# Data Replication-Based Scheduling in Cloud Computing Environment

Bahareh Rahmati[1], Amir Masoud Rahmani[2]

Received (2016-02-02)
Accepted (2017-02-14)

*Abstract* - **High-performance computing and vast storage are two key factors required for executing data-intensive applications. In comparison with traditional distributed systems like data grid, cloud computing provides these factors in a more affordable, scalable and elastic platform. Furthermore, accessing data files is critical for performing such applications. Sometimes accessing data becomes a bottleneck for the whole cloud workflow system and decreases the performance of the system dramatically. Job scheduling and data replication are two important techniques which can enhance the performance of data-intensive applications. It is wise to integrate these techniques into one framework for achieving a single objective. In this paper, we integrate data replication and job scheduling with the aim of reducing response time by reduction of data access time in cloud computing environment. This is called data replication-based scheduling (DRBS). Simulation results show the effectiveness of our algorithm in comparison with well-known algorithms such as random and round-robin.**

*Index Terms* - **Cloud Computing, Data Access Time, Data Replication, Job Scheduling, Response Time**

1- Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran (bahar_rahmatii@yahoo.com)
2- Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

## I. INTRODUCTION

Nowadays data-intensive applications play a prominent role in different scientific fields. These applications produce huge amounts of data. This ever-growing amounts of data affect the systems' performance. Besides the delay of wide area networks and the limitation of network bandwidth make it difficult to access the data. Consequently, they lead to a bottleneck for the whole cloud workflow system. This bottleneck is more sensible when many parallel tasks simultaneously require accessing the same data file on one data center [1].

Since the latency involved in accessing data files increases the jobs' response time, data management is of vital importance [2].

"Data replication" is an appropriate technique to manage data files. Data replication is to create multiple copies of data in multiple sources in order to reduce access time and bandwidth consumption. It also guarantees data reliability and load balancing for the system [3].

Data replication and job scheduling are two effective techniques that can enhance the performance of data-intensive applications.

In one hand, scheduling the jobs without replicating the data files that are required for them needs remote access to data files. Accessing remotely to these data files requires more time than accessing directly so; doing scheduling without replication imposes an overhead of data access time to the system. On the other hand, doing replication without scheduling the jobs fails to enhance the performance of the system in an effective way because moving large-sized data files costs more bandwidth and needs longer time for transfer.

Furthermore, based on new features of cloud, comparing to the traditional distributed computing systems like data grid, a cloud computing system is more cost-effective from different aspects.

Since the data centers on cloud computing systems contain clusters of commodity hardware it is more scalable and cost-effective to provide massive storage and high-performance computing for data-intensive applications [4].

However, a lot of algorithms have been presented in data-intensive applications' field. Most of these algorithms consider job scheduling and data replication as two independent techniques for boosting the grid systems' performance. What distinguishes our algorithm from other algorithms is based on two viewpoints: first, integrating job scheduling and data replication for reducing job response time and increasing load balancing of the system. Second, considering cloud computing is a better and more cost-effective platform for data-intensive applications than data grid. These aspects offer less response time and more load balancing to the system.

The remainder of the paper is as follows. Section 2 presents the related work. Section 3 describes the proposed method. Section 4 shows the simulation results and the final section is the conclusion and future works.

## II. RELATED WORKS

In [5], a Pre-fetching based Dynamic Data Replication algorithm (PDDRA) is presented. PDDRA pre-replicates files based on file access history of the grid sites. This strategy improves job execution time, network usage, hit ratio and storage usage. But the best replica selection has not been studied in this paper. In [6] the authors presented an algorithm which considers the number of file requests and response time to place the replica in the best site within the cluster. By this way, mean job execution time is minimized. In [7] a Bandwidth Hierarchy based Replication (BHR) is presented. The algorithm decreases the data access time by maximizing network-level locality and avoiding network congestions. BHR strategy performs well only when the storage capacity is limited. In [8], Modified BHR is presented. This strategy replicates a file that has been accessed most and it is probable to be used in near future.

