DISTRIBUTED POSITIONING OF REPLICA USING GRASS GROWING STRUCTURE

Mr. Worawit Fankam-ai

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science and Information Technology

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2014

การกระจายตำแหน่งของแบบถอดโดยใช้โครงสร้างการเติบโตของหญ้า

นายวรวิทย์ ฝั้นคำอ้าย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2557

Thesis Title	DISTRIBUTED POSITIONING OF REPLICA USING GRASS GROWING STRUCTURE
By	Mr. Worawit Fankam-ai
Field of Study	Computer Science and Information Technology
Thesis Advisor	Associate Professor Peraphon Sophatsathit, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Science

(Professor Supot Hannongbua, Dr.rer.nat.)

THESIS COMMITTEE

.....Chairman

(Professor Chidchanok Lursinsap, Ph.D.)

.....Thesis Advisor

(Associate Professor Peraphon Sophatsathit, Ph.D.)

.....External Examiner

(Associate Professor Damras Wongsawang, Ph.D.)

วรวิทย์ ฝั้นคำอ้าย : การกระจายตำแหน่งของแบบถอดโดยใช้โครงสร้างการเติบโตของหญ้า (DISTRIBUTED POSITIONING OF REPLICA USING GRASS GROWING STRUCTURE) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. พีระพนธ์ โสพัศสถิตย์ดร., หน้า.

การสำรองซ้ำข้อมูลเป็นเทคนิคหนึ่งที่เพิ่มประสิทธิภาพของการเข้าถึงข้อมูลในระบบการกระจาย วิธีการสำรองซ้ำมีหลากหลายที่ใช้กันในเทคนิคการสำรองซ้ำ เช่น กลยุทธการสำรองซ้ำ กลยุทธ์การเลือกข้อมูลสำรองซ้ำ เพื่อหาขนาดของข้อมูลสำรองซ้ำที่เหมาะสมที่ดีที่สุด วิธีการสำรองซ้ำที่สอดคล้องกัน และกลไกในการวางตำแหน่งของข้อมูล

วิทยานิพนธ์นี้เน้นเทคนิคการกระจายการสำรองซ้ำข้อูลโดยใช้โครงสร้างการเจริญเติบโตของหญ้า เพื่อลดการใช้แบนด์วิธและความล่าช้าในการเข้าถึงข้อมูล การสำรองซ้ำจะถูกกระจายไปสู่โหนดใกล้ที่สุดในระยะทางที่กำหนดล่วงหน้าด้วยระเบียบวิธีจำกัดการสืบค้นเชิงลึก วิธีดังกล่าวนำไปเปรียบเทียบกับระเบียบวิธี centralized, flooding, multi-master และ random walk การประเมินผลใช้ OptorSim ซึ่งทำงานในเครือข่ายมาตรฐานการวัดผล 3 อันคือ EU Data Grid Testbed, CMS Testbed และ GridPP Testbed แล้ววัดประสิทธิภาพจากมาตรวัดการทำงานของเครือข่าย ได้แก่ เวลาเฉลี่ยที่ใช้ ประสิทธิภาพการใช้เครือข่าย การเข้าถึงแฟ้มข้อมูลส่วนท้องถิ่น และการใช้แบนด์วิท สถิติของผลการวัดแสดงให้เห็นว่าวิธีที่เสนอทำงานได้ดีกว่าวิธีอื่น ซึ่งจะเอื้อประโยชน์ต่อการประหยัดพลังงานในการส่งข้อมูลแบบกระจาย

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต .....
สาขาวิชา	วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....
ปีการศึกษา	2557	

5373613423 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORDS: REPLICA LOCATION / DISTRIBUTED POSITIONING / GRASS GROWING STRUCTURE

WORAWIT FANKAM-AI: DISTRIBUTED POSITIONING OF REPLICA USING GRASS GROWING STRUCTURE. ADVISOR: ASSOC. PROF. PERAPHON SOPHATSATHIT, Ph.D., pp.

Data Replication is a common technique used to enhance the performance of data access in distributed systems. There are many efficient replica schemes involved in replication technique such as replication strategy, replica selection strategy to find the best fit replica, replication consistency, and replica positioning mechanisms

This thesis focuses on a distributed data replication technique based on Grass Growing Structure to reduce bandwidth consumption and access latency. The replication will be multicast to the nearest nodes within predefined limiting distance using Depth Limit Search algorithm. The proposed algorithm is then compared with centralized, flooding, multi-master and random walk algorithms. Evaluation is carried out by a simulation using OptorSim to run on standard benchmarking testbeds, namely, EU Data Grid Testbed, CMS Testbed and GridPP Testbed. Performance measurement is done based on a number of network metrics such as mean job time, effective network usage, local file access, and Band width usage. Result statistics show that the Grass Growing Structure performs better than all comparative algorithms which will entail energy conservation in network data distribution.

Department:	Mathematics and Computer Science	Student's Signature	.....
		Advisor's Signature	.....
Field of Study:	Computer Science and Information Technology		
Academic Year:	2014		

ACKNOWLEDGEMENTS

I would like to acknowledge my thesis advisor, Associate Professor Dr. Peraphon Sophatsathit for helpful guidance and encouragement. He has suggested the solutions to many experimental problems and helped me finish this thesis in time.

Finally I would like to thank my family and friends for everything they suggested and supported me.

CONTENTS

Page

Page

List of Figure

List of Tables

CHAPTER 1
Introduction

This chapter presents an introduction of this thesis. The work starts with a high-level overview of key concepts relating to the research problem under investigation. Then the fundamental motivations behind this research are stated and the proposed research challenges are briefly presented. The chapter ends with a discussion on the research contributions.

1.1 Background

Large-scale scientific applications in the areas of high-energy physics, biology science, and earth sciences involve processing of large datasets from simulations or from large-scale experiments. Analysis of these datasets and their dissemination among researchers located over a wide geographical area require high capacity resources such as supercomputers, high bandwidth networks, and mass storage systems. Many of applications may also require new paradigms that address issues such as multi-domain applications, co-operation and co-ordination of resource owners and removing system boundaries. Grid computing is one such paradigm that enables the aggregation of large scale computing, storage, and networking resources.

A grid provides an environment where a widely distributed scientific community can share theirs resources, across organization, to solve large-scale compute and data-intensive problems and collaborate in a wide variety of disciplines. The grid enables the creation of a virtual environment including a pool of physical resources across different administrative domains; these resources are then abstracted into computing or storage units that can be transparently accessed and shared by large numbers of remote users.

The grid computing concepts are not new. The invention of networking and the introduction of network operating systems enable access to resources across geographically distributed locations. More technological advances brought up by

parallel processing and distributed computing allow not only remote access to resources but also the simultaneous sharing of these distributed resources by different remote users. Parallel processing enables different tasks to be run simultaneously on different, usually homogeneous computers, and to compete for access to computational resources. In distributed computing, users can employ widely distributed heterogeneous computers to run jobs that require more resources than may be available in local networks and laboratories. The emerging need for more resources and also for collaborative problem solving in cost efficient ways lead to the development of middleware that transparently provides access to distributed resources and route data from back-end sources to end-user applications in a seamless and relatively scalable manner. Grid computing has the potential to support different kinds of applications. There are various types of grids have been developed to support these applications and have been categorized as follows.

1. Data Grids. These provide the infrastructure to access, transfer and manage large datasets stored in distributed repositories. Experiments in high energy particle physics such as the CMS and ALICE [1, 2] experiments running on the Large Hadron Collider (LHC) produce and collect massive amounts of data and involve thousands of researchers from around the world to analyze the data and initiate future experiments at European Center for Nuclear Research (CERN).

2. Computational Grids. These provide distributed computing facilities for executing compute-intensive applications, such as Monte Carlo simulations[1]

3. Interaction Grids. These provide services and platforms for users to interact with each other in a real-time environment, e.g. Access Grid [3]. This type of grid is suitable for multimedia applications, such as video conferencing, Virtual Reality Application [4] and those that otherwise require fast networks.

4. Application Service Provisioning (ASP) Grids. These focus on providing access to remote applications, modules, and libraries hosted at data centers or on computational grids(e.g. NetSolve[5]).

5. Knowledge Grids. This application works on knowledge acquisition, data processing, and data management. Furthermore, they provide business analytics services driven by integrated data mining services. Some projects in this field are Knowledge Grid [6] and the EU Data Mining Grid [6].

6. Utility Grids. These focus on providing one or more of the above grid services to end-users as information technology (IT) utilities on a pay-to-access basis.eg. Utility Data Center [7], and Gridbus [8].

This thesis focuses specifically on data grid and aims at developing techniques to achieve faster data access and less bandwidth usage. Based on existing data grid applications the size of the data is expected to be multiple terabyte or even petabyte scale for some applications. Storing huge amounts of data in a centralized manner is impractical due to the slowness of remote data access and concerns about a single point of failure. Given the high latency of wide-area networks that underlie many grid systems, and the need to access or manage multiple petabytes of data in data grid environments, data availability and access optimization become key challenges to be addressed.

In most situations, users jobs request the datasets cannot be found at the local nodes in the data grid. In this case, data must be fetched from other nodes in the grid which incurs high access latency. An important technique to speed up access in data grid systems is to replicate data at multiple locations, so a user can access the data from a nearby site[9].

1.2 Research Objectives

1. Establish a new distributed data replication scheme
2. Devise an efficient replication positioning algorithm to govern the above proposed scheme

3. Compare the new distributed data replication scheme with another algorithm in terms of mean job time, network usage, Number of file found in local file access ,bandwidth

1.3 Scope of work

This research will confine the scope with the following constraint

1. Compare performance of propose algorithm with Flooding algorithm [10-12], Centralized Location algorithm [10-14],Multi-Master[15, 16] and Random walk based distributed algorithm[17-19]
2. focus on replication-positioning algorithm
3. The unit of measurement is Mean Job Time, Network Usage, Number of file found in Local File Access and Bandwidth
4. The limit of grid simulation to support only data replication

1.4 Expected Outcome

1. improve speed of data replication
2. increase replication performance efficiency on low latency, good scalability, and reliability

CHAPTER 2
Related Work

This chapter will provide an introduction of distributed system, discuss some issues and challenges in replication for data grids in Section 2.1 and describe replication location, terminology and definition in Section 2.2, 2.3 and 2.4.

2.1 Distributed Systems

Distributed system[14, 20] is a collection of independent entities cooperating to solve the problem that cannot be solve individually on single computer. Distributed systems cover all on computing and information access across multiple processing elements connected by communication network. Distributed Systems can have following characteristic features: no common physical clock, no shared memory, geographical separation, autonomy, and heterogeneity. The motivation for using a distributed system is inherently distributed computations, resource sharing, access to geographically remote data and resource, enhanced reliability, increased performance/cost ratio, scalability, modularity, and incremental expandability. The topology of distributed systems take various forms, namely, grid, star, ring, etc. This research will focus on data grid topology. Data Grid is a very important and useful technique to process large number of data produced by scientific experiments and simulations. However, high latency of the Internet turns to be the bottleneck in accessing the files in the grid can shorten the time of getting the files by creating many replicas and storing the replica in appropriate locations. As such, distribution can achieve higher performance than without replication.

2.2 Data Replication

Data Replication[14, 20] is a common method used to improve the performance of data access in distributed systems. It improves not only data access efficiency, but also data availability and fault tolerance. In order to achieve higher replication performance, there

must be an efficient replica scheme to manage the replication process. Replica scheme mainly includes replication strategy and replica selection strategy to find the best-fit replica, replication consistency, and replica positioning mechanisms. Replication strategy determines when and where to create a replica, taking into account of the factors such as number of data requests, network condition, and storage availability of each replica site.

Data replication can provide good performance, high availability and fault tolerance in distributed systems. When data are stored at a single data server that can cause a potential bottleneck due to too many connections, the whole system can slow down. Moreover, when a data item is stored at a single server and the server crashes, the data item becomes inaccessible. Thus, the availability of data could be increased under replication scheme.

Although data replication is one of the major optimization techniques for promoting high data availability, low bandwidth consumption, increased fault tolerance, and improved scalability, the problem of replica location has not been well studied for large-scale grid environments. To obtain the maximum possible gains from file replication, strategic location of the file replicas in the system is critical. The replica location service is a component of data grid architecture that decides where in the system a file replica should be placed. In fact, different replication strategies can be defined depending on when, where, and how replicas are created and destroyed.

Within grid community, much work have been done on providing the basic infrastructure for a typical grid environment such as Globus [5, 21], Condor [22], and EU Data Grid [23]. These systems have contributed substantially to core grid middle ware services that are available as the basis for further application development. In order to study the complex nature of a typical grid environment and evaluate various replica optimization algorithms, a grid simulator called OptorSim developed by Willian H Bell,David G. Cameron, Luigi Capozza, A. Paul Millar, Kurt Stockinger, Floriano Zini [23, 24] was employed.

2.3 Network Topology Testbeds

A. EU DataGrid Testbed

Network Model of EU DataGrid Testbed [23] sites and their associated network geometry is illustrated in Figure 2.1. Each site is allocated storage resource proportional to their actual hardware allocations. Site S0 is the CERN (European Organization for Nuclear Research) location. The star denotes a router and the circle denotes a site. Each link shows the available bandwidth between two connecting sites. In this experiment, each testbed site, excluding CERN, was assigned computing and storage elements. The CERN was allocated a Storage Element to hold all the master files but was not assigned any Computing Elements (CEs). A CE ran jobs that used data files stored on Storage Elements (SEs). Nodes without Computing or Storage Elements acted as network nodes or routers.

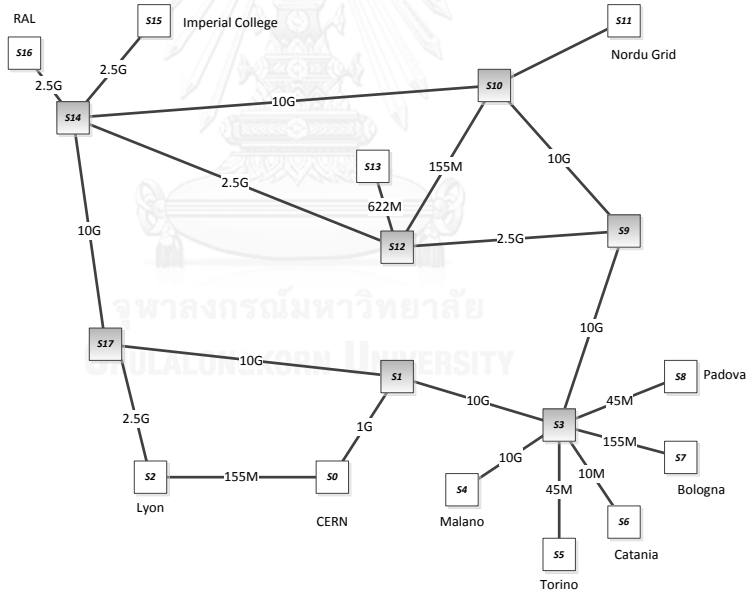


Figure 2.1 EU Data Grid Testbed sites and their associated network topology

The numbers indicate bandwidth between the two ending sites in Mbit/s(M) or Gbit/s(G). Stars denote routers and circular nodes denote replica sites

B. GridPP 2004 Testbed

GridPP 2004 testbed [11] is a collaboration of particle physicists and computer scientists encompassing 17 grid sites in UK and one at CERN in Switzerland. Each UK site has 40 to 1800 processing nodes and has a storage capacity between 5TB and 500TB.

CERN has 1000TB of storage to hold all master files as shown in Figure 2.2

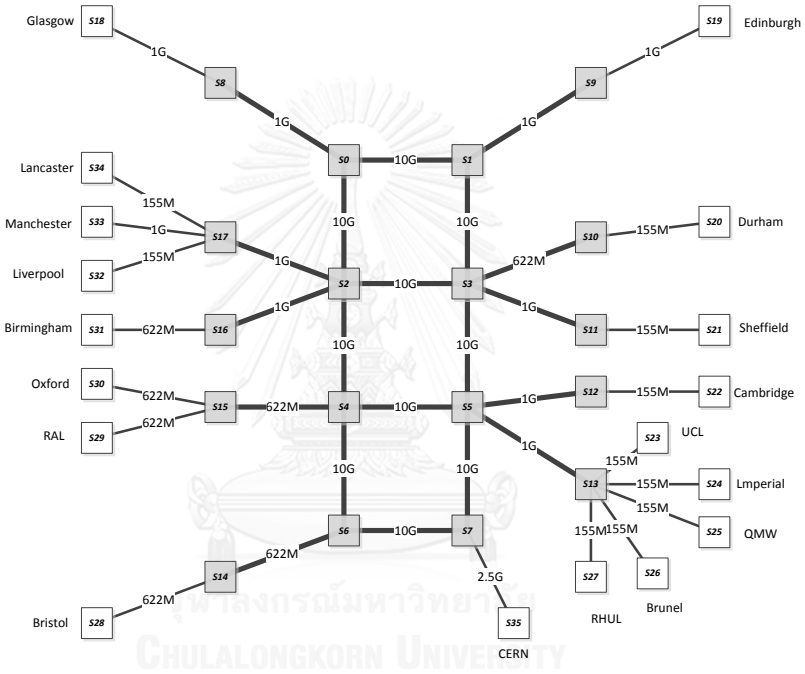


Figure 2.2 show GridPP resources and topology in 2004

The numbers indicate bandwidth between the two ending sites in Mbit/s(M) or Gbit/s(G).

C. CMS Testbed

Compact Muon Solenoid [2] or CMS is a General Purpose Detectors for the Large Hadron Collider (LHC) at CERN, Geneva, Switzerland. CMS Testbed has 20 sites in Europe and the USA. CERN and FNAL have a storage capacity of 100 GB each and a master copy of each file is stored at one of these two sites. Every other site has Computing Element (CE) with 50 GB capacity as shown in Figure 2.3

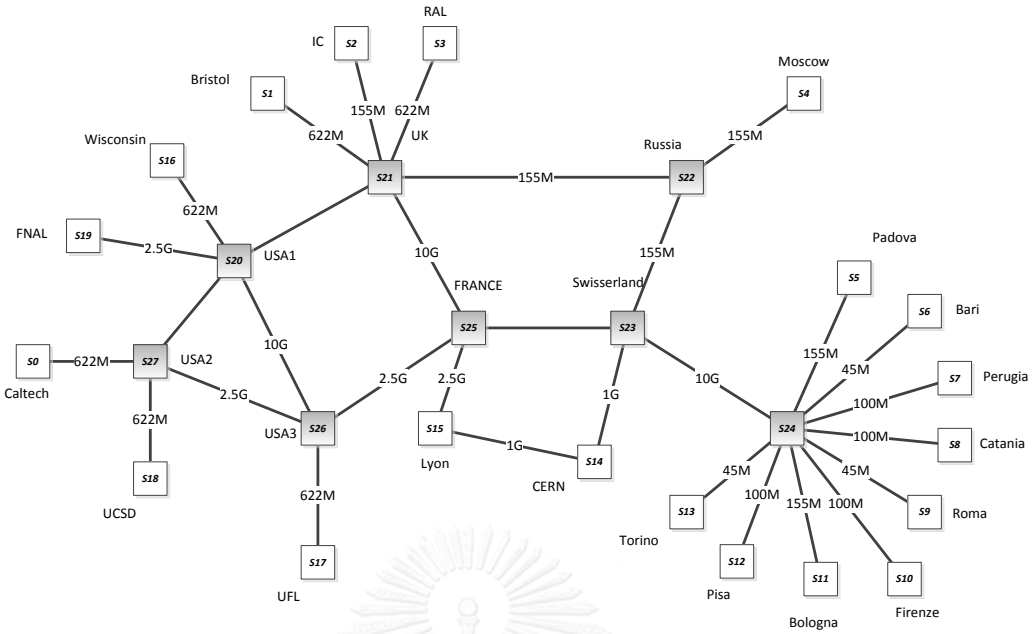


Figure 2.3 CMS 2004 network toplogy

W.H Bell [24] , Wenjuan [20] proposed grid optimization algorithm using EU Data Grid Testbeds. D.G. Cameron [12, 24, 25] proposed CMS Network Topology, while Cole J[11] proposed GridPP network testbed for data grid. Junzhou [10], Li Zeng[20] proposed data grid that compared centralized location by using OptorSim. Runqun [10] compared their proposed algorithm with Flooding Algorithm and Centralized Location Algorithm by using OptorSim with EU Data Grid Testbed as well.

