Performance Evaluation 81 (2014) 20–39

Contents lists available at ScienceDirect

Performance Evaluation

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

Heavy traffic optimal resource allocation algorithms for cloud computing clusters

Siva Theja Maguluri [a,*], R. Srikant [a], Lei Ying [b]

[a] Department of ECE and CSL, University of Illinois at Urbana–Champaign, 1308 W Main Street, Urbana, IL 61801, USA
[b] School of ECEE, 436 Goldwater Center, Arizona State University, Tempe, AZ 85287, USA

ARTICLE INFO

Article history:
Received 11 February 2013
Received in revised form 6 August 2014
Accepted 19 August 2014
Available online 27 August 2014

Keywords:
Scheduling
Load balancing
Cloud computing
Resource allocation

ABSTRACT

Cloud computing is emerging as an important platform for business, personal and mobile computing applications. In this paper, we study a stochastic model of cloud computing, where jobs arrive according to a stochastic process and request resources like CPU, memory and storage space. We consider a model where the resource allocation problem can be separated into a routing or load balancing problem and a scheduling problem. We study the join-the-shortest-queue routing and power-of-two-choices routing algorithms with the MaxWeight scheduling algorithm. It was known that these algorithms are throughput optimal. In this paper, we show that these algorithms are queue length optimal in the heavy traffic limit.

1. Introduction

Cloud computing services are emerging as an important resource for personal as well as commercial computing applications. Several cloud computing systems are now commercially available, including Amazon EC2 system [1], Google's AppEngine [2] and Microsoft's Azure [3]. A comprehensive survey on cloud computing can be found in [4–6].

In this paper, we focus on cloud computing platforms that provide infrastructure as service. Users submit requests for resources in the form of virtual machines (VMs). Each request specifies the amount of resources it needs in terms of processor power, memory, storage space, etc. We call these requests jobs. The cloud service provider first queues these requests and then schedules them on physical machines called servers.

Each server has a limited amount of resources of each kind. This limits the number and types of jobs that can be scheduled on a server. The set of jobs of each type that can be scheduled simultaneously at a server is called a configuration. The convex hull of the possible configurations at a server is the capacity region of the server. The total capacity region of the cloud is then the Minkowski sum of the capacity regions of all servers.

The simplest architecture for serving the jobs is to queue them at a central location. In each time slot, a central scheduler chooses the configuration at each server and allocates jobs to the servers, in a preemptive manner. As pointed out in [7], this problem is then identical to scheduling in an ad hoc wireless network with interference constraints. In practice, however, jobs are routed to servers upon arrival. Thus, queues are maintained at each individual server. It was shown in [7–9] that using join-the-shortest queue-type algorithms for routing, along with the MaxWeight scheduling algorithm [10] at each server is throughput optimal. The focus of this paper is to study the delay, or equivalently, the queue length performance of the algorithms presented in [7].

* Corresponding author. Tel.: +1 2174171631.
E-mail addresses: siva.theja@gmail.com (S.T. Maguluri), rsrikant@illinois.edu (R. Srikant), lei.ying.2@asu.edu (L. Ying).

http://dx.doi.org/10.1016/j.peva.2014.08.002
0166-5316/

Characterizing the exact delay or queue length in general is difficult. So, we study the system in the heavy-traffic regime, i.e., when the exogenous arrival rate is close to the boundary of the capacity region. We say that an algorithm is heavy traffic optimal if it minimizes $\lim_{\epsilon\to 0}\epsilon\mathbb{E}[f(\mathbf{q})]$ where ϵ is the distance of the arrival rate vector from the boundary of the capacity region, $\mathbf{q}$ is the queue length vector and $f(.)$ is a function which we will clearly define later.

In the heavy-traffic regime, for some systems, the multi-dimensional state of the system reduces to a single dimension, called state-space collapse. In [11,12], a method was outlined to use the state-space collapse for studying the diffusion limits of several queuing systems. This procedure has been successfully applied to a variety of multiqueue models served by multiple servers [13–16]. Stolyar [17] generalized this notion of state-space collapse and resource pooling to a generalized switch model, where it is hard to define work-conserving policies. This was used to establish the heavy traffic optimality of the MaxWeight algorithm.

Most of these results are based on considering a scaled version of queue lengths and time, which converges to a regulated Brownian motion, and then show sample-path optimality in the scaled time over a finite time interval. This then allows a natural conjecture about steady state distribution. In [18], the authors present an alternate method to prove heavy traffic optimality that is not only simpler, but shows heavy traffic optimality in unscaled time. In addition, this method directly obtains heavy traffic optimality in steady state. The method consists of the following three steps.

