Replica Management in Data Grid

Husni Hamad E. Al Mistarihi[1], Dr. Chan Huah Yong [2]

School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang

Summary
The emergent of scientific applications which produce a huge volume of data files to be managed and shared require special attention. In large-scale grid, data replication provides a suitable solution for managing data files where data reliability and data availability are enhanced. But data replication causes further problems: 1. How to balance the number of replicas in grid sites. Indeed, increasing number of replicas lead to increase data availability and reliability, however the storage space will be increased as well. Therefore, a good balancing of number of replicas is required. 2. Replica placement problem. Placing the new replicas in the appropriate location site can promote reducing the network bandwidth consumption and reduce the turnaround job time. Replication strategy, replica placement policy, and replica selection are all embodied into our proposed system in order to: reduce job turnaround time, reduce storage cost, and reduce network bandwidth consumption. The simulation results show that our system outperforms other similar systems performance around 15%.

***Keywords*:**
Data Grid, Data Replication, Replication Strategy, Replica Placement Policy

1. Introduction

Data grid provides scalable infrastructure for storage resource and data files management, which supports a variety of scientific applications. Most of scientific applications such as: High Energy Physics (HEP) [5, 17] and climate change modeling [8] require accessing, storing, transferring, analyzing and replicating a large amount of data in geographically distributed locations [12, 13]. These scientific applications face the problem of sharing the distributed datasets. The solution for such problem can be resolved by data replication management.

Typically, grid users would want the required data files in minimum response time, regardless of the grid resources usage. The replica selection service provides grid users with the required replicas in minimum response time. Thus, the replica selection service is a self-interest local objective in short-term optimization that doesn't consider the grid resources usage. However, the replication management is a global objective in long-term optimization that considers the overall grid users, and the overall grid resources usage. The data replication increases the data availability and reliability for the users and thus the job turnaround time will be decreased, but on another hand the replication increases the storage space cost. Moreover, replication strategies influence the network bandwidth consumption usage positively and negatively depending on the efficiency of balancing the replica demand and the available number of the underlying replicas to face such demand. Therein, when the system creates a new replica and places it in a remote site location, some of the network bandwidth is consumed. However, when the system doesn't create a replica while there is a big demand, the number of times the job read this replica remotely also increases the network bandwidth consumption. Thus the replication decisions should be reasonable and justified. Obviously, the global objective and the local objective are contradict to each other. Therefore, we proposed a new solution to the underlying problem encapsulates in our system that termed as Replica Management in Grid (RmGrid), which answers the following questions:

- *When to create / delete replicas and which one to be created / deleted?*
- *How many numbers of replicas should be created and Where to place new replicas?*
- *How to select the best replica among many replicas are available in the grid?*

2. Related Work

Data replication [8, 1, 2] is the process of producing multiple copies for the same data file, and distributes those data files copies (Replicas) into grid sites according to some techniques termed as replication strategies. Indeed, the good replication strategy achieves the following conflicting objectives simultaneously: 1) Reducing the job turnaround time by ensuring data availability. 2) Reducing storage usage. 3) Reducing the network bandwidth consumption. We categorized the replication strategies into three types for simplicity: no

Manuscript received June 5, 2008.
Manuscript revised June 20, 2008.

replication, unconditional replication strategies, and conditional replication strategies.

No Replications: also called "static replication" where the replicas are created and distributed when the system on the offline state. After the system operational, the number of replicas and their locations remain the same. When users' requests change over the time, for example, the demand on a specific replica is increased, but the available number of replicas is insufficient to serve such demand. Thus, the static replication can not adapt to changes in user behavior especially in our scenario where the size of the data files in petabytes and the user community is in the order of thousands around the world [3]. One example of the implemented strategy is the *SimpleOptimizer* algorithm [4], which never performs replication; rather it reads the required replica remotely. *SimpleOptimizer* algorithm is simple to implement and performs the best in relative to other algorithms in terms of the storage space usage, but performs the worst in terms of job turnaround time and network usage as discussed in more detail in section 5.

Unconditional Replications: The algorithms in this strategy perform replication every request. The requested site always replicates the required replica from the replication site. Thus, in this context, unconditional replication acts as caching, and also called plain caching strategy [3]. Indeed, if a client requests a replica while the local storage is full, then a replacement strategy will be triggered to place the new replica by the existing ones.

Least Recently Used (LRU) and Least Frequently Used (LFU) are examples for this kind of replication strategy [3]. In LRU strategy, the requested site caches the required replicas, and if the local storage is full, the oldest replica in the local storage is deleted in order to free the storage. However, if the oldest replica size less than the new replica, the second oldest file is deleted and so on. LFU strategy performs same as LRU but the only different that the LFU deletes the replica which has less demand (less popularity) from the local storage even if the replica is newly stored.

Conditional Replications: Unlike the previous types of replication, in this type of replication, the replica creation mechanism triggers according to some conditions such as replica requests threshold. In general, the conditional replication performs better than other types of replication as discussed in more detailed in section 5. Intelligent algorithms are deployed in this type of replication in order to achieve all the replication objectives at the same time where the objectives might contradict to each others. Our proposed strategy is laid in this type of replication. The most important strategies in this type of replication are: economic model and best-client.