2.4. Terminology

A logical file name (LFN) [12, 24, 25] is a unique logical identifier for desired data content. The replica location service must identify one or more physical copies (replicas) of the logical file. Each physical copy is identified by a physical file name (PFN), which specifies its location on a storage site.

A number of storage sites (SS) [12, 24, 25] collaborate to share their storage capabilities to all users. A replica location node (RLN) [12, 24, 25] aggregates LFN [12, 24, 25] to PFN [12, 24, 25] mappings from one or more SSs and collaborates with other RLNs to build a distributed catalog of LFN mappings. RLNs offer both a query interface

to clients and a registration interface that SSs can enlist PFN to LFN mapping for files stored locally. RLNs also organize into a search network to allow remote searches. Nodes in this network distribute compressed information on the set of LFN mappings stored locally in the form of node digests.

In a nutshell, RLN organize into a flat overlay network and distribute their digests using a soft-state mechanism illustrated in Figure 2.4

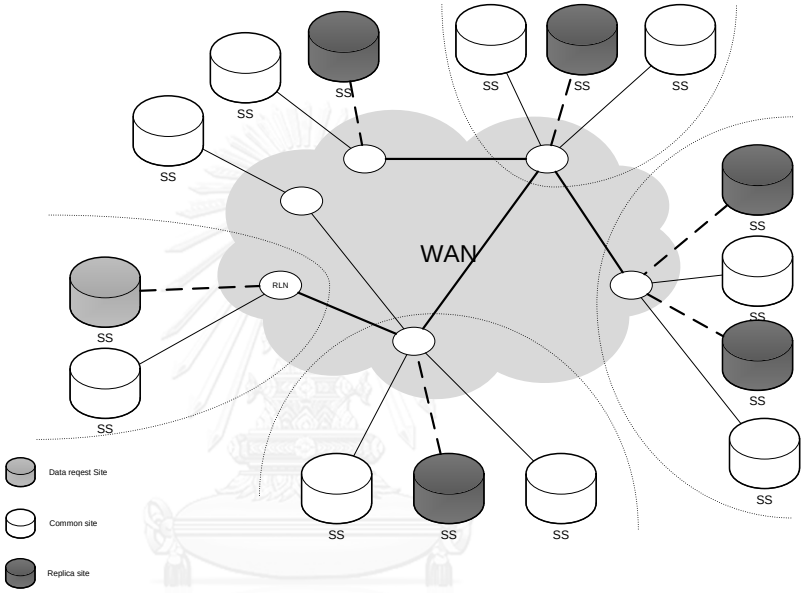


Figure 2.4 Replica location service organization

Depth Limit Search (DLS) [26], like the normal depth-first search, is a uniform search. It works exactly like depth-first search, but avoids the completeness drawbacks by imposing a maximum limit on the depth of the search. Even if the search could still expand a vertex beyond that depth, DLS will not do so and thereby will not follow infinitely deep paths or get stuck in cycles. Therefore depth limited search will find a solution if it is within the depth limit, which guarantees at least completeness on all graphs. The pseudocode of DLS algorithm is shown in Figure 2.5.

```
DLS(node, goal, depth) {
  if ( depth >= 0 ) {
    if ( node == goal )
      return node
   for each child in expand(node)
      DLS(child, goal, depth-1)
  }
}
```

Figure 2.5 Pseudo code of Depth Limit Search

The mean job execution time [12, 24, 25] is defined as the total time to execute all job divide by the number of jobs completed. Grid user would consider it to be one of important measure of how the algorithm is performing.

$$Mean\ Job\ Execution\ time = \frac{Total\ Time}{number\ of\ job\ complete}$$

Effective Network Usage or ENU (r_{ENU}) [12, 24, 25] is defined as network usage after executing all the grid jobs as follows:

$$r_{ENU} = \frac{N_{remote\_file\_access} + N_{file\_replications}}{N_{local\_file\_access}}$$

where $N_{remote\_file\_access}$ is the number of times the CE reads a file from different SE sites. $N_{remote\_file\_access}$ is the number of times a CE reads a file from an SE on the same site. For a given network topology, a low value of r_{ENU} indicates that replication is a better optimization strategy than locating another site.

Number of file found in Local File Access is total number of remote file access that took place and remote file access is total number of remote file accesses that took place.

2.5 Algorithms on Distributed Storage replication

- Centralized Location Algorithm [10] is shown in Figure 2.6. V4 is the only RLN node, denoted by V_{rln} which contains all the information about location of V_{SS} and LFN to PFN mappings

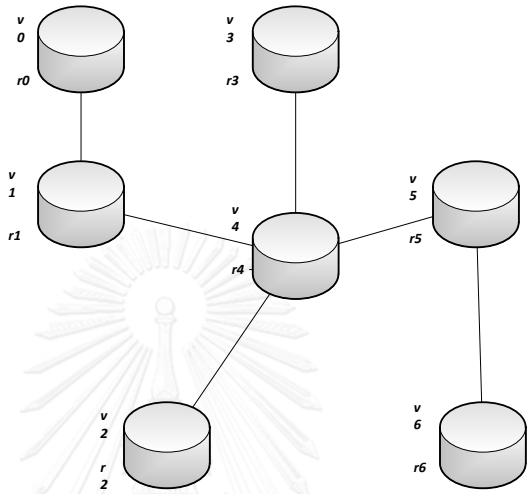


Figure 2.6 Centralized Location Algorithm

- Flooding algorithm [10] progresses the distribution location starting from the node $V_0 \in V_{SS}$ that stores the information about the location of V_{SS} but does not include the corresponding relationship between files and V_{SS} That is to say, the algorithm does not know the location of V_{SS}before locating one replica *r*. In all the information for any node $V_i \in V_{SS}$ stored in the node V_0, the corresponding location l_{vi} is unknown. Thus, this algorithm can be costly in terms of wasted bandwidth while a message may only have one destination to be sent. Moreover, messages can duplicate in the network which increase the load on the network bandwidth. Worse yet, duplicate packets may circulate forever unless certain precautions are taken. The model is depicted in Figure 2.7.

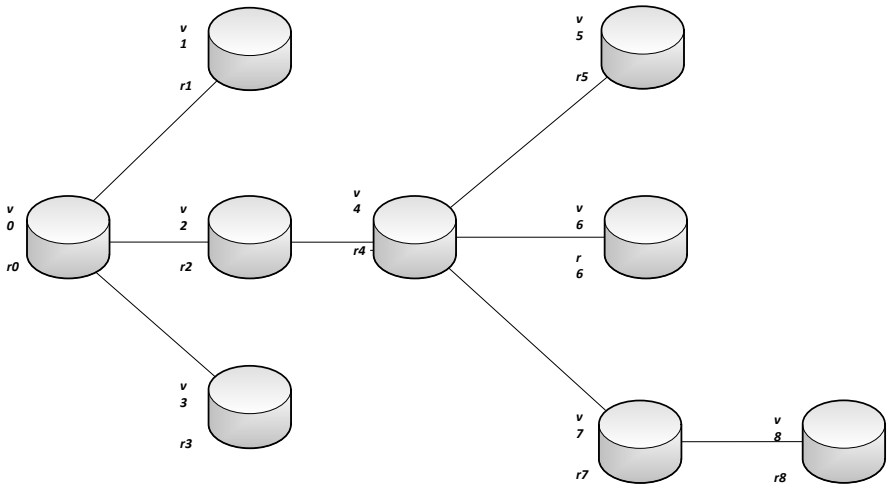


Figure 2.7 Flooding algorithm

- Multi-master replication [15, 16] is a method of data replication which allows data to be stored by a group of computers, and updated by any member of the group. All members are responsive to client data queries. The multi-master replication system is responsible for propagating the data modifications made by each member to the rest of the group, and resolving any conflicts that might arise between concurrent changes made by different members.

- Random Walk based distributed algorithm [17, 18] is an algorithm involving a particular message, the token, that circulates according to random walk scheme. In each step, one node processes the token after a treatment that only the node holding the token is allowed to proceed. The node then chooses one of its neighbors randomly and sends the token to it. Random walk based distribution algorithms do not require any assumption about the topology of the system and are designed to tolerate topological changes.

CHAPTER 3
Design of Research Methodology

This chapter will provide an principles of grass growing algorithm including structure of grass plant, clonal plant and how plant represent as network structure in Section 3.1 and Distributed Positioning of Replica using Grass Growing Structure in 3.2

3.1 Principles of Grass Growing Algorithm

The structure of grass plant [27] is similar among the many species of grasses as shown in Figure 3.1. A grass plant is a collection of tillers or shoots that grow from buds at the base of the plant. Each tiller is composed of a series of repeating units consisting of a leaf, stem node, stem internode, and a bud. Each leaf is attached to the stem at a node. Early in the development of a grass tiller, the distance between nodes (internodes) on the stem is very short and the stem remains compact at the base of the plant. At the top of stem is the growing point where new stem and leaves originate. As long as the growing point remains intact, it is capable of producing new leaves. Later in the development of the tiller growing point undergoes a change. It stops producing leaves and begins to form the immature seed head of the plant. After that, the growing point on this tiller is no longer capable of producing any more new leaves, and grazing or clipping it off has no impact on further new leaf numbers. Once this transition occurs, some of the upper internodes begin to elongate and eventually raise the seed head to the top of the tiller. New tillers will emerge from the plant crown as regrowth.

In Figure 3.2, the stolon is an above-ground, trailing stem that typically produces roots at the nodes where leaves and stems arise. This very invasive perennial grass also produces creeping underground stems called rhizomes. Thus, grass is the plant that spreads by means of stolon.

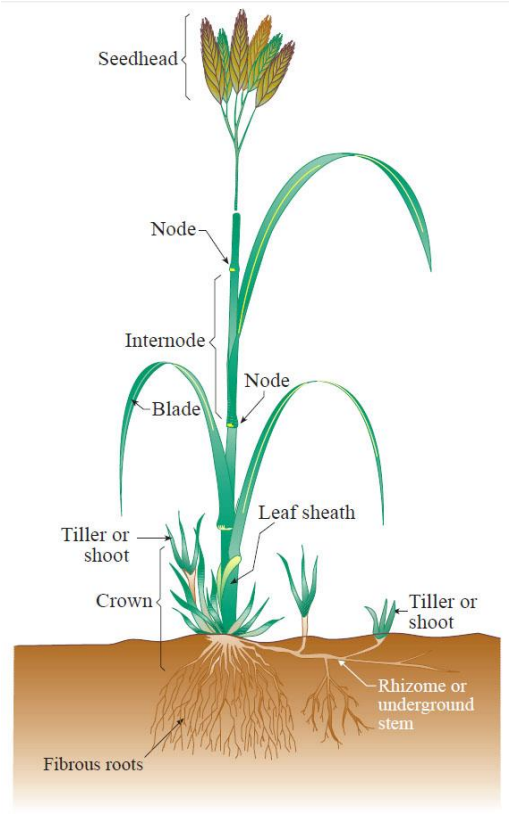


Figure 3.1 A structural illustration of a grass plant

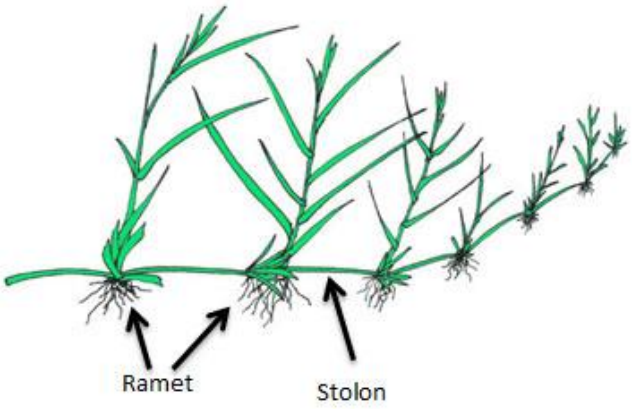


Figure 3.2 Ramets and Stolon

In most clonal plants growth [28] or grass growth can be processed horizontally via the development of modified stems, rhizomes, or stolon, connecting ramets. Ramets are potentially autonomous new individuals and possess aerial and below ground organs in order to sample and uptake resources like water, light, organic nutrients,

nitrates, or phosphorus. This network structure provides the ability to colonize space and allows the exchange of resources and information

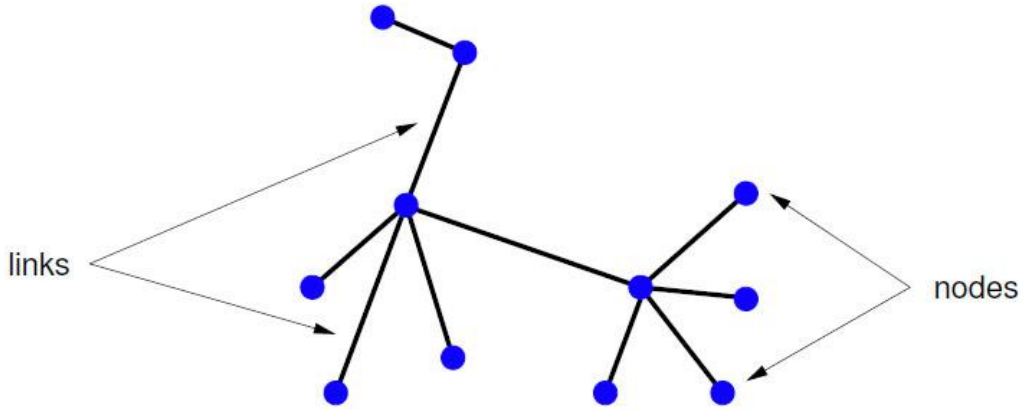


Figure 3.3 The plant is represented as a set of nodes connected by links, where the nodes can be seen as rame

In Figure 3.3, clonal plants is represented as a set of node (ramets) that may be connected by links such as rhizomes or stolons; ramets are represented by points in the plane, and connected by straight lines between two ramets. All ramets and connections are assumed to be identical in terms of demographic parameters and resource consumption.

This thesis focuses on clonal plant or grass that grows on horizontal surface stems termed. Naturally, the areas of growing grass must have adequate irrigation which allows grass to flourish better than the one depleting of water. Grass areas that receive water represent the position of data replicas, having grass trunks as the replica links that form the distribution topology. The principles of Grass Growing Algorithm are approaches to position the replicas that follow grass growing pattern, aka Grass Growing Structure. This proposed topology will be further explored in the sections that follow.

3.2 Distributed Positioning of Replica using Grass Growing Structure

The proposed approach to position the replica will follow the Grass Growing Structure. Figure 3.4 shows an example of grass growing structure. The left figure

represents grass trunk or stolon, wherein water or information flows follow the designated directions, i.e., a directed grass growing structure. The right figure illustrates link distribution with nodes that represent replica positions. Under this topology, suppose the current source node (of information) is 4, the nearest nodes to 4 are 2, 6, 7, 8. Based on grass growing structure, if grass receives flow of water at node 4, node 6, 7, 8 will subsequently flourish as oppose to node 3 and 5. This explanation implies that the replication of information will proceed toward node 6, 7, 8.

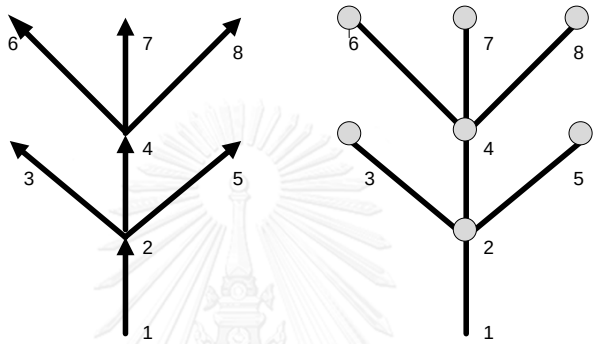


Figure 3.4 Grass Growing Structure

In general, the proposed grass growing structure will be represented by a non-directed topology that implies bi-directional flow of replication.

The first stage positioning of replicas is to set up a random initial source node. Selection of neighboring nodes is carried out by DLS algorithm [18] to find the path between source node and neighboring nodes. The procedure starts by traversing to all neighboring nodes. In the process, it finds the shortest distance and updates the distance cost. The nearest neighboring nodes just visited will be used as the second stage positioning of replica source nodes. This process repeats until one of the nearest neighbor nodes is the destination node. At which point, data replication commences. Thus, considerable network traffic is reduced compared with conventional flooding algorithm. Note that the recursion of replication search may continue indefinitely in a large network configuration. To avoid such a problem, the proposed Grass Growing Structure algorithm imposes a stopping criterion on by limiting the DLS depth to 2 for this study to prevent indefinite depth search. The algorithm is illustrated below.

```
Begin
    SET networktopology is selected network topology
    SET v_0 is initial node in networktopology
// confine depth of search to local replica site
// no deeper (or farther) than 2 hops from the initial
// node, thus taking advantage of local access
    SET depth limit = 2
    SET depth = 0
     // Select a node v_i using DLS algorithm
     WHILE DLS limit != depth
       IF v_i NOT BUSY THEN
             Replicate data on node v_i
       ENDIF
       INCREMENT depth ++;
     ENDWHILE
END;
Function DLS(node ,depth) {
  if ( depth >= 0 ) {
     return node
    for each child in expand(node)
     DLS(child, depth-1)
  }
}
```

Figure 3.5 Grass Growing Algorithm

In actual situation, each replica of the latter stages will be used as the source node of the new grass growing structure for distributed replica positioning sub-problem. The following examples will demonstrate the application of grass growing algorithm on

the three network topology testbeds, namely, the EU Data grid testbed, GridPP 2004 testbed and CMS testbed.

