J. Parallel Distrib. Comput. 93–94 (2016) 87–101

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Data-aware task scheduling for all-to-all comparison problems in heterogeneous distributed systems

Yi-Fan Zhang, Yu-Chu Tian *, Colin Fidge, Wayne Kelly

School of Electrical Engineering and Computer Science, Queensland University of Technology (QUT), GPO Box 2434, Brisbane QLD 4001, Australia

HIGHLIGHTS

- Abstraction of all-to-all comparison computing pattern with big data sets.
- Formulation of all-to-all comparisons in distributed systems as a constrained optimization.
- Data-aware task scheduling approach for solving the all-to-all comparison problem.
- Metaheuristic data pre-scheduling and dynamic task scheduling in the approach.

ARTICLE INFO

Article history:
Received 25 October 2015
Received in revised form
14 February 2016
Accepted 14 April 2016
Available online 25 April 2016

Keywords:
Distributed computing
All-to-all comparison
Data distribution
Task scheduling

ABSTRACT

Solving large-scale all-to-all comparison problems using distributed computing is increasingly significant for various applications. Previous efforts to implement distributed all-to-all comparison frameworks have treated the two phases of data distribution and comparison task scheduling separately. This leads to high storage demands as well as poor data locality for the comparison tasks, thus creating a need to redistribute the data at runtime. Furthermore, most previous methods have been developed for homogeneous computing environments, so their overall performance is degraded even further when they are used in heterogeneous distributed systems. To tackle these challenges, this paper presents a data-aware task scheduling approach for solving all-to-all comparison problems in heterogeneous distributed systems. The approach formulates the requirements for data distribution and comparison task scheduling simultaneously as a constrained optimization problem. Then, metaheuristic data pre-scheduling and dynamic task scheduling strategies are developed along with an algorithmic implementation to solve the problem. The approach provides perfect data locality for all comparison tasks, avoiding rearrangement of data at runtime. It achieves load balancing among heterogeneous computing nodes, thus enhancing the overall computation time. It also reduces data storage requirements across the network. The effectiveness of the approach is demonstrated through experimental studies.

1. Introduction

The size of data sets has grown rapidly across a variety of systems and applications [2,15]. Distributed computing using a distributed data storage architecture has been widely applied in data intensive and computationally intensive problems due to its cost effectiveness, high reliability and high scalability [4,14,21]. It decomposes a single large computing problem into multiple smaller ones and then schedules those smaller ones to distributed worker nodes. Its performance largely depends on the data distribution, task decomposition, and task scheduling strategies. A significant performance degradation may result from inappropriate data distribution, poor data locality for computing tasks, and unbalanced computational loads among the distributed worker nodes. An inappropriate data distribution consumes excessive storage space. Poor data locality means that the data required by a particular worker is unavailable locally, thus creating overheads associated with rearranging data between the nodes at runtime. Load imbalances lengthen the overall computation time due to the need to wait for the slowest nodes. Thus, innovative approaches are required to deal with all these issues for distributed computing of large-scale problems with distributed data.

All-to-all comparison is a type of computing problem with a unique pairwise computing pattern [18,19,33]. It involves comparing two different data items from a data set for all possible pairs of data items. It is widely found in various application domains such

* Corresponding author.
E-mail address: y.tian@qut.edu.au (Y.-C. Tian).

http://dx.doi.org/10.1016/j.jpdc.2016.04.008
0743-7315/

as bioinformatics, biometrics and data mining. For example, in data mining, clustering algorithms use all-to-all comparisons to derive a similarity matrix to characterize the similarities between objects. For instance, in a study of music information retrieval, 3090 pieces of music were pairwise compared to determine the similarity between any pair of items [3].

Existing solutions for general distributed computing have been adapted to distributed processing of all-to-all comparison problems. They generally consider data distribution and comparison task scheduling in two independent phases. There are two representative ideas for data distribution: (1) to copy all data to all nodes [1,11,24]; and (2) to distribute data by using the Hadoop computing framework [28,6]. While considering data distribution and task scheduling in two independent phases simplifies the system's design, significant weaknesses are caused by the lack of coordination between the two phases, leading to poor performance of the overall distributed computation. This is in addition to the inherent drawbacks of these two approaches for all-to-all comparison problems, as is discussed below in Section 2.

To solve these problems, here we present a data-aware task scheduling approach for distributed computation of large-scale, all-to-all comparison problems with distributed storage in heterogeneous distributed systems. Specific contributions of this work include:

1. A formalization of distributed all-to-all comparison computations as a constrained optimization problem, which considers data distribution and task scheduling simultaneously;
2. A metaheuristic pre-scheduling strategy, as a solution to the constrained optimization requirement, for task-oriented data distribution with consideration of storage usage, data locality, and load balancing; and
3. Runtime-scheduling strategies, as a refinement to the pre-scheduling strategy, for static and dynamic scheduling of comparison tasks, with consideration of the computing power of the individual computing nodes in heterogeneous distributed systems.

