# Combination of Replication and Scheduling in Data Grids

**Nhan Nguyen Dang[1,2], Sang Boem Lim[2]**

[1] University of Science and Technology,
52 Eoeun-dong, Yuseong-gu, Daejeon, Republic of Korea, 305-333
[2] Korea Institute of Science and Technology Information,
52-11 Eoeun-dong, Yuseong-gu, Daejeon, Republic of Korea 305-806

**Abstract.** Data Grid environment is a geographically distributed that deal with date-intensive application in scientific and enterprise computing. Dealing with large amount of data makes the requirement for efficiency in data access more critical. The goal of replication is to shorten the data access not only for user accesses but enhancing the job execution performance. In this paper, we proposed a new approach to replication based on organizing the data in Data Grid based on its property. In this paper, we organized the data in to several data categories that it belongs to. And this information is used to help improving data replication placement strategy. We study our approach and evaluate it through simulation. The result shows that our algorithm has improved 30% over the current strategies.

**Keywords:** Data Grids, Scheduling, Replication.

## 1. Introduction

In the increasing demand of scientific and large-scale business application, a large amount of data are generated and spread for using by users around the would. Many good examples can be listed such as High Energy Physics, meteorology, computational genomics which processed and resulted large amount of data. Such data cannot be stored centralized but distributed among centre around the world for processing. Motivation by an integrate architecture for storage, data management and data-intensive application execution, Data Grid is proposed as a solution.

Data Grid is an integrating architecture that allow connect a collection of hundreds of geographically distributed computers and storage resources located in different part of the world to facilitate sharing of data and resources [6]. Size of data that needs to be accessed on the Data Grid may be upto petabytes in the near future. The nature of dealing with large amount of data that geographically spread causes may challenges in Data Grid. One of them is how the scheduling efficiently work with the amount of data need for each job and the impact of replication to the scheduling performance.

### 1.1 Motivation

Replication and scheduling problem has been studied separately for long time, however those in Data Grid have just recently received attention from researchers. Effective scheduling of jobs in Data Grid has its own complicated characteristics since it deal with large amount of input data in the dynamic environment of Grid. The decision of where and when to execute a job is made by considering the job requirement and current status of The Grid, here are computational resources, storage resources and network resources. In Data grid, the execution performance is greatly impacted by the data locality [7]. A good scheduling strategy will allow shortest access to the required data, therefore reduce the data access time. Vise versa, replication strategy that allows place data in a wisely manner will offer a faster access to files require by grid jobs, hence increase the job execution's performance.

### 1.2 Related Works

There are some recent works that address the problem of scheduling and/or replication in Data Grid as well as the combination between them. However, the realization of importance of data locality in job scheduling problem was first proposed by Ranganathan, K. and Foster, I. [7]. The authors had proposed a Data Grid architecture based on 3 main components: External Scheduler (ES), Local Scheduler (LS) and Dataset Scheduler (DS). ES receives job submission from user, then it decides which remote site to which to send the job depend on ES's scheduling strategy. LS of each site decides how to schedule all the jobs assigned to it, on its local resources. DS keeps track of popularity of each dataset currently available make data replicating decision. On this architecture, the authors developed and evaluated various replicating and scheduling strategy to study the impact of the two systems. The result showed the importance of data locality in scheduling job.

Follow on, in OptorSim [2], [3], data replication is combined with job scheduling in a two-stage optimization mechanism. Our proposed architecture is the combination and improvement of both above architectures. More recent works by Chakrabati, A. and colleagues ([6], [7]) or Tang, M et al. [9] improve the older works by integrating the scheduling and replication strategy to improve the scheduling performance.

Analyzing above works, the author realized there were two shortcomings. First one is the relationship among data

and between the data and job. Instead of relying on the grid capability, we approach the problem from the job and data property. By replicating set of files that has high probability to be used together into nearby resources, we expect the job that using those file will be scheduled to that small area. Second issue is the Dataset Scheduler - DS. Instead of just tracking the data popularity, the DS takes its role as an independent scheduler.

