IECON2015-Yokohama
November 9-12, 2015

Distributed Computing of All-to-All Comparison Problems in Heterogeneous Systems

Yi-Fan Zhang, Yu-Chu Tian, Wayne Kelly and Colin Fidge
School of Electrical Engineering and Computer Science
Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia.
Email: y.tian@qut.edu.au.

***Abstract*—The requirement of distributed computing of all-to-all comparison (ATAC) problems in heterogeneous systems is increasingly important in various domains. Though Hadoop-based solutions are widely used, they are inefficient for the ATAC pattern, which is fundamentally different from the MapReduce pattern for which Hadoop is designed. They exhibit poor data locality and unbalanced allocation of comparison tasks, particularly in heterogeneous systems. The results in massive data movement at runtime and ineffective utilization of computing resources, affecting the overall computing performance significantly. To address these problems, a scalable and efficient data and task distribution strategy is presented in this paper for processing large-scale ATAC problems in heterogeneous systems. It not only saves storage space but also achieves load balancing and good data locality for all comparison tasks. Experiments of bioinformatics examples show that about 89% of the ideal performance capacity of the multiple machines have be achieved through using the approach presented in this paper.**

I. Introduction

All-to-all comparison (ATAC) problems represent a specific computing pattern that focuses on pairwise comparisons. They require that each file within a data set be pairwise compared with all other data files. They are important in various domains, such as bioinformatics, biometrics and media analysis. For example, in a video retrieval system [1], two 8717 by 8717 matrices of keyframe similarities are computed by using colour, texture and shape through all-to-all comparisons.

ATAC problems are computed in different platforms, such as distributed systems [2, 3, 4, 5]. Inherited from the scenarios in single machine platforms, data distribution strategies are typically designed to distribute all data sets to every worker node in distributed systems. While this data distribution strategy is easy to implement, the time and storage costs of the data distribution becomes very high. Another way to address these issues is to enhance computing frameworks, such as Hadoop [6], to support ATAC problems [7, 8]. Hadoop is designed for processing small partitions of large data sets independently on large homogeneous clusters. It is developed specifically for the MapReduce computing pattern. However, ATAC is a fundamentally different computing pattern. The data distribution and task allocation from Hadoop become very inefficient in ATAC problems.

Distributed systems are either homogeneous or heterogeneous. Most existing methods for distributed computing of ATAC problems are inherently assumed for homogeneous systems. The methods mentioned above are typical examples. An effective approch for distributed computing of ATAC problems in homogeneous systems is also presented in [9]. Despite these advances, heterogeneous distributed systems have been widely used in data-intensive computing problems. They provide a cost-effective computing environment by integrating diverse sets of resources with different capabilities. However, the heterogeneity scenarios also bring challenges in data distribution, task allocation and load balancing, which are aggravated when processing computing problems with specific patterns, e.g., MapReduce and ATAC. To address those challenges, advanced strategies are required for distributed computing of large-scale and data-intensive problems.

This paper develops a novel data and task distribution strategy for distributed computing of large-scale ATAC problems in heterogeneous systems. It considers storage usage, data locality, task allocation and load balancing explicitly. Through experiments, it is demonstrated to be a scalable, efficient and heterogeneous-support strategy for data distribution and task scheduling of ATAC problems in heterogeneous systems.

The paper is organized as follows. Section II discusses related work and motivations. Section III describes the ATAC problem and its challenges. A data and task distribution strategy is developed in Section IV. Experiments are conducted in Section V to demonstrate our approach. Finally, Section VI concludes the paper.