The paper is organized as follows. Section 2 discusses related work and motivations. Section 3 describes all-to-all comparison problems and their challenges. This is followed by a formalization of distributed all-to-all comparisons as a constrained optimization problem in Section 4. Then, metaheuristic pre-scheduling and runtime-scheduling strategies are presented in Sections 5 and 6, respectively. The strategies are further analysed in Section 7. Section 8 presents experimental results. Finally, Section 9 concludes the paper.

2. Related work and motivations

All-to-all comparison problems occur in various application domains. However, the principle of solving all these problems is the same. A few typical methods are reviewed below.

A brute-force solution to all-to-all comparison problems copies all data onto each node in a distributed system [24,23,32]. Moretti et al. [24] designed a computing framework for all-to-all comparison problems on campus grids. To give each comparison task the required data, they proposed a spanning tree method to deliver the data to every node efficiently. Macedo et al. [23] presented an MPI/OpenMP mixed parallel strategy to run the DIALIGN-TX algorithm in heterogeneous multi-core clusters. They focused on comparing different task allocation policies to show an appropriate choice. Xiao et al. [32] optimized the BLAST algorithm in a GPU–CPU mixed heterogeneous computing system. Their implementation was measured to achieve a six-fold speed-up for the BLAST algorithm. For other big data computing problems other than all-to-all comparisons, distributing all data everywhere was also a widely used data strategy [1].

Distributing all data to all nodes has its advantages and disadvantages. When the data sets are distributed everywhere, scheduling any comparison task to any node will achieve perfect data locality, and load balancing becomes straightforward. However, there are also obvious and major drawbacks. (1) The brute-force replication of data results in worst-case storage usage, the longest time consumption for data transmission, and the highest cost of network communications. For example, a typical all-to-all comparison problem presented by Hess et al. [13] needs to process 268 GB of cow rumen metagenome data, and copying all the data to each node pushes the limits of the available storage resources. In an experiment by Das et al. [9], the average time for deploying 10 GB data sets within a cluster of 14 worker nodes and a 10 Mbps network took nearly 150 min. This long time for data transmission has a significant negative effect on the overall performance of the computing problem. (2) Even if all the data can be duplicated efficiently, much of the data stored in the nodes will never be used in actual comparison tasks, wasting the storage resources considerably. (3) These two drawbacks become even more evident and serious for large-scale problems with big data sets. As all-to-all comparison is a type of combinatorial problem, the complexity of processing big data sets increases exponentially with the size of the data.

Hadoop is a widely-used distributed computing framework for big data problems, based on the MapReduce computing pattern. Therefore, recent attempts have been made to implement domain-specific all-to-all comparison applications using Hadoop [28,6]. Using Hadoop, CloudBurst [28] parallelizes a specific read-mapping algorithm optimized for mapping next-generation sequence data to the human genome and other reference genomes. In a test using a homogeneous cluster of 24 processors, CloudBurst has been shown to achieve an up to 30 times speed-up compared to execution on a single core. Similarly, MRGIS [6] is a parallel and distributed computing platform implemented in Hadoop for geoinformatics applications. A Hadoop environment with 32 homogeneous worker nodes has been investigated for testing the efficiency of the MRGIS system [6]. These solutions benefit from Hadoop's advantages such as scalability, redundancy, automatic monitoring, and distributed computing with simple application programming interfaces (APIs). However, they are domain-specific, and thus not suitable for general all-to-all comparison problems with big data sets from different application domains. Therefore, they were not used as benchmark examples in our experiments to evaluate the performance of our solution for general all-to-all comparison problems.

While Hadoop is widely used, it is inefficient for large-scale all-to-all comparison problems for two reasons. (1) Hadoop is based on the MapReduce computing pattern, which is fundamentally different from the pairwise comparison pattern needed in all-to-all comparison problems. In MapReduce problems, each data item can be processed separately and there is no requirement for pairwise data locality. In Hadoop data items are randomly distributed with a fixed number of replications (the default is 3). Hadoop's data distribution strategy does not consider the data locality requirements for comparison tasks. A naïve attempt to use Hadoop's data distribution for all-to-all comparisons causes a need to redistribute the data between the nodes at runtime [33]. In Qiu et al.'s experiments [27], Hadoop was shown to execute inefficiently for all-to-all comparisons due to frequent switches between 'map' and 'communication' activities. (2) Hadoop does not address load balancing directly, which is also a key requirement for improved performance of the overall execution time in all-to-all comparison problems. This becomes most evident in heterogeneous distributed systems, in which