Application of Simulated Annealing to Data Distribution for All-to-All Comparison Problems in Homogeneous Systems

Yi-Fan Zhang[1], Yu-Chu Tian[1(✉)], Wayne Kelly[1], Colin Fidge[1], and Jing Gao[2]

[1] School of Electrical Engineering and Computer Science,
Queensland University of Technology, GPO Box 2434, Brisbane, QLD 4001, Australia
y.tian@qut.edu.au
[2] College of Computer and Information Engineering,
Inner Mongolia Agricultural University, 306 Zhaowuda Road,
Hohhot 010018, Inner Mongolia, China
gaojing@imau.edu.cn

Abstract. Distributed systems are widely used for solving large-scale and data-intensive computing problems, including all-to-all comparison (ATAC) problems. However, when used for ATAC problems, existing computational frameworks such as Hadoop focus on load balancing for allocating comparison tasks, without careful consideration of data distribution and storage usage. While Hadoop-based solutions provide users with simplicity of implementation, their inherent MapReduce computing pattern does not match the ATAC pattern. This leads to load imbalances and poor data locality when Hadoop's data distribution strategy is used for ATAC problems. Here we present a data distribution strategy which considers data locality, load balancing and storage savings for ATAC computing problems in homogeneous distributed systems. A simulated annealing algorithm is developed for data distribution and task scheduling. Experimental results show a significant performance improvement for our approach over Hadoop-based solutions.

1 Introduction

All-to-all comparison (ATAC) represents an important computing pattern in broad areas such as bioinformatics, biometrics and data mining. In the ATAC computing pattern, each file within a data set is pairwise compared with all other files. For example, Arora et al. evaluated the audio similarity of 3090 music pieces through ATACs [1].

Distributed systems with commodity resources have been widely used in processing data intensive ATAC problems [2–4]. However, the performance of an ATAC computation can be greatly affected by a number of factors, e.g., data transmission and system load balancing.

Strategies for load balancing have been reported in a number of references [2,3]. These have aimed to dispatch a similar number of comparison tasks to the computing nodes in a distributed system. However, because co-location of data needed

S. Arik et al. (Eds.): ICONIP 2015, Part III, LNCS 9491, pp. 683–691, 2015.
DOI: 10.1007/978-3-319-26555-1_77

for the comparisons is not considered during task allocation, poor 'data locality' in the initial distribution and massive consequent movement of data at runtime become major bottlenecks for the overall computation.

Recently, systematic frameworks have been developed to deal with data-intensive ATAC problems. Typical examples are Hadoop-based systems [4,5]. Hadoop simplifies the implementation of ATAC applications from the programmer's perspective and efficiently distributes the data across all nodes. However, its inherent MapReduce pattern does not match the ATAC computing pattern because MapReduce assumes that each data item can be processed independently, whereas ATAC comparisons each require two items. This leads to load imbalances and runtime inefficiencies due to poor data locality when Hadoop's data distribution algorithm is used for ATAC problems [6].

To solve these challenging issues, we present here a data distribution strategy for distributed computing of large-scale ATAC problems in homogeneous distributed computing systems. Our approach is scalable and efficient with full consideration of data locality, load balancing and storage usage. A simulated annealing (SA) algorithm is used for data distribution and static task scheduling.

The paper is organized as follows. Section 2 discusses related work and motivations. Section 3 describes the ATAC problem and its challenges. Our data distribution strategy using an SA algorithm is developed in Sect. 4. Experiments which validate the approach are described in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related Work and Motivations

Distributed computing systems have shown their advantages in processing ATAC problems in various domain areas. An efficient grid scheduler has been designed for parallel implementation of MSA algorithm on a computational grid [2]. It splits a single alignment task into optimal-size subtasks and then distributes the subtasks among multiple processors. However, it assumes that the task running time is much higher than the communication overhead in sending data of the sub-matrix, which makes the method unsuitable for large-scale ATAC problems.

