Future Generation Computer Systems 67 (2017) 152–162

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Scalable and efficient data distribution for distributed computing of all-to-all comparison problems

Yi-Fan Zhang [a], Yu-Chu Tian [a,b,*], Wayne Kelly [a], Colin Fidge [a]

[a] School of Electrical Engineering and Computer Science, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia
[b] College of Information Engineering, Taiyuan University of Technology, Taiyuan, Shanxi 030024, China

HIGHLIGHTS

- New insights into distributed computing of all-to-all comparison problems with big data.
- Formulation of data distribution of all-to-all comparisons as an optimization problem.
- A heuristic algorithm for scalable and efficient data distribution and task scheduling.
- Demonstration of the algorithm in improved storage saving, data locality and execution time.

ARTICLE INFO

Article history:
Received 11 October 2015
Received in revised form
27 May 2016
Accepted 6 August 2016
Available online 9 September 2016

Keywords:
Distributed computing
Big data
All-to-all comparison
Data distribution

ABSTRACT

All-to-all comparison problems represent a class of big data processing problems widely found in many application domains. To achieve high performance for distributed computing of such problems, storage usage, data locality and load balancing should be considered during the data distribution phase in the distributed environment. Existing data distribution strategies, such as the Hadoop one, are designed for problems with MapReduce pattern and do not consider comparison tasks at all. As a result, a huge amount of data must be re-arranged at runtime when the comparison tasks are executed, degrading the overall computing performance significantly. Addressing this problem, a scalable and efficient data distribution strategy is presented in this paper with comparison tasks in mind for distributed computing of all-to-all comparison problems. Specifically designed for problems with all-to-all comparison pattern, it not only saves storage space and data distribution time but also achieves load balancing and good data locality for all comparison tasks of the all-to-all comparison problems. Experiments are conducted to demonstrate the presented approaches. It is shown that about 90% of the ideal performance capacity of the multiple machines can be achieved through using the approach presented in this paper.

1. Introduction

Data-intensive computing [1] with a very large number of big data sets can be a great challenge in service computing for both commercial applications and scientific investigations. For these big data problems, a huge amount of data needs to be stored, retrieved, analyzed and visualized within a period of time acceptable to the users [2,3]. This demands significant resources of computing power, memory and storage spaces, and network communications. While distributed computing systems provide a general platform to solve large-scale computing problems, improving the computing performance for large-scale and data-intensive computing problems is an emerging requirement.

Among various big data processing problems, all-to-all comparison problems are widely found in many application domains such as machine learning, data mining, information management, bioinformatics and biometrics. A typical example is the calculation of covariance matrix for high-dimensional data in machine learning. In data mining, the computation of similarity matrix is a critical step for clustering and classification. It gives all pairwise similarities or dissimilarities between the objects under consideration [4]. The experiments by Perera [5] computed the cosine similarity between 8192 feature vectors. Mapping the ontologies Molecular Functions and Biological Processes from Gene Ontology with 10,000 and 20,000 concepts, respectively, involves comparisons of about 2×10^8 pairs of entities [6]. In bioinformatics,

* Corresponding author at: School of Electrical Engineering and Computer Science, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia.
E-mail address: y.tian@qut.edu.au (Y.-C. Tian).

http://dx.doi.org/10.1016/j.future.2016.08.020
0167-739X/

phylogenetic relationships are inferred through comparing gene sequences of different species [7]. Sequence alignment, clustering analysis [8] and recently investigated global network alignment [9] are typical all-to-all comparison problems in computational biology and bioinformatics.

All-to-all comparisons represent a typical computing pattern, which focuses on processing a large number of large- or small-size data files. In a general formulation of the all-to-all comparison problems, each data item of a data file needs to be compared with all data items of the other data files in the data set. Therefore, the all-to-all comparison computing patten is fundamentally different from the MapReduce computing pattern [10], which has been implemented in Hadoop with wide applications in big data processing [11]. If the number of data files in a data set or the number of data items in a data file is big, the scale of the required computation for all-to-all comparisons becomes massive [12].

Efforts have been made to address all-to-all comparison problems previously [13–18]. Some solutions have been proposed for special-purpose all-to-all comparison problems such as the popularly used BLAST [19] and ClustalW [20], or for different types of computing architectures with GPUs [16] or shared memory [21]. However, these solutions require that all data files be deployed to each of the nodes in the system. While distributing whole data sets everywhere is conceptually easy to implement, it causes significant time consumption and communication cost and demands a huge amount of storage space. Therefore, the data distribution strategy from these previous solutions is not scalable to the big data processing problems considered in this paper.

