# A Hierarchical Approach to Improve Job Scheduling and Data Replication in Data Grid

Somayeh Abdi[1] and Sayyed Hashemi[2]
[1]Department of Computer Engineering, Eslamabad_E_Gharb Branch, Islamic Azad University, Iran
[2]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Iran

**Abstract:** *In dynamic environment of data grid effective job scheduling methods consider location of required data in dispatching jobs to resources. Also, job scheduling methods are combined with data replication mechanisms to reduce remote data access as well as save network bandwidth. In this paper, we combine job scheduling method and dynamic data replication to reduce data access delay and job execution time. Also, we expand our work by applying bloom filter in job scheduling decision. In data grid, appropriate mechanisms for recording, deleting and inquiring information about data files are required for implementing proper job scheduling method. Therefore, we apply counting bloom filter for recording/deleting and inquiring information about data files in Replica Catalogue (RC). Result of simulation indicates that proposed job scheduling and data replication methods reduce job execution time, also using bloom filter saves network bandwidth and reduces time of gathering information for selecting appropriate resources in job scheduling.*

**Keywords:** *Job scheduling, data replication, RC, replica manager, resource discovery, counting bloom filter.*

*Received August 11, 2012; accepted March 19, 2014, published online June 26, 2014*

## 1. Introduction

Scientific applications generate large amount of data that could not be stored in centralized manner. Data grid is best solution for managing and storing huge amount of data. A data grid is a distributed collection of computers and storage resources maintained to enable resource sharing and coordinate problem solving in dynamic, multi institutional virtual organizations [11]. Data intensive applications are dependent on large amount of data; in this applications job scheduling and data management are most important because the performance of executing these applications depends on data access time. Replication mechanisms are applied to manage data; in data management field, the replication mechanism determines which file should be replicated, when to create new replicas and where the new replicas should be placed [5]. The Grid Scheduler (GS) is one of the most critical components of the resource management systems, since it has the responsibility of assigning resources to jobs by considering the job's requirements and resources status [20]. There are different type of data replication and scheduling strategies [14]. Trust-based, market-based and performance-based strategies are important type of scheduling. We use performance-based scheduling to decrease data access time and job execution time. The data GS approach has three phases: Resource discovery, resource selection and job execution. Resource discovery identify resources that can be used with their capability when the data GS has to find a suitable set of resources for job execution [15]. For job scheduling decision, job scheduler should get information about the location of required data. Since, time of inquiring and accessing to data is the primary cause of job execution delay in data grid. Applying efficient method for file indexing and querying can reduce time of gathering information for scheduling decision. From another hand, data replication mechanisms are applied in data grid for reducing data access time. Most applications require special batch files. We define a batch files as a set of files that are accessed in executing special application with each other. In data replication structure, replica manager and Replica Catalogue (RC) are important components. Replica manager manages data movement between storage resources. RC indexes data files that are stored in resources. For each file, RC keeps a list of resources stored it. Lists can be used to keep characteristic of elements in the network, but searching in list is too time-consuming and resource discovery by using them has very low efficiency [15]. Bloom filter is an alternative to store and search within characteristic of a set of elements, representing a method to keep characteristic of elements which reduces space utilization [19]. In this paper we develop centralized job scheduling and distributed dynamic data replication that consider the changes of data access in the grid environment and it automatically creates new replicas for popular data files or moves the replicas to other sites when is needed to improve the performance. Also, we apply counting bloom filter to reduce resource discovery time in the first phase of scheduling.

## 2. Related Work

There are some related works on job scheduling or data replication in data grid as well as combining them. Our previous work [2] only considers the most

available data at site level and does not distinguish the importance of hierarchical bandwidth in Scheduling Strategy (SS) in real networks. Literature [1] proposed two level job scheduling and data replication algorithms, but the whole computing for selecting the best cluster and site has been done at GS and GS became a bottleneck point in grid system and could not tolerate in increasing the resources and users. From other hand, centralized structure of RC decreases the scalability of handling more data files. In [17] an algorithm for a 2-level hierarchical structure based on internet hierarchy has been introduced which only considers dynamic replication and does not consider scheduling. Literature [8] organized the data into several data categories and this information is used to help improving data replication placement and job scheduling decision. Resource SS is based on the estimation of time of executing a job in each grid site. Literature [22] proposed centralized and distributed dynamic replication. Also, it used shortest turnaround time and data present as scheduling policies and evaluated combination of scheduling and replication strategies. In [12] bloom filter is used to implement service discovery protocol for ad hoc networks. Literature [18] used the bloom filter data structure for memory efficiency. In this paper, we develop job scheduling and data replication as well as applying bloom filter to improve job execution in data grid.