Economic model adapts to the market place. In this context, the data files represent goods that purchased by Computing Element (CE) for running jobs and by Storage Element (SE) to make an investment for expected future revenue. Therein, the Replica Manager (RM) keeps the data files requests number as a historical data, and use these data for deciding when to replicate, and how many number of replicas should replicate. RM decisions are based on the replica request demand and the threshold which related to some data file distribution methods such as: Normal distribution and Binomial distribution. Each node in the grid has RM, thus all decisions are made by each grid node independently to obtain local optimum that expecting to lead for global optimum.

The authors [10, 11] adapt the economic model for replication strategies. They focus on optimizing the replication of data in grid environment to achieve the final goal which is reducing the job turnaround time in long term. Each grid site/node evaluates the local file values according to popularity of the files, and decides whether to replace the more popular files with the less popular files. Since each node in the grid is responsible for manage its storage system, there is no global vision on the overall grid nodes. Therefore, each node optimize the local storage in short term optimization which expected to leads for long term optimization, and as a result, more time required to reach the global optimum state. Moreover, only the data popularity is considered for the replication decisions while the network status and the number of replicas for each file are not considered [14].

In best-client strategy [3], the data files which exceed the threshold will be replicated and placed at the best client location. In this context, best client is the client who issues more requests on a specific replica. However, when the best-client exists very far away from other clients in the grid related region space, other clients will require more response time to get their replicas from the best-client location. Indeed, considering only one client from many clients is insufficient strategy that causes the job turnaround time to be increased.

A new replication strategy, a new replica placement policy termed as "best-location", and a replica selection process are all embodied into RmGrid, which consists of two main components namely: Global Optimizer and Local Optimizer. The Global Optimizer performs on behalf of the system in order to best utilize grid

resources efficiently, and deployed both replication strategy and “best-location” policy. However, the Local Optimizer performs on behalf of the users for selecting the best replica among many replicas distributed across the globe. In our proposed system, the file's popularity, replicas locations, network status, replica placement policy, and the number of replicas for each file are all considered in order to improve system resources usage and turnaround job time.

3. System Design

RmGrid leverages on many numbers of existing successfully data grid core services, such as the Replica Location Service (RLS) which provides RmGrid with the physical file locations, Metacomputing Directory Service (MDS) and Network Weather Service (NWS), which provides information about the network status and the storage system. Since the prediction process based on historical log file contains the data transmission time and other information, the RmGrid makes use other services, such as the GridFTP [6] in order to transport replicas to grid users securely, and to logs the end-to-end transfer data. These mentioned grid services (RLS, NWS, MDS, GridFTP, …) act as an information provider to our system as shown in Fig. 1.

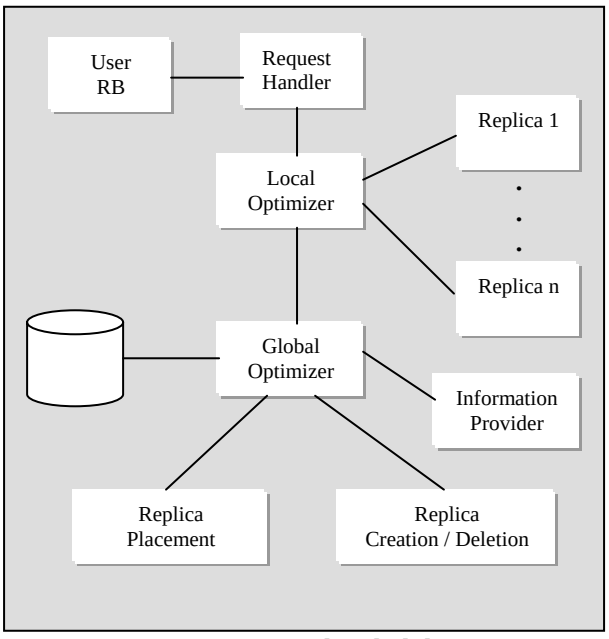


Fig 1. System detailed design

Typically, grid users submit their jobs to the Resource Broker (RB) which select the best site to execute jobs. Jobs under execution required data files. The request handler gets these requests and passes it to the Local Optimizer which selects the best replica for the underlying job. The Global Optimizer gets a feedback from the Local Optimizer such as the replica requests demand, and gets some related information from the information provider such as network status. Consequently, The Global Optimizer triggers both of replica creation / deletion function and the replica placement function in order to optimize the number of the replicas and their locations in the grid sites.

3.2 Local Optimizer

Local Optimizer solves the underlying local optimization problem, which is how to select the best replica for the job under execution among many replicas distributes over the grid sites. In this context, the best replica means the replica that has minimum response time which include the data transfer time and the storage access latency [15]. The data transfer time is computed by dividing the replica size by the network bandwidth. In this paper, we don’t concentrate on the local objective since it’s our ongoing work.

3.3 Global Optimizer

Global Optimizer deploys a replica placement policy and a replication strategy. Therefore, the appropriate number of replicas and their locations is optimized in order to reduce grid resources usage and the job turnaround time, which defined as a period of time between the beginning of operation of a job until the reception of output. In particular our system reduces the response time which also called data access cost and thus the total job turnaround time will be reduced accordingly, because the job turnaround time includes the response time and the job processing time. In this context the response time is the time elapsed from when the job request replicas until received the replicas in the local machine. The response time includes the data transfer time and the storage access latency [15].