3.2.1 Example Use on EU Data Grid testbed

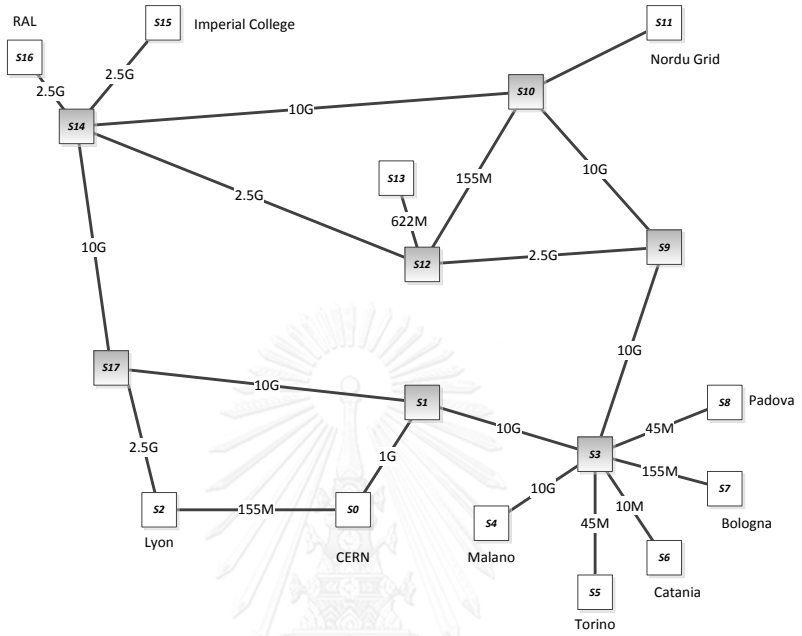


Figure 3.6 EU Data Grid Network Topology Testbed

In Figure 3.6, the initial source node is S0 (CERN). The grass growing structure algorithm replicates the data in the following order: (S0), (S2), S17, S14, S1, S3, (S4), (S5), (S6), (S7), (S8), and (S9).

3.2.2 Example Use on GridPP 2004 Testbed

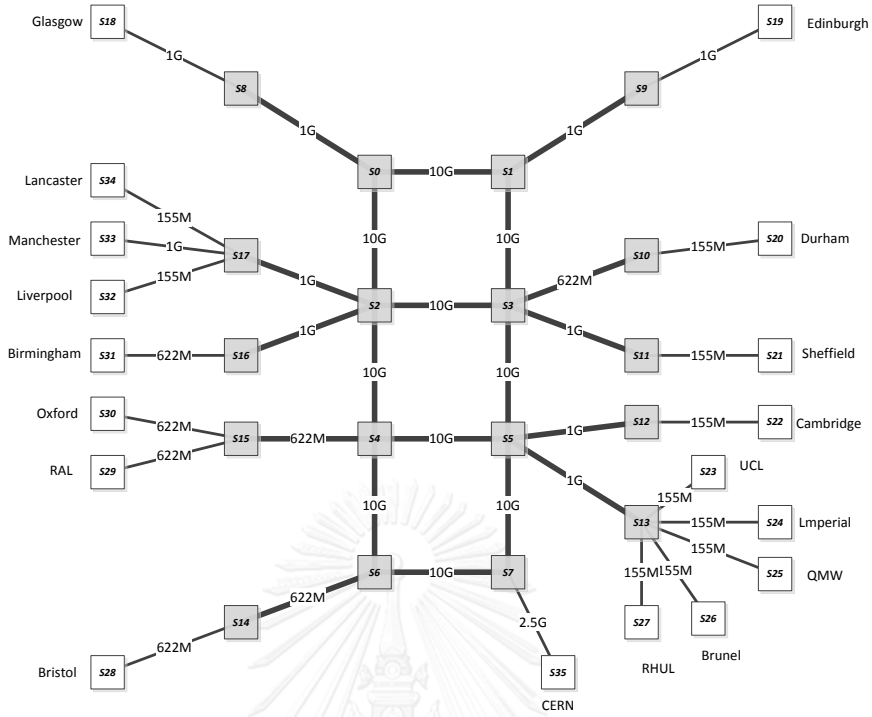


Figure 3.7 GridPP 2004 Testbed

In Figure 3.7, the initial source node is S27.The grass growing structure algorithm replicates the data in the following order: (27),13,(26),(25),(24),(23),5,7,4,3.

3.2.3 Example Use on CMS Testbed

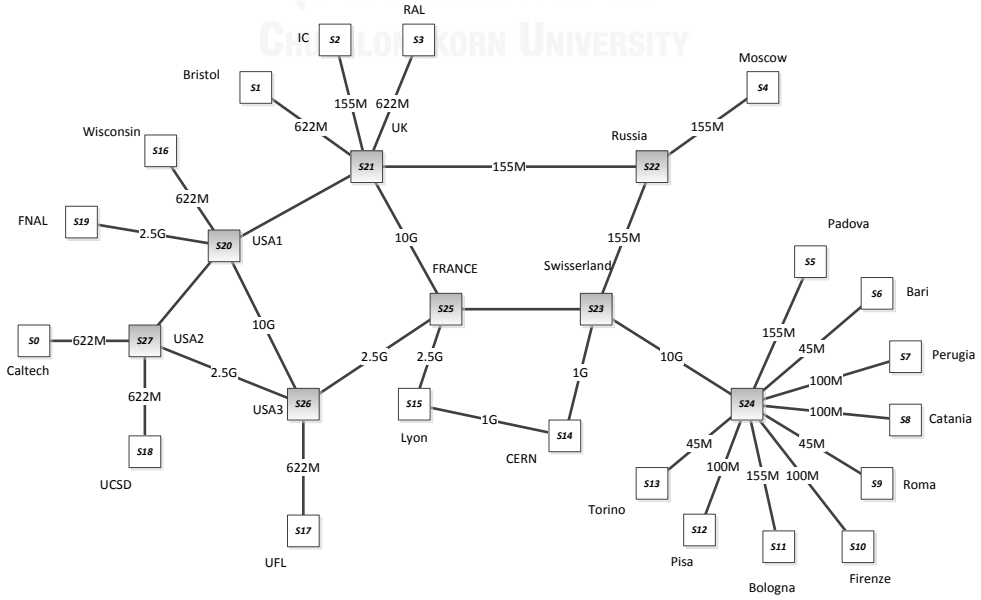


Figure 3.8 CMS Testbed

In Figure 3.8, the initial source node is S13 The grass growing structure algorithm replicates the data in the following order: (13),(12),(11),(10),(9),(8),(7),(6),(5),14,25,3.

CHAPTER 4
Simulation and evaluation

This chapter describes the experiments and analysis of experimental results on the performance of Grass Growing Structure algorithm, as well as other comparable algorithms. They are Flooding, Centralized Location, Multi-Master Location, and Random Walk.

4.1 The Experiments

The proposed method was carried out by a simulation using OptorSim [12][13][14] as the data grid simulator to simulate real data grid environment. This simulator is developed in Java under the funding of the European Data Grid project (EU Data Grid) [6]. The OptorSim simulator runs on a set of predetermined hardware specifications as follows:

- Intel Core i5 2.5GHz
- 8 GB Ram
- Java JDK 1.7

This experiment used 3 Network Topology testbeds for performance comparison, namely, EU Data Grid Testbed [2] [16], GridPP Testbed[11], and CMS Testbed[2]. Performance measurement is carried out by

- Means job execution time [12][13][14],
- Effective network usage (ENU) [12][13][14],
- Number of file found in Local File Access [12][13][14], and
- Bandwidth [12][13][14].

Each experiment used Optimistic replication method:

- No Replication
- Least Recently Used (LRU) – always replicates by deleting least recently created file

- Least Frequently Used (LFU) - always replicates by deleting least frequently accessed file

and the access patterns:

- Random Access[12, 23-25] : Files were accessed via uniform random distribution.

- Random Gaussian random walk[12, 23-25]: Files were accessed via Gaussian random walk. The starting file was chosen by uniform random distribution.

- Random with Zipf [12, 23-25, 29, 30]: Files were accessed via Zipf distribution.

4.1 1 OptorSim

OptorSim is a simulation tool for simulating grid network topology. The jobs are submitted to a grid testbed consisting of a number of sites, each of which may provide computational and data storage resources. Each site consists of zero or more Computing Elements (CE) and zero or more Storage Elements (SE). CEs run jobs which require data from the files stored on SEs. A Resource Broker controls scheduling of the jobs. Sties without SE or CE act as network nodes or routers. Figure 4.1 shows user interface of OptorSim.

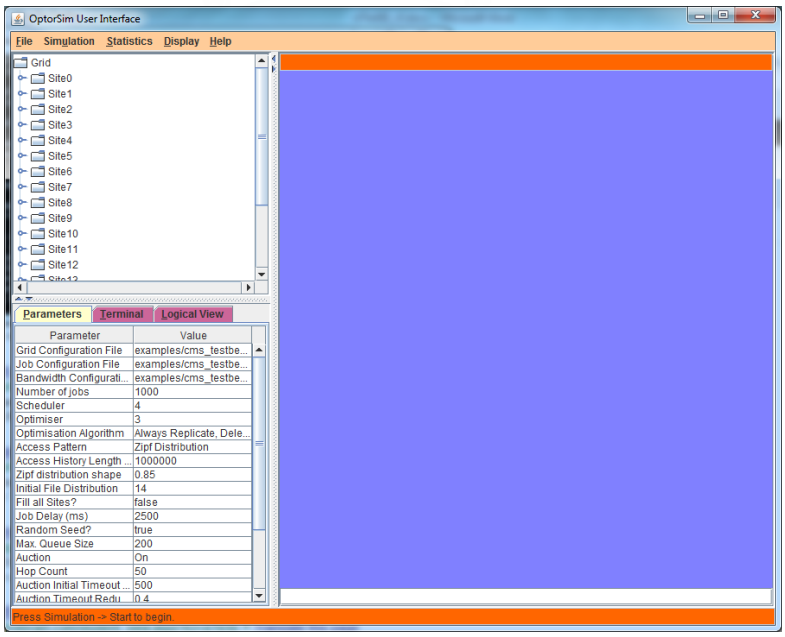


Figure 4.1 OptorSim User Interface

4.1.2 Grid Configuration

There are several configuration files used to control various inputs to OptorSim. A network topology configuration file describes links between different sites, the available network bandwidths, and size of disk storage on each site. The job configuration file describes jobs and the site policies. The simulation parameter file contains various simulation parameters to control the number of tests, network topology, access patterns, and data placement methods. The Grid Configuration file describes the status of resources of each site and layout of the grid being simulated. Table 4.1 depicts all parameters associating with a network configuration.

Table 4.1 A Sample network configuration based on 10 sites two of which have CEs

		No of CEs	No of SEs	SE Sizes	Site vs Site bandwidth									
Site and Site Bandwidth					0	1	2	3	4	5	6	7	7	9
	0	0	1	100	000	000	100	000	000	000	000	000	000	000
	1	0	1	100	000	100	000	000	000	000	000	000	000	000
	2	5	1	100	100	100	000	100	100	000	000	000	000	000
	3	0	1	100	000	000	100	000	000	000	100	000	000	000
	4	0	1	100	000	000	100	000	000	000	100	000	000	000
	5	0	1	100	000	000	000	000	000	000	000	100	000	000
	6	0	1	100	000	000	000	000	000	000	000	100	000	000
	7	2	1	100	000	000	000	000	000	100	100	000	100	100
	8	0	1	100	000	000	000	000	000	000	000	100	000	000
	9	0	1	000	000	000	000	000	000	000	000	100	000	000

The information is transferred into the simulation configuration file as shown in Figure 4.2. Each row holds the information of one site.

```
# A sample network configuration based on 10 sites, two of which have CEs.

# no of CEs, no of SEs, SE sizes, site vs site bandwidth

#

0 1 100   000.   000.   100.   000.   000.   000.   000.   000.   000.   000.

0 1 100   000.   000.   100.   000.   000.   000.   000.   000.   000.   000.
```

```
5 1 100   100.   100.   000.   100.   100.   000.   000.   000.   000.   000.
0 1 100   000.   000.   100.   000.   000.   000.   100.   000.   000.   000.
0 1 100   000.   000.   100.   000.   000.   000.   000.   000.   000.   000.
0 1 100   000.   000.   000.   000.   000.   000.   000.   100.   000.   000.
0 1 100   000.   000.   000.   100.   000.   000.   000.   100.   000.   000.
2 1 100   000.   000.   000.   000.   000.   100.   100.   000.   100.   100.
0 1 100   000.   000.   000.   000.   000.   000.   000.   100.   000.   000.
0 1 100   000.   000.   000.   000.   000.   000.   000.   100.   000.   000.
```

Figure 4.2 Sample Grid Configuration file

The first column denotes the number of CEs. The second column denotes the number of SEs. The third column denotes the size of SE in Megabyte (MB). The rest of the columns denote a site and site matrix giving the maximum bandwidth or link capacity between each site in Mb/s. The matrix is diagonally symmetric. Figure 4.4 shows the network diagram of the above configuration file.

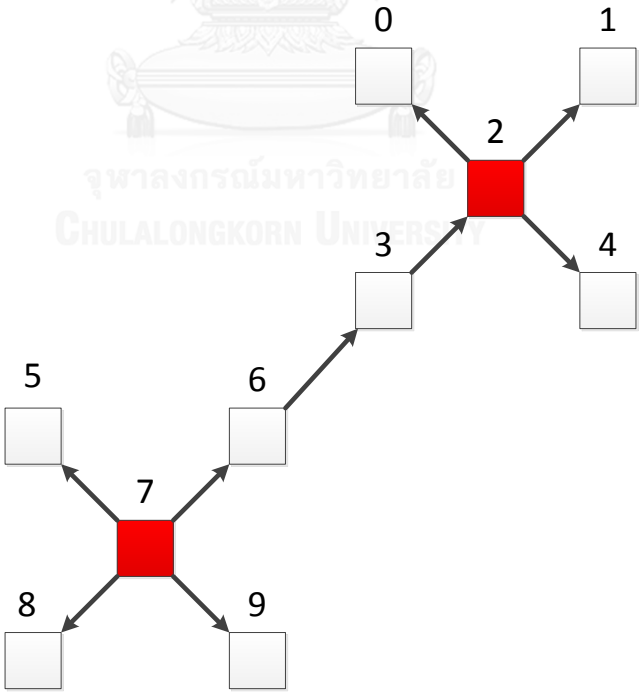


Figure 4.3 Network diagram from sample configuration file

The simulation parameter file contains several simulation parameters which can be set for grid configuration file, job configuration file, bandwidth configuration file, number of jobs, and access patterns.

Configuration	Description
grid.configuration.file = examples/cms_testbed_grid.conf job.configuration.file = examples/cms_testbed_jobs_grass.conf	Setting grid configuration file path and job configuration file path
number.jobs = 1000	Number of requests for simulation run
optimiser = 3	The choices of optimisers are: 1. SimpleOptimiser - no replication. 2. LruOptimiser - always replicates, deleting least recently created file. 3. LfuOptimiser - always replicates, deleting least frequently accessed file.
access.pattern.generator = 1	The choices of access pattern generators are: 1. RandomAccessGenerator - Files are accessed using a flat random distribution. 2. RandomWalkGaussianAccessGenerator - Files are accessed using a Gaussian random walk. 3. RandomZipfAccessGenerator - Files are accessed using a Zipf distribution

Figure 4.4 Sample Properties Files

4.1.3. Network Topology Testbeds

All the testbeds used in the experiment are described in detail as follows.

A. EU Data Grid.

Figure 4.6 EU Data Grid Testbed sites and their associated network topology. The numbers indicate bandwidth between the two ending sites in Mbit/s(M) or Gbit/s(G). Gray Boxes denote routers and White Box nodes denote replica sites.

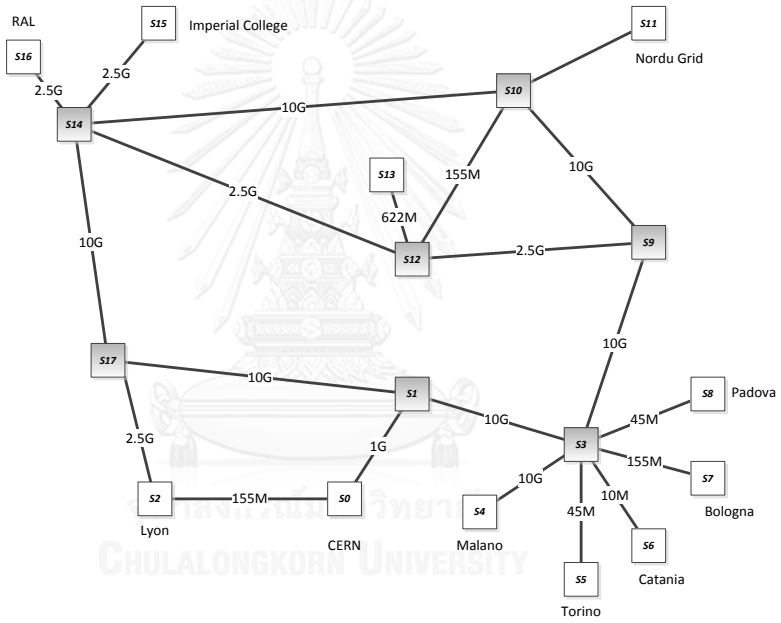


Figure 4.5 EU Data Grid Testbed sites and their associated network topology

```
# The network configuration for the EU Data grid testbed of 1 site
# no of CEs, no of SEs, SE sizes, site vs site bandwidth
#
1 1 80000 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 02500.0 00000.0 10000.0 00000.0 02500.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 02500.0 00000.0
0 0000000 00000.0 10000.0 00000.0 00000.0 01550.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 01550.0
1 1 33000 00000.0 00000.0 00000.0 00000.0 00622.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 02500.0 01550.0 00622.0 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 00000.0 10000.0 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 02500.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 02500.0 00000.0 01550.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 1 100000000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 01550.0 00000.0 01000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 10000.0 00000.0 01000.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 00000.0 00000.0 00000.0 00000.0 10000.0 00000.0 00000.0 00000.0 10000.0 00000.0 00450.0 01550.0 00100.0 00450.0 00100.0 00000.0 00000.0
1 1 63000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00450.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
```

```
1 1 30000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 01550.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 30000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00100.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00450.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00100.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 70000 00000.0 00000.0 01550.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
```

Figure 4.6 Network configuration of EU Data Grid

B. GridPP 2004 testbed

GridPP 2004 testbed [11] is a collaboration of particle physicist and computer having 17 grid sites located in UK and one in CERN of Switzerland. Each UK site has 40 and 1800 processing nodes with storage capacity between 5TB and 500TB. The CERN site has 1000TB of storage capacity holding all master files as shown in Figure 4.7.