## 3. The Proposed Structure

In this section, we propose hierarchical network structure and apply two strategies for job scheduling and data replication base on the hierarchical network structure.

### 3.1. Network Structure

We depict our structure in Figure 1. In this structure we categorized sites into different regions and defined internal link as a link inside a region and external links as links between different regions. Each site comprised of several storage nodes that store replicas and process nodes that process data operations. Also, for each site we assign a replica manager to control data replication and deletion and Local Scheduler (LS) to manage CPU time. Other essential components in this structure are GS, Data Grid Information System (DGIS) and RC. At the start of establishing the network each site registered itself in DGIS so, GS can query DGIS for available resource and resource status information. We add RC for each region to keep track a list of available replica in the region. GS query RCs for available data and location of them in the regions. At each site replica manager provide a mechanism for accessing the RC on local region. When each site store a new replica, RM send a replica register request to RC and it add this site to the list of sites that hold the replica. Also, when RM deletes a replica in the site, send a replica delete request to local RC at region and it deletes this site from list of sites that hold the replica. All jobs submitted to GS and for each job, GS according to the job scheduling policy and information that gets from RC and DGIS select appropriate region and site respectively and then submits the job to LS in the selected site. When a job assigned to LS, RM at site is responsible for preparing data for job execution.

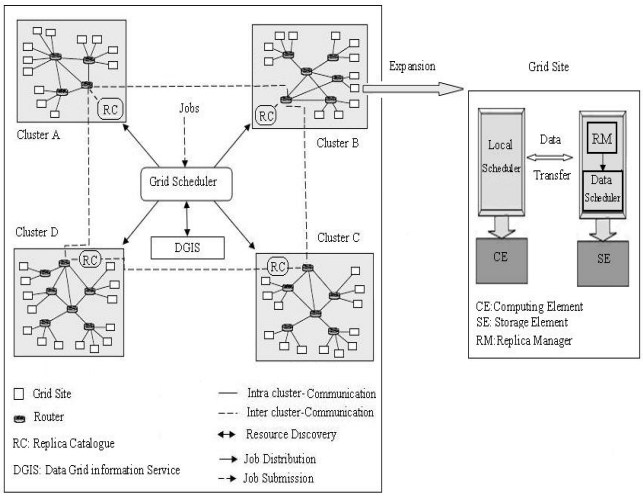



Figure 1. Data grid architecture.

### 3.2. Scheduling and Replication Policy

In proposed structure we apply centralized job scheduling that use the overall information of data grid system for job assignment. We apply distributed RC to eliminate the bottleneck and increase the scalability in this structure. RC in regions tolerates the load of calculating available data on regions and sites. So, GS acts as a coordinator for job assignment.

SS considers the locations of required data at region and site level in job assignment decision. Algorithms 1 and 2 indicate pseudo code at GS and Regional RC sides. According to Algorithm 1 at First SS determines the appropriate region (with the most available data); this will reduce the external link congestion and remote data access as well as save network bandwidth. Then, it selects the appropriate site in the elected region. It will diminish the amount of internal data movement too. When a job is accepted to data grid environment the GS sends job's required data to all RCs in regions. Each RC by comparing the required data with list of available data computes the size of available data and sends the result to GS, so GS select the region with maximum available data. After selecting the best region, GS send the query to associated RC and it computes available data for each site and sends result for GS to select the site with the most available data. Finally, when a job is submitted to LS at selected site, RM according to replication policy manages required data transferring for job execution. DRS as the replication strategy determines which replica will be transferred for job execution and how to handle this new replica. DRS avoid remote data access and external link communications. Our DRS has two phases: DRS select the replica for