Gunturu et al. [3] presented a load scheduling strategy with near optimal processing time for parallel DNA sequence alignment. In their study, the load distribution depends on the length of the sequence and the number of processors. They also assume that all the comparison tasks have local data, so how to achieve the data locality needed for ATAC problems in general is not answered.

Recently, computing frameworks have been developed for ATAC problems. CloudBlast [4] is a distributed implementation of NCBI BLAST. It integrates a number of technologies, e.g., Hadoop, virtual workspaces and ViNe, together to parallelize, deploy and manage bioinformatics applications. Compared to MPI-based solutions such as mpiBLAST, CloudBLAST shows improvement. It also simplifies the development and management of the computing applications.

Addressing Hadoop's weakness, improved methods have been proposed. Bi-Hadoop [7] is an extension of Hadoop to better support binary-input applications. By providing a binary-input aware task scheduler and a caching subsystem, it outperforms Hadoop by up to 3.3 times and a 48 % reduction in remote data reads for binary-input applications. The caching mechanism improves the computing performance, but all data still needs to be distributed through Hadoop's data strategy. Thus, these improved methods still have limitations.

Among existing approaches for distributed processing of ATAC problems, many methods for load balancing do not simultaneously consider the need for data co-location. Also, Hadoop-based solutions are inefficient because of the unmatched data and task strategies. While attention has been paid to improving Hadoop for ATAC problems, the performance improvement is still limited due to Hadoop's fundamental basis in the MapReduce computing model.

To address these issues, we present here a substantial extension of our own previous work in the area [6]. A large-scale, high-performance data distribution strategy is developed for ATAC problems, based on simulated annealing.

3 Problem Statement and Challenges

An ATAC problem is a specific Cartesian product of a data set. Let A, A_i, $C(A_i, A_j)$, $M[i, j]$ represent the set of input data items, a single data item in set A, the comparison operation between data items A_i and A_j, and an output similarity matrix element, respectively. The ATAC problem is to calculate

$$M[i, j] = C(A_i, A_j) \quad \text{for} \quad i, j = 1, 2, \ldots, |A| \ . \tag{1}$$

For distributed processing of ATAC problems, both data set A and all comparison tasks $C(A_i, A_j)$ need to be distributed to different worker nodes. While different strategies have been developed previously, performance issues still exist.

Task Balancing Causes Data Storage Issues. Comparison tasks are usually allocated by rows or columns [8,9]. Though load balancing is considered, unoptimized data distribution causes severe data imbalances and high storage usage. Consider an example of 6 data items and 3 nodes. The workload is divided by the rows. The result in Fig. 1 shows that although each of the three nodes has 5 different comparison tasks, the data files are stored in inefficiently. Node 1 has to store copies of all the data items, but this should be avoided when for data-intensive computing. Moreover, three worker nodes have 6, 5 and 4 data items, respectively, implying a system data imbalance.

Storage Saving Causes Task Issues. When Hadoop-based solutions have been used to solve ATAC problems, each data item is randomly distributed to the worker nodes with a fixed number of replications. Although this achieves high data reliability due to replication, poor performance is inevitable due to the lack of consideration of comparison task allocation. Take 6 data items and 4 worker nodes for example. In Fig. 2, each data item has three copies in three different

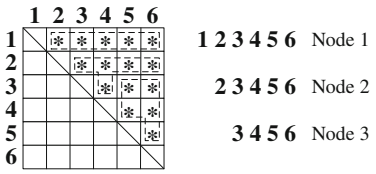

Fig. 1. A data imbalance

nodes for reliability. Each node stores 50 % of the data. However, 9 comparison tasks do not have local data, requiring massive data movement at runtime to complete the comparisons and consequently poor overall performance.

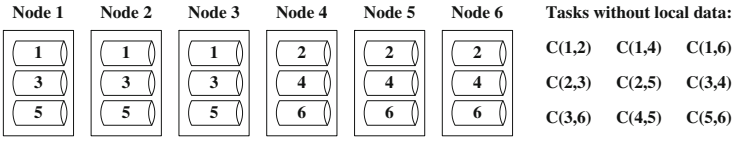

Fig. 2. Poor data locality for comparison tasks

4 Data Distribution Strategy with Simulated Annealing

This section presents our metaheuristic data distribution strategy for distributed computing of ATAC problems in homogeneous systems. For efficient derivation of data distribution and task scheduling, a simulated annealing (SA) algorithm is developed with specific methods for generating and selecting solutions.