3.3.1 Replica Request Demand

Replica Request Demand (RRD) is number of times a specific replica is requested clustered on time, which also known as file popularity. The replication strategies are looking for the potential popular files because it is believed that popular files in past time window will be popular in future time window [9]. Therefore, RRD is computed for each file as:

$$RRD = \frac{\sum N_{\text{Re quests}}}{(T_{Current} - T_{Stored})} \qquad (1)$$

$N_{Requets}$: Number of times the file is requested.
$T_{Current}$-T_{stored}: Time starting from the day the replica created until the current day.

3.3.2 Replica Creation and Deletion Mechanism

In this mechanism, RmGrid triggers replica creation or deletion function according to RRD, therein, the system triggers the replica creation function when the RRD exceeds the threshold to face the increasing of the replica demand. Thus, the appropriate number of replicas for each file is determined. RmGrid performs one of the following tasks according to the RRD and the threshold:-**Task 1:** Create a new replica, when the RRD for the specific replica exceeds the upper threshold, this replica said to be valuable replica and requires more number of copies in the system. In this context, the system creates another copy of the underlying replica and thus performs the "best-location" policy in order to find the best location site to store the underlying replica. **Task 2:** Delete an existing replica, when the RRD exceeds the lower threshold, the system based on RRD finds the less valuable replica to be deleted. **Task 3:** No action occurred, when the system in stable states where the RRD is located between the lower threshold and the upper threshold.

3.3.3 Determine the Replicas to be Created or Deleted

The system finds the valuable file to increase its number of copies and the less valuable file to decrease its number of copies. Given the number of requests for each file, and the period of time the file exist on the grid, the RRD can be computed by using Eq. (2). For example: consider a grid with 7 sites and 5 files as shown in table 1, each field in the table is the RRD for the specified file. The intersection of File1 and Site 3 is the RRD value for file1 indicating that file1 has requested 23 times by site 3 in a specific period of time, for example per day.

The file value for each file can be computed as:

$$File\ \text{Value} = \frac{\text{RRD}}{\text{Number of Copies}} \qquad (2)$$

Table 1: RRD values for 5 files requested by 7 sites

Names of files	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7
File1	12	9	23	16	10	15	29
File2	20	8	12	12	12	21	20
File3	23	7	14	13	19	25	24
File4	32	17	5	16	3	17	25
File5	8	22	8	7	20	18	20

All file values in the example are computed as shown in table 2.

Table 2: Solution to the example in table 1

Names of files	RRD	Copies	File Value
File 1	114	55	2.072727
File 2	105	53	1.981132
File 3	125	54	2.314815
File 4	115	56	2.053571
File 5	103	57	1.807018

Therefore, if the RRD exceeds the upper threshold, then the system will create a new replica that has the maximum file value. Likewise, if the RRD exceeds the lower threshold, then the system will delete the existing replica that has the minimum file value. The threshold is specified by the system administrator. Suppose the threshold used in the example is 10%. The average of the RRD values = (114+105+125+115+103)/5 = 112.4. Therefore the threshold upper value = 112.4 + (10% × 112.4) = 123.64. The threshold lower value = 112.4 - (10% × 112.4) = 101.16. Since file 3 exceeds the upper threshold, the replica creation function will be triggered, and file 3 will be replicated because it has the maximum file value (2.314815). It is not necessary that the file which has the maximum RRD and exceeds the upper threshold will be the same file to be created. For example, suppose that in the mentioned example above that the number of copies of file 3 is = 60 instead of 54. The file 3 value become 125/61 = 2.049180 and thus the file 1 will be replicated because it has the maximum file value (2.072727). There is no any RRD exceeds the lower threshold in the example, thus no action occurred for the replica deletion function.

3.3.4 Determine the Locations for the New Replicas

Based on best-location policy, the site location must serve all grid sites, and thus a central point must be found, which we name it Location Cost (LC). The best-

location site is the site that has the minimum LC as follow:

$$\text{Best-Location} = \text{Min}\ (\ LC_1, LC_2 \ldots\ LCn\ ) \qquad (3)$$

n: number of candidate sites in the grid that don't have the underlying replica.

While the LC is computed as:

$$LC = \sum_{i=1}^{n} (SP_{Site(i)} \times \text{RT}) - \sum_{i=1}^{m} (\text{Avg}(SP) \times \text{RT}) \qquad (4)$$

Where,
RT: Response Time.
n: total number of the sites in the grid.
m: total number of the replication sites, which have the underlying replica.
SP: Site's Power.

The best location policy designed based on two rules namely: 1) the best site is the site that most service all other sites in the grid as a central point location. On another word the site that has the minimum access total cost of the requested sites in the grid. 2) The best site is the site that located as far as possible from other replication sites, since the replication site itself never request a replica that it is already stored on it. Thus, the replicas will be distributed in a manner that services all requested sites. The number of requests issued by a site is represented by the SP, therefore, if any site in the grid doesn't request a specific replica, then the SP value is equal to zero (SP=0), and thus its transfer time cost is also equal to zero. Thereby, the site that has SP=0 is excluded from the total LC. The first part of the Eq. (5) represents the first rule and the second part represents the second rule. Obviously, the LC depends on three parameters namely: 1) SP, 2) Transfer time, and 3) the replica distribution among the sites.

1) Site's Power (SP)

SP is the number of times the site requests a specific replica clustered on time, and computed as:

$$SP = \frac{\sum N_{\text{Re quests}}}{(T_{Current} - T_{Join})} \qquad (5)$$

$N_{Requets}$: Number of times the client requests the underlying replica.
$T_{Current}$-T_{join}: Time starting from the day the client joins the grid until the current day.