In [9], different replica placements are discussed. The authors consider a set of parameters such as access cost, bandwidth consumption and scalability for evaluating the different replica placement strategies. In [10], replica placement in a hierarchical data grid is studied. The authors consider the hierarchical data grid as a tree structure. They attempted to avoid unnecessary replications and enhance the performance of the system by a new labeling scheme called Dewey encoding. The authors believed that frequent insertion can be optimized. Moreover, replicas' placement can be modified to guarantee service quality and it is proper to boost the system performance with a new file placement algorithm. In [11], a replacement algorithm for data grid is proposed. This algorithm has two levels and uses fuzzy logic to evaluate the value of the replicas. This algorithm enhances the performance of the system by minimizing the jobs' execution time and the numbers of replications.

In [12], research issues in cloud data management for scientific workflow systems are discussed. Three research directions are data storage, data placement and data replication. In each direction, the existing research problems are analyzed and promising methodologies are introduced. In [13], the importance of migrating scientific workflow management systems from traditional grid to cloud is focused. It shows that cloud computing offers a cost-effective solution for data-intensive applications. Furthermore, cloud computing systems provide a new platform for scientists from all over the world to do their research together. The authors have integrated Swift scientific workflow management system with the Open Nebula cloud platform, which supports high-throughput task scheduling and efficient resource management in the cloud. They are working on an interface that will improve the integration of Swift with other Cloud platforms such as Amazon EC2 and Open Stack for future. In [14], the authors investigated the QOS data replication for big data applications in cloud computing. To improve the performance of the cloud computing QOS-aware data replication algorithm is proposed. This algorithm operates based on the idea of minimum-cost maximum-flow problem. The algorithm reduces the total replication cost and the total time. Reducing storage space and energy consumption is the case that can be improved about this algorithm.

## III. THE PROPOSED ALGORITHM

### *Problem Assumptions*

The system contains independent jobs. Each job is composed of dependent tasks and requires an input data file. Therefore each job requires a set of input data files for its execution. It is possible that a data file in a job being used by more than one task of that job.

Data files have different sizes, they are read only and they cannot be modified.

Jobs execute in data centers. There is a database for each data center which keeps the data files (replicas) of the data center. The database of each data center is placed in that data center.

Since each job requires a set of data files. The data files are distributed on data centers. The data file distribution is random proportional to bandwidth and capacity of the data centers.

Each data center is connected to other data centers through cloud network links. Each link has a specific bandwidth and data files are sent sequentially (in a queue) over network links. Jobs are executed simultaneously in the data centers with considering the data center workload. Job response time is proportional to its data access time.

Each data center has an independent scheduler which keeps the neighbor data centers' information.

The DRBS goal is to reduce job response time by reduction of data access time.

### *Data Replication-Based Scheduling*

DRBS algorithm contains three steps. These three steps are summarized as follows:

1) Scheduling the jobs with considering the location of required input data files.

2) Data replication in data centers.

3) Replacement of the data files with the arrival of new jobs and lack of storage capacity.

#### *1. Scheduling*

In scheduling part of the algorithm when a job arrives at the system an appropriate virtual machine (VM) is chosen as below:

The job is sent to a data center with maximum files matching number (maximum number of data files for the job available at the data center) in compare with neighbor data centers. This helps to reduce data access latency involved in moving the data file which is not present in the scheduled data center.

If more than one data center has the same condition in terms of files matching number "job access time" is checked. Job access time is the cost of moving the data files that are required for the job to the scheduled data center. To calculate job access time first, we calculate the access time of each required data files for a job then we get the sum of all "access times". Finally, the job is sent to the data center with minimum job access time. Job access time calculation is shown by (2).

$$\text{Access time } (f_i) = \frac{S(i)}{Bw(i)} \quad (1)$$

*S(i)*: size of file (fi).
*Bw(i)*: bandwidth.

$$\text{Job access time (j)} = \sum_{i \epsilon Mj} access\ time(f_i) \quad (2)$$

$J_i \epsilon J$, *J*: set of jobs.
$M_j$ : set of file indexes in $j$, *i*: file number.

It is probable that when data replicated to data centers more than one data center contains the requested data file. In this situation, minimum access time is accepted.

Finally between different virtual machines in a data center the job schedules to a virtual machine with less queuing size with considering the number of jobs to the number of processors.

