Future Generation Computer Systems 37 (2014) 162–177

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Dynamic tuning of the workload partition factor and the resource utilization in data-intensive applications

Claudia Rosas [a,*], Anna Sikora [a], Josep Jorba [b], Andreu Moreno [c], Antonio Espinosa [a], Eduardo César [a]

[a] Computer Architecture and Operating Systems Department, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain
[b] Estudis d'Informatica, Multimedia i Telecomunicacio, Universitat Oberta de Catalunya, 08018 Barcelona, Spain
[c] Escola Universitaria Salesiana de Sarria, 08017 Barcelona, Spain

HIGHLIGHTS

- A methodology to improve performance in data-intensive applications has been designed.
- It adapts at run time the workload partition factor and the number of resources to be used.
- Adaptation of the partition factor enables load balancing and overall time reduction.
- Adaptation of the number of resources used enables execution without large idle times.
- Obtained results using real data-intensive applications are encouraging and positive.

ARTICLE INFO

Article history:
Received 15 April 2012
Received in revised form
23 October 2013
Accepted 3 December 2013
Available online 14 December 2013

Keywords:
Load balancing
Dynamic tuning
Data-intensive applications
Divisible Load Theory (DLT)

ABSTRACT

The recent data deluge needing to be processed represents one of the major challenges in the computational field. This fact led to the growth of specially-designed applications known as data-intensive applications. In general, in order to ease the parallel execution of data-intensive applications input data is divided into smaller data chunks that can be processed separately. However, in many cases, these applications show severe performance problems mainly due to the load imbalance, inefficient use of available resources, and improper data partition policies. In addition, the impact of these performance problems can depend on the dynamic behavior of the application.

This work proposes a methodology to dynamically improve the performance of data-intensive applications based on: (i) adapting the size and the number of data partitions to reduce the overall execution time; and (ii) adapting the number of processing nodes to achieve an efficient execution. We propose to monitor the application behavior for each exploration (query) and use gathered data to dynamically tune the performance of the application. The methodology assumes that a single execution includes multiple related queries on the same partitioned workload.

The adaptation of the workload partition factor is addressed through the definition of the initial size for the data chunks; the modification of the scheduling policy to send first data chunks with large processing times; dividing of the data chunks with the biggest associated computation times; and joining of data chunks with small computation times. The criteria for dividing or gathering chunks are based on the chunks' associated execution time (average and standard deviation) and the number of processing elements being used. Additionally, the resources utilization is addressed through the dynamic evaluation of the application performance and the estimation and modification of the number of processing nodes that can be efficiently used.

We have evaluated our strategy using as cases of study a real and a synthetic data-intensive application. Analytical expressions have been analyzed through simulation. Applying our methodology, we have obtained encouraging results reducing total execution times and efficient use of resources.

1. Introduction

Nowadays, one of the biggest challenges in the computational field is the continuous growth of data that needs to be processed. The data flow coming from sensors, results of biological and physical experiments [1], and even from the information generated by

* Corresponding author. Tel.: +34 934137244.
E-mail addresses: crosas@caos.uab.es (C. Rosas), ania@caos.uab.es (A. Sikora), jjorbae@uoc.edu (J. Jorba), amoreno@euss.cat (A. Moreno), aespinosa@caos.uab.es (A. Espinosa), eduardo@caos.uab.es (E. César).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.002

users, are surpassing the capacities of the systems and algorithms recently designed. This led to a new type of applications known as *data-intensive applications* [2], or *big-data computing* [3].

In the era of data-intensive applications, computational systems are not only intended to compute but also to store and manage data. Given the current volume of data, those tasks increase the challenge and complexity of developing a suitable solution. Moreover, the efficient data processing is not only a matter of having a large number of processing units because it also depends on characteristics of the workload of the application.

In order to improve performance, there are many studies that have obtained good results, ranging from approaches that analyze the effectiveness of I/O systems, to the design of appropriate strategies to define and access data structures [4]. In many cases, it has been necessary to divide the workload of data-intensive applications into smaller data chunks (according to *Divisible Load Theory*, DLT [5]) to ensure that the workload of the application can be manageable. This has been done to reduce the size of the workload and enable parallelism, but once the workload has been divided, other issues rise like those related to disk access or load balancing.

The execution of data-intensive applications that involves a large number of queries or iterations, may lead to variations in the overall execution time between iterations. For this reason, performance analysis and load balancing techniques must be adapted to particular characteristics of the application. In most cases, given the variability between (or within) iterations, performance analysis must be carried out at run time. This is an extremely complex process because it is carried out during the execution of the application without incurring in excessive overheads. If not, the proposed solution may be obsolete from one iteration to the other.