Considering the challenges involved in solving ATAC problems, a data distribution strategy is presented below to meet the following requirements:

1. The system has good static load balancing (Load balancing);
2. All comparison tasks have the data they need locally (Data locality); and
3. The maximum number of data among all nodes is minimized (Storage saving).

Simulated Annealing [10] is a probabilistic optimization technique derived from the physical process of crystallization. It is widely used to solve global optimization problems. We adopt it here to solve the data distribution and task scheduling problems for ATAC computations. To use an SA approach, we must: (1) determine the Annealing module and Acceptance Probability module; and (2) determine an initial solution, the neighbourhood selection method and the fitness equation.

Annealing Module and Acceptance Probability Module. The setting of the SA module has significant effects on the final result [11]. As one of the fastest

Table 1. SA module parameter settings (k represents the iteration step)

Item	Setting
Temperature decreasing function	$t_k = T_0/k$
Starting temperature	1.0
Ending temperature	10^{-5}
Inner loop iteration threshold	100
Acceptance probability function	$P(\Delta E) = \exp(-\Delta E/t)$

decreasing temperature methods, we use Cauchy scheduling [12]. Parameters used for the example in Sect. 5 are shown in Table 1.

Initial Solution. For ATAC problems with M data items, $M(M-1)/2$ comparison tasks must be allocated. Hence, for a homogeneous system with N nodes, an initial solution can be generated by randomly and evenly allocating all comparison tasks and related data items. Let M, N, D_i, T_i and U represent the number of data files to be processed, the number of nodes in the system, the set of data files stored on node i, the comparison task set allocated to node i and the set of tasks that have not yet been scheduled, respectively. An initial solution is generated as follows:

1. Keep picking up comparison tasks from set U and allocating them to each node $i \in \{1, 2, \ldots, N\}$ until all have been allocated, i.e., $|T_i| = \left\lceil \frac{M(M-1)}{2N} \right\rceil$ or $U = \emptyset$; and
2. Based on each task set T_i, distribute all related data files to data set D_i.

The solution $S = \{(T_1, D_1), (T_2, D_2), ..., (T_N, D_N)\}$ is then a feasible solution, which meets both our initial requirements.

Neighbourhood Selection Method. Following the design of the initial solution, a new neighbourhood solution S' can be generated from a solution S from the following steps:

1. Randomly choose two nodes i and j;
2. Randomly pick up two comparison tasks $t_k \in T_i$ and $t_l \in T_j$ and swap them; and
3. Update related data files in data set D_i and D_j.

Considering that all the nodes in a homogeneous system can be treated as indistinguishable, this method promises that each new solution has the capability to solve the ATAC problem and all possible solutions can be generated theoretically.

Fitness Equation. Considering the requirements mentioned at the beginning of this section, the fitness equation $F(S)$ for a solution S is defined as the set of the number of data files allocated to each of the worker nodes:

$$F(S) = \{|D_1|, |D_2|, \ldots, |D_N|\} . \tag{2}$$

The difference ΔF between two different solutions S and S' is calculated as follows. Firstly, the elements in both $F(S)$ and $F(S')$ are sorted in descending order. Then, ΔF is obtained as:

$$\Delta F = F(S) - F(S') = \{(|D_1| - |D'_1|), \ldots, (|D_N| - |D'_N|)\} . \tag{3}$$

Finally, the value of the cost change Δf is defined as:

$$\Delta f = \begin{cases} \text{the 1st non-zero element value in Eq. (3)}, & \text{if one exists} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

This method promises that solution S with smaller maximum values in $F(S)$ always be accepted as required by SA. Moreover, unlike only comparing the maximum values in $F(S)$ and $F(S')$, this method utilizes much more information from other elements. Hence, the SA algorithm has a higher efficiency in searching for better solutions.