If there was any unscheduled job at this point we repeat the mentioned steps. Fig. 1 shows the pseudo code of scheduling algorithm.

```
Algorithm of scheduling
Input: J: set of jobs.
Output: all jobs sequenced to an appropriate virtual machines.
Description:
1. for each $J_i \epsilon J$
2. Send the job to a DC with maximum files matching number
3. if more than one DC exists with same file matching number
Send the job to a DC with less job access time
//Access time ($f_i$) = $\frac{S(i)}{Bw(i)}$ , Job access time (j) = $\sum_{i \epsilon Mj} access\ time(f_i)$
//S(i)= file size, Bw(i)= bandwidth
4. end if
5. In each DC send the job to a virtual machine with less waiting time
6. end for
```

**Fig.1. Scheduling algorithm**

#### *2. Data Replication*

When jobs scheduled to the appropriate data centers we replicate data files as follow:

First, we list the required data files for all jobs in a data center that do not exist in that data center. Then we calculate file frequency of each data file in the list.

"File frequency" is the number of times that tasks of one job require a shared data file and sum of all jobs' file frequencies for each file in one data center is called "demand".

$$\text{Demand}\ (f_i) = \sum_{j \in M} \text{file frequency}\ (j, i) \qquad (3)$$

$J_i \in J$, $J$: set of jobs.
$M_j$ : set of file indexes in $j$, $i$: file number.

We calculate the demand of files in the list and also the access time of each file. In this step we compute the multiplication of these two parameters as in (4):

$$\text{Demand}\ (f_i) * \text{Access time}\ (f_i) \qquad (4)$$

We sort the required data files in descending order according to the multiplication results. At last, we replicate the sorted data files till the storage is full. Note that we will not have more than one replica of a data file in each data center after replication. Fig. 2 shows the replication algorithm.

```
Algorithm of replication
---------------------------------------------------------------------------
Input: L: list of files that are required by jobs in each DC but
don't exist in it.
Output: replicated files.
Description:
//i = file index
//M = job indexs in each DC
1: for each DC
2: for i ϵ L
3:calculate Demand(f_i)
// Demand (f_i) = $\sum_{j \in M}$ file frequency (j, i)
4:y_i = Demand (f_i) * Access time (f_i) // Access time (f_i) = $\frac{S(i)}{Bw(i)}$
5:end for
6: sort y_i in descending order
7: Replicate files according to their orders.
8: if (new jobs arrive && storage is full)
9: Replace files          //replacement algorithm
10: end if
11: end for
```

**Fig.2. Replication algorithm**

*3. Replacement*

With the arrival of new jobs and lack of storage capacity replacement is done as follow:

We consider two important factors for choosing a new replica and replacing it with the old ones. These factors are access time and demand. If the value in (4) for the new replica be greater than the mean of this multiplication for all the existing replicas on the data center, the new replica is a candidate for replacement.

Note that access time for the existing replica in here is the minimum access time from the nearest data center which contains the data file.

The candidate data file for deletion is chosen by (4) but in ascending order. It is probable that one or more replicas need to be removed from the storage in order to store new replica. In this condition, we compare the importance of the new replica with the replicas which are supposed to be deleted (Lines 13 and 14 of the replacement algorithm). Fig. 3 shows the replacement algorithm.

```
Algorithm of replacement
--------------------------------------------------------------------------------
-----
Input: L: list of new files
Output: Replaced files
1: for x ϵ DC
2: calculate p= $\sum_{i \in S}$(Demand (fi) * min(Access time (fi) from nearest DC))
//S= list of file indexes that exist in that DC, Access time (f_i) = $\frac{S(i)}{Bw(i)}$
3:for t ϵ L
4: if Demand(f_t) * Access time (f_t) is greater than mean(p)
// (Demand (f_t) * Access time (f_t) > $\frac{P}{n}$), n= number of existing
files in DC
5: Candidate this file for replacing in that DC
6:end if
7:end for
8: calculate y_i = (Demand (f_i) * min (Access time (f_i) from nearest
DC)
// Access time (f_i) = $\frac{S(i)}{Bw(i)}$, i= file indexes that exist in that DC
9: sort y_i in ascending order
10: Replace the file with highest priority in sorted form with f_t
11: if there is not enough space for replacement
12: Select files for removing in sorted list till enough space is
available
13:if (Demand (f_t) * Access time (f_t)) is greater than sum of
(Demand (f_k) * Access time (f_k))
// k =list of selected file indexes in sorted list
14:Replace new file with selected files
15:end if
16: end if
17:end for
```

**Fig.3. Replacement algorithm**

## IV. SIMULATION AND RESULTS

In order to demonstrate the performance improvement of DRBS, we used "MATLAB R2014a" to evaluate our algorithm.