In addition to these specific solutions, some computing frameworks are also widely used to process big data problems in distributed environments [22–25]. Enabling distributed processing of large data sets across clusters of commodity servers, Hadoop [11] is designed to scale up from a single server to thousands of machines. With fault tolerance, it can achieve high computing performance for distributed computation that well matches the MapReduce computing pattern. However, the Hadoop distributed file system (HDFS) and its data distribution strategy are inefficient for all-to-all comparison problems due to the completely different computing pattern involved.

The main contribution of this paper is a scalable and efficient data distribution strategy for a class of all-to-all comparison problems, in which all input data files have the same or similar size and thus comparison tasks have the same or similar execution time. The strategy is developed with consideration of storage usage, data locality and load balancing of distributed computing tasks. This paper has substantially extended our preliminary studies in a conference paper [12] from four aspects: new insights into the challenges, reformulation of the problem, refinement of the data distribution strategy, and more experimental studies. This will be summarized at the end of Section 2 on Related Work.

The paper is organized as follows. Section 2 reviews related work and motivates the research. Section 3 formalizes all-to-all comparison problems and identifies the challenges of data distribution. This is followed by Section 4 to formulate the data distribution problem as an optimization problem. Providing a heuristic solution to the optimization problem, our new data distribution strategy is presented in Section 5, and is experimentally demonstrated in Section 6. Finally, Section 7 concludes the paper.

2. Related work and motivations

Several approaches have been developed to address specific all-to-all comparison problems in bioinformatics. All-to-all comparisons are a key calculation stage in Multiple Sequence Alignment (MSA) [26] and studying of phylogenetic diversity in protein families [27]. In general, the computing of these bioinformatics problems includes the calculation of a cross-similarity matrix between each pair of sequences [28,29].

Data intensiveness describes those applications that are I/O bound or with a need to process large volumes of data [2,10]. Compared with computing-intensive problems, applications of all-to-all comparison problems devote most of the processing time to I/O and movement for a massive amount of data [30]. Therefore, to improve the performance of data-intensive problems, the distribution of all the data sets needs to be well considered.

To process all-to-all comparison problems, various distributed systems and runtime libraries have been used. Heitor and Guilherme [13] proposed a methodology to parallelize a multiple sequence alignment algorithm by using a homogeneous set of computers with the Parallel Virtual Machine (PVM) library. In their work, a detailed description of the modules was provided and a special attention was paid to the execution of the multiple sequence comparison algorithm in parallel. Macedo et al. [14] proposed an MPI/OpenMP master/slave parallel strategy to run the DIALIGN-TX algorithm in heterogeneous multi-core clusters. In their research, different task allocation policies were compared to determine the appropriate choice. All these approaches distribute all data to each of the computing nodes in the cluster, leading to inefficiencies in data distribution and storage. Our work in this paper aims to avoid such inefficiencies.

Meng and Chaudhary [15] presented a heterogeneous computing platform through a Message Passing Interface (MPI) enabled enterprise computing infrastructure for high-throughput biological sequence analysis. In order to achieve load balancing, they distributed the workload based on the hardware configuration. The whole database is split into multiple nearly-equal sized fragments; and then each of the computing nodes is assigned a number of database fragments according to its processing capacity. However, in practical computing of all-to-all comparison problems using their approach, data transmissions among the computing nodes cannot be avoided at runtime. This drawback will be overcome in our work presented in this paper.

Xiao et al. [16] proposed a design and optimization of the BLAST algorithm in a GPU–CPU mixed heterogeneous system. Due to the specific architecture of the GPU, their implementation can achieve a six-fold speed-up for the BLAST algorithm. GPUs were also used for a parallel implementation of MAFFT for MSA analysis [26]. In the work by Torres et al. [31], they were configured for exact alignment of short-read genetic sequences. To accelerate the next generation long read mapping, Chen et al. made use of FPGA hardware to speed up sequence alignment [32]. In comparison with all those hardware-dependent implementations, our work in this paper does not rely on specific hardware.

Several approaches were also developed for load balancing in distributed computing of all-to-all comparison problems. One version of parallel all-to-all comparison of genome sequences was carried out by Hill et al. [17]. The main intention of the work was to provide load balancing among the clusters by dividing the comparison matrix into rows and then dynamically assigning these rows to different nodes. Gunturu [18] proposed a load scheduling strategy, which depends on the length of the sequence and the number of processors in the network. They assumed that all the processors in the network already had both sequences to be compared in their local memory. Our work in this paper distributes data to the computing nodes with consideration of the computing tasks, thus avoiding this assumption.

Recently, efforts have been made to use computing frameworks for all-to-all comparison problems. All-pairs is an abstraction designed by Moretti et al. [33] for data-intensive computing on campus grids. It focuses on providing an abstraction for users to deal with all-to-all comparison problems. To give each comparison task the required data, a spanning tree method is proposed to deliver all data to every node efficiently.