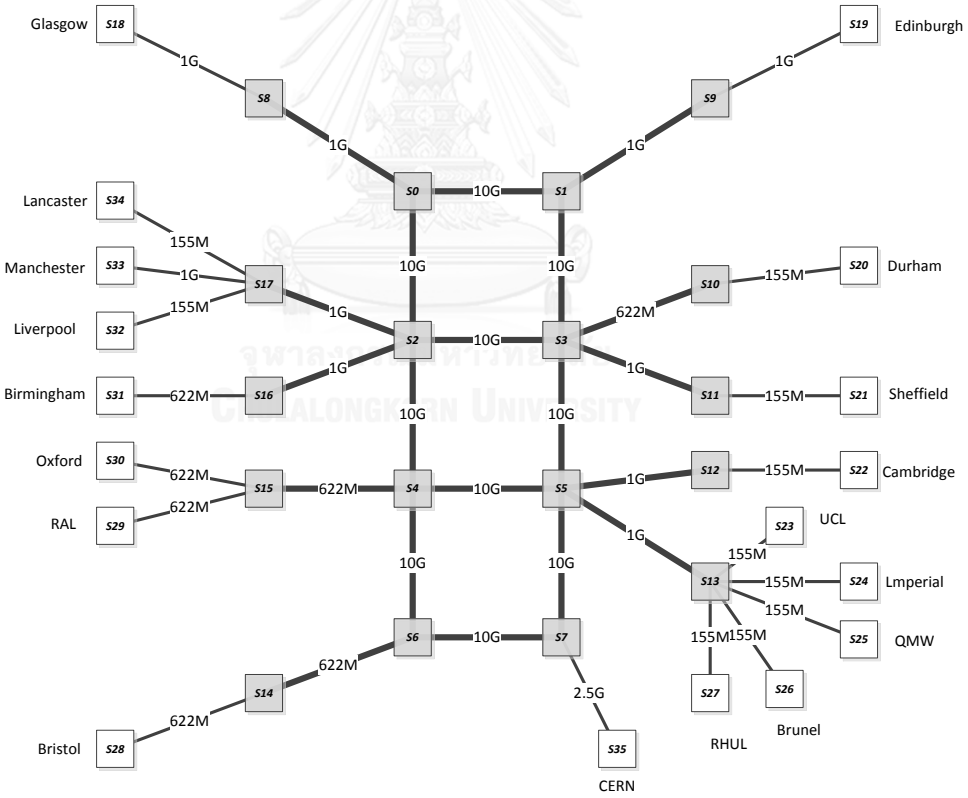


Figure 4.7 GridPP resources and topology in 2004

```
# The network configuration for the EU Data grid testbed of 1 site
# no of CEs, no of SEs, SE sizes, site vs site bandwidth
0 0 0. 10000. 10000. 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 10000. 0. 0. 10000. 0. 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 10000. 0. 0. 10000. 10000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1000. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 10000. 10000. 0. 0. 10000. 0. 0. 0. 0. 622. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 10000. 0. 0. 10000. 10000. 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 10000. 10000. 0. 0. 10000. 0. 0. 0. 0. 1000. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 0. 10000. 0. 0. 10000. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 0. 0. 10000. 10000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 2500.
0 0 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 155. 155. 155. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 622. 622. 0. 0. 0. 0. 0.
0 0 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0.
0 0 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 1000. 155. 0.
200 1 100000 0. 0. 0. 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
40 1 640000 0. 0. 0. 0. 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
78 1 53000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
365 1 136000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
40 1 33000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
150 1 150000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
1100 1 400000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
220 1 200000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
306 1 280000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
120 1 60000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
300 1 200000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
972 1 573000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
412 1 380000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
150 1 90000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 622. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
1890 1 163000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
225 1 100000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1000. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
320 1 640000 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 155. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0 1 10000000 0. 0. 0. 0. 0. 0. 0. 2500. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

Figure 4.8 GridPP 2004 Network configurations

C. CMS Testbed

Compact Muon Solenoid [2] is a General Purpose Detectors for the Large Hadron Collider (LHC) at CERN, Geneva, Switzerland. CMS Testbed has 20 sites in Europe and the USA. CERN and FNAL have storage capacity of 100 GB each and a set of master files stored at one of these two sites. Other sites have CEs having 50 GB capacity each as shown in Figure 4.9

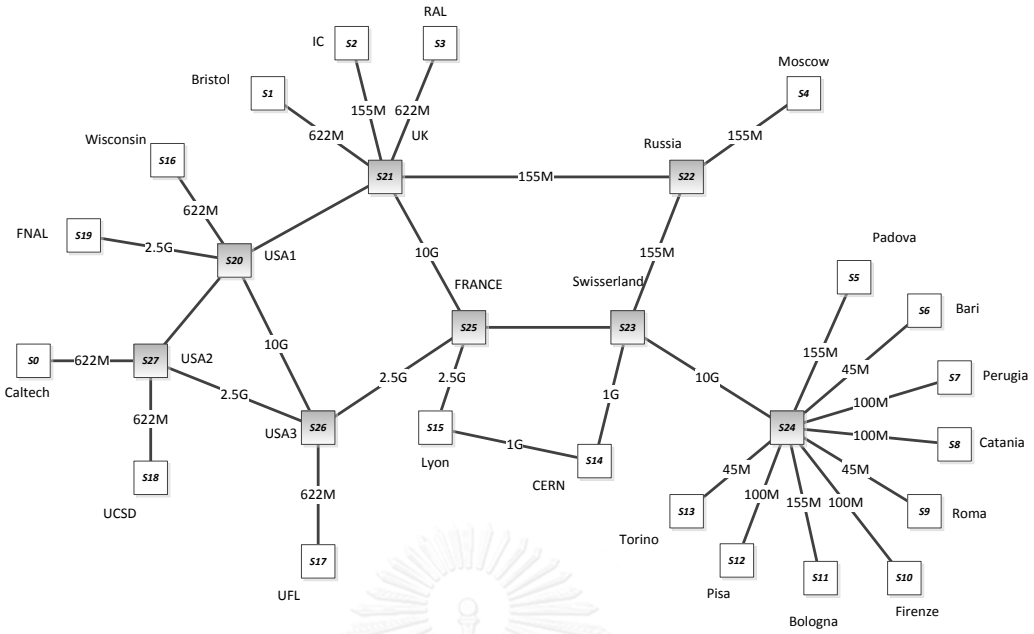


Figure 4.9 CMS 2004 network Topology