transferring to local site and DRS decide how to hold this new replica. For the first phase, DRS search replica in local region that job should be executed. If the replica does not exist in this region, DRS search other regions and select the replica from the region that the maximum external link bandwidth between local region and it is established. In second phase, if there is enough space in the local site, new replica is stored, otherwise DRS delete the replicas in are existence in local region to make space for new replica. Therefore, DRS distinguish between internal and external replicas and avoid from deleting the replica that did not exist on the local region before and it has been replicated from other regions recently. BHR strategy, in the first phase, searches all regions to find the replica and did not distinguish between local region and external regions. In the second phase, BHR deleted files based on LFU and did not consider region level data locality.

*Algorithm 1: SS* (*Job J*)

```
Required_Data=Get_required_data(Job j);
RCs[n]= Get_Available_RCs(From DGIS)
#n indicates number of regions
   for (i=0 to n-1)
      {   Region_available_data[i]=Inquiry_RC-
          Total_data(RCs[i], Required_Data);
      }
      index = Select_Best-Region(Region_available_Data[n]);
      Site-available_data[m]=Inquiry_RC-Site-data(index,
      Required_Data, Job j)
      Resource_id=Select_Best_Site(Site-available_data[m])
      Submit(job j, region index, resource Resource_id);
```

*Algorithm 2: SS_communication_ReplicaCatalogue(Job J Data)*

```
Inquiry_RC-Total_data (Required_Data);
{
      Size_of_available_data=Compare-with-available-
      data(Required_Data, List_of_data);
      Send_Total_available_data(GS), Size_of_available_data)
}
Inquiry_RC-Site-data(Required_Data)
{
  # m indicates number of sites in selected region
       for( i=0 to m-1)
      {
          available_data-on-
          sites[i]=Compare(List_Available_data site[i],
          Required_Data);
       }
        Send_available_data-on-sites(GS, available_data-on-
sites[m])
}
```

## 4. Applying Counting Bloom Filter to Proposed Structure

Bloom filter can be used to register membership of elements and investigate their membership. We develop counting bloom filter to improve categorized data (batch files) indexing and inquiring.

### 4.1. Data structure of Counting Bloom Filter

Bloom filter consist of array with n bits that are zero initially. In this structure, characteristic of elements are coded by using hash functions and these codes are used to register membership of elements into bloom filter array [4]. In grid systems, efficiency of resource discovery mechanisms with bloom filter depends on size of bloom filter, number of resources as well as number and type of hashing functions. Thus, bloom filter can be employed in cases where a huge amount of information is sent in order to record characteristic or to request resources within the network. For implementing element's membership in bloom filter, $k$ different hash functions are applied on element's characteristic. These hash functions create $k$ numbers between 1 and bloom filter size (n) for any element $xi$. Then, these numbers are used as indexes of array locations for element $xi$. During process of registering element $xi$ membership in bloom filter, zero bits in specified indexes are setting to 1. In contrast, in process of removing element $xi$ membership from bloom filter, the bits of specified indexes are setting to zero. Primary defect of this method is that content of indexes of element $x_i$ may be changed during registering other elements in bloom filter. This problem is called "positive error" in bloom filter. From other hand, the content of element $xi$ indexes may be changed in removing other elements in bloom filter. This problem is called "negative error" in bloom filter. To avoid these errors "counting bloom filter" is proposed. Counting bloom filter contains an array of n counters that all cells are zero initially. During process of registering element's membership in bloom filter, one unit is added to counters of these locations while in process of removing elements membership from bloom filter, one unit is reduced from mentioned location counters. During the process of replying the requests by scheduler, bloom filter structure is searched to find resource(s) with recorded characteristics. In the process of inquiring information, counters of the locations obtained from hash functions are checked to see whether counters are zero or not. If one or more counters are zero, then, absolutely this element does not exist in the set [3]. But, if all counters are not zero, then this element exists in the set probably. It is slightly probable that tested counters are not zero due to addition of different elements. Therefore, we will have a positive error. In order to avoid positive errors, codes obtained from hash functions are converted into a string, next, such string codes are stored in bloom structure. In order to, implement counting bloom filter structure effectively, one list can be attached to any of array element in which string codes of hash functions and resource characteristics are stored. Counting bloom filter backs removal elements membership from the structure and there is no error in the process of deletion of the set elements membership.