This paper will be organized as follow: Section 2 will include the proposed system model. Section 3 describes the scheduling issue and section 4 goes in detail the replication strategy, following by the simulation results in Section 5. Section 6 will summarize the paper and the end will be preferences - section 7.

## 2. System Architecture

The Data Grid system model for job scheduling and replication problem is described in OptorSim – EU Data Grid project 2. Basically, we follow that model OptorSim's model and architecture describe in [7] (Fig. 1). However, we add into that model a centralized ActiveReplicator as part of ReplicaManager and also modify the role of Data Scheduler. There are some terms:

- LS/DS/CE/SE/RB: Local Scheduler/ DataScheduler/ ComputerElement/ StorageElement/ ResourceBroker - [2], [7].
- RC: ReplicaCatalo-gue store the list of all replicas on the grid.

For every predefined interval, the replicator will collect the replica information and data usage information over the Grid. Then it decides whether to replicate a data file. Operation of the active replicator will be described in section.

When a job is assigned to a LS, the DS will responsible

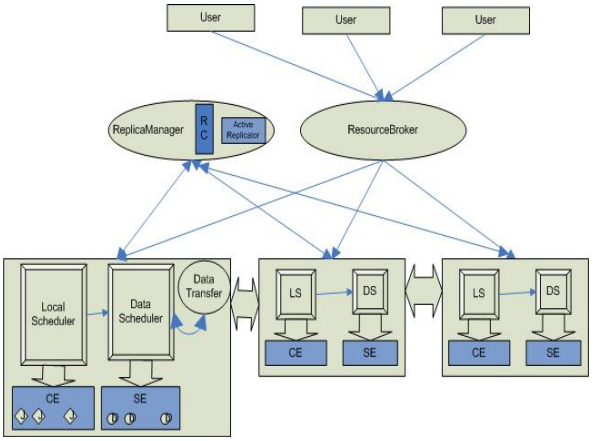


**Fig. 1:** Data Grid Architecture

for all the data requests by the LS. This data request is generated as soon as job is scheduled into LS queue. The objective of this DS is to obtain to the local site as much data required by a job as possible before that job is executed.

## 3. Scheduling Strategy

We proposed a scheduling strategy for the RB that makes use of Local Scheduler and Data Scheduler in each Grid Site. The Resource Scheduling strategy based on the estimation of cost (time) of executing a job in each grid site. It is possible to assume that job is submitted to RB one by one. When receiving a job submission, the RB will estimate the time for completing executing (ETTC) a job in a grid site i:

$$ETTC_{j,i} = \max\{DT(f(j),i), QT(i)\} + EET_{j,i} \quad (1)$$

This estimation equation is similar to what was introduced in [9]. In the real case, the work of obtain QT(i) – Queuing time in site I - is quite simple. Suppose that j-1 is the last job in site I's queue. We can easily realize that QT(i) = ETTCj-1,i.. ETTCi-1,j can easily obtain if the LS allow such a service for RB. DT(f(j), i) also can be estimated based on the Grid status information as describe in [9].

We define a method to optimize the data obtain for job before it is executed. Once a job j is submit to site i, all the data files required by j (i.e. f(j)) is submitted to DS. The data scheduler can organized the data request by queue and processed the request based on its own strategy (e.g. first come first serve). By communicate with NWS or MDS, the Data Scheduler can know whether a request to a file is ready to process Since requested file spread among various grid site, data request can be process simultaneously. If one file is completed, it can inform the LS so that the LS know about the situation of data input. By using a DS, we can optimize the data transfer time for job (DT(j)), hence improve the job's execution performance.

## 4. Dynamic Replication

We assume that data in Data Grid belongs to a field of study, e.g. Physical, Biology, Chemical, Meteorology, etc. They are the fist level of a hierarchical tree. Going further down, we can divide the field to more specific, for example biology can be divided to cell biology, molecular of biology, cell technology, proteomics. One more step, we can say that a gene data $g_d$ of a chicken is categorized to Animal gene, rather than Plant genes. The reason of this assumption is that data in one field usually rarely or cannot be used in other fields, an experiment on animal gene never used gene data from a tree. By doing like this, we can form a hierarchical tree of data type. On that tree, we can define the relationship between data in same category and relationship between nearby categories. Through that, we can extract a correlation between data. However, due to

current data and scenario, we can just define a flat category system, including a set of category. Each data entry belongs to one system and has a close relation with data entries in that category rather than other categories. Because the work of replication of data takes place before and without information about the job that used the data, if we can gather files that have high possibility to be used together, the performance of job execution will increase.