```
# The network configuration for the EU Data grid testbed of 1 site
# no of CEs, no of SEs, SE sizes, site vs site bandwidth
#
#       Imp Coll   UK    Swed   NIKHEF   Holl    Germ    Fran    Lyon    CERN    Swis   Italy   Padova  Bolog   Catan   Torino  Milano    RAL   Nordu
1 1 80000 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 02500.0 00000.0 10000.0 00000.0 02500.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 02500.0 00000.0
0 0000000 00000.0 10000.0 00000.0 00000.0 01550.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 01550.0
1 1 33000 00000.0 00000.0 00000.0 00000.0 00622.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 02500.0 01550.0 00622.0 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 00000.0 10000.0 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 02500.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 02500.0 00000.0 01550.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 1 100000000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 01550.0 00000.0 01000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 10000.0 00000.0 01000.0 00000.0 10000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
0 0000000 00000.0 00000.0 00000.0 00000.0 00000.0 10000.0 00000.0 00000.0 00000.0 10000.0 00000.0 00450.0 01550.0 00100.0 00450.0 00100.0 00000.0 00000.0 00000.0
1 1 63000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00450.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 30000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 01550.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 30000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00100.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00450.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00100.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 50000 00000.0 02500.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
1 1 70000 00000.0 00000.0 01550.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0 00000.0
```

Figure 4.10 CMS Network Configuration File

4.2. Simulation Results and Discussion

The following results were obtained from simulation runs using the above set up on all three testbeds. Statistics on mean job time, number of file found in local file

access, effective network usage, number of hops, bandwidth and time were collected and shown below.

4.2.1 EU Data Grid

The results of EU Data Grid Testbed after run OptorSim Simulation by using optimizer with Least Recently Used (LRU) and Normal Random access pattern are shown in Table 4.2, comparing the value of mean job time in Grass, Flooding, Centralized, Multi-master and Random Walk algorithms. Figure 4.11 shows the corresponding plot.

Table 4.2 Mean Job Time (in seconds) of EU Data Grid Testbed from OptorSim Simulation with LRU and Normal Random access pattern

Time(second)	5	10	15	20	25	30
Grass	21,000	39,000	54,000	65,000	73,000	81,000
Flooding	20,000	39,000	58,000	65,000	72,000	81,000
Centralized	24,000	49,000	75,000	95,000	120,000	145,000
Multi-Master	25,000	50,000	70,000	90,000	120,000	145,000
Random Walk	24,000	50,000	75,000	95,000	120,000	135,000

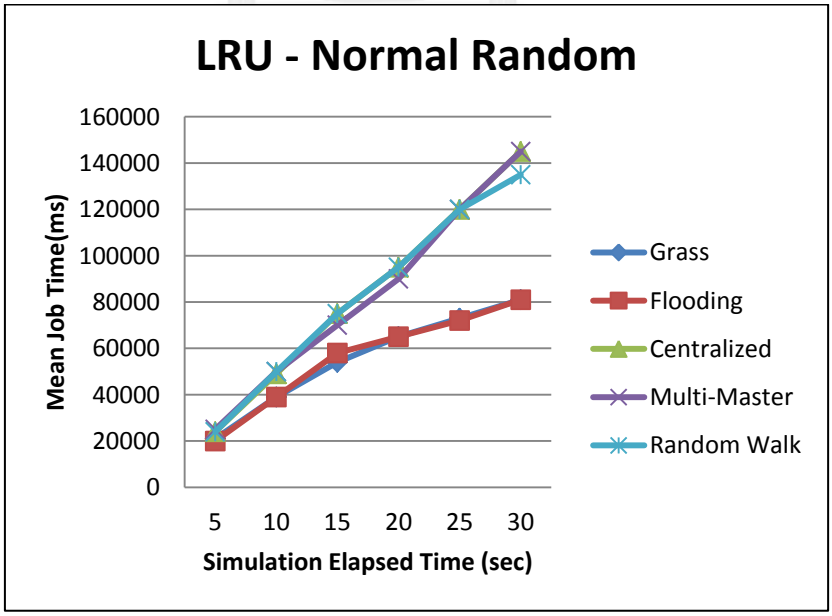


Figure 4.11 Mean Job Time of EU Data Grid Testbed plot

Note that Grass and Flooding algorithms are equally lower than other comparative algorithms, yielding faster replication.

Table 4.3 No. of file found in Local File Access of EU Data Grid Testbed from OptorSim Simulation with LRU and Normal Random access pattern

Time(second)	5	10	15	20	25	30
Grass	2,900	5,800	8,000	9,800	10,500	11,200
Flooding	3,000	5,900	8,100	9,900	11,200	12,200
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,500	3,000	4,500	6,000	7,300	9,000

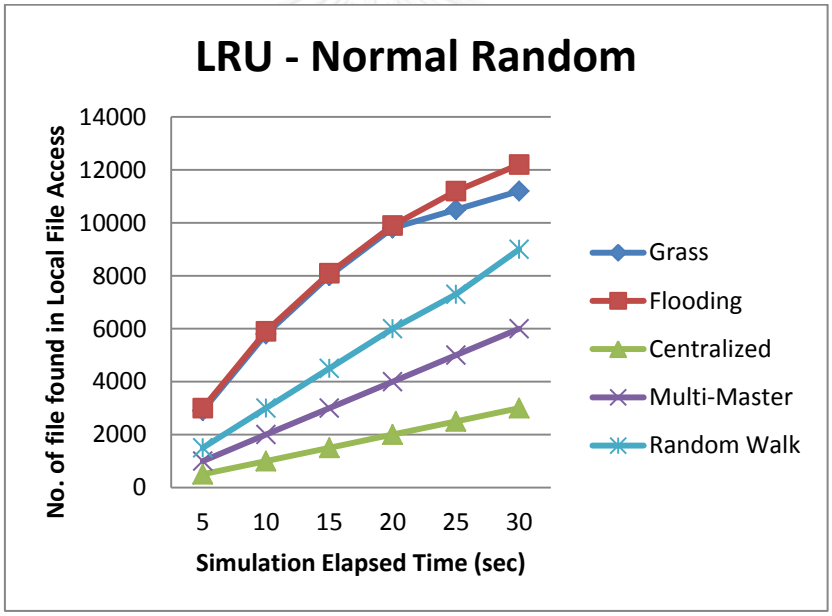


Figure 4.12 No. of file found in Local File Access of EU Data Grid Testbed plot

Table 4.3 and Figure 4.12 show the number of local file accesses of Grass and Flooding algorithms that are equally greater than other comparative algorithms, implying higher replication yields.

Table 4.4 Effective Network Usage (in seconds) of EU Data Grid Testbed from OptorSim Simulation with LRU and Normal Random access pattern

Time(second)	5	10	15	20	25	30
Grass	0.29	0.26	0.25	0.25	0.26	0.26
Flooding	0.29	0.25	0.24	0.24	0.25	0.26
Centralized	0.41	0.41	0.40	0.41	0.41	0.41
Multi-Master	0.53	0.50	0.50	0.49	0.49	0.49
Random Walk	0.40	0.38	0.35	0.34	0.34	0.34

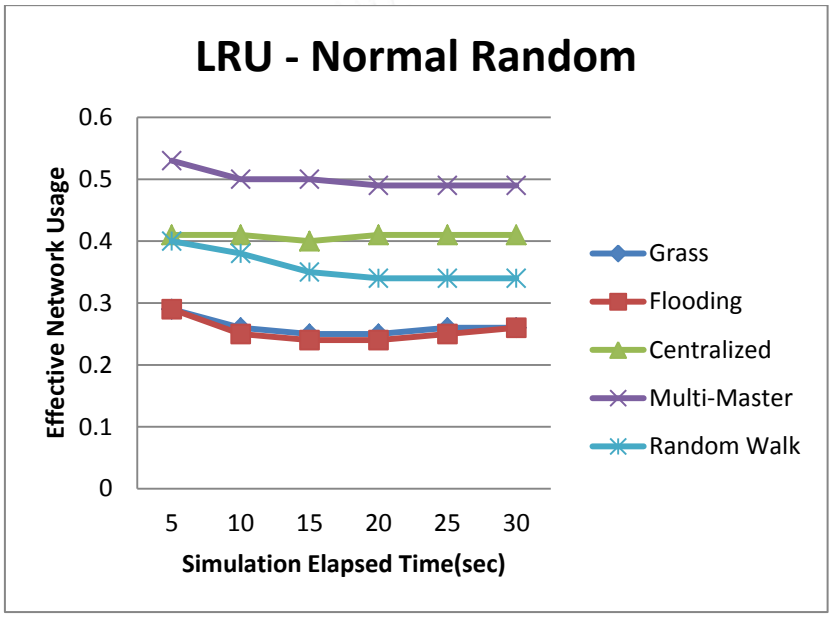


Figure 4.13 Effective Network Usage of EU Data Grid Testbed plot

Table 4.4 and Figure 4.13 show effective network usage of Grass and Flooding algorithms that are equally less than other comparative algorithms, implying less network usage for replication.

Table 4.5 Number of hops to replicate data in EU Data Grid Testbed

Algorithm	Number of hop
Grass	13
Flooding	18

Centralized	1
Multi-Master	6
Random Walk	9

The replication took somewhat higher number of hops for both Grass and Flooding algorithms due to the propagation to reach all the nodes. This measure will be subsequently considered along other measures to convey the overall performance of each algorithm.

The bandwidth and time used to placing data on each node for each algorithm are shown below.

Table 4.6 Bandwidth and Time Usage of Grass Algorithm in EU Data Grid Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S0	1000	10.00
S0	S2	155	64.52
S2	S17	2500	4.00
S17	S14	10000	1.00
S17	S1	10000	1.00
S0	S1	1000	10.00
S1	S3	1000	10.00
S3	S9	10000	1.00
S3	S8	45	222.22
S3	S7	155	64.52
S3	S6	10	1000.00
S3	S5	45	222.22
S3	S4	10000	1.00
Total Time			1601.48

Table 4.7 Bandwidth and Time Usage of Flooding Algorithm in EU Data Grid Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S0	1000	10.00
S0	S1	1000	10.00
S0	S2	155	64.52
S2	S17	2500	4.00
S1	S3	10000	1.00
S17	S14	10000	1.00
S3	S8	45	222.22
S3	S7	155	64.52
S3	S6	10	1000.00
S3	S5	45	222.22
S3	S4	10000	1.00
S3	S9	10000	1.00
S14	S16	2500	4.00
S14	S15	2500	4.00
S14	S12	2500	4.00
S12	S13	622	16.08
S9	S10	10000	1.00
S10	S11	155	64.52
Total Time			1685.07

Table 4.8 Bandwidth and Time Usage of Centralized Algorithm in EU Data Grid

From	To	Bandwidth (Mbps)	Time(s)
source	S0	1000	10.00
Total Time			10.00

Table 4.9 Bandwidth and Time Usage of Multi-Master Algorithm in EU Data Grid

From	To	Bandwidth (Mbps)	Time(s)
source	S0	1000	10.00
S0	S1	1000	10.00
S1	S3	10000	1.00
S3	S9	10000	1.00
S9	S10	10000	1.00
S10	S11	155	64.52
Total Time			77.52

Table 4.10 Bandwidth and Time Usage of Random Walk Algorithm in EU Data Grid Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S0	1000	10.00
S0	S1	1000	10.00
S1	S3	10000	1.00
S3	S8	45	222.22
S3	S9	10000	1.00
S9	S12	2500	4.00
S12	S13	622	16.08
S12	S10	155	64.52
S10	S11	155	64.52
Total Time			383.33

Note that the bandwidth between nodes and time usage measures assumed that 10GB data were replicated from the source node in Centralized setup and in Multi-Master sources before the simulation run. The results in Table 4.6, 4.7, 4.8, 4.9 and 4.10 show that the centralized algorithm uses the least time for replication, while Flooding algorithm performs the worst.

4.2.2 GridPP

Below are the results from GridPP testbed by using optimizer with Least Recently Used (LRU) and Normal Random access pattern. Statistics on mean job time, number of file found in local file access, effective network usage, number of hops, bandwidth and time were collected and shown below.

Table 4.11 Mean Job Time (in seconds) of GridPP Testbed from OptorSim Simulation with LRU and Normal Random access pattern

Time (second)	5	10	15	20	25	30
Grass	2,000	3,000	5,000	7,000	8,000	10,000
Flooding	20	100	200	300	400	500
Centralized	2,500	3,000	6,000	7,500	9,000	11,000
Multi-Master	2,000	4,000	5,000	7,500	10,000	13,000
Random Walk	1,500	2,600	4,500	6,500	8,000	10,000

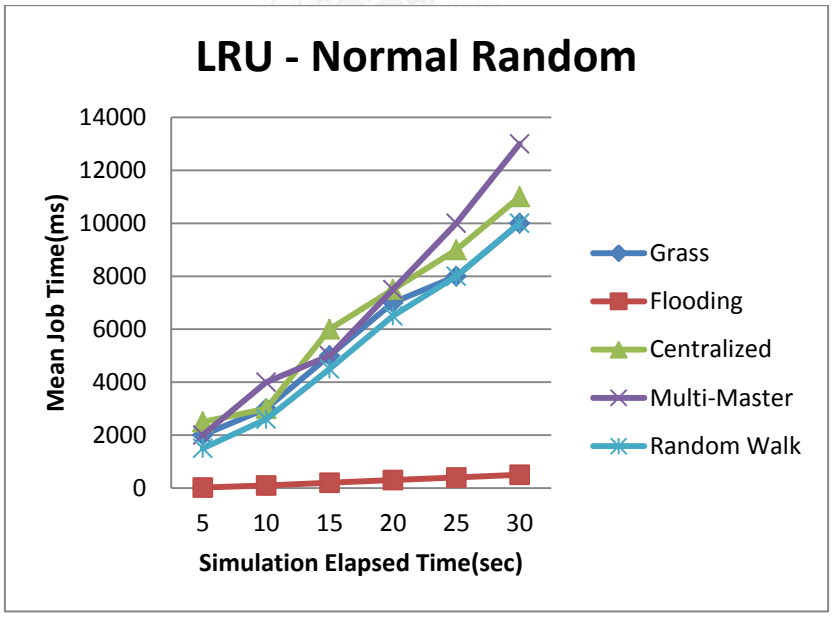


Figure 4.14 Mean Job Time of GridPP Testbed plot

Table 4.11 and Figure 4.14 show the mean job time of Flooding algorithm that is lower than other comparative algorithms, yielding fastest replication. Notice that Grass and Random Walk algorithms perform slightly better than Centralized and Multi-Master algorithms.

Table 4.12 No. of file found in Local File Access (in seconds) of GridPP Testbed from OptorSim Simulation with LRU and Normal Random access pattern

Time(second)	5	10	15	20	25	30
Grass	1,250	2,500	4,000	6,000	7,000	8,000
Flooding	2,500	6,000	11,000	16,000	20,000	24,000
Centralized	1,000	1,800	2,800	3,500	4,500	5,500
Multi-Master	1,000	2,000	3,000	4,000	6,000	6,500
Random Walk	1,500	2,000	4,000	6,500	8,000	10,000

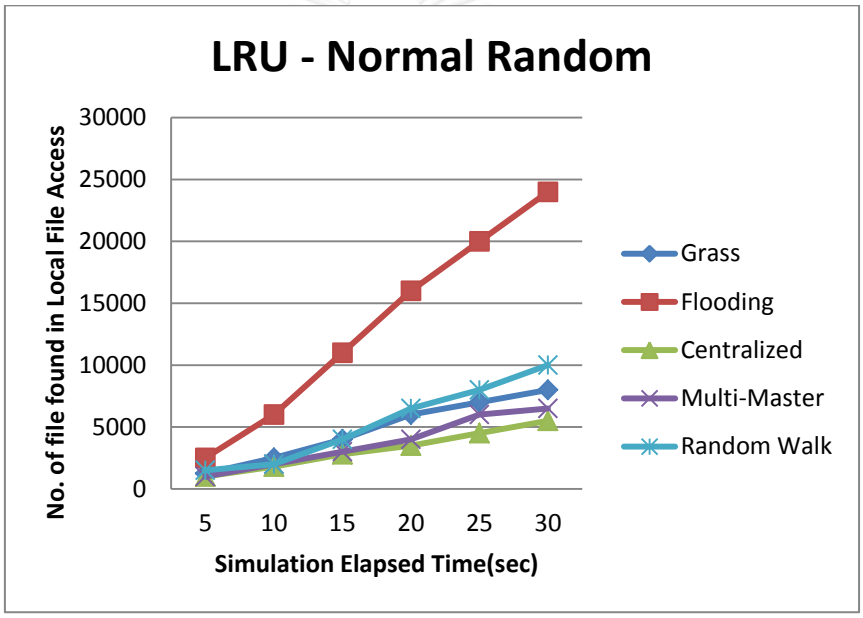


Figure 4.15 No. of file found in Local File Access of GridPP Testbed plot

Table 4.12 and Figure 4.15 show the number of local file accesses of Flooding algorithm that is greater than other comparative algorithms, implying the highest replication yield. Grass and Random Walk also perform slightly better than Centralized and Multi-Master algorithms.

Table 4.13 Effective Network Usage (in seconds) of GridPP Testbed from OptorSim Simulation with LRU and Normal Random access pattern

Time(second)	5	10	15	20	25	30
Grass	0.64	0.6	0.56	0.53	0.53	0.53
Flooding	0.57	0.51	0.48	0.44	0.43	0.43
Centralized	0.62	0.62	0.62	0.62	0.62	0.63
Multi-Master	0.62	0.61	0.61	0.61	0.61	0.60
Random Walk	0.63	0.58	0.57	0.56	0.54	0.54

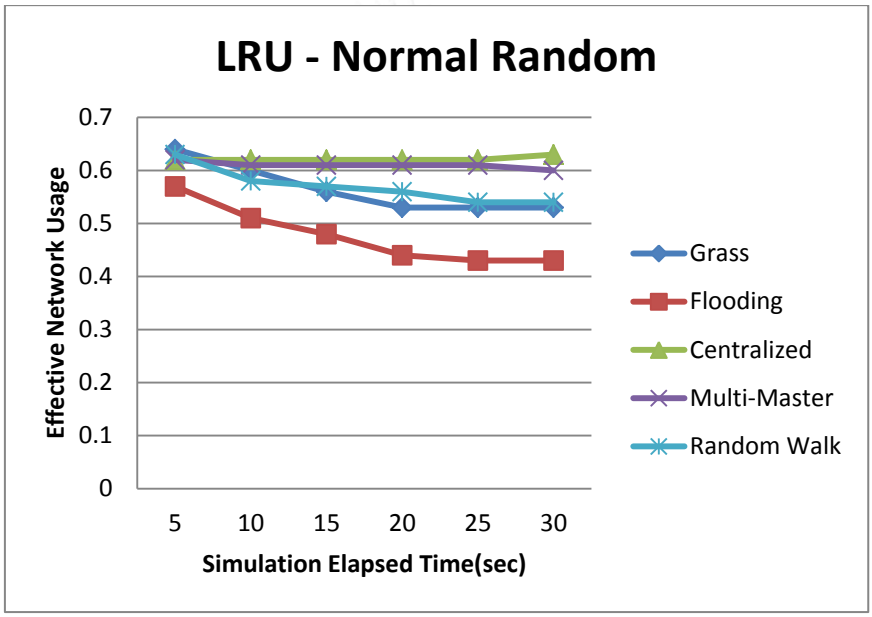


Figure 4.16 Effective Network Usage of GridPP Testbed plot

Table 4.13 and Figure 4.16 show effective network usage of Flooding algorithm that is less than other comparative algorithms, implying the least network usage for replication. Grass and Random Walk algorithms remain in the moderate range.

Table 4.14 Number of hops to replicate data in GridPP Testbed

Algorithm	Number of hop
Grass	10
Flooding	36
Centralized	1

Multi-Master	8
Random Walk	7

The number of hops for Grass algorithm remains competitive with other comparative algorithms with the exception of Flooding and Centralized algorithms.

Table 4.15 Bandwidth and Time Usage of Grass Algorithm in GridPP Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S27	1000	10.00
S27	S13	155	64.52
S13	S26	155	64.52
S13	S25	155	64.52
S13	S24	155	64.52
S13	S23	155	64.52
S13	S5	1000	10.00
S5	S7	10000	1.00
S5	S4	10000	1.00
S5	S3	10000	1.00
Total Time			335.58

Table 4.16 Bandwidth and Time Usage of Flooding Algorithm in GridPP Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S27	1000	10.00
S27	S13	155	64.52
S13	S5	1000	10.00
S13	S23	155	64.52
S13	S24	155	64.52
S13	S25	155	64.52
S13	S26	155	64.52
S5	S7	10000	1.00