2) Response Time

The response time is the time elapsed for the job when requested a replica until the time the job received the replica, and include storage access latency and data transfer time which computed by dividing the replica size by the network bandwidth. When the bandwidth increases between two sites, the response time is decreases. In order to represent the sites and their bandwidth consider the graph shown in Fig. 2(A). The vertices represent the sites and the edges represent the bandwidth among the sites. In this context, the bandwidth represent the distance in time between two sites, such that the big bandwidth represent small distance and the vice versa.

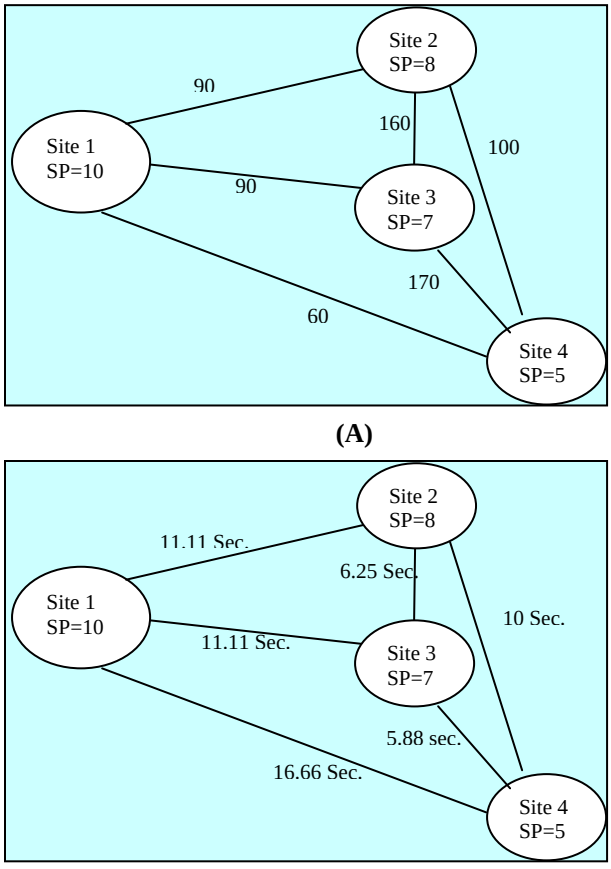


Fig. 2 Example of sites and the links among the sites

As a comparison between the best-client and the best-location policy, consider the following simple example: Suppose that we have four sites in the grid: site1, site2,

site3, and site 4. The SP values for the sites are: 10, 8, 7, and 5 respectively. The bottleneck bandwidth is considered as shown in Fig. 3 (B). In best-client the new replica will be placed on site1, which has the highest SP. whilst in best-location policy, the replica will be placed on site 3, which has the minimum location cost and thus the central location point that provide minimum transfer time to all other sites. The feasibility of the best-location policy compared to the best-client can be approved mathematically as follow: Suppose a new replica with size = 1000 MB is created by the system and we need to place this replica on the best site. After computing the response time, the graph became as shown in Fig. 2 (B), where the vertices represent the sites and its SP and the edges represent the response time between the links in seconds.

Therefore, placing the replica at site 3 is less cost than placing the replica at site 1which explained below by computed the LC for all sites using Eq. 5

site1 = (8 × 11.11) + (7 × 11.11) + (5 ×16.66)-0 =249.95
site 2 = (10× 11.11) + (7 × 6.25) + (5 × 10) - 0 = 204.85
site 3 = (10× 11.11) + (8 × 6.25) + (5 × 5.88)-0 = 190.50
site 4 = (10 × 16.66) + (8 × 10) + (7 × 5.88)-0 = 287.76

3) Replica Distribution

Replication sites degrade the LC points of the candidate sites, such that the candidate site that is very close to the replication sites is getting more impact. We aim to place the new replicas in a central point of the grid sites that provide as much as possible service for all other sites, and the central point should be as far as possible from the replication sites since the replication sites will never request the underlying replica and the replica distribution should balanced among the sites. Since we represent the transfer time as a distance among the sites in time, the site that located far away from the replication site relatively is the site that has big transfer time between the underlying candidate site and the underlying replication site. Thus, when the transfer time increases, then the LC of the site decreases in big rate, and causing more possibility for this site to be the best-location site. However, when the transfer time decreases, then the LC of the site decreases in low rate, and causing less possibility for this site to be the best-location site. The second part of the Eq. (5) which is $\sum_{i=1}^{m} Avg(SP) \times \text{Response Time}$ decreases the LC of the underlying site and represents the best analogy of the LC value which influenced by the replication sites. The average of the SP is multiplied by the transfer time in order to magnitude the important of the transfer time in uniform scale.

In order to implement the Eq. (5), we considered the same example mentioned in previous section. Refer to the Fig. 2 (B), site 3 now become a replication site, and we want again to create the same replica. The LC for the candidate sites: site1, site2, and site 4 are computed as follow:-

The average of the SP = (10+8+7+5) / 4 = 7.5, the LC for all the candidate sites are computed as:

site1= (8 × 11.11) + (5 × 16.66) - (7.5 × 11.11) = 88.855
site 2 = (10× 11.11) + (5 × 10) - (7.5 × 6.25) = 114.225
site 4 = (10 × 16.66) + (8 × 10) - (7.5 × 5.88) = 202.5

The best site is the site that has the minimum LC; therefore, site 1 is the best site to store the underlying replica.

