Computer Networks 94 (2016) 231–249

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Auction-based cloud service differentiation with service level objectives

Jianbing Ding [a,b,*], Zhenjie Zhang [c], Richard T. B. Ma [d], Yin Yang [e]

[a] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[b] SYSU-CMU Shunde International Joint Research Institute, Shunde, China
[c] Advanced Digital Sciences Center, Illinois at Singapore Pte. Ltd., Singapore
[d] School of Computing, National University of Singapore, Singapore
[e] College of Science and Engineering, Hamad Bin Khalifa University, Qatar

ARTICLE INFO

Article history:
Received 28 April 2015
Revised 14 August 2015
Accepted 4 November 2015
Available online 10 November 2015

Keywords:
Cloud computing
Service differentiation
Auction
MapReduce

ABSTRACT

The emergence of the cloud computing paradigm has greatly enabled innovative service models, such as Platform as a Service (PaaS), and distributed computing frameworks, such as MapReduce. However, most existing cloud systems fail to distinguish users with different preferences, or jobs of different natures. Consequently, they are unable to provide *service differentiation*, leading to inefficient allocations of cloud resources. Moreover, contentions on the resources exacerbate this inefficiency, when prioritizing crucial jobs is necessary, but impossible. Motivated by this, we propose *Abacus*, a generic resource management framework addressing this problem. Abacus interacts with users through an *auction mechanism*, which allows users to specify their priorities using *budgets*, and job characteristics via *utility functions*. Based on this information, Abacus computes the optimal allocation and scheduling of resources. Meanwhile, the auction mechanism in Abacus possesses important properties including incentive compatibility (i.e., the users' best strategy is to simply bid their true budgets and job utilities) and monotonicity (i.e., users are motivated to increase their budgets in order to receive better services). In addition, when the user is unclear about her utility function, Abacus automatically learns this function based on statistics of her previous jobs. Extensive experiments, running Hadoop on a private cluster and Amazon EC2, demonstrate the high performance and other desirable properties of Abacus.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Cloud computing is a cost-effective paradigm for both the cloud providers and the users. The providers benefit by effectively multiplexing dynamic user demands of various computing resources, e.g., CPU, storage, bandwidth, etc., through virtualization techniques. Meanwhile, the users are liberated from large capital outlays in hardware deployment and maintenance. Successful cloud models include Infrastructure as a Service (e.g., Amazon's EC2), and data-intensive distributed computing paradigm (e.g., Map-Reduce), both of which are well adopted for a wide spectrum of web services and data management tasks [13,15].

Cloud systems can be categorized into private clouds, which are used exclusively by one organization, and public ones, which rent out their computational capacities to customers. For private clouds, one of the most important objectives is *efficiency*, meaning that the overall utility derived from the jobs executed with the limited cloud resources. Although public clouds might put profitability before efficiency, both objectives are often aligned and achieved by serving the most valued jobs under resource competition. In order to maximize efficiency, jobs should be

* Corresponding author. Tel.: +86 18925190471.
E-mail addresses: dingsword@gmail.com (J. Ding), zhenjie@adsc.com.sg (Z. Zhang), tbma@comp.nus.edu.sg (R. T. B. Ma), yyang@qf.org.qa (Y. Yang).

http://dx.doi.org/10.1016/j.comnet.2015.11.007
1389-1286/© 2015 Elsevier B.V. All rights reserved.

differentiated based on their characteristics, including the utilities they generate and the distinct resources they require. For instance, computation-intensive jobs may need powerful CPUs more than other resources, whereas bandwidth is often the most important resource for delay-sensitive applications. Intuitively, resources should be allocated to jobs of high importance, and to jobs that need them the most. Existing cloud systems generally do not provide adequate capability for such service differentiation. In particular, private clouds mainly use simple resource allocation strategies, such as first-in-first-out and fair-share. Public clouds essentially differentiate users based on the amount of money they pay for each type of resources. For instance, Amazon EC2 bundles resources into virtual machines (VMs), and each type of VM has its unique configuration and unit-time price. Based on these prices and the status of their applications, the users decide by themselves the type and number of VM-hours to purchase. Moreover, such prices for computational resources in public clouds often fluctuate continuously, forcing users to monitor these prices and adjust their VM portfolios accordingly, if they want to maximize overall utility within budget limits.