S5	S4	10000	1.00

S5	S3	10000	1.00
S5	S12	1000	10.00
S7	S35	2500	4.00
S7	S6	10000	1.00
S4	S15	622	16.08
S4	S2	10000	1.00
S3	S1	10000	1.00
S3	S10	622	16.08
S3	S11	1000	10.00
S12	S22	155	64.52
S6	S14	622	16.08
S15	S29	622	16.08
S15	S30	622	16.08
S2	S16	1000	10.00
S2	S17	1000	10.00
S2	S0	10000	1.00
S1	S9	1000	10.00
S10	S20	155	64.52
S11	S21	155	64.52
S14	S28	622	16.08
S16	S31	622	16.08
S17	S32	155	64.52
S17	S33	1000	10.00
S17	S34	155	64.52
S0	S8	1000	10.00
S9	S19	1000	10.00
S8	S18	1000	10.00
Total Time			868.70

Table 4.17 Bandwidth and Time Usage of Centralized Algorithm in GridPP Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S27	1000	10.00
Total Time			10.00

Table 4.18 Bandwidth and Time Usage of Multi-Master in GridPP Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S27	1000	10.00
S27	S13	155	64.52
S13	S5	1000	10.00
S5	S3	10000	1.00
S3	S1	10000	1.00
S1	S0	10000	1.00
S0	S8	1000	10.00
S8	S18	1000	10.00
Total Time			97.52

Table 4.19 Bandwidth and Time Usage of Random Walk Algorithm in GridPP Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S27	1000	10.00
S27	S13	155	64.52
S13	S5	1000	10.00
S5	S7	10000	1.00
S7	S6	10000	1.00
S6	S14	622	16.08
S14	S28	622	16.08
Total Time			108.67

Table 4.15, 4.16, 4.17, 4.18 and 4.19 show bandwidth between nodes and time usage measures undergo the same assumption as that of the EU data grid. Flooding performs the worst and Centralized is the best.

4.2.3 CMS

Below are the results from CMS testbed by using optimizer with Least Recently Used (LRU) and Normal Random access pattern. Statistics on mean job time, number of file found in local file access, effective network usage, number of hops, bandwidth and time were collected and shown below.

Table 4.20 Mean Job Time (in seconds) of CMS Testbed from OptorSim Simulation with LRU and Normal Radom access pattern

Time(second)	10	20	30	40	50	60
Grass	500	1,000	2,000	2,300	3,200	3,400
Flooding	400	650	1,100	1,550	1,800	1,900
Centralized	500	1,000	2,000	2,900	3,400	4,000
Multi-Master	500	1,000	1,200	1,900	2,800	3,100
Random Walk	500	1,000	2,000	2,750	3,300	3,800

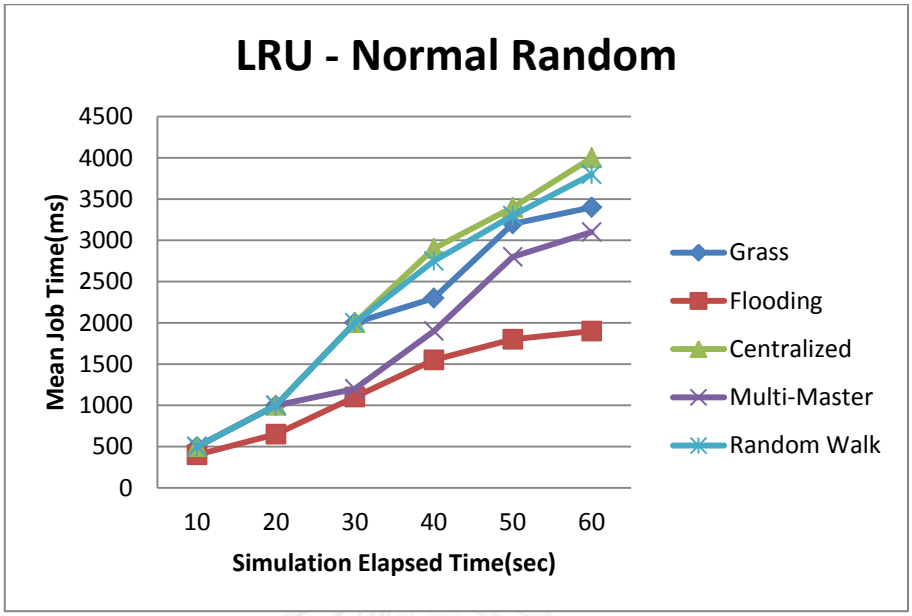


Figure 4.17 Mean Job Time of CMS Testbed plot

Table 4.20 and Figure 4.17 show the mean job time of Flooding, Grass, and Random Walk algorithms that are equally lower than other comparative algorithms, yielding fastest replication.

Table 4.21 No. of file found in Local File Access (in seconds) of CMS Testbed from OptorSim Simulation with LRU and Normal Radom access pattern

Time(second)	10	20	30	40	50	60
Grass	500	1,000	3,500	4,500	6,500	7,200
Flooding	1,100	2,800	7,900	10,100	11,800	11,900
Centralized	500	700	1,000	1,200	1,800	2,000
Multi-Master	500	1,100	3,800	5,500	6,000	6,600
Random Walk	700	1,800	3,500	5,200	7,500	9,000

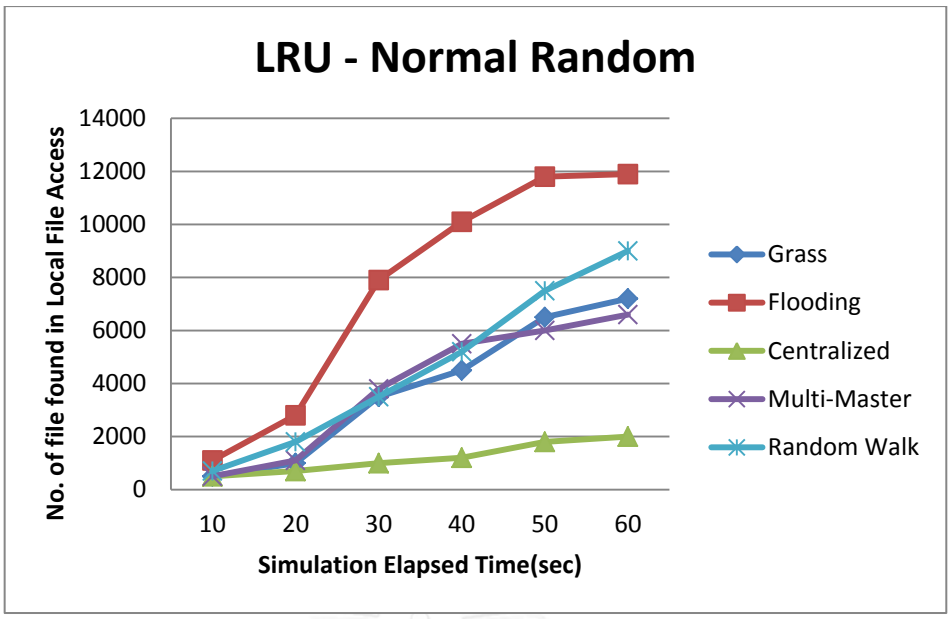


Figure 4.18 Number of file found in Local File Access of CMS Testbed plot

Table 4.21 and Figure 4.18 show the number of local file accesses of Grass, Flooding and Random Walk equally greater than other comparative algorithms as the number of simulation runs increase, implying the highest replication yield.

Table 4.22 Effective Network Usage (in seconds) of CMS Testbed from OptorSim Simulation with LRU and Normal Radom access pattern

Time(second)	10	20	30	40	50	60
Grass	0.75	0.30	0.23	0.17	0.17	0.18
Flooding	0.60	0.30	0.23	0.17	0.20	0.23
Centralized	0.90	0.35	0.33	0.34	0.35	0.35
Multi-Master	0.75	0.23	0.15	0.16	0.17	0.18
Random Walk	1.00	0.40	0.25	0.23	0.20	0.20

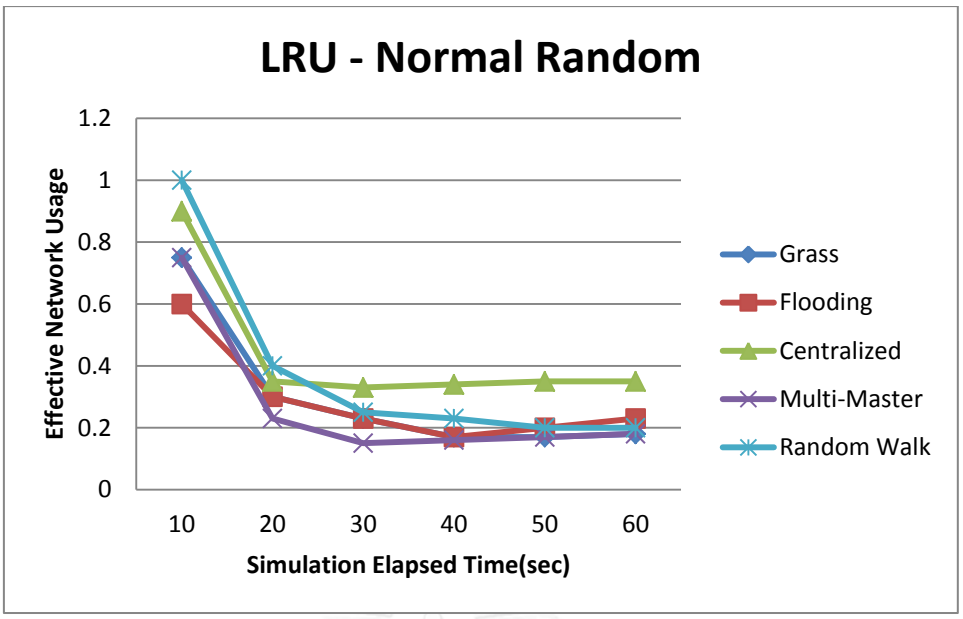


Figure 4.19 Effective Network Usage of CMS Testbed plot

Table 4.22 and Figure 4.19 show effective network usage of Grass algorithm that is less than other comparative algorithms, implying the least network usage for replication. Random Walk and Multi-Master algorithms remain close to Grass.

Table 4.23 Number of hops to replicate data in CMS Testbed

Algorithm	Number of hop
Grass	18
Flooding	28
Centralized	1
Multi-Master	5
Random Walk	8

Note that both Grass and Flooding algorithms expend considerably high hops to replicate the data in CMS Testbed.

Table 4.24 Bandwidth and Time Usage of Grass Algorithm in CMS Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S14	1000	10.00
S14	S15	1000	10.00

S15	S25	2500	4.00
S25	S26	2500	4.00
S25	S21	10000	1.00
S25	S23	10000	1.00
S23	S22	155	64.52
S22	S4	155	64.52
S23	S24	10000	1.00
S24	S5	155	64.52
S24	S6	45	222.22
S24	S7	100	100.00
S24	S8	100	100.00
S24	S9	45	222.22
S24	S10	100	100.00
S24	S11	155	64.52
S24	S12	100	100.00
S24	s13	45	222.22
Total Time			1345.73

Table 4.25 Bandwidth and Time Usage of Flooding Algorithm in CMS Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S14	1000	10.00
S14	S15	1000	10.00
S14	S23	1000	10.00
S15	S25	2500	4.00
S23	S24	10000	1.00
S25	S26	2500	4.00
S25	S21	10000	1.00
S23	S22	155	64.52
S22	S4	155	64.52

S24	S6	155	64.52
S24	S7	45	222.22
S24	S8	100	100.00
S24	S9	100	100.00
S24	S10	45	222.22
S24	S11	100	100.00
S24	S12	155	64.52
S24	S13	100	100.00
S22	S4	45	222.22
S21	S1	622	16.08
S21	S2	155	64.52
S21	S3	622	16.08
S26	S20	10000	1.00
S20	S17	10000	1.00
S20	S16	622	16.08
S20	S19	2500	4.00
S20	S27	10000	1.00
S27	S0	622	16.08
S27	S18	622	16.08
Total Time			1506.63

Table 4.26 Bandwidth and Time Usage of Centralized Algorithm in CMS Testbed

From	To	Bandwidth (Mbps)	Time(s)
source	S14	1000	10.00
Total Time			10.00

Table 4.27 Bandwidth and Time Usage of Multi-Master in CMS Testbed

From	To	Bandwidth (Mbps)	Time(s)
	S14	1000	10.00
S14	S15	1000	10.00

S15	S25	2500	4.00
S25	S26	2500	4.00
S26	S20	10000	1.00
S20	S19	2500	4.00
Total Time			23.00

Table 4.28 Bandwidth and Time Usage of Random Walk Algorithm in CMS Testbed

From	To	Bandwidth (Mbps)	Time(s)
	S14	1000	10.00
S14	S23	1000	10.00
S23	S22	155	64.52
S22	S4	155	64.52
S22	S21	155	64.52
S21	S3	622	16.08
S21	S2	155	64.52
S21	S14	622	16.08
Total Time			300.22

Table 4.24, 4.25, 4.26, 4.27 and 4.28 show bandwidth between nodes and time usage measures undergo the same assumption as that of the EU data grid. Flooding performs the worst and Centralized is the best.

4.2 Discussion

The results obtained from OptorSim for Mean Job Time, Local Fie Access, and Effective Network Usage on EU Data Grid, CMS, and GridPP Network Topology Testbeds show that Grass and Flooding are equally comparable and better than Centralized, Multi-Master, and Random Walk algorithms. However, in real situation, Flooding will occupy the entire network during the replication process that could cause temporarily traffic jam.

This makes it unattractive since no network service can afford such a suspension in normal operation.

In addition, the number of hops and bandwidth and time in replication are in favor of Grass over Flooding algorithm as shown in Figure 4.20 and Figure 4.21.

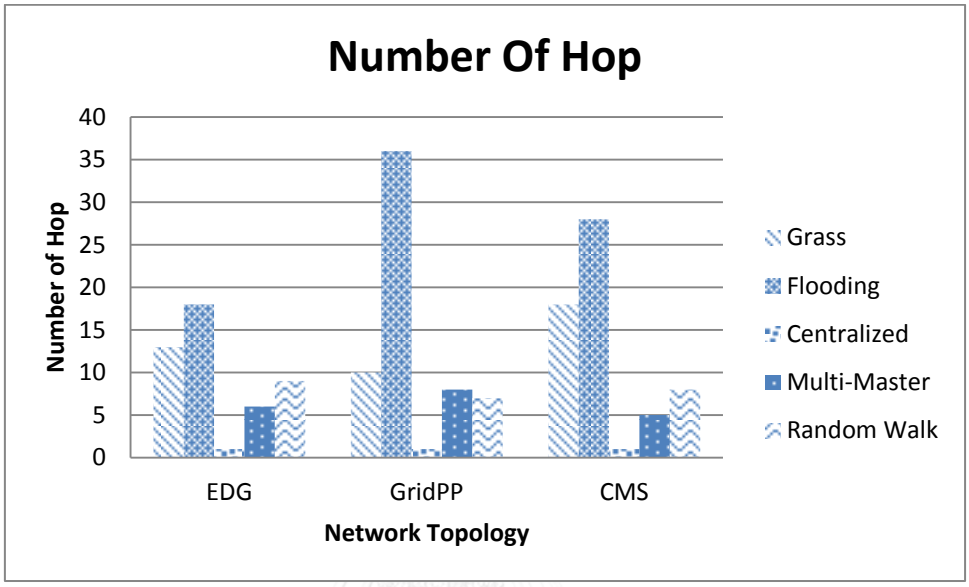


Figure 4.20 Number of hops for Grass, Flooding, Centralized, Multi-Master, and Random Walk on each network topology

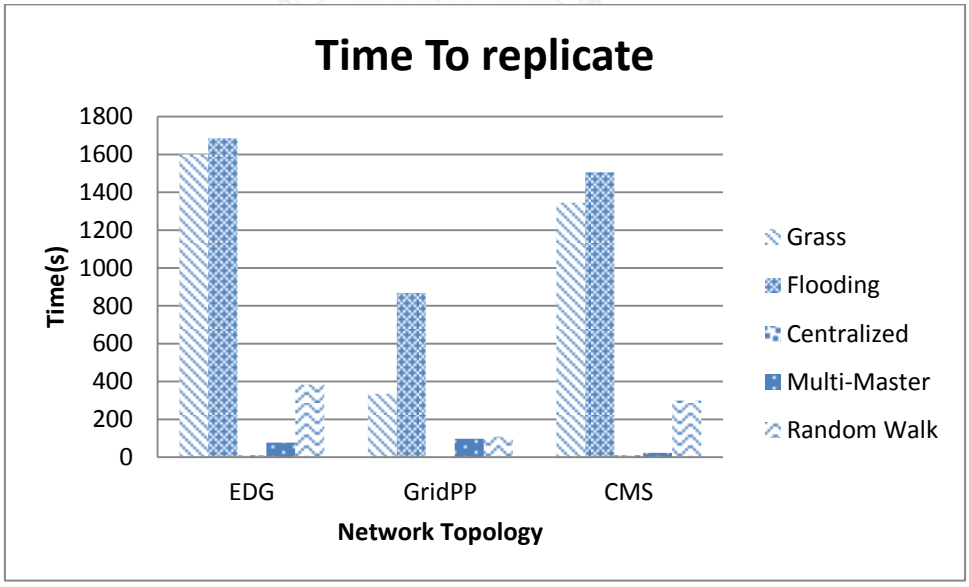


Figure 4.21 Time to Replicate for Grass, Flooding, Centralized, Multi-Master, and Random Walk on each network topology

CHAPTER 5
CONCLUSION AND FUTURE WORK

In this thesis, the Grass Growing Structure algorithm for replica positioning in a distributed environment is proposed based on natural grass growing process. The main idea rests on the pattern and amount of water irrigation that nourish grass growing. Thus, data replication process to the designated locations mimics the idea by introducing simple yet effective selection and replication of data over the network. Performance of the proposed algorithm is assessed in comparison with conventional Centralized Location, Multi-Master, Random-Walk, and Flooding algorithms. Evaluation was simulated using 3 standard benchmark testbeds, namely, EU Data Grid Network Topology, GridPP Network Topology, and CMS Network Topology. The results were satisfactory in terms of mean job time, effective network usage, number of file found in local file access, and bandwidth usage. The proposed Grass Growing Structure algorithm is better than Flooding algorithm in terms of bandwidth usage and other algorithms in terms of mean job time, effective network usage, and number of file found in local file access. Such contributions will entail a means to conserve the energy consumption in distributed processing as the information superhighway has proliferated in recent years.

Future improvement can be conducted in a real environment such as a local area network to gauge if the above performance measures are valid, where and how fine-tuning should be done. In addition, more distribution algorithms such as fragment transmission to reduce payload and enhance speed or delayed distribution to reduce traffic congestion could be incorporated. As such, energy consumption will be kept to minimal.

REFERENCES

1. Abramson, D., J. Giddy, and L. Kotler. *High performance parametric modeling with Nimrod/G: killer application for the global grid?* in *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*. 2000.
2. Holtman, K., *CMS Data Grid System Overview and Requirements.* 2001.
3. Childers, L., et al., *Access Grid: Immersive Group-to-Group Collaborative Visualization*. 2000.
4. Bierbaum, A., et al. *VR Juggler: a virtual platform for virtual reality application development*. in *Virtual Reality, 2001. Proceedings. IEEE*. 2001.
5. Brzezniak, M., T. Makiela, and N. Meyer. *Integration of NetSolve with Globus-based grids*. in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*. 2004.
6. *EU Data Mining Grid*. [cited 2013 2013,4th September]; Available from: http://www.datamininggrid.org.
7. Llorente, I.M., et al. *A Grid Infrastructure for Utility Computing*. in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE '06. 15th IEEE International Workshops on*. 2006.
8. Buyya, R. and S. Venugopal. *The Gridbus toolkit for service oriented grid and utility computing: an overview and status report*. in *Grid Economics and Business Models, 2004. GECON 2004. 1st IEEE International Workshop on*. 2004.
9. Venugopal, S., R. Buyya, and K. Ramamohanarao, *A taxonomy of Data Grids for distributed data sharing, management, and processing.* ACM Comput. Surv., 2006. **38**(1): p. 3.

10. Runqun, X., L. Junzhou, and S. Aibo. *An Effective Replica Location Algorithm Based on Routing-Forward in Data Grid*. in *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*. 2010.

11. Coles, J. *The evolving grid deployment and operations model within EGEE, LCG and GridPP*. in *e-Science and Grid Computing, 2005. First International Conference on*. 2005.