1. *Lower bound:* First a lower bound is obtained on the weighted sum of expected queue lengths by comparing with a single-server queue. A lower bound for the single-server queue, similar to the Kingman bound [19], then gives a lower bound to the original system. This lower bound is a universal lower bound satisfied by any joint routing and scheduling algorithm.
2. *State-space collapse*: The second step is to show that the state of the system collapses to a single dimension. Here, it is not a complete state-space collapse, as in the Brownian limit approach, but an approximate one. In particular, this step is to show that the queue length along a certain direction increases as the exogenous arrival rate gets closer to the boundary of the capacity region but the queue length in any perpendicular direction is bounded.
3. *Upper bound*: The state-space collapse is then used to obtain an upper bound on the weighted queue length. This is obtained by using a natural Lyapunov function suggested by the resource pooling. Heavy traffic optimality can be obtained if the upper bound coincides with the lower bound.

In this paper, we apply the above three step procedure to study the resource allocation algorithms presented in [7]. We briefly review the results in [7] now. Jobs are first routed to the servers, and are then queued at the servers, and a scheduler schedules jobs at each server. So, we need an algorithm that has two components, viz.,

1. a *routing algorithm* that routes new jobs to servers in each time slot (we assume that the jobs are assigned to a server upon arrival and they cannot be moved to a different server) and
2. a *scheduling algorithm* that chooses the configuration of each server, i.e., in each time slot, it decides which jobs to serve. Here we assume that jobs can be preempted, i.e., a job can be served in a time slot, and then be preempted if it is not scheduled in the next time slot. Its service can be resumed in the next time it is scheduled. Such a model is applicable in situations where job sizes are typically large.

It was shown in [7] that using the join-the-shortest-queue (JSQ) routing and MaxWeight scheduling algorithm is throughput optimal. In Section 3, we show that this policy is queue length optimal in the heavy traffic limit when all the servers are identical. We use the three step procedure described above to prove the heavy traffic optimality. The lower bound in this case is identical to the case of the MaxWeight scheduling problem. However, state-space collapse does not directly follow from the corresponding results for the MaxWeight algorithm in [18] due to the additional routing step here. We use this to obtain an upper bound that coincides with the lower bound in the heavy traffic limit.

JSQ needs queue length information of all servers at the router. In practice, this communication overhead can be quite significant when the number of servers is large. An alternative algorithm is the power-of-two-choices routing algorithm. In each time slot, two servers are chosen uniformly at random and new arrivals are routed to the server with the shorter queue. It was shown in [7] that the power-of-two-choices routing algorithm with the MaxWeight scheduling is throughput optimal if all the servers are identical. Here, we show that the heavy traffic optimality in this case is a minor modification of the corresponding result for JSQ routing and MaxWeight scheduling.

A special case of the resource allocation problem is when all the jobs are of the same type. In this case, scheduling is not required at each server. The problem reduces to a routing-only problem which is well studied [20–24]. For reasons to be explained later, the results from Section 3 cannot be applied in this case since the capacity region is along a single dimension (of the form $\lambda < \mu$). In Section 4, we show the heavy traffic optimality of the power-of-two-choices routing algorithm. The lower and upper bounds in this case are identical to the case of JSQ routing in [18]. The main contribution here is to show state-space collapse, which is somewhat different compared to [18]. The results here complement the heavy traffic optimality results in [22,23] which were obtained using Brownian motion limits.

We note that this paper is a longer version of [25]. In [25], certain details were omitted in the proofs in Sections 3.2, 3.3 and 4.3 due to space limitations. Here we provide the missing proofs.

Note on Notation: The set of real numbers, the set of nonnegative real numbers and the set of positive real numbers are denoted by $\mathbb{R}$, $\mathbb{R}_+$ and $\mathbb{R}_{++}$ respectively. We denote vectors in $\mathbb{R}^J$ or $\mathbb{R}^M$ by x, in normal font. We use bold font $\mathbf{x}$ to denote vectors in $\mathbb{R}^{JM}$. The dot product in the vector spaces $\mathbb{R}^J$ or $\mathbb{R}^M$ is denoted by $\langle x, y\rangle$ and the dot product in $\mathbb{R}^{JM}$ is denoted by $\langle \mathbf{x}, \mathbf{y}\rangle$.