Most load balancing methods, such as *factoring* [6,7], are based on the idea of distributing the workload of the application in chunks of decreasing size. For the purpose of improving total computation time of scientific applications, these methods try to determine a good *partition factor* to obtain the chunks. When doing this, parameters such as computation time, communication time, and overall performance of the application are taken into consideration.

This work proposes a methodology that dynamically identifies and tunes load imbalances in parallel data-intensive applications. This proposal is oriented to: applications that perform several related explorations or queries[1] on a large workload; and the possibility of arbitrarily dividing or concatenating the workload of the application into data chunks of different size. These assumptions are sound because large-scale data processing usually consists on launching several related explorations on the data, and processing can be performed on data chunks of arbitrary size.

To improve performance in parallel data-intensive applications with arbitrarily divisible workloads, our methodology considers: (i) the adaptation of the partition factor for the workload to reduce the overall execution time and avoid load imbalances; and (ii) the modification of the number of processing nodes that can be used efficiently. This proposal works for homogeneous clusters and uses an application performance model that allows for dynamically adjusting the tuning parameters according to the current application behavior.

The methodology is based on monitoring the computation time of generated data chunks to determine the order in which they should be scheduled in future explorations. The proposal includes the dynamic division and gathering of data chunks (when the partitioning cost is low); and the possibility of dynamically choosing among previously generated partitions (when the partition cost is too high). In both cases, the calculation of the partition factor will take into consideration the communication cost, memory use, and the number of available computing nodes (besides the computation time).

Our methodology assumes that a single execution includes multiple related explorations on the same partitioned workload. Thus, previously collected data for one exploration can be used to dynamically adapt the number of resources (processing nodes) for subsequent explorations. As our method is based on the execution of applications in homogeneous clusters of workstations, the computation capacity is constant and, in most cases, the disk and network latency are stable. Moreover, in order to make easier the initial design we used a *shared nothing* [8] processing approach. Under this approach, each node (consisting of processor, local memory, and disk resources) shares nothing with other nodes in the cluster.

Summarizing, our strategy proposes:

(a) generating multiple representative workload partitions prior to the execution of the application when the cost of partitioning data is too high;
(b) monitoring the computation time of every exploration or query on every data chunk;
(c) ordering and allocation of data chunks along the execution of the application according to their associated computation times;
(d) tuning of the partition factor of the data chunks with the highest (partitioning) and lowest (grouping) associated computation times according to the observed efficiency (relation between computation time and the number of computation nodes);
(e) distributing newly generated data chunks in subsequent explorations;
(f) estimating the number of processing nodes to be effectively used by the application.

The evaluation of our proposal has been carried out in a real and widely used data-intensive application: the computation/data-intensive bioinformatics tool *Basic Local Alignment Sequence Tool* (BLAST) [9], as well as on a distributed merge sort. Results obtained from both applications are encouraging in terms of total execution time reduction and the efficient use of resources.

Moreover, an analytical simulator has been used to evaluate the analytical expressions of the methodology. These expressions are used to estimate the modifications in the size of the data chunks and in the number of processing nodes to be used. In order to analyze the behavior of the methodology we used the simulation for a wider range of scenarios.

The rest of the paper is organized as follows. First, Section 2 provides an overview of related work. Next, Section 3 describes the proposed methodology for balancing the load and improving performance of data-intensive applications. Section 4 shows the most relevance characteristics of the selected scenarios to evaluate our methodology: (i) a real data-intensive application, the bioinformatics tool BLAST; (ii) a synthetic application based on a distributed sorting algorithm; and (iii) an analytical simulator of data-intensive applications. In addition, Section 4 explains the reasons for using these applications to test the proposed methodology. Section 5 where the experimental evaluation is described and results are discussed. Finally, Section 6 shows the conclusions and outlines future work.

2. Related work

A divisible workload is such that can be divided into several independent pieces or chunks of arbitrary size to be processed in parallel by a set of compute nodes. The Divisible Load Theory (DLT) [5] was introduced in the late 1980s. Later on, DLT has branched in many new directions covering scheduling problems and performance modeling for various types of computational environments, such as Grid and Cloud systems [10], systems with memory limitations [11] or with computation time restrictions [12].

[1] We consider terms: exploration, query and iteration as synonyms; and they may be used interchangeably along this work.