Big Data Research 2 (2015) 110–116

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Multi-Tier Resource Allocation for Data-Intensive Computing ☆

Thomas Ryan [a], Young Choon Lee [b,*]

[a] School of Information Technologies, The University of Sydney, NSW, 2006, Australia
[b] Department of Computing, Macquarie University, NSW, 2109, Australia

ARTICLE INFO

Article history:
Received 22 December 2014
Received in revised form 9 February 2015
Accepted 2 March 2015
Available online 23 March 2015

Keywords:
Resource allocation
Data-intensive computing
Cloud computing
Big data
Application heterogeneity
MapReduce

ABSTRACT

As distributed computing systems are used more widely, driven by trends such as 'big data' and cloud computing, they are being used for an increasingly wide range of applications. With this massive increase in application heterogeneity, the ability to have a general purpose resource management technique that performs well in heterogeneous environments is becoming increasingly important.

In this paper, we present Multi-Tier Resource Allocation (MTRA) as a novel fine-grained resource management technique for distributed systems. The core idea is based on allocating resources to individual tasks in a tiered or layered approach. To account for heterogeneity, we propose a dynamic resource allocation method that adjusts resource allocations to individual tasks on a cluster node based on resource utilisation levels. We demonstrate the efficacy of this technique in a data-intensive computing environment, MapReduce data processing framework in Hadoop YARN. Our results demonstrate that MTRA is an effective general purpose resource management technique particularly for data-intensive computing environments. On a range of MapReduce benchmarks in a Hadoop YARN environment, our MTRA technique improves performance by up to 18%. In a Facebook workload model it improves job execution times by 10% on average, and up to 56% for individual jobs.

1. Introduction

As the scale and size of applications continues to increase with the explosive growth in data volume (dubbed 'Big Data'), distributed processing/computing has become rather essential. The availability of virtually unlimited resource capacity with cloud computing has greatly enabled such distributed computing. As a result, a wide range of distributed systems has been developed. Some popular examples are MapReduce data processing framework [1], Pregel graph processing system [2] and Montage astronomical image mosaic engine [3,4]. Applications in these distributed systems exhibit much heterogeneity in terms particularly of resource usage characteristics, e.g., CPU-intensive applications and data-intensive applications.

Within distributed systems, resource allocation and management is an ongoing research concern; it is well known that improving resource allocation mechanisms can result in considerable real-world improvements in performance and efficiency. However, different applications have different resource requirements, meaning that effective resource management in heterogeneous environments is much more difficult. While there has been a large amount of research into resource allocation, most resource allocation strategies do not account for application heterogeneity. In particular, the majority of classic optimisation techniques for resource management assume homogeneity in either application or resource, or both. And thus, they often do not perform optimally in heterogeneous environments.

In this paper, we address the problem of fine-grained resource allocation to distributed systems where jobs are composed of many small tasks taking into account application heterogeneity. The traditional approach to resource management in these systems has been to divide resources into logical partitions (called 'slots' or 'containers'), and allocate tasks to partitions [1,5]. However, since different jobs have different resource usage characteristics, this approach can lead to both resource under-utilisation and resource contention, resulting in decreased performance. A general purpose, scheduler-independent solution to this problem is highly desirable, especially as we see increasing heterogeneity of both applications and execution environments.

To this end, we develop Multi-Tier Resource Allocation (MTRA) as a novel resource management technique that dynamically adjusts resource allocations to heterogeneous, individual tasks in distributed systems. The dynamic adjustments are based on the

☆ This article belongs to BDA-HPC.

* Corresponding author.
E-mail addresses: trya3473@uni.sydney.edu.au (T. Ryan), young.lee@mq.edu.au (Y.C. Lee).

http://dx.doi.org/10.1016/j.bdr.2015.03.001
2214-5796/

resource requirements of each task as well as current levels of resource utilisation and resource contention on each node in a given distributed computing system, or simply cluster. The idea is that a common set of resources can be dynamically multiplexed in several fine-grained resource allocation tiers to enable multiple tasks with different resource usage characteristics to "harmoniously" share the resource set. Rather than relying on a centralised scheduler, resource allocations are adjusted locally on each node, allowing for very fine-grain control. This distributed resource management approach can be combined with any existing scheduler. It also decreases scheduler complexity, allowing for greater scalability. Note that our MTRA is a resource management technique underneath any schedulers that deal with the globalisation of the resource allocation. The current implementation of MTRA primarily deals with CPU and IO resources although it potentially capable of dealing with other resources, such as network resources.