Data Distribution Algorithm. By integrating all the above designs, our data distribution algorithm using SA is presented as Algorithm 1.

Algorithm 1. Data Distribution Algorithm

Initialisation:
Randomly generate initial solution S using the method in **Initial Solution**;
Set parameters based on Table 1;
Set the current temperature t to be the starting temperature.
Distribution:
while The current temperature $t >$ the ending temperature **do**
  while The iteration step is below the inner loop iteration threshold **do**
    Generate a new solution S' from S (using the **Neighbourhood Selection Method**);
    Calculate the change of fitness, Δf, from Equation (4)
      (The fitness method for F, ΔF and Δf are developed in **Fitness Equation**);
    if $\exp(-\Delta f/t) > random[0, 1)$ **then**
      Accept the new Solution: $S \leftarrow S'$
    Increment the iteration step by 1;
  Lower the current temperature t based on the function in Table 1;
Return final solution S.

5 Experiments

We conducted experiments to evaluate the following aspects of our algorithm: storage savings, task allocations, data scalability and computing performance.

Storage Savings, Task Allocations and Data scalability. An example with 4 nodes and 8 data items is used to show the effectiveness of our data distribution strategy. The results are summarized below:

It can be seen from these results that data balancing and static load balancing are achieved. Each worker node only stores 5 data items. Moreover, each node is allocated 7 comparison tasks all with good data locality.

Node	Distributed data files	Allocated comparison tasks
A	0, 2, 3, 6, 7	(0,2) (0,3) (0,6) (0,7) (2,3) (2,6) (2,7)
B	1, 3, 5, 6, 7	(1,3) (1,5) (1,6) (1,7) (3,7) (5,7) (6,7)
C	0, 1, 2, 4, 5	(0,1) (0,4) (0,5) (1,2) (1,4) (2,4) (2,5)
D	3, 4, 5, 6, 7	(3,4) (3,5) (3,6) (4,5) (4,6) (4,7) (5.6)

As the numbers of data items and nodes increases, Table 2 shows our strategy still achieves good results in storage saving, load balancing and data locality, compared with Hadoop's strategy (using 3 copies of each data item). Each node has an equal number of comparison tasks with good data locality in our solution. Although Hadoop's strategy uses less storage space overall, runtime performance issues are inevitable due to the poor data locality for comparison tasks. Figure 3 shows that our approach still has good data scalability, and is far better than the brute-force ATAC solution of copying all data items onto every node.

Table 2. Storage usage and storage savings of our work versus Hadoop for 256 files

No. of	Max. # of files on a node		Storage space saving		# of tasks on each node
nodes	This work	Hadoop	This work	Hadoop	This work
8	150	96	41 %	63 %	4080
16	116	48	55 %	81 %	2040
32	83	24	68 %	91 %	1020
64	56	12	78 %	95 %	510

Computing Performance. We also conducted experiments with a bioinformatics ATAC application. The experimental environment was set as:

- A homogeneous cluster with 5 machines, all running Redhat Linux. One acts as the master node, and all the nodes have one core and 64 GB of RAM.
- Sequential and distributed versions of the CVTree application, which is a typical ATAC problem in bioinformatics [13].
- DsDNA data files from the National Center for Biotech Information (NCBI).

Figure 4 shows the different computation times between our data distribution strategy and Hadoop's strategy. By considering the three requirements summarized in Sect. 4, our data distribution strategy achieves much higher computing performance than Hadoop's strategy. As we discussed in Sect. 3, this is because Hadoop's strategy needs to move numerous data items between nodes during the computation, due to poor data locality.

