_i^{used} * t_i^k, k) \tag{2}$$

$$c_i^{new} = \sum_{k \in \Gamma} f(e_{min} + e_{core} * pc_i^{new} * t_i^k, k) \tag{3}$$

where α is a coefficient for some basic fixed costs including staffing and amortized server costs, Γ is a set of electricity rates (e.g., Table I), e_{core} and e_{min} are per-core energy consumption in watts and the minimum/base power when a resource is active (and yet running any job), respectively, pc_i^{used} and pc_i^{new} are the number of cores used in already-being-used resources and that in newly active resources, respectively, and t_i^k is the amount of time j_i is run during the time period of electricity rate k.

In the meantime, the cost of running a job j_i in the public cloud c_i^{public} is calculated based on charges for new resources (instances in Amazon EC2 terminology, see Table II for instance types and their prices) and charges for additional usage (in hours) of already-rented/existing resources. For example, if a job running on two resources—one being an existing resource and the other being a newly rented resource—only incurs the usage cost of the new resource unless the existing resource exceeds the current charge period in hours.

Note that although a single job may require multiple cores, it runs only in either private cloud or public cloud, but not across both.

B. Job Scheduling

In essence, CBS schedules a job (Algorithm 1) with the most cost effective allocation of resources considering three main factors: (1) cost of under-utilized (or idle) resources, (2) different electricity rates and (3) job deadline. Specifically, for a given partly used resource, the number of spare cores is identified accounting for finish times of currently running jobs and the deadline of a job being scheduled; such identification also applies to public cloud resources.

When a job j_i is considered for running on a public cloud and there are one or more private resources are partly used, the cost of running j_i is calculated accounting for the cost of idle cores in private resources (lines 7 – 8) since these cores are wasted. However, no extra costs incur with the opposite case (idling some cores of public cloud resources) since those (existing) public cloud resources within the current hour have been already charged either when they were initially rented or when previous jobs were/are using those resources.

At the end of each hour, public cloud resources are check if there are any jobs running or waiting. Public cloud resources are terminated (line 16) if they have no jobs to run.

V. Performance Evaluation

In this section, we describe our evaluation of CBS.

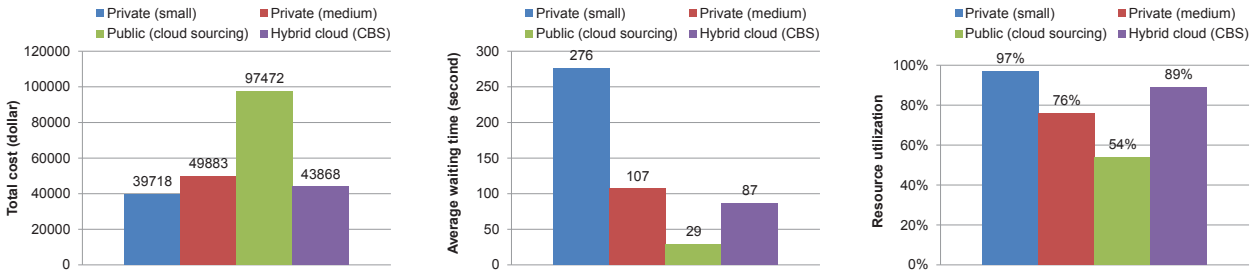
A. Simulation Setup

The private cloud is set up in two different scales: (1) small with 15 resources/nodes, and (2) medium with 32 resources; each resource in either setup contains 8 cores. In the meantime, the public cloud consists of 11 different resource/instance types as shown in Table II. The maximum number of resources for each of those 11 types is limited 20 as in the case with Amazon EC2 unless a request is made.

Jobs in our simulations come from workload traces—that are maintained and made available by the Experimental Systems Lab at The Hebrew University [6]. In particular, we have used *DAS2-fs0-2003-1* trace log (or simply *fs0*) with 225,711 jobs and the total time period of 365 days for the job distribution of fs0). Among 18 log elements in the trace, we use five: job ID, submission time, requested time (deadline), requested processor count and requested RAM size.