Our idea is gather the data that is "related" to the small region so that the job use such data will be execute inside that region. We believe that doing such thing will reduce the cost to transfer data to the job execution site, therefore improve the job execution performance. For simplicity, we define data that are "related" are data belongs to the same category.

In particular, the replication problem includes two questions to be answer:

- Which data to be replicated?
- Where to put the new replica?

We will define algorithms to solve each problem that we call Dynamic Data Replication Algorithm (DP) in following sections.

### 4.1 Replica decision

In order to decide which file needs to be replicate, we use a metric call average number of access of the as indicated in [9]. In replication mechanism, each replication server maintains data accesses record. When it is time to replica data, all replication servers send the access record to the central replication manager. The manager will aggregate and create a summarized access record for every unique file identifier (FID). The result is a record NOA which each item NOA(f) indicates the times that a file with unique ID f is accessed on the whole grid system.

Once the average number of accesses is calculated, if a replica of a file is accessed more than the average access, then the file needs to be replicated. But we do not use the original number of access in general. Instead, we calculate the amount of data access by taking into account the file size as a parameter when making decision.

The algorithm to decide which data file to be replicated is described below:

- Compute average number of access:

$$\overline{NOA} = \frac{\sum_{\forall f} NOA(f)}{N} \quad (2)$$

*N*: number of distinguish data file (number of FID) in Grid system. *NOA(f)*: number of access of file f

- For every file f that satisfies:

$$\frac{NOA(f)}{NOR(f)} \times |f| > \overline{NOA} \times \overline{|f|} \quad (3)$$

*(NOR(f): number of replicas of f on the whole grid system and |f| is average file size of all data files in the system)*

invoking replication replacement algorithm to create a replica for f.

Since the cost of using a file is in direct ration to file size, our algorithm uses the average number of access with taking into account the file size when making replication decision. We call it Dynamic Replication Decision (DS)

### 4.2 Replica replacement

As above, our strategies is replicating file that belong to the same category close to each other so that job belongs to that category will be executed nearby. When the job is executed nearby, the cost for file transfer will be reduce. We call this strategy Dymamic Replication Placement (RP)

To measure the how close a replica is to the data in the same category, we define a new concept: Dis(Distance).

- Distance is a measurement from D to D1 for a file f ( of category C) is defined as the time to transfer all files that belong to category C on site D1 to site D:
  - If D is the same as D1, then Dis(f, D1) = 0.
  - Else:

$$Dis(f_D, D_1) = \pm \frac{\sum_{f_i \in D_1, f_i \in C} |f_i|}{BW_{D,D_1}} \quad (5)$$

*Dis*($f_D$, $D_1$) is positive/ negative (+/-) when $D_1$ does/ does not contain a replica of f. Why there is negative distance? It means the further the physical distance of the two replica of the same file, the better it is. However, it still must be close enough to other files of the same category.

- Similarly, distance for a replica f (of category C) on site D to all files of C is the time to transfer all files belongs to category C on the Grid system to site D:

$$Dis(f_D) = \sum_{\forall D_i} Dis(f_D, D_i) \quad (7)$$

*Dis*($f_D$) will evaluate how close $f_D$ to other files in the same category. A file that locates at a site with lower *Dis*($f_D$) is said to be close to all of files its category.

To choose a site to place the newly created replica, we evaluate the Dis(f, D) for all sites D in the Grid system. The site which offers the lowest Dis(f, D) will be chosen to store the new replica.