### 4.2. Applying Bloom Filter to Proposed Structure

In data grid, structures of replica manager and RC can be implemented in the form of centralized, distributed

and hierarchical architecture. Considering files with specific accessibility (categorized data files), counting bloom filter structure can be applied to the process of batch files replication/deletion. Figure 1 shows proposed network structure. In this structure, each site contains a replica manager for sending information on files replication/deletion in the site to the RC within the same region and for managing inters site data transmission. In proposed structure, we applied centralized job scheduling. Figure 2 indicate distributed structure of RC. Counting bloom filter can be used at local and global RC levels in order to speed up process of record, search and reply the requests from job scheduler. Algorithm 3 shows the process of files searching in regional RC. To execute each job; GS inquires RC of the region that job is started for information on data locations needed by jobs. As the Algorithm 3 shows, during the process of job scheduling, at first, scheduler get the name of files required to perform the job and searches for location of needed data. In proposed model, scheduler convert names of (batch) files needed by jobs into one string and applies *K* hash functions on this string and sends obtained codes to the local RC ( in the same region) which searches received locations in bloom filter array.

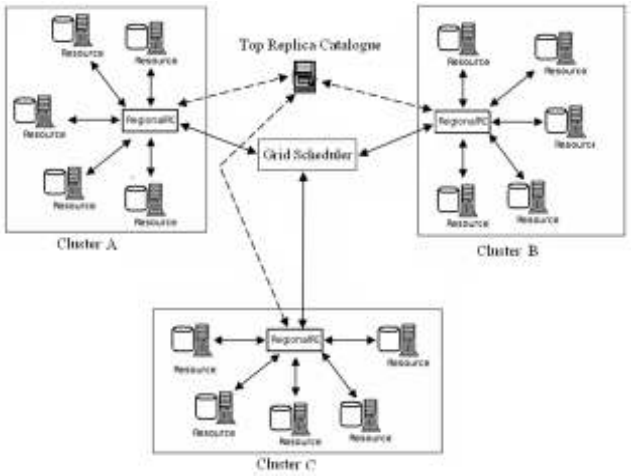



Figure 2. Hierarchical structure for applying bloom filter.

*Algorithm 3: Search_localReplicaCatalogue()*

```
Receive (index , id) from GS ;
bloom_index= extract(index);
 j=0;
str=null;
GS _id, job_id) = extract(id);
for( i=0 to k-1)
   {   j=bloom_index[i];
       if ( bloomfilter[j].counter==0 )
          { send (index , id, RegionalRC_id) to Top RC;
            return;
          }
       else  if( i<k-1)    str=str+'j';

           LocalResourcelist=search( bloomfilter[j].list , str);
              if (LocalResourcelist ==nu    # false positive
                   { send (index , id, RegionalGIS_id) to Top RC;
                return;
                 }
           else
          send (LocalResourcelist, job_id) to GS;
```

It sends resource lists to scheduler in the case that all counters are not zero and that code string of hash functions exist on the list attached to the last location of hash functions, therefore, batch files required by jobs exists in local resources in the same region and avoids remote data access. In the case that counters are not zero, but there is no code string of hash functions on the list attached to the last location, positive error will be detected while name of batch files required by jobs does not exist in local resources in the same region. For this reason, local RC sends scheduler request to global RC in order to search global resources in other regions. Global RC does this search similarly; and if resources having requested files exist, it sends their list to the local RC which sends reply to respective scheduler. Otherwise, the message of 'No resources' is sent to scheduler so that they apply other scheduling criteria based on scheduling policy to execute the job. After assigning the job to a site, replica manager transmit needed files to the site by applying replication mechanisms. So, replica manager decides about replicating batch files in the same site and region to have access to these files without remote data access in future requests.