4. Performance Evaluation and Metrics

We have considered the simulation OptorSim [7] as it is more appropriate to our research work. The performance evaluation metrics used as a benchmark for our system are as follow:

1) Mean Job Turnaround Time (MJTT).

The job turnaround time is the time that elapses from when a job arrive the queue line in a site waiting for its service, until the time when the job finish processing and left the site. The job turnaround time includes the response time of the required replicas by the job and the job processing time. Therefore, reducing the response time cause the job turnaround time to be reduced. In order to evaluate the overall system performance, the average time of all jobs have executed in the grid is measured and can be computed as:

$$\text{MJTT} = \frac{\sum_{i=1}^{n} T_{Arrive} - T_{Departure}}{n} \qquad (6)$$

$T_{Arriva:}$ The time when the job started execution.
$T_{Departure:}$ The time when the job finished execution.
n: Total number of jobs processed through the system.

2) Effective Network Usage (ENU)

Our system is optimized to minimized the usage of the network in terms of minimizes the bandwidth consumption and thus reducing the network traffic. Placing the replicas in appropriate sites can eliminate the

number of remote file read and increasing the number of times the site read the replicas locally. Number of times the system creates replicas should be minimized in order to reduce the network bandwidth consumption. Therefore, the ENU is a suitable metric in our case and can be measures by using the following equation [7]:-

$$ENU = \frac{N_{\text{remote file access}} + N_{\text{file replication}}}{N_{\text{remote file access}} + N_{\text{local file access}}} \qquad (7)$$

$N_{\text{remote file access}}$: Number of times the jobs read the required replicas from remote sites.
$N_{\text{local file access}}$: Number of times the jobs read the required replicas from the local site.
$N_{\text{file replication}}$: Number of times the system creates new replicas.

3) Average Storage Usage (ASU)

Storage usage can be calculated for each site as a percentage of capacity reserved by files according to the total capacity for the underlying storage. The average of the all storage elements in the grid can reflect the total system storage cost, which computed as:

$$\text{ASU} = \frac{\sum_{i=1}^{n} \frac{U}{C}(site\ \text{i})}{n} \text{ X } 100\% \qquad (8)$$

Where,
U: Is the storage usage calculated by the capacity reserved in the storage for each site in MB.
n: Number of sites in the Grid.
C: Total capacity of the storage medium.

4.1 Simulation Setup

In order to simulate different replication optimization strategies, the simulation configuration must be very closed to reality, consequently, OptorSim adapts EU DataGrid topology and configuration, the grid topology as an input to OptorSim compromises 20 sites in USA and Europe that were used during a data production form CMS experiment [16] as shown in Fig. 3 and the other input is simulated the grid jobs and data files configuration [7].

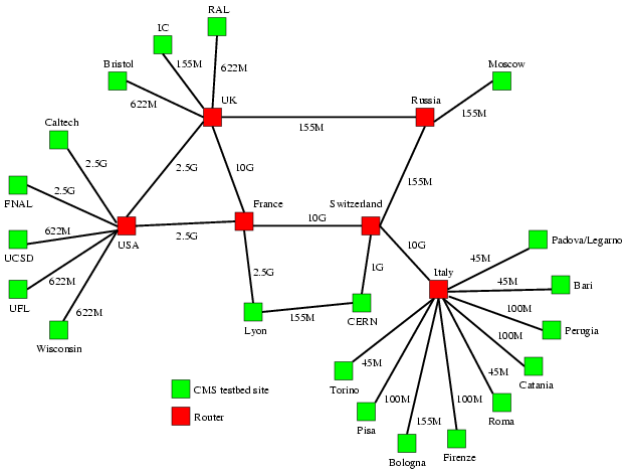


Fig. 3 Grid topology for CMS

CERN and FNAL are producing the original files and store them at their local storage of capacity 100GB each, and other sites has at least one CE and storage capacity 50GB each.

Before presenting the simulation results, we provided the summary of our system functionalities and the correspondence equations as shown in table 3.

Table 3: System functionalities and the correspondence equations

System Functions	Equation Numbers
Replica Creation and Deletion Function	1, 2
Replica Placement Function	3, 4, 5

5. Results and Discussion

RmGrid is compared to other similar algorithms exist in the literature with different scenarios. The existing algorithms that termed as replication strategies and tend to achieve the three objectives as discussed in section 2 are: 1) No replication, such as SimpleOptimizer, 2) Always replicate such as: LFU, 3) Replicate with conditions such as: Best-Client and Economic Approach. The figures listed below in this section show different replications systems which labeled in numbers as shown in table 4.