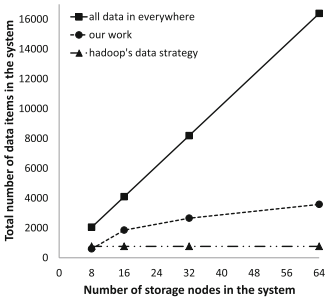

Fig. 3. Data Scalability

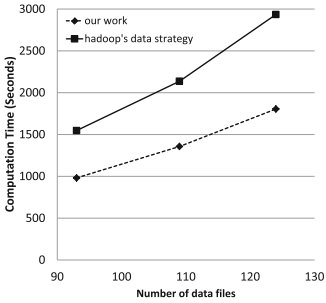

Fig. 4. Computation time performance

6 Conclusion

A scalable and efficient data distribution strategy using simulated annealing has been presented for distributed computing of all-to-all comparison problems in homogeneous distributed systems. It is designed to use as little storage space as possible while still achieving system load balancing and good data locality. Experiments have shown that although our approach uses more overall storage than Hadoop's, we achieve greatly reduced computation times.

Acknowledgments. Author J. Gao would like to acknowledge the support from the National Natural Science Foundation of China under the Grant Number 61462070, and the Inner Mongolia Government under the Science and Technology Plan Grant Number 20130364.

References

1. Arora, R., Gupta, M.R., Kapila, A., Fazel, M.: Similarity-based clustering by left-stochastic matrix factorization. J. Mach. Learn. Res. **14**, 1715–1746 (2013)
2. Somasundaram, K., Karthikeyan, S., Nayagam, M.G., RadhaKrishnan, S.: Efficient resource scheduler for parallel implementation of MSA algorithm on computational grid. In: International Conference on Recent Trends in Information, Telecommunication and Computing, Kochi, Kerala, pp. 365–368, IEEE, 12–13 March 2010
3. Gunturu, S., Li, X., Yang, L.: Load scheduling strategies for parallel DNA sequencing applications. In: Proceedings of the 11th IEEE International Conference on High Performance Computing & Communication Seoul, pp. 124–131. IEEE, 25–27 June 2009
4. Matsunaga, A., Tsugawa, M., Fortes, J.: Cloudblast: combining mapreduce and virtualization on distributed resources for bioinformatics applications. In: IEEE 4th International Conference on eScience, Indianapolis, IN, pp. 222–229, 7–12 December 2008
5. Pireddu, L., Leo, S., Zanetti, G.: Seal: a distributed short read mapping and duplicate removal tool. Bioinformatics **27**(15), 2159–2160 (2011)

6. Zhang, Y.F., Tian, Y.C., Fidge, C., Kelly, W.: A distributed computing framework for all-to-all comparison problems. In: The 40th Annual Conference of the IEEE Industrial Electronics Society (IECON 2014), Dallas, TX, USA. IEEE, 29 October–1 November 2014
7. Yu, X., Hong, B.: Bi-hadoop: Extending hadoop to improve support for binary-input applications. In: 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Delft, IE, pp. 245–252, 13–16 May 2013
8. Moretti, C., Bui, H., Hollingsworth, K., Rich, B., Flynn, P., Thain, D.: All-pairs: an abstraction for data-intensive computing on campus grids. IEEE Trans. Parallel Distrib. Syst. **21**, 33–46 (2010)
9. Pedersen, E., Raknes, I.A., Ernstsen, M., Bongo, L.A.: Integrating data-intensive computing systems with biological data analysis frameworks. In: 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Turku, pp. 733–740. IEEE, 4–6 March 2015
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
11. Wu, W., Li, L., Yao, X.: Improved simulated annealing algorithm for task allocation inreal-time distributed systems. In: IEEE International Conference on Signal Processing Communications & Computing (ICSPCC), Guilin, pp. 50–54. IEEE, 5–8 August 2014
12. Keikha, M.: Improved simulated annealing using momentum terms. In: 2011 Second International Conference on Intelligent Systems, Modelling and Simulation (ISMS), Kuala Lumpur, pp. 44–48. IEEE, 25–27 January 2011
13. Hao, B., Qi, J., Wang, B.: Prokaryotic phylogeny based on complete genomes without sequence alignment. Modern Phy. Lett. **2**(4), 14–15 (2003)