Algorithm 4 shows the process of files replication/ deletion. Replica managers apply *k* different hash functions on the files name string and send indexes obtained from hash functions to the local RC, which add/reduce one unit to/from counters of locations obtained from hash functions and add/remove resource arguments to/from the lists attached to the last location of resulted codes. After that, local RC sends hash codes to global RC to be performed in the same way as that done in local RC. In this structure local RC is responsible for storing information on location of files existing in the same region and we applied global RC, in order to, increase efficiency of job scheduling with global information.

*Algorithm 4:ReplicaManager_Storefile()*

```
str =String(files name);
for ( i=0 to k-1 )
     bloom_index[i]=hash_i(str);
     index = compress(bloom_index);
     send (index , Resource_id) to related RC;
```

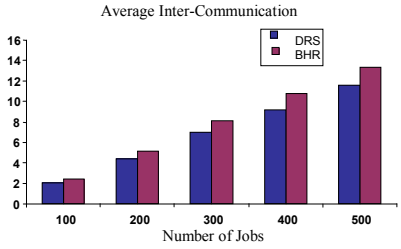
## 5. Simulation

Efficiency of hierarchical job scheduling and data replication and using counting bloom filter in selecting resources has been evaluated by Gridsim 5.2. Table 1 specifies the parameters that used in simulation. For evaluating our work proposed data replication strategy will be compared with Bandwidth Hierarchy based Replication (BHR) and proposed job scheduling policy will be compared with Data Present Relative Load (DPRL) and relative load scheduling strategies.

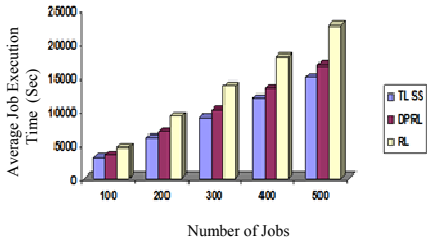
Table1.Simulation parameters.

| | |
|---|---|
| **Number of Regions in Network** | 4 |
| **Average Number of Sites in each Region** | 20 |
| **Each Storage Element Capacity** | 10GB |
| **External Link Bandwidth** | 10 Mbps |
| **Internal Link Bandwidth** | 100 Mbps |
| **Size of a File** | 500 Mbps |
| **Files Modification Grant** | Read-Only |
| **Location of Original Files** | Spread Randomly at Starting Simulation |

Figure 3 has shown the comparison between different job replication strategies. In DRS, the successfully received replicas from other regions must be stored locally to avoid between regions congestion and bottleneck. DRS with considering the data locality at region and site level, reduce the number of external communication and respectively internal communication to reduce data preparation time for job execution. Therefore, DRS have less inter region communication than BHR. Figure 3-b has shown the comparison between different job scheduling and replication strategies. Since, our SS schedules jobs to certain specific region and sites according to required data files, jobs would be executed on the region with most required data, so data access latency and average job execution time are decreased than other scheduling strategies.




a) Average inter-communication comparison between different replication strategies.

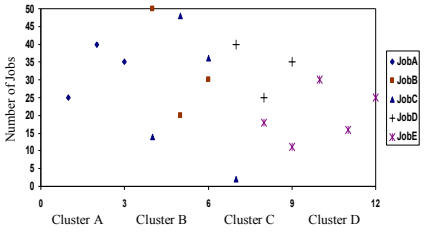



b) Average job execution time based on different job scheduling and replication strategies.

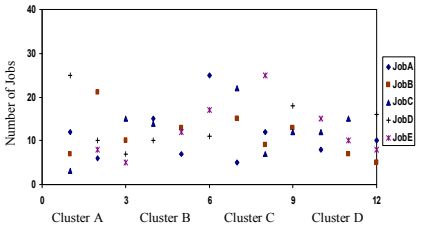
Figure 3. Make a comparison of job scheduling and replication strategies based on average job execution time and inter-communication.

Figures 4-a and b showed the distribution of where jobs are executed. Since, SS schedule jobs to certain specific sites and region according to requested data files, jobs would be executed on a region with the most required files. Therefore, the same types of jobs are executed at the same region as shown in Figure 4-a.