II. Related Work and Motivations

Many sequential ATAC applications in different areas have been paralleled on distributed platforms with specific hardware configurations. Mendonca and Melo [3] proposed and evaluated a method for biological sequence comparisons. They implemented the method by adjusting workload in hybrid platforms composed of GPUs and multicores with SIMD extensions. Providing different task allocation strategies, their implementation achieved good performance benefits in executing the Smith-Waterman algorithm. Singh et al. [4] implemented Smith-Waterman applicaitons by using desktop grids composed of CPUs and GPUs. In their implementation, two levels of parallel scheduling were provided: 1) a desktop grid for coarse-grained parallelization, and 2) GPUs for fine-grained parallelization. In their experiments running on a GPU-accelerated BOINC framework, over 10 times speedup were achieved in comparison with CPU-only systems. Safia [5] proposed SWDUAL, an implementation of the Smith-Waterman algorithm on hybrid platforms consisting of multiple processors and GPUs. SWDUAL is based on a fast dual approximation scheduling algorithm. The algorithm selects the most suitable tasks to execute on the GPUs while keeping a good balance of the computational load over the platform. Different from those hardware-dependent methods, our work presented in this paper does not rely on specific Hardware such as GPU and SSE CPUs.

Heterogeneous computing platforms have been employed for biological sequence analysis, which involves ATAC computation. Meng and Chaudhary [2] have used such a platform composed of CPU and FPGA for high-throughput computing. In their platform, a master node splits the whole database into multiple nearly-equal sized fragments and distributes task load proportionally (WFixed) among all worker nodes to achieve load balancing. However, the batching process causes massive data transmissions among the nodes at runtime. This problem will be overcome in our work of this paper.

Recently, computing frameworks have been used to deal with ATAC problems systematically. Galaxy CloudMan [7] is such a batch queue processing system. It enables individual researchers to compose and control an arbitrarily sized compute cluster on Amazons EC2 cloud infrastructure without any informatics requirements. An entire suite of biological tools is available in CloudMan for immediate consumption. In the architecture of CloudMan, all data sets are stored in persistent storage components, and are delivered to different nodes from there. However, as a resource management system, CloudMan does not consider optimization of data distribution explicitly. In comparison, our work in this paper investigates the optimization problem.

CloudBurst [8] is a new parallel read-mapping algorithm optimized for next-generation sequence data. It uses the Hadoop implementation of MapReduce to execute tasks in parallel on multiple compute nodes, thus making it feasible to perform highly sensitive alignments on large read sets. CloudBurst scales linearly as the number of reads increases. It also exhibits a near linear parallel speedup as the size of the cluster increases.

All-pairs designed by Moretti et al. [10] is an abstraction for data-intensive computing of ATAC problems on campus grids. To give each comparison task the required data, a spanning tree method is proposed to deliver the data to each node efficiently. However, each worker node still has to store all the entire data sets in this approach. In comparison, our work in this paper does not deliver the data sets to everywhere.

Though contributions have been made in distributed computing of ATAC problems, limitations still exist in this area. Most existing methods do not pay much attention to the efficiency of data distribution and data locality of comparison tasks. Some widely used distributed computing frameworks, such as Hadoop, are nonetheless inefficient when processing large-scale ATAC comparison problems.

To address the distributed computing of ATAC comparison problems, this paper presents a scalable and efficient strategy for data distribution and task allocation in heterogeneous systems. It is a further development of our previous work reported in [9] that focuses on homogeneous platforms. We will start with problem statement in the following section.

III. 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, single data item in set A, the comparison operation between data items A_i and A_j, and output similarity matrix element, respectively. The ATAC problem is formulated as:

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

Hadoop-based solutions are widely used for distributed computing of ATAC problems, they are inefficient for the ATAC pattern. However, with the focus on the MapReduce pattern, Hadoop is inefficient for the ATAC pattern of the ATAC problems. Challenges of executing ATAC problems by using Hadoop are summarised in the following three aspects:

Challenge 1. Poor data locality of comparison tasks. Good data locality is one of the principles in processing data intensive problems. It is often better to move the computation close to where the data is located [11].

For Hadoop, good data locality for Map tasks are almost promised due to two reasons: 1) Hadoop assumes that a Map task is expected to process a single data split; and 2) In Hadoop's data distribution, each data with a fixed number of replications is randomly distributed into different worker nodes. However, for ATAC problems, each comparison task needs to process two different data items. Therefore, Hadoop's data distribution becomes inefficient.