***Simulation Environment Assumptions***

The parameters values in the simulation appear as below:

Our simulation platform contains 12 to 100 independent jobs with 8 to 80 different input data files. The size of data files is in the range of

[100 GB-900 GB]. There are 4 data centers and 24 virtual machines in our system. Instruction number per job is in the range of [3000-7000] instructions and VM MIPS are in range of [2000-3000]. The data centers have 1.5TB to 28TB storage capacities and 50 TB/S to 300 TB/S bandwidths.

Response time and standard deviation are our evaluation criteria.

We compare DRBS with the time when no replication is used in this algorithm and also standard algorithms: random and round-robin (with and without replication). As mentioned DRBS algorithm contains two parts: scheduling and replication. We refer to the scheduling part as DRBS without replication.

***Simulation and Results***

Fig. 4 shows the mean job response time of the three scheduling strategies when no replication is used for 12 to 100 jobs with 8 to 80 different input data files on 4 data centers. The mean response time improvements of DRBS without replication in comparison with random and round-robin algorithms are 41.94% and 37.80%.

This is because in the proposed algorithm, at first step of scheduling each job schedules to a data center with maximum files matching number. Hence as we expected the job response time is reduced by reducing data movement between data centers.

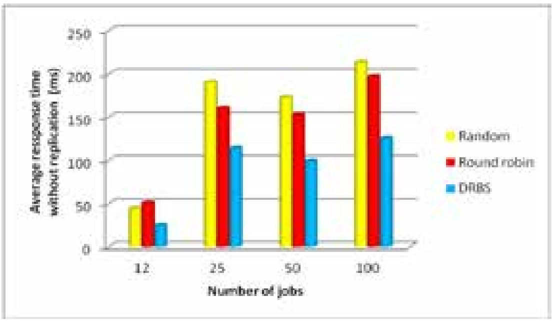

**Fig.4. Mean jobs response time of three strategies without replication**

Fig.5 also shows this comparison when data are replicated to data centers. In this case, the DRBS mean response time improvements increased to 55.67%, 51.74% in comparison with random round-robin algorithms. This is justified by the primary scheduling strategy and considering the files that are required by the jobs.

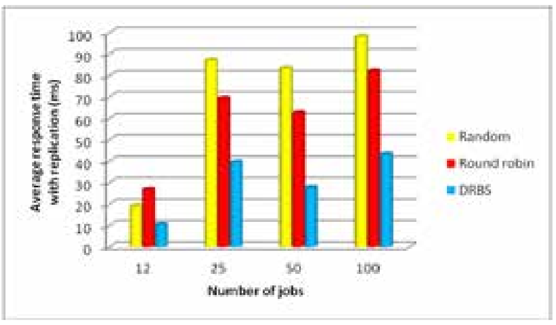

**Fig.5. Mean jobs response time of three strategies with replication**

Fig. 6 and 7 show the comparison of STD (standard deviation) of the three strategies (with and without replication). The STD of DRBS is less than others which indicate better load balancing of our algorithm.