12. Cameron, D., et al., *Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using OptorSim.* Journal of Grid Computing, 2004. **2**(1): p. 57-69.

13. *CMS*. [cited 2014; Available from: http://cms.web.cern.ch/.

14. Tanenbaum, A.S. and M.v. Steen, *Distributed Systems: Principles and Paradigms (2nd Edition)*. 2006: Prentice-Hall, Inc.

15. Opocenska, et al. *Dynamic Setup for Clusters with Multi-master Architecture*. in *Software Science, Technology and Engineering (SWSTE), 2010 IEEE International Conference on*. 2010.

16. Matejka, L., J. Safarik, and L. Pesicka. *Distributed File System with Online Multi-master Replicas*. in *Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on the*. 2011.

17. Bui, A., A. Kudireti, and D. Sohier. *A Fully Distributed Clustering Algorithm Based on Random Walks*. in *Parallel and Distributed Computing, 2009. ISPDC '09. Eighth International Symposium on*. 2009.

18. Bernard, T., et al. *A new method to automatically compute processing times for random walks based distributed algorithms*. in *Parallel and Distributed Computing, 2003. Proceedings. Second International Symposium on*. 2003.

19. Kogias, D., K. Oikonomou, and I. Stavrakakis. *Study of randomly replicated random walks for information dissemination over various network topologies*. in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*. 2009.

20. Li, Z., et al. *Distributed computing environment: Approaches and applications*. in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*. 2007.

21. Foster, I. and C. Kesselman, *Globus: a Metacomputing Infrastructure Toolkit.* International Journal of High Performance Computing Applications, 1997. **11**(2): p. 115-128.

22. Chang, L., Z. Zhiwen, and L. Fang. *An Insight into the Architecture of Condor - A Distributed Scheduler*. in *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on*. 2009.

23. David G. Cameron, R.C.-S., A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, and Floriano Zini, *UK Grid Simulation with OptorSim*, in *In e-Science All-Hands Meeting*. 2005.

24. Bell, W., et al., *Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies.* International Journal of High Performance Computing Applications, 2003. **17**: p. 403-416.

25. Cameron, D.G., et al. *Evaluating scheduling and replica optimisation strategies in OptorSim*. in *Grid Computing, 2003. Proceedings. Fourth International Workshop on*. 2003.

26. *Depth-limited search*. [cited 2013 2013,4 September]; Available from: http://en.wikipedia.org/wiki/Depth-limited_search.

27. Barnhart, S.K. and Iowa State University. Cooperative Extension Service., *How pasture plants grow*. Pm 1791. 1999, Ames, Iowa: Iowa State University, University Extension. 1 folded sheet (4 p.).

28. Campillo, F. and N. Champagnat, *Simulation and analysis of an individual-based model for graph-structured plant dynamics.* Ecological Modelling, 2012. **234**(0): p. 93-105.

29. Li, W., *Random texts exhibit Zipf's-law-like word frequency distribution.* Information Theory, IEEE Transactions on, 1992. **38**(6): p. 1842-1845.

30. Levitin, L.B. and B. Schapiro. *Zipf's Law and Information Complexity in an Evolutionary System*. in *Information Theory, 1993. Proceedings. 1993 IEEE International Symposium on*. 1993.

APPENDIX

A.1 EU Data Grid - Mean Job Time

Replication : No Replication — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	20,000	39,000	55,000	71,000	86,000	99,000
Flooding	20,000	39,000	55,000	71,000	89,000	108,000
Centralized	20,000	49,000	75,000	90,000	110,000	140,000
Multi-Master	20,000	50,000	70,000	95,000	120,000	145,000
Random Walk	24,000	45,000	74,000	95,000	120,000	140,000

Replication : No Replication — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	20,000	39,000	58,000	71,000	89,000	109,000
Flooding	20,000	39,000	55,000	71,000	88,000	109,000
Centralized	20,000	45,000	74,000	95,000	120,000	140,000
Multi-Master	20,000	45,000	70,000	95,000	120,000	140,000
Random Walk	24,000	48,000	73,000	95,000	120,000	145,000

Replication : No Replication — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	20,000	39,000	56,000	71,000	89,000	106,000
Flooding	19,000	38,000	58,000	71,000	88,000	108,000
Centralized	20,000	46,000	74,000	95,000	120,000	145,000
Multi-Master	25,000	49,000	70,000	90,000	120,000	140,000
Random Walk	25,000	49,000	74,000	90,000	120,000	145,000

Replication : LRU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	21,000	39,000	54,000	65,000	73,000	81,000
Flooding	20,000	39,000	58,000	65,000	72,000	81,000
Centralized	24,000	49,000	75,000	95,000	120,000	145,000
Multi-Master	25,000	50,000	70,000	90,000	120,000	145,000
Random Walk	24,000	50,000	75,000	95,000	120,000	135,000

Replication : LRU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	20,000	41,000	60,000	66,000	73,000	78,000
Flooding	20,000	40,000	58,000	70,000	76,000	81,000
Centralized	24,000	45,000	75,000	100,000	124,000	145,000
Multi-Master	24,000	45,000	70,000	100,000	120,000	140,000
Random Walk	25,000	45,000	70,000	95,000	115,000	145,000

Replication : LRU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	20,000	36,000	51,000	65,000	74,000	81,000
Flooding	19,000	39,000	50,000	59,000	68,000	76,000
Centralized	20000	35,000	65,000	100,000	120,000	150,000
Multi-Master	25,000	50,000	70,000	95,000	120,000	145,000
Random Walk	25,000	45,000	70,000	95,000	120,000	145,000

Replication : LFU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	20,000	40,000	56,000	65,000	75,000	88,000
Flooding	20,000	40,000	57,000	73,000	86,000	91,000
Centralized	20,000	45,000	70,000	95,000	120,000	145,000
Multi-Master	25,000	45,000	70,000	95,000	120,000	145,000
Random Walk	20,000	45,000	75,000	95,000	120,000	140,000

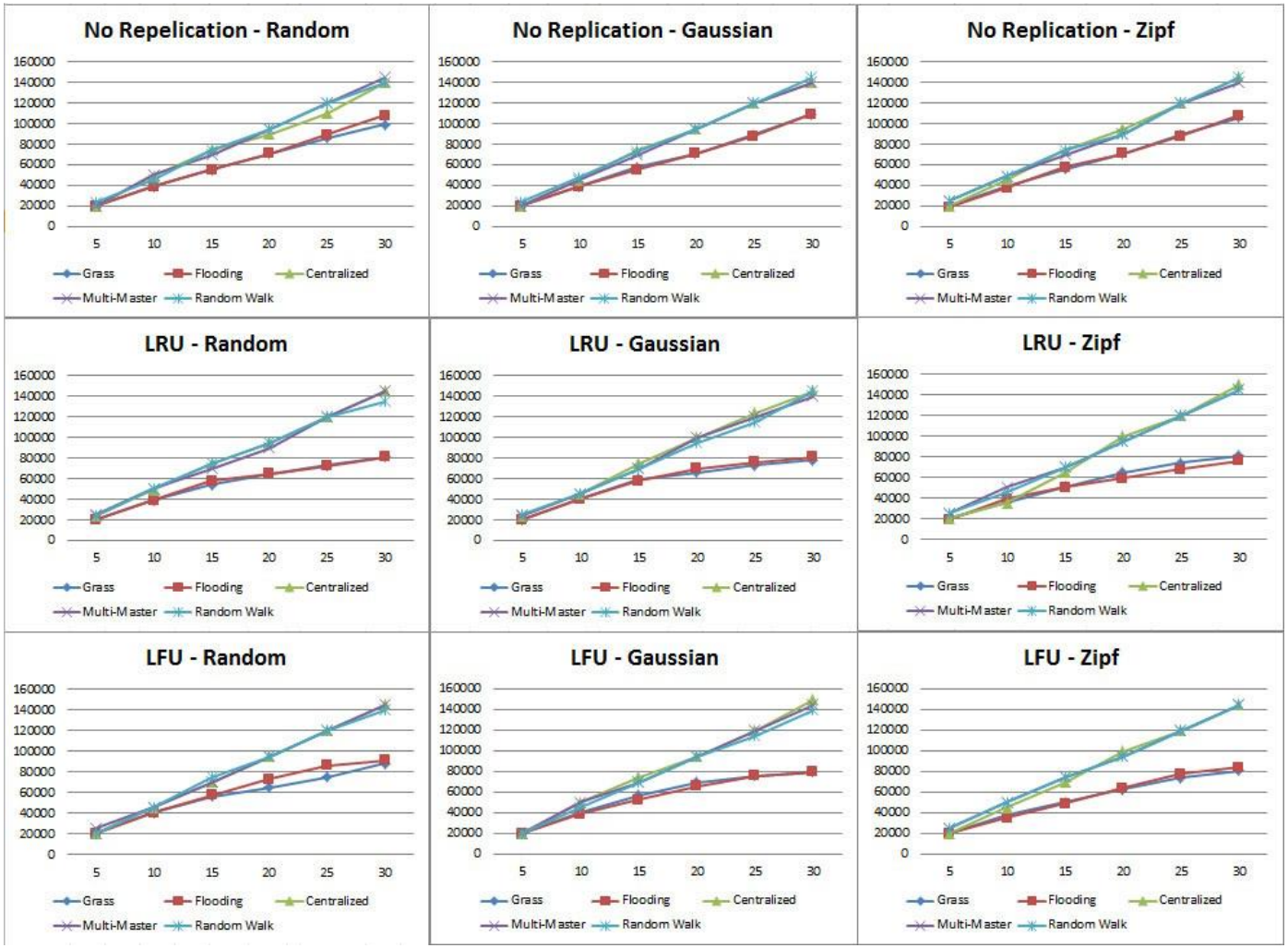
Replication : LFU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	20,000	40,000	57,000	70,000	76,000	80,000
Flooding	20,000	39,000	53,000	66,000	76,000	80,000
Centralized	20,000	50,000	75,000	95,000	120,000	150,000
Multi-Master	20,000	50,000	70,000	95,000	120,000	145,000
Random Walk	20,000	45,000	70,000	95,000	115,000	140,000

Replication : LFU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	20,000	38,000	50,000	63,000	74,000	81,000
Flooding	20,000	35,000	49,000	64,000	78,000	84,000
Centralized	20,000	45,000	70,000	100,000	120,000	145,000
Multi-Master	25,000	50,000	75,000	95,000	120,000	145,000
Random Walk	25,000	50,000	75,000	95,000	120,000	145,000

A.2 EU Data Grid - Mean Job Time Plot



A.3 EU Data Grid - No. of file found in Local File Access

Replication : No Replication — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	2,400	4,900	6,800	8,500	10,100	11,100
Flooding	2,200	4,900	6,900	8,500	10,200	12,100
Centralized	400	900	1,400	1,900	2,400	2,900
Multi-Master	400	900	1,400	1,900	2,400	3,900
Random Walk	1,500	2,900	4,200	5,900	7,100	8,800

Replication : No Replication — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	2,400	4,800	6,800	8,500	10,200	12,100
Flooding	2,400	4,800	6,800	8,500	10,200	12,100
Centralized	500	900	1,400	1,900	2,400	2,800
Multi-Master	1000	2000	2,900	3,900	4,800	5,800
Random Walk	1500	2900	4,200	5,800	7,200	8,800

Replication : No Replication — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	2,400	4,900	6,900	8,500	10,100	11,900
Flooding	2,400	4,800	6,800	8,700	10,300	12,100
Centralized	500	1,000	1,400	1,900	2,400	2,800
Multi-Master	1,000	2,000	3,000	3,900	4,900	5,900
Random Walk	1,400	2,900	4,200	5,800	7,100	8,700

Replication : LRU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	2,900	5,800	8,000	9,800	10,500	11,200
Flooding	3,000	5,900	8,100	9,900	11,200	12,200
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,500	3,000	4,500	6,000	7,300	9,000

Replication : LRU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	2,900	5,800	8,500	9,600	10,600	11,500
Flooding	2,900	5,900	8,500	10,500	11,200	12,100
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,600	3,000	4,500	6,000	7,300	9,000

Replication : LRU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	3,000	5,400	7,600	9,900	11,100	12,100
Flooding	3,000	5,900	7,900	9,300	10,500	11,800
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,800	3,000	4,700	6,000	7,700	9,000

Replication : LFU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	3,000	5,900	8,300	10,000	11,200	12,900
Flooding	3,000	5,900	8,200	10,200	12,100	12,900
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,500	3,000	4,500	6,000	7,500	9,000

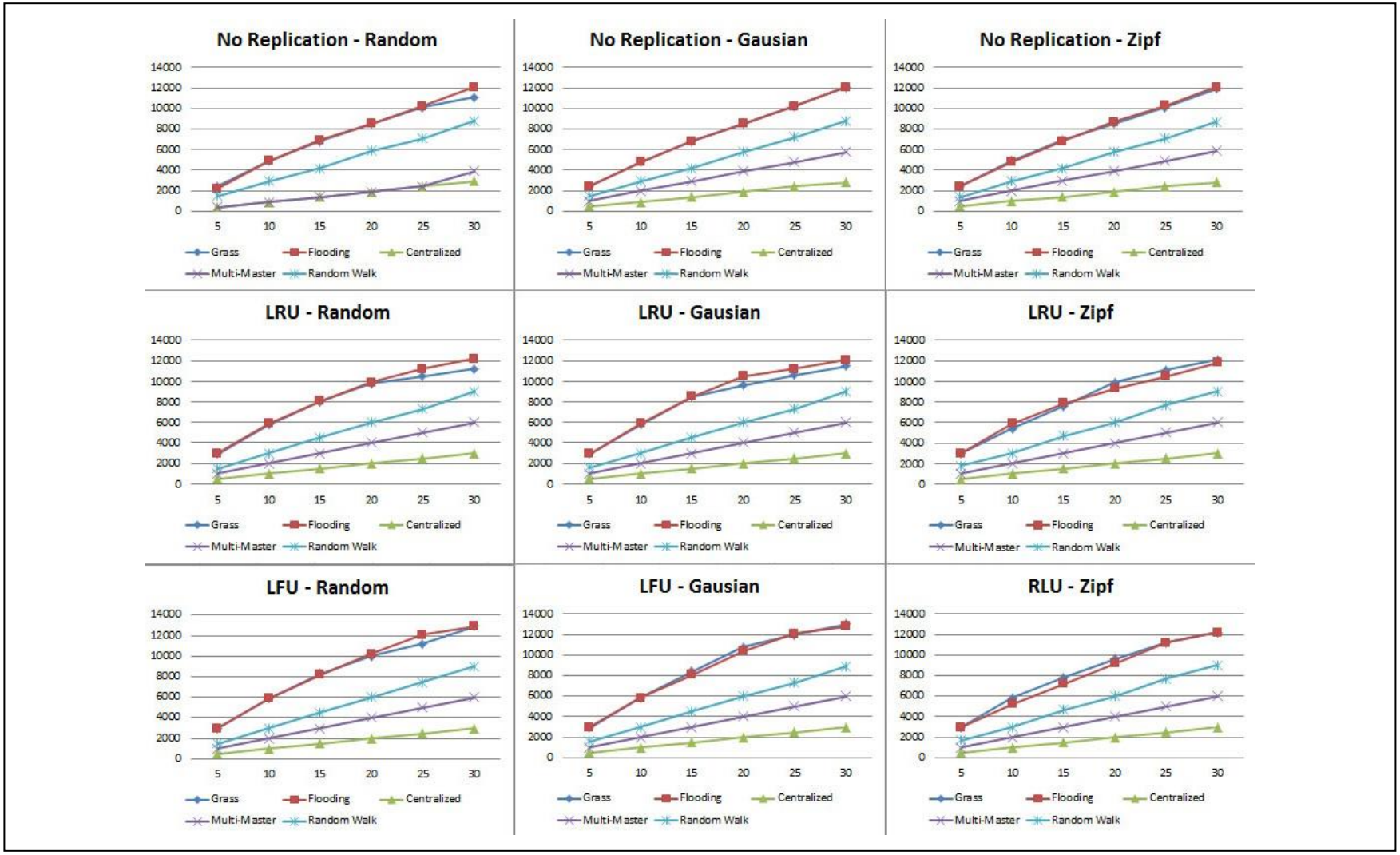
Replication : LFU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	2,900	5,800	8,400	10,800	12,000	13,000
Flooding	3,000	5,800	8,100	10,400	12,100	12,800
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,600	3,000	4,500	6,000	7,300	8,900

Replication : LFU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	3,000	5,800	7,800	9,600	11,200	12,200
Flooding	3,000	5,200	7,200	9,200	11,200	12,200
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,700	3,000	4,700	6,000	7,700	9,000

A.4 EU Data Grid - No. of file found in Local File Access Plot



A.5 EU Data Grid - Effective Network Usage

Replication : No Replication — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : No Replication — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : No Replication — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : LRU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	0.29	0.26	0.25	0.25	0.26	0.26
Flooding	0.29	0.25	0.24	0.24	0.25	0.26
Centralized	0.41	0.41	0.40	0.41	0.41	0.41
Multi-Master	0.53	0.50	0.50	0.49	0.49	0.49
Random Walk	0.40	0.38	0.35	0.34	0.34	0.34

Replication : LRU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	0.29	0.26	0.25	0.25	0.25	0.25
Flooding	0.29	0.26	0.24	0.24	0.25	0.25
Centralized	0.36	0.35	0.35	0.36	0.36	0.36
Multi-Master	0.48	0.44	0.43	0.43	0.45	0.45
Random Walk	0.38	0.36	0.33	0.32	0.32	0.32

Replication : LRU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	0.09	0.08	0.07	0.06	0.06	0.06
Flooding	0.10	0.06	0.06	0.06	0.06	0.06
Centralized	0.65	0.33	0.06	0.06	0.06	0.06
Multi-Master	0.15	0.12	0.11	0.11	0.11	0.11
Random Walk	0.13	0.11	0.10	0.10	0.10	0.09

Replication : LFU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	0.29	0.25	0.26	0.26	0.26	0.27
Flooding	0.27	0.25	0.25	0.26	0.27	0.27
Centralized	0.40	0.41	0.42	0.42	0.41	0.41
Multi-Master	0.58	0.50	0.49	0.49	0.48	0.48
Random Walk	0.34	0.33	0.33	0.31	0.31	0.32

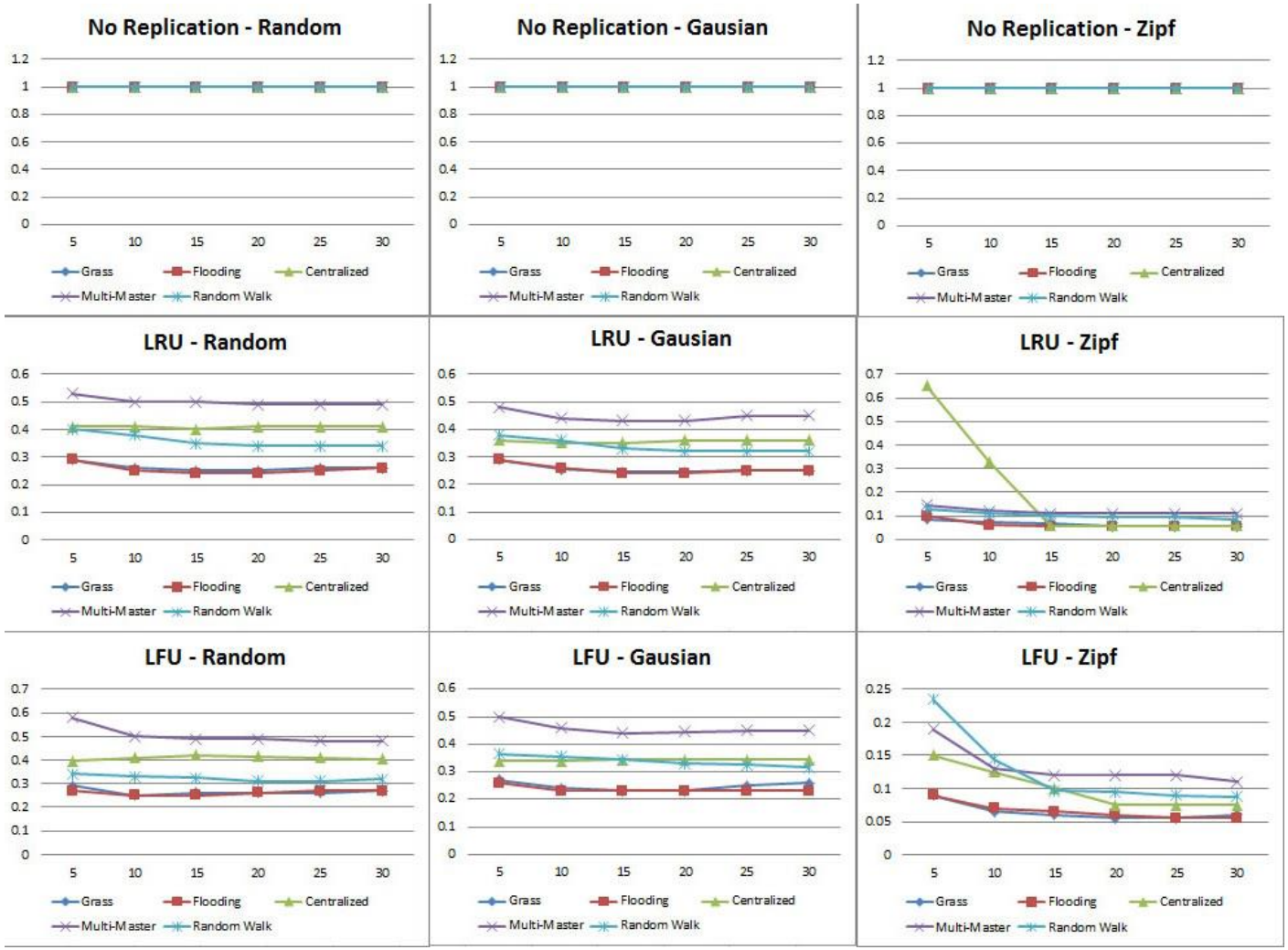
Replication : LFU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	0.27	0.24	0.23	0.23	0.25	0.26