Let us consider an example of 4 data items and 6 worker nodes. If each data stored in HDFS has 3 copies (default setting), a possible data distribution is depicted in Fig. 1. It is seen from Fig. 1 that poor data locality exists for four comparison tasks $C(1,2), C(2,3), C(1,4)$ and $C(3,4)$. To execute these four tasks, data items must be transferred at runtime among the nodes. For large-scale ATAC problems, moving massive data around at runtime causes significant degradation of the overall computing performance.

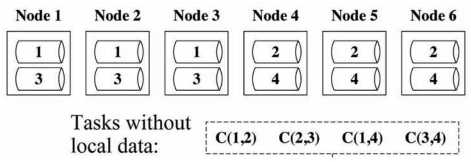

Fig. 1. Poor data locality for comparison tasks.

Challenge 2. Invalid locality-aware scheduling of comparison tasks. Data locality is important in processing a large volume of data sets. In Hadoop, data locality information is used to schedule Map tasks. Map tasks are executed in the worker nodes with required data. However, the data locality for

Hadoop's Map tasks is lost when dealing with ATAC problems. For example, again for the scenario shown in Fig. 1, it is a problem for Hadoop's task scheduler to decide in which worker nodes the tasks $C(1,2), C(2,3), C(1,4)$ and $C(3,4)$ are executed because each of the nodes stores only half of the required data items for each of these comparisons.

Another example is shown in Fig. 2. Assume that two comparison tasks $C(2,4)$ and $C(2,6)$ are ready to dispatch for execution and only work nodes 3 and 4 are idle at the moment. If $C(2,4)$ and $C(2,6)$ are scheduled to worker nodes 3 and 4, respectively, both tasks have local data to complete the execution. However, if these two tasks are swapped between worker nodes 3 and 4, task $C(2,6)$ will have no local data for execution on node 3. Hadoop's task scheduling does not guarantee that the later occasion does not happen.

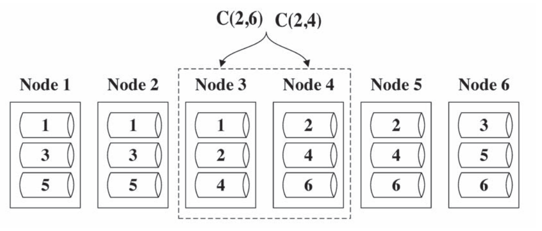

Fig. 2. Task scheduling with globe consideration requires $C(2,4)$ and $C(2,6)$ are allocated to worker nodes 3 and 4, respectively, for data locality when these two nodes are idle for more tasks.

Hadoop is not effective in the above two examples because Hadoop is not designed for ATAC computing pattern. It has a lack of the capability to make a global decision to schedule comparison tasks with consideration of all available idle nodes in ATAC computing. Therefore, task scheduling in Hadoop is not data locality aware in general for ATAC problems.

Challenge 3. Inefficiency in heterogeneous systems. Hadoop inherently assumes all the machines have the same processing capabilities. Though this simplifies the system modeling and implementation, it leads to poor use of the computing resources of a heterogeneous system.

For instance, in a cluster with four worker nodes, two nodes have twice as much computing power as that in other nodes. If there are 18 comparison tasks waiting for execution, each of the four nodes are allocated three tasks first. Then, the remaining six tasks should be evenly distributed to nodes 3 and 4 for the best use of the computing resources. This is clearly shown in Fig. 3. However, Hadoop does not have such a mechanism to dispatch tasks with full consideration of computing resources in heterogeneous systems.

The above three challenges indicate the ineffectiveness of Hadoop's strategy in solving ATAC problems in heterogeneous environments. The following section develops a new data and scheduling strategy to address these challenges.

IV. Data and Task Distribution Strategy

In this section, an abstraction is developed for the requirements of data distribution and task distribution from the challenges discussed in Section III. Then, a heuristic strategy is presented for data distribution and task scheduling for distributed computing of ATAC problems in heterogeneous systems. Examples will also be given for further analysis of the strategy.

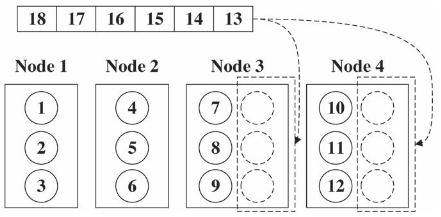

Fig. 3. The best use of computing resources requires tasks 13 to 18 are evenly distributed to worker nodes 3 and 4, which have twice as much computing power as that in the other two nodes.