Table 4: Replication Systems with Data Labeled numbers

Replication System Name	Data Label
No Replication	1
Always Replicate	2
Economy	3
Best Client	4
RmGrid	5

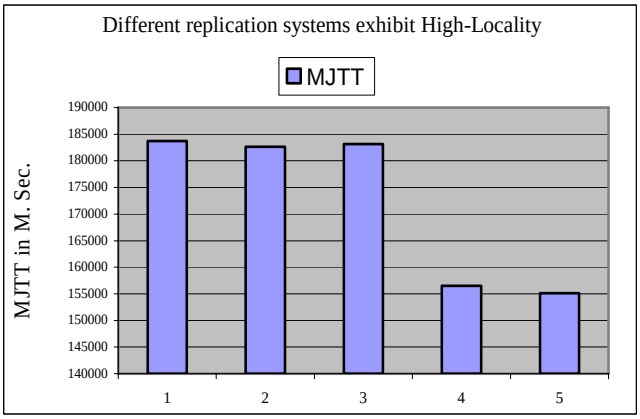
The main parameter that substantially influences the replication strategies is the locality of the request patterns. Kavitha and Ian Foster [3] show that the requests patterns for the data files can exhibit different locality properties, because the scientists tend to work in groups on projects. Moreover, Yi-Fang et al [13] ensure that the data files in data grid systems has special data access patterns that not exists in traditional parallel and distributed systems. Therefore, we run the simulation in three scenarios: High-Locality, Medium-Locality, and No-Locality. The sizes of the files are vary and ranged between 100 MB and 10000 MB. The simulation run with 200 jobs, and the storage access latency assigned randomly to the sites and ranged between 1000 and 10000 (millisecond). The evaluation criteria: MJTT, ASU, and the ENU are used in order to evaluate the overall system performance and ensure that the three mentioned replication objectives are achieved.

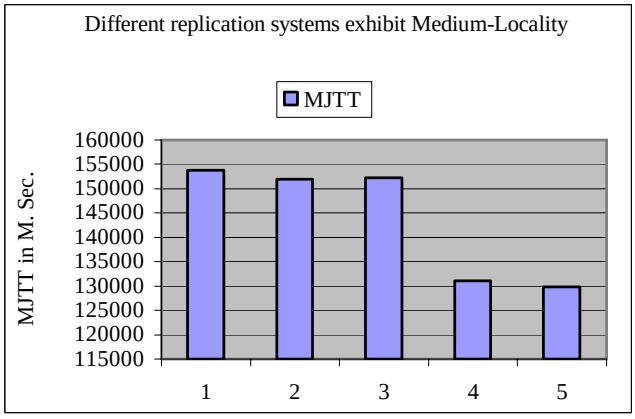
5.3.1 Verifying the Job Turnaround Time

Obviously, the "No Replication" strategy performs the worst as shown in table 5 and Fig. 4, because the data files are not replicated in our experiment; rather only the master files are stored in the grid, and thus the requesting sites have limited choices to select their datasets which lead the delay of the job turnaround time. "Best Client" and RmGrid perform better than others because they achieve local and global optimization, while The "Always Replicate" and the "Economy" achieves only the local optimum as discussed in section 2. However, the MJTT is degraded for all replication systems as locality increases, because the RB tends to submit some kinds of jobs to specific sites even these sites have an over workload, and thus the increasing number of jobs waiting in the queue in these sites is increased as well as the MJTT. Conversely, when the data files have no locality, the RB has more choices when submitting the jobs to the grid sites and can balance the number of jobs at each site.

Table 5: MJTT simulation results for different replication systems

Locality level	No Replication	Always Replicate	Economy	Best Client	RmGrid
High-Locality	283718	182657	183153	156527	155126
Medium-Locality	253855	151950	152240	131150	129898
No-Locality	200520	95898	96287	84100	83374





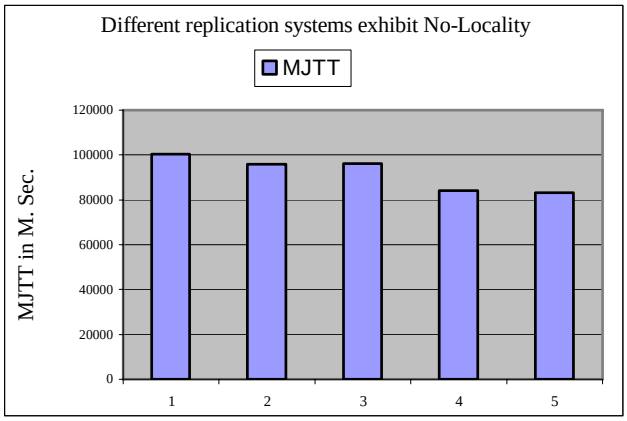


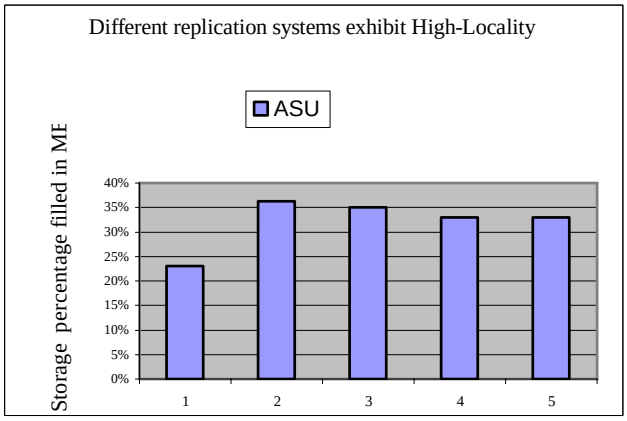
Fig. 4 MJTT in millisecond for different replication systems with different data locality.