## 5. Performance Studies

In order to evaluate the performance of the replication strategy, the OptorSim simulation tool was used. The EU Data Grid as described in OptorSim was used for this purpose. The grid job is submitted to the RB for every 2.5 seconds. Each computing node has a processing speed of 0.1 second/ GB. Which job to be submitted to the RB is randomly generated based on the job execution probability which is predefined. The initial file distribution among the grid sites is random. Each node has 0 or 1 Storage Element of size 15GB to 100GB. We performed 2 kinds of test:

- Replica placement algorithm test
- Replication algorithm test

### 5.1 Replica placement test

The replica placement strategy was tested to measure its performance against the random placement strategy. In this test, we maintain only one dynamic algorithm for choosing which data files to be replicated. The site to place the

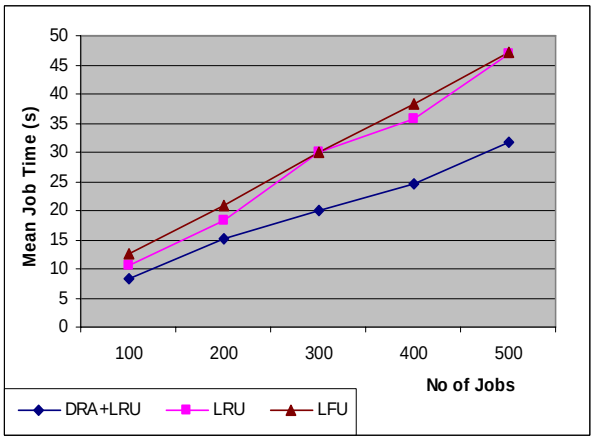


**Fig. 4:** Scheduling performance in various replication strategies (CCS scheduling strategy is used)

newly created replica will be chosen randomly (DR – Dynamic Replication Decision + Random Placement) or by the algorithm that is described in Section 4.2 (DP – Dynamic Replication + Dynamic Placement algorithm). The scheduling strategies set up for this test were the Random scheduling (RS) and Combined-cost Scheduling (CCS). With RS, the RB choose a site for a job execution randomly on the grid, while with CCS, site with minimum of combination of access cost and queuing cost will be chosen for job execution. These two scheduling strategies are provided with OptorSim. The parameter to be varied was number of submitted jobs.

For each combination of methods and parameters, the mean job execution time was measured (Fig. 2). The

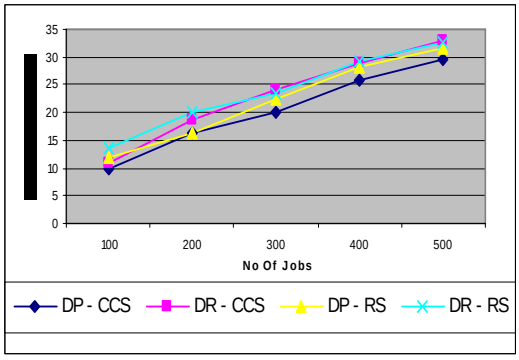


**Fig. 2**: Performance of different replication placement algorithms with scheduling strategies

Dynamic Replication Decision with Placement Algorithm (DP) is outperformed that with random placement. Another measurement of replication placement efficiency is that number of replicas (Fig. 3: ). The figure shows with the same replica decision strategy, that DP has fewer replicas than that of DR, therefore the replication placement algorithm (DP) are more efficient.