A. Strategy Targets

To address the challenges discussed in Section III, a data and task scheduling strategy will be designed to meet the following requirements:

1) Data are distributed in a way to enable all comparison tasks to have good data locality (Challenge 1).
2) Comparison tasks are always scheduled with good data locality (Challenge 2).
3) Each of the worker node is assigned comparison task load proportional to its computing power (Challenge 3).
4) For storage saving, data replications are as fewer as possible, and the maximum number of data items among all worker nodes is minimized.

Distributing data items to worker nodes in a heterogeneous system can be treated as a complex combinatorial mathematics problem. The optimization of a combinatorial mathematics problem is difficult, even in a homogeneous system as we have investigated previously [9]. Therefore, a heuristic algorithm will be developed to address this problem in the following.

B. Strategy Design

For the data and task scheduling problem, a reasonable idea is to pre-consider task allocation during data distribution. In this way, the task scheduling will become data-aware. Consequently, runtime data re-arrangement can be avoided.

In actual heuristic task scheduling, the computing tasks are assigned to the work nodes one after another. Therefore, choosing which node to assign a task most appropriately is critical. This paper uses a concept of node completeness to quantify the degree of the appropriateness for a node to have the next task assignment. Then, a full process is designed to assign all tasks to the worker nodes.

Completeness of a node. Let $|T_i|$ denote the total number of comparison tasks that should be allocated to node i. From the load balancing requirement 3) listed above, $|T_i|$ should be proportional to the computing power of node i. As an ATAC problem has a finite number of comparison tasks, $|T_i|$ can be determined before data distribution is conducted. Now, let

$|A_i|$ denote the number of comparison tasks that have already be assigned to node i. The following ratio R_i is defined to describe the completeness of node i:

$$R_i = |A_i| \,/\, |T_i|\,,\ 0 \leq R_i \leq 1 \tag{2}$$

When $R_i = 0$, no tasks have been assigned to node i. If node i has been fully assigned tasks in comparison with all other nodes in terms of their computing power, $R_i = 1$.

With the definition of node completeness, the next task should be assigned to a node with the smallest completeness value among all nodes. Using R_i instead of $|A_i|$ in determination of a node for assignment of the next task will help avoid load imbalance among the nodes in a heterogeneous system.

Rules for Data and Task Distribution. Fig. 4 shows the distribution of a new data d to node i that has already been assigned p data files $(1, 2, \cdots, p)$. The connecting arrows represent new comparison tasks available at node i due to the distribution of the new data. There are two types of comparison tasks: those that have never been allocated (dotted arrows), and those that have already been allocated previously (solid arrows). We have developed rules for both types of tasks:

Rule 1. For the comparison tasks that have never been allocated, allocate as many tasks as possible to node i by meeting the load balancing requirement 3).

Rule 2. For the comparison tasks that have already been allocated previously, re-allocate each of them by meeting the load balancing requirement 3). During data distribution, for a comparison task $C(x, y)$, there could be more than one worker nodes with both data items x and y. In this case, compare the completeness of all these worker nodes and re-allocate the comparison task $C(x, y)$ to the node with the lowest completeness value defined in Eq. (2).

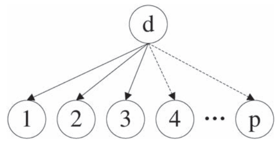

Fig. 4. New available comparison tasks for distributing new data d to node i. They include those that have never been allocated (dotted arrows) and those that have been allocated previously (solid arrows).

For an ATAC problem with M data items, the total number of comparison tasks is $M(M-1)/2$. All these tasks are allocated by meeting the load balancing requirement 3). If the task allocation is further constrained by meeting the data locality requirements 2) and 3) discussed previously, a solution to the problem of data distribution and task scheduling is obtained to fulfil all of our design targets. Furthermore, for each new distributed data file d to node i, by allocating as many newly introduced tasks as possible to node i and re-locating those tasks that have already been allocated before, the total number of data files need to be distributed can be minimized. This tends to give a good solution to meet the requirement 4) discussed before.

