Simulation Modelling Practice and Theory 44 (2014) 63–77

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

Load balancing for cluster systems under heavy-tailed and temporal dependent workloads

Jianzhe Tai *, Zhen Li, Jiahui Chen, Ningfang Mi

Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, United States

A R T I C L E   I N F O

Article history:
Received 11 December 2013
Received in revised form 14 March 2014
Accepted 15 March 2014
Available online 14 April 2014

Keywords:
Resource allocation
Temporal dependence
Heavy tailed workloads
Size-based load balancing
Cluster systems

A B S T R A C T

Large-scaled cluster systems have been employed in various areas by offering pools of fundamental resources. Efficient allocation of the shared resources in a cluster system is a critical but challenging issue, which has been extensively studied in the past few years. Despite the fact that existing load balancing policies, such as Random, Join Shortest Queue and size-based polices, are widely implemented in actual systems due to their simplicity and efficiency, the performance benefits of these policies diminish when workloads are highly variable and temporally correlated. In this paper, we propose a new load balancing policy, named ADuS, which attempts to partition jobs according to their present sizes and further rank the servers based on their loads. By dispatching jobs of similar sizes to the corresponding ranked servers, ADuS can adaptively balance user traffic and system load in a cluster and thus achieve significant performance benefits. Extensive trace-driven simulations using both synthetic and real traces show the effectiveness and robustness of ADuS under many different environments.

1. Introduction

Large-scale cluster computing environments are being employed in an increasing number of application areas these days. Examples of these systems include High Performance Computing (HPC), enterprise information systems, data centers, cloud computing and cluster servers. In particular, a cluster environment provides shared resources as an unified hosting pool, which requires a central mechanism for resource provisioning and resource allocation based on the demands of multiple remote clients [1,2]. A lot of research works have been carried out on the study of multi-server clusters with a single system image, i.e., a set of homogeneous hosts behave as a single resource pool [3–5]. In such a multi-server cluster, a front-end dispatcher plays a roll in distributing incoming jobs among those back-end servers. Such a front-end dispatcher is featured as a redirect buffer without central waiting queue while each back-end server has its own queue for waiting jobs and a separate processor that operates under the first-come first-serve (FCFS) queuing discipline. In this paper, we focus on the load balancing design for a front-end dispatcher in such a multi-server cluster system.

A lot of previous studies have been focusing on developing load balancing policies for a large-scale cluster computing system over the past decades [3–7]. Examples of these policies include Join Shortest Queue (JSQ) and the size-based AdaptLoad [8]. JSQ has been proven to be optimal [9] for a cluster with homogeneous servers, when there is not a priori knowledge of job sizes and the job sizes are exponentially distributed. However, [10] evaluated the performance of JSQ under various

* Corresponding author. Tel.: +1 617 373 3028.
E-mail addresses: jtai@ece.neu.edu (J. Tai), zli@ece.neu.edu (Z. Li), jchen@ece.neu.edu (J. Chen), ningfang@ece.neu.edu (N. Mi).

http://dx.doi.org/10.1016/j.simpat.2014.03.006
1569-190X/© 2014 Elsevier B.V. All rights reserved.

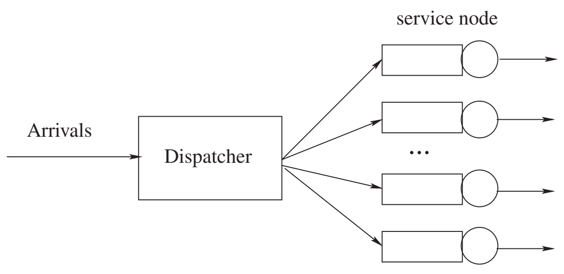

Fig. 1. The model of a cluster system with N service nodes.

workloads by measuring the mean response times and found that the performance of JSQ clearly varies with the characteristics of different service time distributions. The optimality of JSQ quickly disappears when job service times are highly variable and heavy-tailed [11–13].

Recently, size-based policies were proposed to balance the load in a cluster system, only using the knowledge of the incoming job sizes. The literatures in [14,15,6,12,8] have shown that such size-based policies can be deployed in cluster systems to achieve the minimum job response times and job slowdowns. The AdaptLoad policy being a representative example of size-based policies, has been developed to improve average job response time and average job slowdown by on-the-fly building the histogram of job sizes and distributing jobs of similar sizes to the same server [8]. However, such size-based solutions are not adequate if the job service times are temporally dependent[1] [4].

Motivated by the limitations of the existing load balancing policies, we propose a new load balancing policy, named ADuS, which adaptively distributes work among all servers by taking account of both *user traffic* and *system load*. In detail, ADuS ranks all servers based on their present system loads and updates job size boundaries on-the-fly, allowing the dispatcher to direct jobs of similar sizes to the servers which have the same ranking. We expect that our new policy can improve the overall system performance by inheriting the effectiveness of both JSQ and AdaptLoad and meanwhile overcoming the limitations of these two policies. Trace-driven simulations are used to evaluate the performance of ADuS. Extensive experimental results indicate that ADuS significantly improves the system performance, e.g., job response times and job slowdowns, under heavy-tailed and temporal dependent workloads. Sensitivity analysis with respect to system loads, variation and temporal dependence in job sizes, and the number of servers in the system demonstrates that ADuS is effective and robust in various environments. We also use the case study of real traces, e.g., TCP, BC-pOct89, WorldCup98, and Cambridge-hm_0 to further validate the benefits of ADuS in actual systems.

In summary, the main contributions of this paper include:

- We demonstrate that the existing loading balancing policies, e.g., JSQ, and size-based approaches, become ineffective when the workloads are heavy-tailed and bursty.
- We develop a new load balancing policy which distributes the workloads to the servers by taking account of both user traffic and system load.
- We conduct trace-driven simulations using both synthetic and real traces to evaluate the performance of our new policy.

We expect that our new load balancing policy can be adopted to balance the computational load across a set of instances of the same application in the cloud. For example, in Amazon EC2, when an application receives multiple incoming requests, these requests should be each assigned to one of its EC2 instances. Elastic Load Balancing (ELB) has been used in Amazon EC2 to redirect all the incoming application requests across multiple EC2 instances of the same application. In this case, ADuS can be implemented in Amazon EC2 to replace ELB for balancing an application's load across this application's EC2 instances.

The rest of the paper is organized as follows. Section 2 gives an overview of load balancing in cluster systems. Section 3 discusses the limitations of the existing load balancing policies under highly variable and/or heavily dependent workloads. Section 4 presents a new load balancing policy which aims to distribute the jobs among all servers by taking account of both user traffic and system load. Section 5 uses trace-driven simulations to evaluate the performance improvement of the proposed policy. Related work is presented in Section 6 and conclusions and future work are summarized in Section 7.

2. Background

A cluster system contains a group of homogeneous or heterogeneous service nodes, which are linked together and behave as a single virtual server in many respects. Either a centralized front-end dispatcher or a decentralized decision making algorithm is responsible for distributing incoming requests among server nodes. The performance of a cluster system is typically much more cost-effective than that of a single computer with comparable speed or availability. Fig. 1 illustrates the model of a simple cluster system where a dispatcher is responsible to forward the incoming jobs (or requests) to an appropriate server

[1] Temporally dependent or bursty workloads are often found in multi-tier architectures, large storage systems, and grid services [16,17].