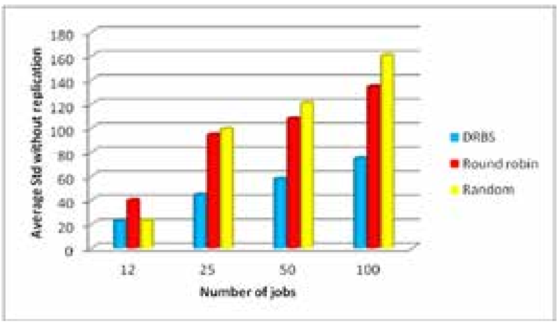

**Fig.6. Mean job STD of three strategies without replication**

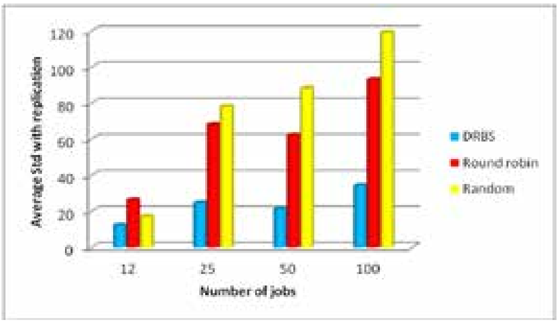

**Fig.7. Mean job STD of three strategies with replication**

## V. CONCLUSION

In this work, we proposed data replication based scheduling algorithm. The DRBS integrates scheduling and replication with the aim of reducing response time by reduction of data access time in cloud computing environment. We compared the DRBS strategy with well-known algorithms: random and round-robin. Experimental results demonstrate that our DRBS algorithm can achieve a significant improvement over random and round-robin algorithms in case of response time and load balancing which indicates the effectiveness of our algorithm.

In future, we will propose a dynamic data replication algorithm and we will also consider a threshold for replicating data. Since the DRBS algorithm focuses on scheduling independent jobs; the algorithm can be extended to incorporate job dependencies in workflow environment and data files can be pre-fetched on the data centers based on the dependency between data files.

## REFERENCES

[1] Djebbar, E.I. and Belalem, G., 2013, December. Optimization of tasks scheduling by an efficacy data placement and replication in cloud computing. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 22-29). Springer International Publishing.

[2] Mansouri, N., 2014. A threshold-based dynamic data replication and parallel job scheduling strategy to enhance Data Grid. Cluster Computing, 17(3), pp.957-977.

[3] Ma, J., Liu, W. and Glatard, T., 2013. A classification of file placement and replication methods on grids. Future Generation Computer Systems, 29(6), pp.1395-1406.

[4] Yuan, D., Yang, Y., Liu, X. and Chen, J., 2010. A data placement strategy in scientific cloud workflows. Future Generation Computer Systems, 26(8), pp.1200-1214.

[5] Saadat, N. and Rahmani, A.M., 2012. PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids. Future Generation Computer Systems, 28(4), pp.666-681.

[6] Sashi, K. and Thanamani, A.S., 2010. Dynamic replica management for data grid. International Journal of Engineering and Technology, 2(4), p.329.

[7] Park, S.M., Kim, J.H., Ko, Y.B. and Yoon, W.S., 2003, December. Dynamic data grid replication strategy based on Internet hierarchy. In International Conference on Grid and Cooperative Computing (pp. 838-846). Springer Berlin Heidelberg.

[8] Sashi, K. and Thanamani, A.S., 2011. Dynamic replication in a data grid using a Modified BHR Region Based Algorithm. Future Generation Computer Systems, 27(2), pp.202-210.

[9] Souri, A. and Rahmani, A.M., 2014. A survey for replica placement techniques in data grid environment. International Journal of Modern Education and Computer Science, 6(5), p.46.

[10] Rahmani, A.M., Fadaie, Z. and Chronopoulos, A.T., 2015. Data placement using Dewey Encoding in a hierarchical data grid. Journal of Network and Computer Applications, 49, pp.88-98.

[11] Saadat, N. and Rahmani, A.M., 2016. A Two-Level Fuzzy Value-Based Replica Replacement Algorithm in Data Grids. International Journal of Grid and High Performance Computing (IJGHPC), 8(4), pp.78-99.

[12] Yuan, D., Cui, L. and Liu, X., 2014, August. Cloud data management for scientific workflows: Research issues, methodologies, and state-of-the-art. In Semantics, Knowledge and Grids (SKG), 2014 10th International Conference on (pp. 21-28). IEEE.

[13] Zhao, Y., Li, Y., Raicu, I., Lin, C., Tian, W. and Xue, R., 2014. Migrating Scientific Workflow Management Systems from the Grid to the Cloud. In Cloud Computing for Data-Intensive Applications (pp. 231-256). Springer New York.

[14] Vijaya-Kumar-C, D.G., 2014, Optimization of Large Data in Cloud computing using Replication Methods. International Journal of Computer Science & Information Technologies, 5(3).