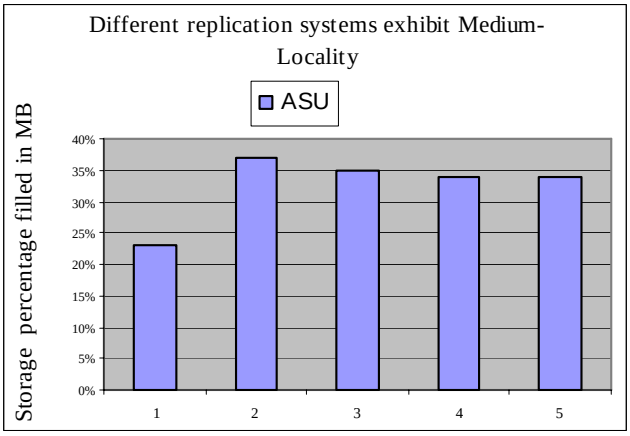
5.3.2 Verifying the Storage Space Usage

As shown in table 6 and Fig. 5, the "No Replication" strategy performs the best in the ASU metric. However, it performs the worst in MJTT as discussed in the previous section and the worst also in ENU as discussed later in next section. Thus as a combination of the three evaluation metrics, the "No Replication" performs the worst. Clearly there is no any impact of the locality on the ASU, but the locality influences the evaluation metrics: MJTT and ENU.

Table 6: ASU simulation results for different replication systems

Locality Level	No Replication	Always Replicate	Economy	Best Client	RmGrid
High-Locality	23%	36%	35%	33%	33%
Medium-Locality	23%	37%	35%	34%	34%
No-Locality	23%	38%	38%	33%	34%





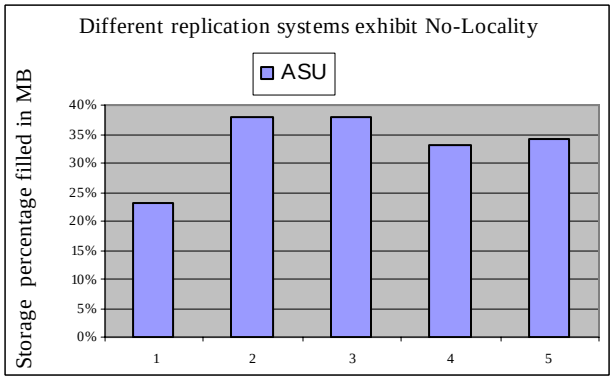


Fig. 5 ASU in MB for different replication systems with different data locality.

5.3.3 Verifying the Effective Network Usage.

The ENU values are ranged between 0 and 1, and we multiplied the results by 100% for more clarity. The less ENU the better performance is, thus the "No Replication" strategy performs the worst and consumed the maximum network bandwidth available in the network. The ENU decreases as the data locality increases for both "Best Client" and RmGrid systems as shown in table 7 and Fig. 6, and the vice versa for "Always Replicate" and "Economy", because global decisions are considered only in "Best Client" and RmGrid systems. Global decisions are based on the overall sites knowledge as a global optimum, thus the global decisions beneficial from theses knowledge in placing new replicas. However, without the global knowledge, the site decides based on its local storage information.

Table 7: ENU simulation results for different replication systems

Locality Level	No Replication	Always Replicate	Economy	Best Client	RmGrid
High-Locality	100%	81%	77%	71%	68%
Medium-Locality	100%	80%	74%	73%	69%
No-Locality	100%	76%	75%	74%	74%

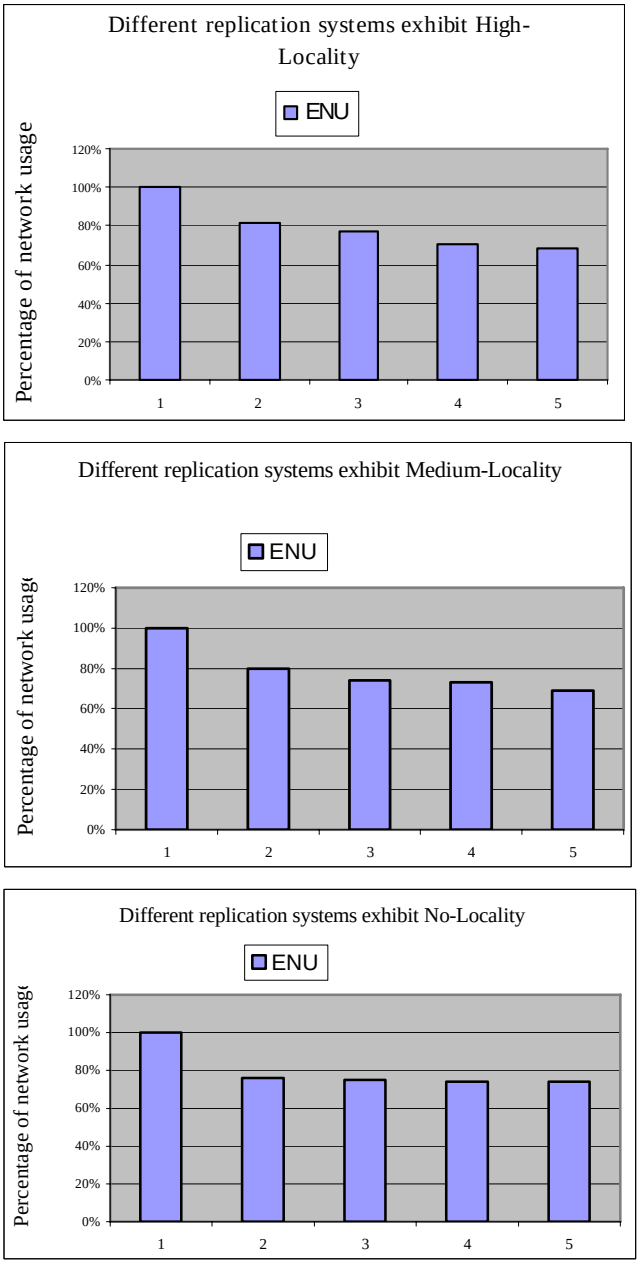


Fig. 6 ENU in percentage form for different replication systems with different data locality.