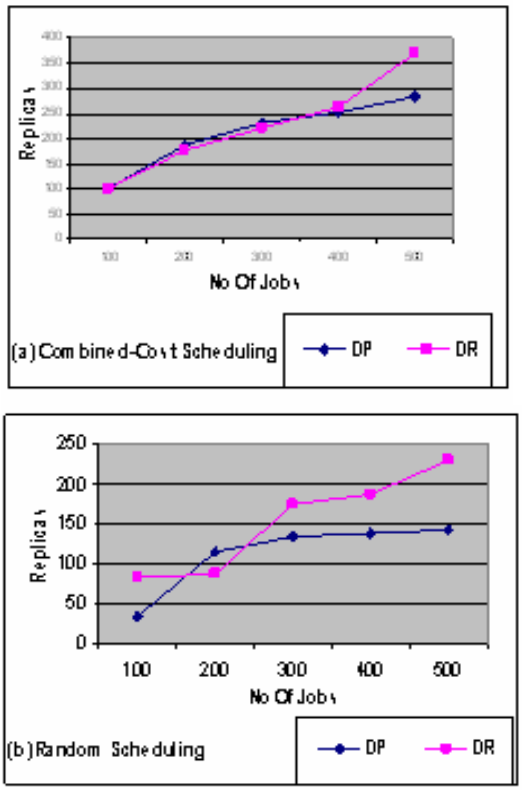


**Fig. 3:** Number of replicas in related to number of jobs submitted

### 5.2 Replication algorithm test

The whole replication algorithm is evaluated against the OptorSim's Least Frequently Used (LFU) and LRU

(Leaset Recently Used) replication. Once again, the OptorSim's CCS scheduling strategy is used.

In OptorSim, LRU and LFU replication strategies perform a replication action of a file when it is required by a job. These strategy is similar to caching, except for the created file is registered to Replication Catalogue for later use. The DRA strategy is combined with LRU to form a new replicating strategy that is call DRA+LRU. DRA performs dynamic replication algorithm over all the data files in the Grid system for every predefined interval. However, when a job need some input files that is located at remote site, the files will be replicated to the job execution's site. In case the replicating action cannot be completed, the files will be accessed remotely.

The simulation result (Fig. 4) shows that by combining Dynamic Replication Algorithm with the replicating optimizer at runtime, the performance is significantly increased of 30%.

## 6. Conclusion

In this paper, we have proposed our improvement over current Grid scheduling and replication problem. In replication issue, we have proposed a new approach to replication problem in Data Grid. The simulation result showed that our replication placement strategy overcomes the random placement strategy. Also, the dynamic replicating algorithm makes an improvement and can be used with OptorSim's replication optimization. The result is quite promising. In the future work, we will do more simulation test and improve the replication strategy. Meanwhile, the scheduling component needs to be completed for integrating with replication mechanism to perform a whole system simulation.

## 7. References


1. William H. Bell, et al. OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies. *International Journal of High Performance Computing Applications*, 17(4), 2003.
2. William H. Bell, et al. Evaluation of an Economy-Based File Replication Strategy for a Data Grid. In *International Workshop on Agent based Cluster and Grid Computing at CCGrid 2003*, Tokyo, Japan, May 2003. IEEE Computer Society Press.
3. David G. Cameron, et al. Evaluating Scheduling and Replica Optimisation Strategies in OptorSim. In *4th International Workshop on Grid Computing (Grid2003)*, Phoenix, Arizona, November 17, 2003. IEEE Computer Society Press.
4. K. Ranganathan and I. Foster. Decoupling computation and data scheduling in distributed data-intensive applications. In Proceedings of the Eleventh IEEE Symposium on High Performance Distributed Computing (HPDC), Edinburgh, Scotland, July 2002.
5. David G. Cameron, et al. Replica Management in the European DataGrid Project. Journal of Grid Computing. 2(4): 341-351 (2004)
6. H. Lamehamedi, et al., Simulation of Dynamic Data Replication Strategies in Data Grids. In Proc. of 12th Heterogeneous Computing Workshop (HCW2003), Nice, France, Apr 2003. IEEE-CS Press.
7. K. Ranganathan and I. Foster. Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids. Journal of Grid Computing, Volume 1, Number 1, 2003, pp. 53-62(10).
8. Anirban Chakrabarti, R. A. Dheepak, Shubhashis Sengupta: Integration of Scheduling and Replication in Data Grids. HiPC 2004: 375-385
9. Tang, M., Lee, B., Tang, X., and Yeo, C. 2006. The impact of data replication on job scheduling performance in the Data Grid. Future Generation Computing System 22, 3 (Feb. 2006), 254-268. DOI= http://dx.doi.org/10.1016/j.future.2005.08.004
10. David G. Cameron, et al. , OptorSim v2.1: Installation and User Guide, OptorSim official Sourceforge website: http://sourceforge.net/projects/optorsim , Oct 2006.