C. Strategy for Data Distribution and Task Scheduling

From the discussions above, a heuristic strategy is formally given in the following algorithm for data distribution and task scheduling of ATAC problems in heterogeneous systems:

1) Calculate the number of comparison tasks that should be allocated to each worker node (Requirement 3).
2) Find all unallocated comparison tasks.
3) Find all data items needed for these unallocated tasks; Put them in set I, which is initially empty.
4) From the set I, find the data item which is needed by the greatest number of unallocated comparison tasks. Let d denote this data file.
5) Choose a set of nodes (denoted by C) that
   - do not have the data file d,
   - have the lowest completeness computed from Eq. (2),
   - have stored the least number of data files.
6) Check **Rule 1** in Section IV-B for all nodes in set C. If none of the nodes meet the requirement 3), remove this data file d from set I and go back to Step 4.
7) Find a node i in set C such that the node is empty or can be allocated the largest number of new comparison tasks that are introduced by adding the data file d and have not been allocated before. Distribute data file d to this node i.
8) For comparison tasks that are introduced by adding data file d in Step 7 and have already been allocated to other nodes before, use **Rule 2** in Section IV-B to re-allocate these tasks.
9) Repeating Steps 2 to 8 until all comparison tasks have been allocated.

By following the above steps, allocating all comparison tasks with as less number of data items as possible will help compress the maximum number of data among all worker nodes. Also, allocating comparison tasks according to the computing power of each node leads to good load balancing.

D. Analysis of the Presented Strategy

Consider an example with seven data files numbered $\{0, 1, 2, 3, 4, 5, 6\}$ and five nodes labelled $\{A, B, C, D, E\}$. Assume that worker nodes A and B have twice as much computing power as that in the other nodes. A solution obtained from our strategy is shown in Table I.

TABLE I
DISTRIBUTION OF SEVEN DATA FILES TO FIVE WORKER NODES.

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

It is seen from Table I that the task scheduling solution obtained from our strategy allocates twice as many tasks to each of nodes A and B as those to each of the other nodes. This is consistence with the computing power of the worker nodes. Also, all allocated tasks to each node have good data locality. Therefore, good load balancing can be achieved during actual task execution. This implies that the three challenges discussed in Section III have been well addressed.

V. Experiments

In this section, experiments are conducted to evaluate our strategy for data distribution and task scheduling strategy from three aspects: 1) storage saving and task allocation; 2) somputing performance; and 3) scalability.

A. Storage Saving and Task Allocation

An ATAC problem with a data set with 256 data files is considered in a heterogeneous system. The heterogeneous system is composed of a number of nodes with the number changing between 2 and 64. Half of the nodes (Type B) have twice as much computing power as that of the other nodes (Type A). For Hadoop's strategy, the number of replication is set to the default value 3.

With the number of nodes growing, Table II shows at least how much storage space our strategy and Hadoop's data strategy can save in comparison with the method to distribute the whole date set to every node. The allocation of comparison tasks by using our strategy is also presented in Table II.

TABLE II
Storage usage, storage saving and task allocation.

No. of	Max. no. of files on a node		Storage space saving (%)		No. of tasks on each node (Ours)	
nodes	Our	Hadoop	Ours	Hadoop	Type A	Type B
4	225	192	12	25	5440	10880
8	207	96	19	63	2720	5440
16	163	48	36	81	1360	2720
32	113	24	56	91	680	1360
64	79	12	69	95	340	680

It is seen from Table II that Hadoop saves the storage space the most. But this is achieved with significant sacrifice of data locality [9]. Our strategy also save much storage space, especially when the mount of nodes becomes great. Although our strategy does not save as much storage space as the Hadoop strategy, all allocated comparison tasks from our strategy have 100% data locality. This improves the overall computing performance of the ATAC problem significantly.

Results in Table II also show a static task scheduling obtained from our strategy. Worker nodes of B Type with more processing power are allocated more comparison tasks. This is not implicitly considered in Hadoop's strategy.

B. Computing Performance

Both computing performance and scalability are important in processing big data problems in heterogeneous systems. Experiments are designed with a real ATAC application to evaluate our data and task distribution strategy.