The job distribution of DPRL is shown in Figure 4-b. On the contrary, DPRL does not consider region information and it schedules jobs to certain specific site, therefore different job types would be executed on a region and the number of remote data access would be increased. It would lead to more overhead in transferring file replicas between region and increase data access latency.




a) Job distribution based on job type for 500 jobs for combination of SS with DRS.




b) Job distribution based on job type for 500 jobs for combination of DPRL with LRU.

Figure 4. Make a comparison between scheduling and replication combination based on Job distribution in regions.

During this simulation, measures of response time average and the number of bytes transmitted over the network were compared in the scheduling with using Bloom filter and lists to discover resources. In this simulation, we applied one global RC in data grid. For following simulation, resources and users of data grid were placed within 4 different regions stochastically. Figure 5 illustrates changed average of bytes used by any request with increasing number of resources. In the process of bloom filter base search, scheduler sends locations to be checked in bloom filter to the local RC by applying hash functions on files name instead of send list of files name, So, the number of bytes sent over network is decreased compared to the method not using bloom filter. Also, as simulation results show, with increased number of resources in the network, average of bytes transmitted for both list and counting bloom filter remain almost unchanged. This signifies scalability of proposed structure.

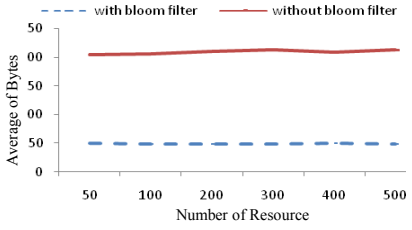



Figure 5. Average of bytes used by any request with increasing number of resource.

Figure 6 indicated changed total of bytes used by any request with increasing number of users in data grid. In proposed method, schedulers request for these same locations obtained from hash functions, therefore, the number of bytes sent between schedulers, local and global RC is much less than that

in the method not using bloom filters which sends files name.

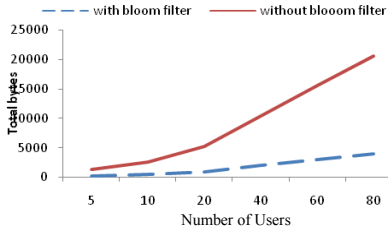



Figure 6. Total of bytes used by any request with increasing number of users.

Figure 7 indicates average of bytes used for each request with increasing Bloom filter size according to the number of resources. Size of bloom filters is one of the important factors lowering the rate of positive error while increasing the number of bytes sent over the network. As seen in Figure 7, size of bloom filter changes as a function of the number of resources and horizontal x-axis indicate this size coefficient of increase in accordance with number of resources. This simulation shows that average of bytes transmitted for each request increases as the size of bloom filter grows bigger. For this reason, selection the appropriate size of bloom filter for establishing balance between positive error reduction and reduction of bytes sent over the network has an effect on the efficiency of entire system and on bandwidth saving.

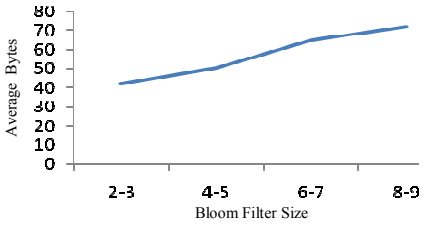



Figure 7. Total of bytes used by any request with increasing number of users

Figure 8 shows comparison of average changes of used bytes with increased number of regions for modes with and without bloom filter of resource discovery in the first phase of scheduling. Results of simulation indicate that with increased number of regions, the number of bytes sent over the network is much less than that in the method not using bloom filter.

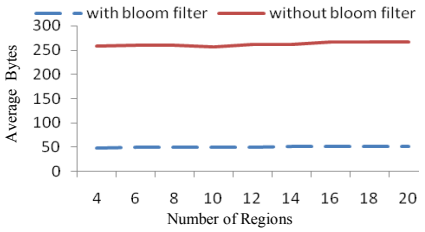



Figure 8. Average of bytes used with increasing number of regions.