While the above pricing scheme can be seen as a kind of manual service differentiation, to our knowledge, currently there is no solution for *automatic* service differentiation. Providing this functionality is challenging in several aspects. First, only users (possibly) know the utilities and resource usage patterns of their own jobs; hence, the cloud system needs an effective way to obtain this information from the users. Second, an *incentive compatible* mechanism is needed to prevent any user from misreporting information so as to increase its own utility, as this may hurt the performance of other jobs as well the overall utility of the entire cloud. Third, realizing an abstract service differentiation solution on a real cloud system is non-trivial, as the implementation must work seamlessly with the existing resource allocation and scheduling modules.

Facing these challenges, we propose a novel cloud resource allocation framework called *Abacus*, which enables service differentiation for clouds. Abacus acquires information on users' job characteristics through a novel auction mechanism. Resources are then dynamically allocated according to the auction results and the availability of each type of resources. Abacus' auction mechanism is *incentive-compatible*, meaning that every user's dominating strategy is to simply bid its true job characteristics. This also implies that the auction is *stable*, i.e., no user can benefit from unilateral bid changes. These properties ensure that as long as a user's job characteristics remain the same, there is no need to monitor the auction or to change bids. In this aspect, Abacus is much easier to use compared to the price-based service differentiation as in Amazon EC2. Further, the auction is *monotonic*, which guarantees fairness in the sense that users paying more for a type of resource are always allocated higher quantities of it. Finally, Abacus is *efficient*, which achieves high overall system utility under the above constraints, as confirmed by our experimental evaluations.

Abacus can be used in various cloud systems, including public and privacy ones, and clouds running on different platforms. To demonstrate the practical impact of Abacus, we implement it on top of Hadoop, and evaluate its effectiveness and efficiency using Map-Reduce workloads. To further improve the usability of Abacus, especially for cloud system users without clear knowledge on the utility model of their own *repeated* jobs, we extend our standard auction mechanism to enable users to submit bids with only budget information. After running the jobs for a number of rounds under default utility functions, the utility prediction component is capable of recovering the true utility function, using regression techniques. Experiments using a large-scale cluster confirm that Abacus successfully achieves high overall utility, high performance, and all the designed desirable properties.

The main contributions of the paper are listed below

- We present a new study on service differentiation techniques for general cloud system. Our solution potentially opens new business models for cloud systems in the future, and enables ordinary users to exploit the benefits of clouds.
- We propose *Abacus*, an auction based approach to cloud system resource allocation and scheduling, with enticing features such as *incentive-compatibility*, *system stability* and *system efficiency*.
- We simplify the auction procedure by allowing the users to skip the utility function when the user is unsure or unaware of the exact utility model of his own repeated jobs.
- We implement Abacus by modifying the scheduling algorithm in Hadoop, and test it on a large-scale cloud platform. Our experimental results verify the truthfulness of our auction-based mechanism, system efficiency, as well as the accuracy of our utility prediction algorithm.

A preliminary version of Abacus appears in [41]. The main difference between [41] and the full version is that the former mainly focuses on the Abacus model and its *theoretical properties*, whereas the latter also presents solutions and results that are crucial for applying Abacus *in practice*. These include (i) a novel algorithm to handle jobs with Service Level Objectives (SLOs), presented in Section 6, which enables Abacus to support jobs running on cloud systems with performance requirements, e.g. maximum response time, (ii) a large-scale experimental evaluation of Abacus on a public cloud (i.e., Amazon EC2) with real-world workloads, presented in Section 7.2, (iii) performance comparison between Abacus and ARIA [37], a state-of-the-art solution for SLO-based scheduling, also presented in Section 7.2. In addition, the full version provides detailed proofs for our theoretical results on the robustness and soundness of the proposed algorithms.

In the following, Section 2 reviews related work. Section 3 provides problem definition and assumptions. Section 4 details the auction mechanism. Section 5 discusses automatic optimization for users not knowing their own utility functions. Section 6 extends Abacus to handle jobs with Service Level Objective. Section 7 contains an extensive experimental evaluation. Finally, Section 8 concludes the paper.

2. Related work

2.1. Grid/Cloud resource allocation

Resource allocation [12,25] and scheduling [17,27] in distributed systems have been extensively studied in the

Download English Version:

https://daneshyari.com/en/article/452808

Download Persian Version:

https://daneshyari.com/article/452808

Daneshyari.com