A heterogeneous cluster with 5 machines is built for experiments. All the machines run 64-bit Redhat Enterprise Linux. One node acts as the master node, the other four are worker nodes. Two of the worker nodes have dual cores and 64 GB RAM, while the other two worker nodes have a single core and 32 GB RAM.

We have implemented a CVTree application, which is a typical ATAC problem in bioinformatics [12]. In the CVTree problem, a set of dsDNA viruses files from the National Center for Biotechnology Information [13] are chosen as the input data. A sequential version of the CVTree computing is also developed as the basis for scalability evaluation.

Fig. 5 shows comparisons of the computation time performance between our strategy and Hadoop's one. By considering the three requirements summarized in Section IV-A, our data and task distribution strategy achieves much higher computing performance than Hadoop's strategy. This is because our strategy guarantees good data locality and load balancing. However, Hadoop's strategy causes runtime movement of a large amount of data during computation. Also, it does not make full use of the computing resources in the heterogeneous environment, leading to poor load balancing. Therefore, the computation time performance of the ATAC problem by using Hadoop's strategy becomes much poorer.

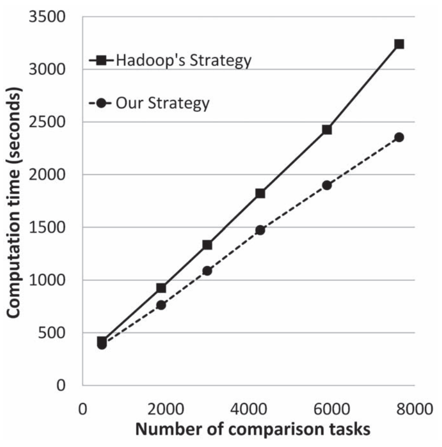

Fig. 5. Comparisons of the computation time performance.

The good load balancing of our strategy is confirmed by the experimental results shown in Fig. 6. This results from the fact that the worker nodes are allocated the number of comparison tasks proportional to their respective computing power. Therefore, the computing resources of the heterogeneous system are fully utilized.

C. Scalability

Scalability is the capability of a system to have a linearly increased performance when the system ensemble size is scaled up. It is widely used to predict the performance of distributed systems at a large system size based on their performance at small size [14]. In general, a system exhibits a linear speedup as the number of processors increases if the communication overhead, load imbalance and extra computation are not considered [15]. In Fig. 7, the 1:1 linear speedup in dot line can be considered as an ideal speedup.

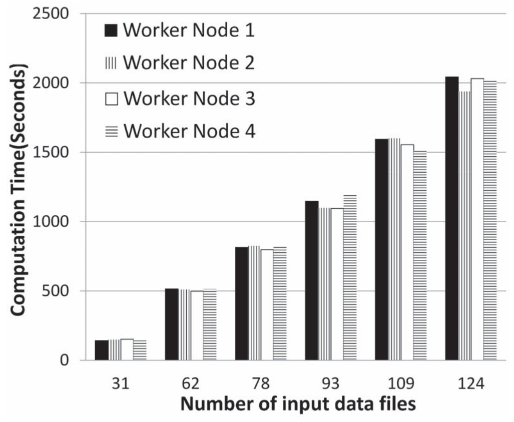

Fig. 6. Demonstration of load balancing from our strategy.

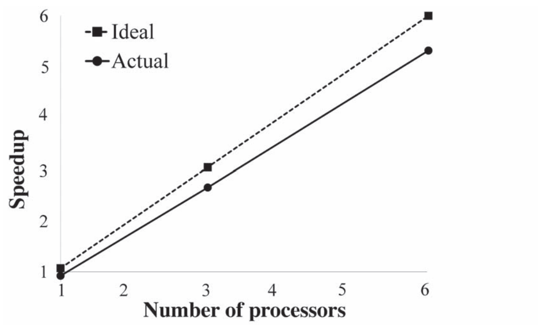

Fig. 7. Speed-up achieved by the data and task distribution strategy.