B. Results

Evaluation results are presented using three metrics: total cost, waiting time, and resource utilization (Figure 1). In essence, for any scenario, the total cost is explained by average waiting time and resource utilization. Since jobs in our experiments are associated with deadlines that mostly give sufficient time cushion compared with the actual runtime, jobs may put on hold (waiting time) as long as they are deemed to meet their deadline. As a result, resource utilization is improved. In other words, jobs wait for some resources already being used to be available, i.e., finishing running jobs; this would avoid or alleviate the excessive use of resources.



(a) Total costs. (b) Average waiting time. (c) Average utilization.

Fig. 1: Performance results. The total cost with the private cloud with 15 nodes (small) is the smallest only because 3393 jobs passed their deadlines.

Clearly, CBS outperforms other solutions by a substantial margin. In particular, the total cost is reduced by 12% and 55% compared with those of private (medium) and public cloud solutions, respectively. Although the total cost of the private (small) solution is lower than that of CBS, it is inconsiderate of job deadline as 3393 jobs violate deadline constraint; this is due to the lack of resource capacity.

As CBS only activates (new) resources when neither private cloud resources nor public cloud resources can complete jobs within their deadlines. For this reason, the average resource utilization is significantly improved with CBS (89%) compared with the private (medium) cloud only solution. Note that the poor resource utilization of the public cloud only (cloud bursting) solution (54%) is due to relentless use of new resources; although it reduces waiting time (29 seconds on average, Figure 1b), such reduction is not essential since most jobs can afford some waiting time within their deadlines.

VI. Conclusion

In this paper, we have studied cloud bursting in comparison with traditional in-house systems and public clouds. As cloud computing and public clouds are increasingly adopted for cost effectively dealing with ever increasing computing and data processing needs, the efficient resource usage of public clouds in addition to existing private cloud resources must be sought. Our cloud bursting scheduler (CBS) makes judicious decisions on scheduling and resource allocation explicitly taking into account different electricity rates over time for private clouds and the hourly pricing of public clouds. Based on our results obtained from extensive simulations using a workload trace, CBS confidently demonstrates its efficacy in cost efficiency, performance and resource utilization.

We plan to investigate the security/privacy issues in the context of cloud bursting.

Acknowledgment

Dr. Young Choon Lee's work is supported by ARC Linkage Grant LP140100980.

References

[1] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011, pp. 49:1–49:12.

[2] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ser. ICDCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 559–570.

[3] Y. C. Lee and A. Y. Zomaya, "Stretch Out and Compact: Workflow Scheduling with Resource Abundance," in *Proceedings of the International Symposium on Cluster Cloud and the Grid (CCGRID)*, May 2013, pp. 219–226.

[4] L. A. Barroso, "The price of performance," *Queue*, vol. 3, no. 7, pp. 48–53, 2005.

[5] J. Koomey, "Growth in data center electricity use 2005 to 2010," August 2011.

[6] "Logs of real parallel workloads from production systems," http://www.cs.huji.ac.il/labs/parallel/workload/logs.html, 2014.

[7] H. Li, D. Groep, and L. Wolters, "Workload characteristics of a multi-cluster supercomputer," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., vol. 3277. Springer Berlin Heidelberg, 2005, pp. 176–193.

[8] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[9] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[10] R. N. Calheiros and R. Buyya, "Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds," in *Proceedings of the 13th International Conference on Web Information Systems Engineering*, ser. WISE'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 171–184.

[11] Y. Chen and R. Sion, "To cloud or not to cloud?: Musings on costs and viability," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 29:1–29:7.

[12] O. A. Bar-Yehuda, M. Silberstein, A. Sharov, A. Iosup, and A. Schuster, "Expert: Pareto-efficient task replication on grids and a cloud," in *International Parallel and Distributed Processing Symposium, IPDPS12*, China, 2012.

[13] M. R. HoseinyFarahabady, Y. C. Lee, and A. Y. Zomaya, "Pareto-optimal cloud bursting," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2670–2682.

[14] *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[15] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou, "Profiling applications for virtual machine placement in clouds," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, ser. CLOUD '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 660–667.

[16] "Energy australia: Energy price fact sheet nsw business (electricity)," https://secure.energyaustralia.com.au/EnergyPriceFactSheets/Docs/EPFS/E_B_N_BBAS_EA_6_02-01-2017.pdf, 2017.