Figure 9 shows change in average of response time with an increase in resources number in the whole grid system for both modes of resource discovery with and without bloom filter. This diagram indicates that responses time in resource discovery with bloom filter is much less than without bloom filter. In the cases that requested resources exist in local region, location obtained from hash functions are searched only in bloom filter of local RC, with responses being sent to schedulers. Otherwise, obtained locations are also checked in global RC, with time of responding to scheduler being increased. In conventional method, on the other hand, requested file characteristics are searched initially in the list of resources available in local region and, secondly, they are searched in global resources in the case of absence of resources in local region. Since, list search is done more slowly than check of locations in bloom filter. Therefore, in proposed method, average of response time is much less than that in conventional method.

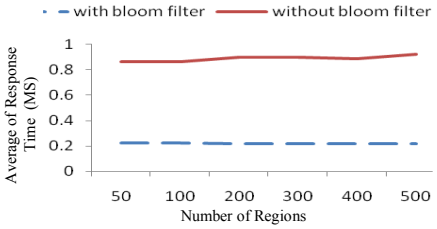



Figure 9. Average of response time with an increase in resources number.

## 6. Discussion

We proposed a structure for distributed replication and effective job scheduling in data grids. To save network bandwidth and reduce data access time, we propose a job scheduling policy SS and distributed replication mechanism that consider regional information in job placement decision. We apply RC for each region to facilitate selecting best region and site for job execution. We can conclude that combination of SS and DRS strategy can be effectively utilized when hierarchy of bandwidth appears. Also, this structure is scalable for resource management while it deals with increasing number of resources and files. Proposed hierarchical structure reduces the average of bytes transmitted over the network and decreases time of responding to schedulers' request by using counting bloom filter. In proposed structure, application of hash functions on scheduler side and replica manager in the site reduces the number of bytes transmitted in processes of data replication/deletion and searching files from resources between scheduler, RC and replica manager components in the network. If requested data be available in resources within local regions, local resources list are sent to schedulers, reducing response time to scheduler's requests considerably. Otherwise, scheduler's requests are searched in bloom filter of global resources. Proposed structure, increases scalability of resource management as well as decreases average of bytes transmitted over the network for resource discovery. The most important advantage of using bloom filter is reducing the time to gathering information for job

scheduling decision. Although, proposed structure avoids of bottleneck in schedulers, local and global RC's by hierarchical approach.

## Acknowledgements

This research is pecuniary supported by "Eslamabad-E-Gharb Branch, Islamic Azad University".

## References

[1] Abdi S., Pedram H., and Mohammadi S., "The Impact of Data Replication on Job Scheduling Performance in Hierarchical Data Grid," *International Journal on Applications of Graph Theory in Wireless ad hoc Networks and Sensor Networks*, vol. 2, no. 3, pp. 15-25, 2010.

[2] Abdi S., Pedram H., and Mohammadi S., "Two Level Job Scheduling and Data Replication in Data Grid," *International Journal of Grid Computing and Applications*, vol. 1, no. 1, pp 23-37, 2010.

[3] Bonomi F., Mitzenmacher M., Panigrahy R., Singh S., and Varghese G., "An Improved Construction for Counting Bloom Filters," *in Proceedings of the 14th Annual European Symposium on Algorithms*, London, UK, pp. 684-695, 2006.

[4] Broder A. and Mitzenmacher M., "Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485-509, 2002.

[5] Chang S., Chang S., and Lin Y., "Job Scheduling and Data Replication on Data Grids," *Future Generation Computer Systems*, vol. 23, no. 7, pp. 846-860, 2007.

[6] Cheng S., Chang C., and Zhang L., "An Efficient Service Discovery Algorithm for Counting Bloom Filter-Based Service Registry," *in Proceedings of IEEE International Conference on Web Services*, Los Angeles, California, USA, pp. 157-164, 2009.

[7] Cokuslu D., Hameurlain A., and Erciyes K., "Grid Resource Discovery Based on Centralized and Hierarchical Architectures," *International Journal for Infonomics*, vol. 3, no. 1, pp. 227-233, 2010.

[8] Dang N. and Lim B., "Combination of Replication and Scheduling in Data Grids," *International Journal of Computer Science and Network Security*, vol. 7, no. 3, pp. 304-308, 2007.

[9] Elghirani A., Subrata R., Zomaya Y., and Al Mazari A., "Performance Enhancement through Hybrid Replication and Genetic Algorithm Co-Scheduling in Data Grids," *in Proceedings of IEEE/ACS International Conference on Computer Systems and Applications*, Doha, Qatar, pp. 436-443, 2008.