6. Conclusions and Future Work

Replication strategies are important mechanism in order to face the huge volume of data required by many grid users. Replication management increases data availability and reliability for the jobs which required these data, therefore, the replicas must be managed properly in terms of replica creation, deletion, selection, and placement. We proposed a replication system provides good solution to the underlying problem and outperform the previous traditional replication systems as a combination of the three evaluation criteria. But it may not perform the best in one criterion in some cases such as the ASU criteria.

As a result our system is feasible and applicable in grid environment where the data exhibit some localities. As shown in the results the data locality degraded the job turnaround time, such that, as the locality increases the MJTT increases for all systems, but the data locality degrades our system very little in relative to other system. Therefore, our system is more applicable and feasible in data grid environment where the data exhibit some level of locality. However, the data locality degrade the network usage for other systems, while in our system the network usage improves as the locality increases, because our system best utilize the locality properties by considering the global knowledge of the data replicas and their locations.

As a future work, the previous replica replacement strategy depends on the file popularity or on the file age. But we see that there are other important factors that influence the replacement performance. Moreover, the replacement strategy will be dynamic and depends on these factors.

Acknowledgements

This work was supported by the Universiti Sains Malaysia (USM), Malaysia-Penang

References

[1] R. Brennan, G. O'Gorman, C. Doherty, N. Hurley, C. McArdle, Autonomic Replication of Management Data Evaluation of a Market-based Approach, 1-4244-0143-7/06/$20.00©2006 IEEE.

[2] Henry Lin, J. H. Abawajy, Rajkumar B, Economy-Based Data Replication Broker, Proceedings of the Second IEEE International Conference on e-Science and grid Computing (e-Science'06) 0-7695-2734-5/06 $20.00 © 2006 IEEE.

[3] Kavitha Ranganathan and Ian Foster. "Identifying Dynamic Replication Strategies for a High-Performance Data grid",

Proceedings of the Second International Workshop on grid Computing, Lecture Notes In Computer Science;Vol.2242, 2001, ISBN:3-540-42949-2, pp:75-86.
[4] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Optorsim - a grid simulator for studying dynamic data replication strategies. International Journal of High Performance Computing Applications, 17(4), 2003
[5] K. Holtman, CMS data grid system overview and requirements, Tech. report, CERN, July 2001, CMS Note 2001/037.
[6] Globus Toolkit web site at: http://www.globus.org/
[7] OptorSim. http://grid-data.management.web.cern.ch/grid-data-management/optimisation/optor/
[8] Tim Ho and David Abramson, The GriddLeS Data Replication Service, Proceeding of the first international conference on e-science and grid computing (e-science'05), 0-7695-2448-6/05 $20.00 © 2005 IEEE.
[9] Ming Tang, et al. "Dynamic Replication Algorithms for the Multi-Tier Data Grid", Future Generation Computer System 21(2005) 775-790, Science Direct Journal © 2004 Elsevier.
[10] William H. Bell1, David G. Cameron1, Ruben Carvajal-Schiaffino2, A. Paul Millar, Kurt Stockinger, Floriano Zini, Evaluation of an Economy-Based File Replication Strategy for a Data grid, Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the grid (CCGRID.03) 0--7695-1919-9/03 $17.00 © 2003 IEEE.
[11] Mark Carman, Floriano Zini, Luciano Serafini, Kurt Stockinger, Towards an Economy-Based Optimisation of File Access and Replication on a Data grid, Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID.02) 0-7695-1582-7/02 $17.00 © 2002 IEEE.
[12] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, Journal of Network and Computer Applications 23 (2001) 187-200.
[13] A. Chervenak et. al, Giggle: A framework for constructing scalable replica location services, Proc. of the ACM/ IEEE Super Computing Conference, November 2002.
[14] Kavitha Ranganathan, Adriana Iamnitchi, Ian Foster, "Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities", Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID.02) 0-7695-1582-7/02 $17.00 © 2002 IEEE.
[15] Replica Optimization Service (ROS) can be found at link http://edg-wp2.web.cern.ch/edg-wp2/optimization/ros.html
[16] CMS Data Challenge 2004. http://www.uscms.org/s&c/dc04/
[17] Kihyeon Cho, "A test of the interoperability of grid middleware for the Korean High Energy Physics Data Grid system", IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.3, March 2007.

Mr. Husni Hamad Al-Mistarihi is a PhD candidate in the School of Computer Sciences, Universiti Sains Malaysia (USM). He is interesting in grid computing, in particular data grid. His research spans from grid computing, to a more specific issues such as data grid architecture, replica selection service, replication management, resource discovery, and resource load balancing, allocation.

Dr. Chan Huah Yong is a senior lecturer in the School of Computer Sciences, Universiti Sains Malaysia (USM). He is actively involved in grid computing research activities, both, at the national and international level. He is also one of the project leaders of the *e-Science Grid* pioneer project in Malaysia, funded by the Ministry of Science, Technology and Innovation. His research spans from grid computing, to a more specific issues such as resource allocation, load balancing, software agents, and middleware engineering.