Results from our experiments are depicted in Fig. 7. It is seen from Fig. 7 that that with the increase of the number of processors, our data and task distribution strategy achieves a linear speedup. This implies that though ATAC problems incur inevitable costs in network communications, memory and disk I/O, good scalability is still achieved on the overall distributed computing. Our strategy presented in this paper achieves about 88.5% of the performance capacity of the ideal 1:1 linear speedup. The is measured by $5.31/6 = 88.5\%$ at 6 cores from the results shown in Fig. 7.

VI. Conclusion

A new data and task distribution strategy has been presented in this paper for distributed computing of ATAC problems with big data in heterogeneous systems. It is designed to distribute as less number of data items as possible to the worker nodes while still addressing the challenges of data locality and load balancing of comparison tasks. Experiments have been conducted to show the good results of the data distribution and task scheduling from the strategy. Further improvement of the quality of the solution to the problem of data distribution and task scheduling will be our future work.

References

[1] S. Sav, G. J. F. Jones, H. Lee, N. E. OConnor, and A. F. Smeaton, "Interactive experiments in object-based retrieval," in *Int Conf on Image and Video Retrieval*, Tempe, AZ, USA, 13-15, July 2006, pp. 1–10.

[2] X. Meng and V. Chaudhary, "A high-performance heterogeneous computing platform for biological sequence analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 9, pp. 1267–1280, 2010.

[3] F. M. Mendonca and A. C. M. A. de Melo, "Biological sequence comparison on hybrid platforms with dynamic workload adjustment," in *27th Int Symp on Parallel & Distributed Processing Workshops and PhD Forum*. Cambridge, MA: IEEE, 20-24 May 2013, pp. 501 – 509.

[4] A. Singh, C. Chen, W. Liu, W. Mitchell, and B. Schmidt, "A hybrid computational grid architecture for comparative genomics," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 2, pp. 218 – 225, March 2008.

[5] S. Kedad-Sidhoum, F. Mendonca, F. Monna, G. Mounie, and D. Trystram, "Fast biological sequence comparison on hybrid platforms," in *43rd Int Conf on Parallel Processing*. Minneapolis: IEEE, 9-12 Sept. 2014, pp. 501 – 509.

[6] http://hadoop.apache.org, 2005, accessed: 3 July 2014.

[7] E. Afgan, D. Baker, N. Coraor, B. Chapman, A. Nekrutenko, and J. Taylor, "Galaxy cloudman: delivering cloud compute clusters," *BMC Bioinformatics*, vol. 11, no. 12, pp. 4–10, 2010.

[8] M. C.schatz, "Cloudburst: highly sensitive read mapping with mapreduce," *Bioinformatics*, vol. 25, no. 11, pp. 1363–1369, Apr. 2009.

[9] Y.-F. Zhang, Y.-C. Tian, C. Fidge, and W. Kelly, "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 Oct - 1 Nov 2014.

[10] C. Moretti, H. Bui, K. Hollingsworth, B. Rich, P. Flynn, and D. Thain, "All-pairs: An abstraction for data-intensive computing on campus grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 33–46, 2010.

[11] M. Khan, Y. Liu, and M. Li, "Data locality in hadoop cluster systems," in *11th International Conference on Fuzzy Systems and Knowledge Discovery*. Xiamen: IEEE, 19-21 Aug. 2014, pp. 720 – 724.

[12] B. Hao, J. Qi, and B. Wang, "Prokaryotic phylogeny based on complete genomes without sequence alignment," *Modern Phy. Lett.*, vol. 2, no. 4, pp. 14–15, 2003.

[13] NCBI, "National Center for Biotechnology Information," http://www.ncbi.nlm.nih.gov/, accessed: 3 July 2014.

[14] X.-H. Sun, Y. Chen, and M. Wu, "Scalability of heterogeneous computing," in *Int Conf on Parallel Processing*. Oslo, Norway: IEEE, 14-17, June 2005, pp. 557 – 564.

[15] K. Li, Y. Pan, H. Shen, and S. Zhang, "A study of average-case speedup and scalability of parallel computations on static networks," *Mathematical and Computer Modelling*, vol. 29, no. 9, pp. 83–94, May 1999.