[10] Esmaeelimanesh R., Jamshidi M., Zareie A., Abdi S., Parseh F., and Parandin F., "Peresentation an Approach for Useful Availability Servers Cloud Computing in Schedule List Algorithm," *International Journal of Computer Science Issue*, vol. 9, no. 3, pp. 465-470, 2012.

[11] Foster I. and Ranganathan K., "Design and Evaluation of Dynamic Replication Strategies for High Performance Data Grids," *in Proceedings of International Conference on Computing in High Energy and Nuclear Physics*, Beijing, China, pp. 1-17,2001.

[12] Goering P. and Heijenk G., "Service Discovery using Bloom Filters," *in Proceedings 12th Annual Conference of the Advanced School for Computing and Imaging*, Belgium, pp. 219-227, 2006.

[13] Jianhua J., Huifang I., Gaochao X., and Xiaohui W., "Scheduling Algorithm with Potential Behaviors," *Journal of Computers*, vol. 3, no. 12, pp. 1-9, 2008.

[14] Krauter K., Buyya R., and Maheswaran M., "A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing," *Software Practice and Experience*, vol. 32, no. 2, pp. 135-164, 2002.

[15] Krauter K. and Murshed M., "GridSim: A Toolkit for the Modelling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," *Concurrency And Computation: Practice and Experience*, vol. 14, no. 13, pp. 1175-1220, 2002.

[16] Mohamed H. and Epema J., "An Evaluation of the Close-to-files Processor and Data Co-allocation Policy in Multiclusters," *in Proceedings of IEEE International Conference on Cluster Computing*, California, USA, pp. 287-298. 2004,

[17] Park M., Kim H., Ko B., and Yoon S., *Dynamic Grid Replication Strategy based on Internet Hierarchy*, Springer-Verlag, Berlin Heidelberg, 2004.

[18] Parthasarathy S. and kundur D., "Bloom Filter based Intrusion Detection for Smart Grid SCADA," *in Proceedings of the 25th IEEE Canadian Conference on Electrical & Computer Engineering*, Montreal, pp. 1-6, 2012.

[19] Por Y., Ong Y., Beh D., and Ismail M., "A Grid Enabled E-Theses and Dissertations Repository System," *the International Arab Journal of Information Technology*, vol. 9, no. 4, pp. 392-401, 2012.

[20] Ranganathan K. and Foster I., "Identifying Dynamic Replication Strategies for a High Performance Data Grid," *in Proceedings of the 2nd International Workshop on Grid Computing*, London, UK, pp. 75-86, 2001.

[21] Ranganathan K. and Foster I., "Computation Scheduling and Data Replication Algorithm for Data Grid," available at:

http://www.mcs.anl.gov/papers/P1081.pdf, last visited 2003.

[22] Tang M., Lee S., Tang X., and Yeo K., "The Impact of Data Replication on Job Scheduling Performance in the Data Grid," *Future Generation Computer Systems*, vol. 22, no. 3, pp. 254-268 2006.

**Somayeh Abdi** received her BS degree in software engineering from Razi University in Kermanshah in 2005 and MS degree in the same course from Science and Research Branch, Azad University, Tehran, in 2010. Currently, she is pursuing her PhD degree in software engineering in Science and Research Branch, Azad University, Tehran. Her research interests include distributed system; cloud computing, resource management and resource discovery in Grid Computing. She is currently a faculty member at Eslam_Abade_Gharb Branch, Azad University. Her PhD is under supervision of Dr. Hashemi.

**Seyyed Hashemi** received the MS degree in computer science from Amirkabir University of Technology (Tehran Polytechnic University) in 2003 and the PhD degree in computer science from the Azad University in 2009. Both of them ware under supervision of Professor Mohammadreza Razzazi. Moreover, he is currently a faculty member at Science and Research Branch, Azad University, Tehran. His current research interests include software intensive systems, e-x systems (e-commerce, e-government, e-business and so on), global village services, grid computing, ibm ssme, business modeling and agile enterprise architecting through isrup.