Our main focus is on MapReduce applications within a Hadoop YARN environment. Resource management strategies in Hadoop are based on the idea of logical partitions of resources. Traditionally, these have been statically configured, a process which is itself an ongoing research issue. Since different tasks require different amounts and kinds of resources, configuration in heterogeneous environments is even more problematic. Our preliminary solution to this problem is Local Resource Shaper (LRS) [6] that enables local resources (CPU core and disks) to be shared primarily by two tiers allowing dynamic movement of tasks between resource allocation tiers. While this model is shown to improve resource utilisation while minimising resource contention in Classic Hadoop, we show that it does not account well for task heterogeneity. With regard to both application and environment heterogeneity, MTRA enables optimal performance by adjusting allocations to maximise resource utilisation and minimise resource contention for the resource requirements of each task.

Our evaluation shows that by accounting for heterogeneity, our MTRA technique improves system performance. MTRA is comparable or better than the previous best resource management alternative for a range of MapReduce benchmarks with different resource usage characteristics, including outperforming the two-tier allocation model [6]. For individual MapReduce benchmarks, we observe performance improvements of up to 18% compared to Hadoop YARN. We show that for a workload model based on a Facebook production workload, MTRA reduces individual job execution times by 10% on average and up to 56% for individual jobs in the workload. These results prove that MTRA improves performance by a significant margin in real-world, heterogeneous environments.

The rest of this paper is organised as follows. Section 2 gives background. Section 3 presents our multi-tier resource allocation technique and describes the application of MTRA to MapReduce in Hadoop YARN. Section 4 presents our evaluation on the efficacy of MTRA. Section 5 discusses the related work. Finally, Section 6 concludes the paper.

2. Background

In this section, we begin by some background on big data and cloud computing, and then provide essential works and application used in this paper.

2.1. Big data and cloud computing

The recent trend of so-called 'Big Data' is expected to be rather norm as the volume of data exponentially increases literally in every area—business, science, and daily life to name a few. Today, some claim that data (more specifically, data-intensive science/computing) are the fourth paradigm in scientific research after experimentation/observation, theory, and computational simulation [7]. Efficient data storage and timely data processing is of great practical importance. Clearly, this requires both massive storage and processing capacity. While some data processing deals with simple retrieval or search (IO-intensive), other case involves much computation on data (mixture usage of IO and CPU resources); hence, application heterogeneity.

Cloud computing has emerged as a new computing paradigm with its strengths in elasticity and pay-as-you-go pricing, and it is a viable solution to many ICT services including big data processing. Cloud computing in this context refers to the Infrastructure-as-a-Service model, such as that provided by Amazon Elastic Compute Cloud (EC2) [8] or Google Compute Engine [9]. This model has many advantages, including elasticity, scalability, and using a pay-as-you-go pricing system [10]. All of these properties have implications for cluster environments, in terms of economics (only paying for what you need) and performance (such as the shared-tenant hardware and unpredictable datacentre loads). The use of public cloud providers is associated with performance overheads due to a range of factors. Individual nodes, or 'instances', are based in virtual machines (VMs) which do not necessarily correspond to physical machines. This representation of instances provides a lot of control, however information such as the network topology within the datacentre is unavailable; this has means we cannot guarantee our nodes will be provisioned on the same server or rack, leading to inconsistent communication times between nodes. Additionally, it is likely that instances will "statically" share a physical machine with other cloud users, with potential performance impacts if an instance is co-located with a resource-greedy neighbour.

2.2. MapReduce and Hadoop

The MapReduce framework is a widely used programming paradigm for distributed environments [1]. MapReduce provides an abstraction away from the details of parallelising computation; the framework automatically divides a job into individual tasks, handles scheduling of individual tasks, distributes data and deals with machine failures. The basic MapReduce model expresses computations as a 'Map' and a 'Reduce' function. Hadoop is a framework for the execution of MapReduce jobs. Classic Hadoop has been widely used and studied since its release, but we focus on the more recently developed Hadoop YARN [5]. Both Classic Hadoop and Hadoop YARN use the idea of dividing resources into logical partitions (called 'slots' and 'containers' respectively) which are assigned to executing tasks.

2.3. Local resource shaper for MapReduce

The Local Resource Shaper (LRS) for MapReduce [6] modifies the slot-based resource allocation approach used in Classic Hadoop. Rather than statically configured Map and Reduce slots, LRS introduces the idea of a dual purpose task slot. LRS 'shapes' task resource usage by allocating resources to tasks in a tier-based model. Tasks are split into Active and Passive tiers, with resources allocated such that the Active task uses as much resources as possible to maintain its original usage, while the Passive task uses resources unused by the Active task. In Classic Hadoop, resources are allocated based on a slot model, with each slot representing a partition of resources. LRS pairs slots such that for each Active task (i.e., slot), there is an associated Passive task. This means that within each pair, the two tasks have complementary resource usage; for example, while the Active task waits on I/O operations to complete, the Passive task is able to use the otherwise wasted CPU resources. This approach is shown to significantly increase resource utilisation while minimising resource contention, resulting