Flooding	0.26	0.23	0.23	0.23	0.23	0.23
Centralized	0.34	0.34	0.35	0.35	0.35	0.35
Multi-Master	0.50	0.46	0.44	0.45	0.45	0.45
Random Walk	0.37	0.36	0.35	0.33	0.33	0.32

Replication : LFU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	0.09	0.07	0.06	0.06	0.06	0.06
Flooding	0.09	0.07	0.07	0.06	0.06	0.06
Centralized	0.15	0.13	0.10	0.08	0.08	0.08
Multi-Master	0.19	0.13	0.12	0.12	0.12	0.11
Random Walk	0.24	0.15	0.10	0.10	0.09	0.09

A.6 EU Data Grid - Effective Network Usage Plot



A.7 GridPP - Mean Job Time

Replication : No Replication — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	2,500	5,000	6,500	8,000	10,000	12,500
Flooding	1,500	3,000	4,500	7,000	7,500	10,500
Centralized	2,000	4,000	6,500	8,000	10,000	12,000
Multi-Master	2,000	4,000	6,000	8,000	10,000	12,000
Random Walk	2,500	4,500	6,000	7,000	10,000	11,500

Replication : No Replication — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	2,500	4,000	5,000	7,500	10,000	11,500
Flooding	1,500	3,000	4,500	6,500	8,000	10,000
Centralized	2,000	4,500	6,500	8,500	10,500	12,500
Multi-Master	2,000	4,000	6,000	8,000	10,000	12,500
Random Walk	2,000	4,000	6,000	8,000	9,000	12,000

Replication : No Replication — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	2,500	5,000	6,000	7,500	9,500	11,000
Flooding	1,250	3,000	5,000	7,000	8,500	10,000
Centralized	2,000	4,000	6,500	8,000	10,000	12,500
Multi-Master	2,000	4,000	6,000	8,000	10,000	12,000
Random Walk	1,500	2,500	5,000	7,500	8,000	12,000

Replication : LRU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	2,000	3,000	5,000	7,000	8,000	10,000
Flooding	20	100	200	300	400	500
Centralized	2,500	3,000	6,000	7,500	9,000	11,000
Multi-Master	2,000	4,000	5,000	7,500	10,000	13,000
Random Walk	1,500	2,600	4,500	6,500	8,000	10,000

Replication : LRU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	1,500	3,000	5,000	7,000	8,000	11,000
Flooding	900	2,000	3,100	4,100	5,100	5,900
Centralized	2,400	3,500	6,000	7,500	9,000	11,500
Multi-Master	2,000	4,000	5,000	5,200	8,000	10,000
Random Walk	2,000	3,000	6,000	6,500	8,000	9,500

Replication : LRU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	1,000	1,900	2,900	3,900	4,600	5,100
Flooding	96	98	94	93	94	94
Centralized	1,400	2,900	4,500	6,200	7,800	9,400
Multi-Master	1,500	2,800	4,000	5,500	6,800	8,000
Random Walk	750	1,300	1,800	2,400	2,800	3,300

Replication : LFU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	2,000	3,000	5,500	7,000	8,500	10,500
Flooding	900	1,900	2,200	3,400	4,300	5,500
Centralized	2,000	3,000	6,000	7,500	10,000	12,000
Multi-Master	2,000	4,000	5,000	7,500	9,000	11,000
Random Walk	1,250	3,500	5,000	6,500	8,000	10,000

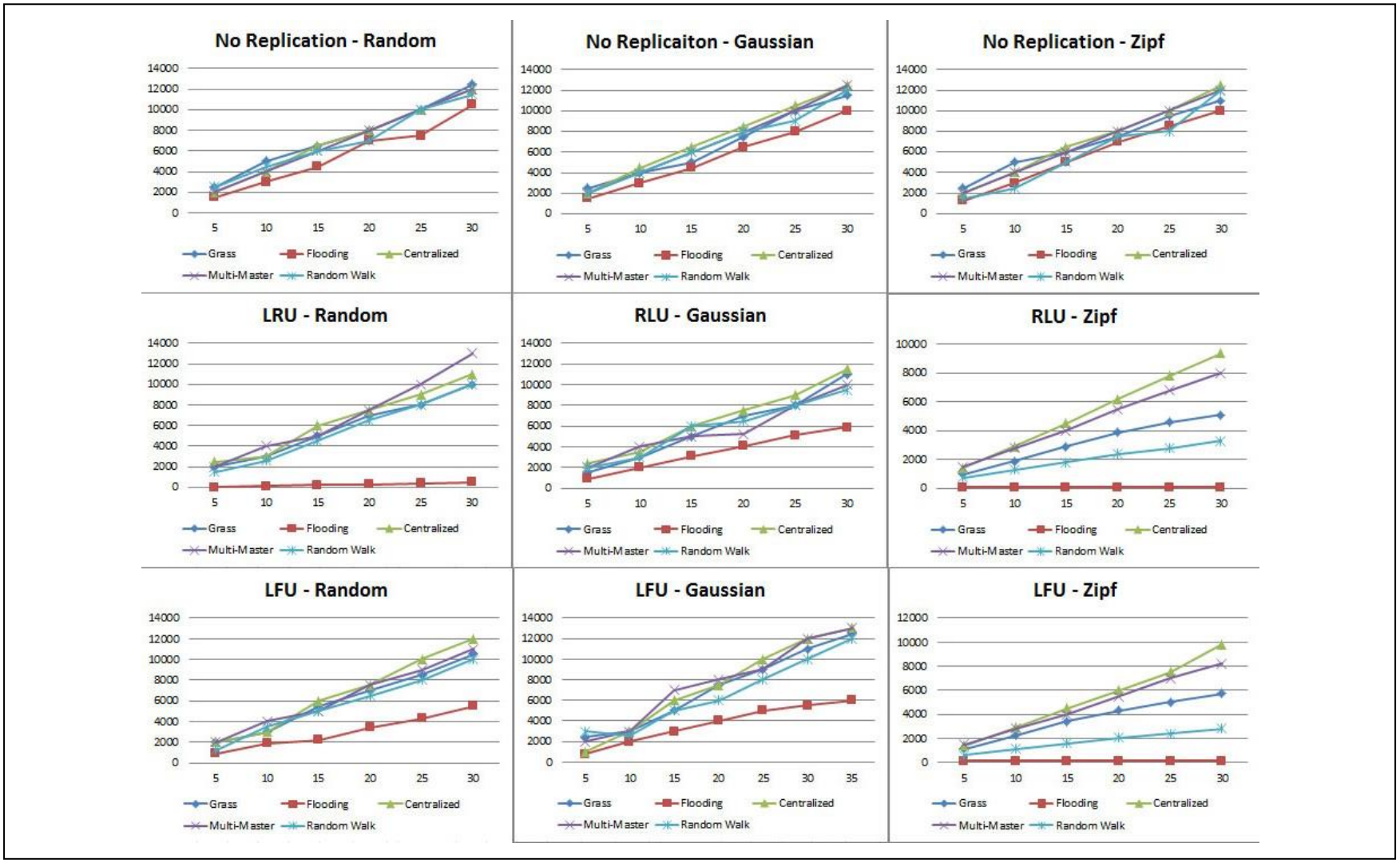
Replication : LFU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	2,400	3,000	5,000	7,500	9,000	11,000
Flooding	750	2,000	3,000	4,000	5,000	5,500
Centralized	1,000	3,000	6,000	7,500	10,000	12,000
Multi-Master	2,000	3,000	7,000	8,000	9,000	12,000
Random Walk	3,000	2,600	5,000	6,000	8,000	10,000

Replication : LFU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	1,100	2,250	3,400	4,300	5,000	5,700
Flooding	112	115	108	104	102	101
Centralized	1,400	2,900	4,500	6,000	7,500	9,800
Multi-Master	1,500	2,800	4,000	5,500	7,000	8,200
Random Walk	600	1,100	1,600	2,050	2,400	2,800

A.8 GridPP - Mean Job Time Plot



A.9 GridPP - Local File Access

Replication : No Replication — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	1,000	2,000	2,400	3,000	4,000	5,000
Flooding	1,500	3,000	4,500	6,000	7,500	9,000
Centralized	500	1,200	1,800	2,400	3,000	3,600
Multi-Master	1,000	1,500	2,000	2,500	3,000	4,000
Random Walk	1,000	2,000	2,500	3,000	4,000	5,500

Replication : No Replication — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	1,000	1,500	2,500	3,000	4,000	4,500
Flooding	1,500	3,000	4,500	6,500	7,500	9,000
Centralized	650	1,200	1,800	2,500	3,000	3,500
Multi-Master	500	1,500	2,000	2,500	3,000	4,000
Random Walk	1,000	2,000	3,000	3,500	4,000	5,500

Replication : No Replication — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	1,500	2,000	2,500	3,000	3,500	5,000
Flooding	1,250	3,000	4,000	6,500	7,500	8,500
Centralized	500	1,200	2,000	2,500	3,000	3,500
Multi-Master	500	1,000	1,500	2,000	3,500	4,000
Random Walk	1,000	2,000	2,500	3,500	4,500	5,500

Replication : LRU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	1,250	2,500	4,000	6,000	7,000	8,000
Flooding	2,500	6,000	11,000	16,000	20,000	24,000
Centralized	1,000	1,800	2,800	3,500	4,500	5,500
Multi-Master	1,000	2,000	3,000	4,000	6,000	6,500
Random Walk	1,500	2,000	4,000	6,500	8,000	10,000

Replication : LRU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	1,500	2,500	3,300	5,500	8,800	8,500
Flooding	2,500	6,000	10,000	14,000	19,000	23,000
Centralized	1,000	1,800	2,800	3,600	4,600	5,500
Multi-Master	1,000	2,500	4,000	5,000	6,000	7,000
Random Walk	2,000	3,500	5,500	7,000	8,500	11,000

Replication : LRU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	3,000	7,500	12,000	17,000	21,000	23,000
Flooding	6,000	12,500	19,000	25,500	32,000	33,000
Centralized	2,100	4,200	6,800	9,000	11,400	13,500
Multi-Master	2,500	5,000	8,000	11,000	13,000	16,500
Random Walk	3,500	8,000	15,000	20,000	25,000	26,000

Replication : LFU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	1,250	2,500	4,500	6,000	8,000	9,000
Flooding	3,000	6,000	10,000	14,000	18,000	22,500
Centralized	1,000	1,900	3,000	3,600	4,500	5,500
Multi-Master	1,250	2,500	3,000	4,000	6,000	7,000
Random Walk	1,300	3,000	5,000	6,000	8,000	10,000

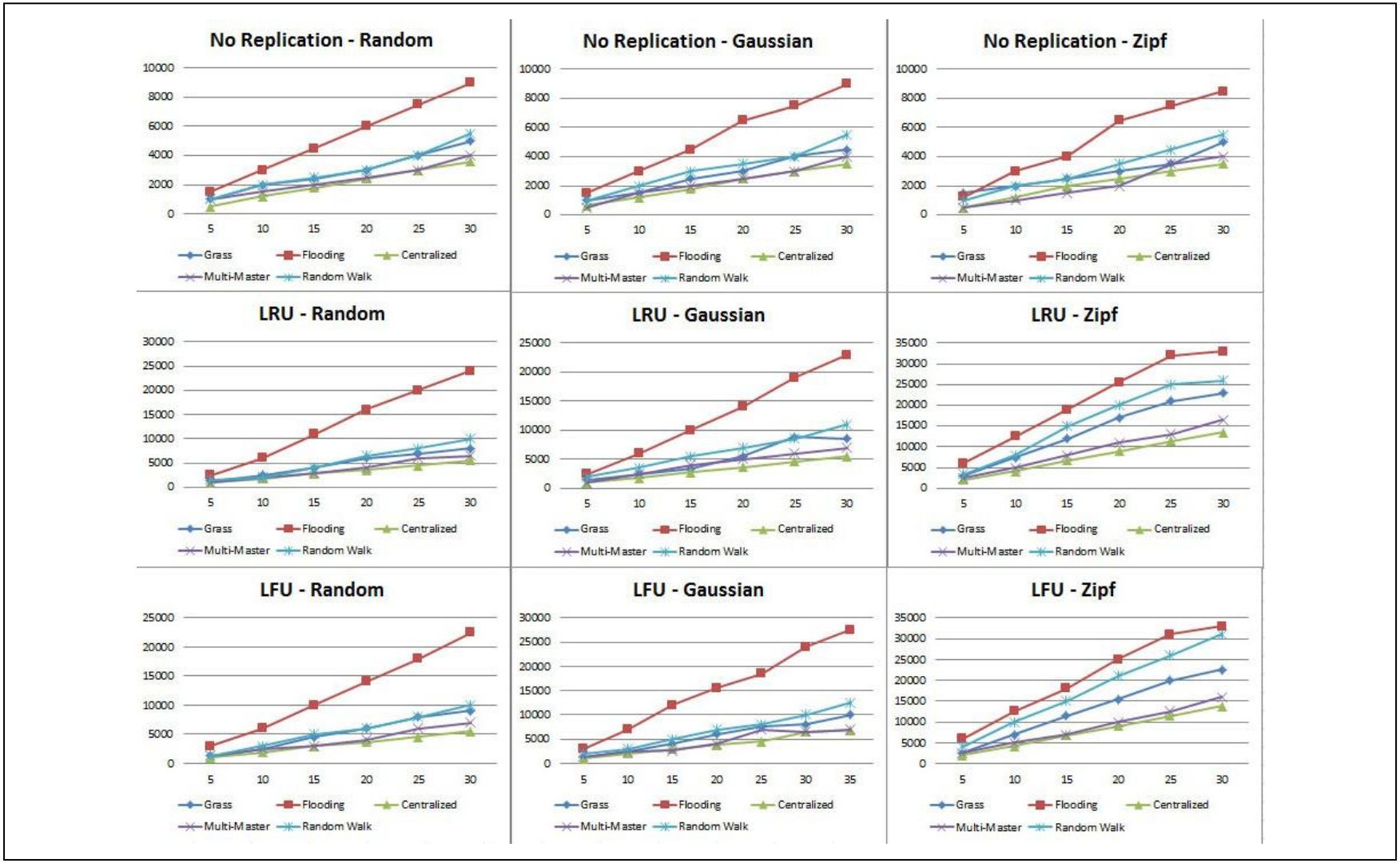
Replication : LFU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	1,250	2,500	4,000	6,000	7,500	8,000
Flooding	3,000	7,000	12,000	15,500	18,500	24,000
Centralized	1,000	2,000	3,000	3,800	4,500	6,500
Multi-Master	1,250	2,400	2,600	4,000	7,000	6,500
Random Walk	2,000	3,000	5,000	7,000	8,000	10,000

Replication : LFU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	2,500	7,000	11,500	15,500	20,000	22,500
Flooding	6,000	12,500	18,000	25,000	31,000	33,000
Centralized	2,000	4,200	6,800	9,000	11,500	13,800
Multi-Master	2,500	5,000	7,000	10,000	12,500	16,000
Random Walk	4,000	10,000	15,000	21,000	26,000	31,000

A.10 GridPP - Local File Access Plot



A.11 GridPP - Effective Network Usage

Replication : No Replication — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : No Replication — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : No Replication — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : LRU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	0.64	0.60	0.56	0.53	0.53	0.53
Flooding	0.57	0.51	0.48	0.44	0.43	0.43
Centralized	0.62	0.62	0.62	0.62	0.62	0.63
Multi-Master	0.62	0.61	0.61	0.61	0.61	0.60
Random Walk	0.63	0.58	0.57	0.56	0.54	0.54

Replication : LRU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	0.62	0.61	0.58	0.56	0.55	0.54
Flooding	0.58	0.52	0.50	0.47	0.46	0.47
Centralized	0.61	0.61	0.62	0.62	0.62	0.62
Multi-Master	0.60	0.58	0.58	0.58	0.59	0.59
Random Walk	0.58	0.52	0.51	0.51	0.51	0.50

Replication : LRU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	0.23	0.19	0.18	0.17	0.17	0.17
Flooding	0.20	0.15	0.13	0.12	0.11	0.10
Centralized	0.25	0.24	0.23	0.23	0.23	0.23
Multi-Master	0.25	0.23	0.22	0.22	0.22	0.22
Random Walk	0.21	0.18	0.17	0.17	0.17	0.17

Replication : LFU — Access Pattern : Random

time(s)	5	10	15	20	25	30
Grass	0.64	0.60	0.55	0.52	0.49	0.49
Flooding	0.61	0.55	0.53	0.51	0.48	0.48
Centralized	0.64	0.63	0.63	0.62	0.62	0.63
Multi-Master	0.63	0.63	0.62	0.62	0.62	0.62
Random Walk	0.62	0.52	0.53	0.53	0.53	0.53

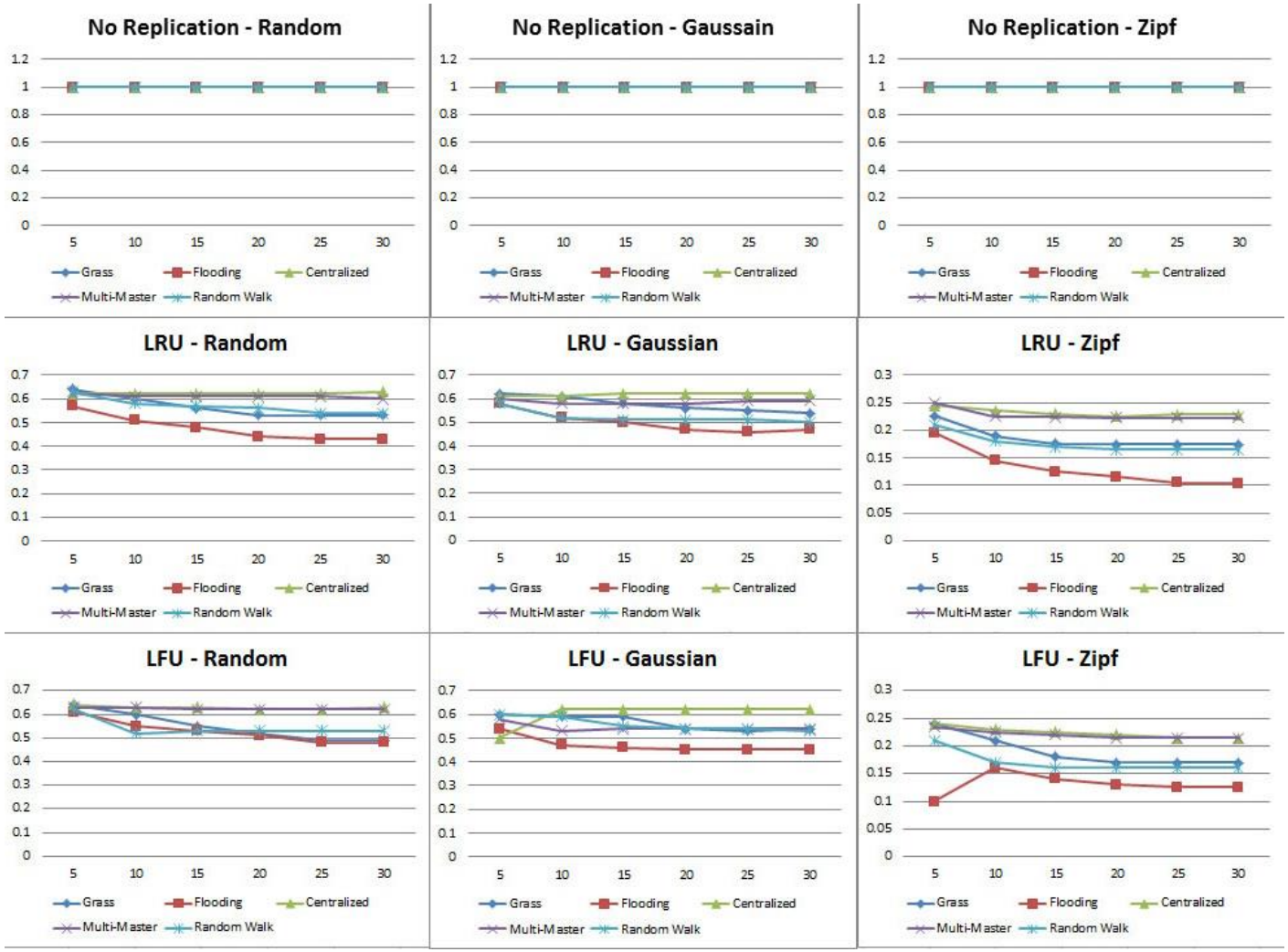
Replication : LFU — Access Pattern : Gausian

time(s)	5	10	15	20	25	30
Grass	0.60	0.59	0.59	0.54	0.53	0.54
Flooding	0.54	0.47	0.46	0.45	0.45	0.45
Centralized	0.50	0.62	0.62	0.62	0.62	0.62
Multi-Master	0.58	0.53	0.54	0.54	0.54	0.54
Random Walk	0.60	0.59	0.55	0.54	0.54	0.53

Replication : LFU — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	0.24	0.21	0.18	0.17	0.17	0.17
Flooding	0.10	0.16	0.14	0.13	0.13	0.13
Centralized	0.24	0.23	0.23	0.22	0.22	0.22
Multi-Master	0.24	0.23	0.22	0.22	0.22	0.22
Random Walk	0.21	0.17	0.16	0.16	0.16	0.16

A.12 GridPP - Effective Network Usage Plot



A.13 CMS - Mean Job Time Plot

Replication : No Replication — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	5,000	9,000	14,000	19,000	22,500	26,000
Flooding	5,000	9,000	14,000	19,000	21,500	23,500
Centralized	5,000	10,000	15,000	20,000	25,000	30,000
Multi-Master	4,000	10,000	15,000	20,000	24,000	27,000
Random Walk	5,000	10,000	15,000	20,000	25,000	30,000

Replication : No Replication — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	4,500	9,000	14,000	19,000	22,000	24,000
Flooding	5,000	10,500	15,000	19,500	23,000	26,000
Centralized	5,000	10,000	15,000	20,000	25,000	30,000
Multi-Master	2,500	10,000	13,000	20,000	23,000	25,000
Random Walk	5,000	10,000	15,000	20,000	24,000	27,500

Replication : No Replication — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	5,000	10,000	14,000	18,000	21,500	24,000
Flooding	5,000	10,000	14,500	19,000	22,500	24,000
Centralized	5,000	10,000	20,000	25,000	30,000	35,000
Multi-Master	5,000	10,000	15,000	22,000	24,000	25,500
Random Walk	5,000	10,000	15,000	19,000	24,000	26,000

Replication : LRU — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	500	1,000	2,000	2,300	3,200	3,400
Flooding	400	650	1,100	1,550	1,800	1,900
Centralized	500	1,000	2,000	2,900	3,400	4,000
Multi-Master	500	1,000	1,200	1,900	2,800	3,100
Random Walk	500	1,000	2,000	2,750	3,300	3,800

Replication : LRU — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	500	1,000	2,200	2,500	3,500	4,100
Flooding	500	780	900	1,200	1,320	1,320
Centralized	500	1,000	2,000	3,000	4,000	5,000
Multi-Master	500	1,000	1,700	3,000	3,500	4,300
Random Walk	500	1,000	1,500	2,000	2,600	3,250

Replication : LRU — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	75	225	375	450	460	460
Flooding	115	210	350	460	425	375
Centralized	300	600	900	1,200	1,500	1,800
Multi-Master	220	340	570	850	920	1,040
Random Walk	125	240	375	520	550	510

Replication : LFU — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	500	1,100	1,900	2,100	2,300	2,600
Flooding	400	600	960	1,250	1,550	1,600
Centralized	500	1,000	1,500	2,000	2,500	3,000
Multi-Master	500	1,000	1,500	2,200	3,500	4,000
Random Walk	500	1,100	1,500	2,000	2,750	2,450

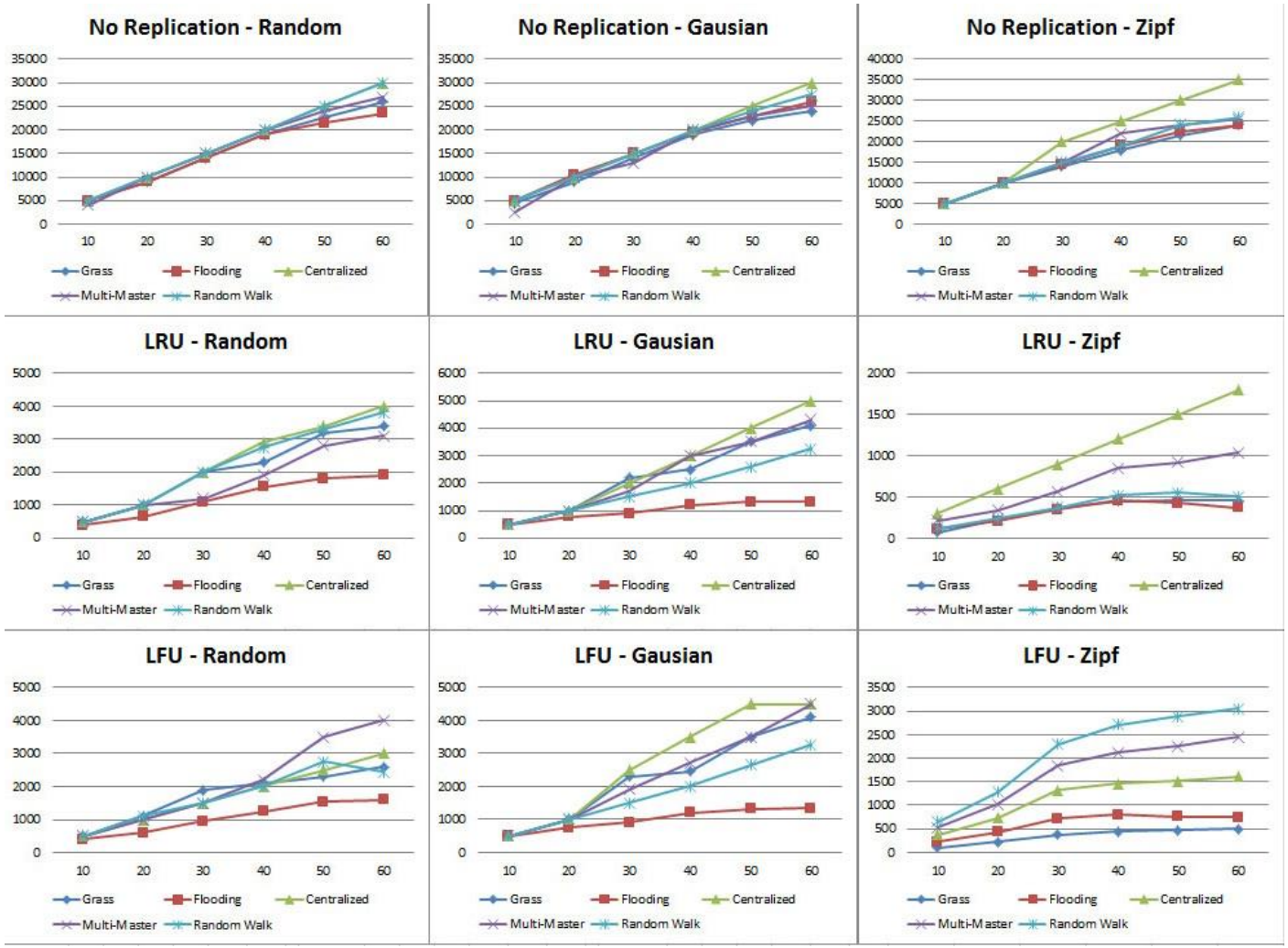
Replication : LFU — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	500	1,000	2,300	2,450	3,500	4,100
Flooding	500	750	900	1,200	1,320	1,340
Centralized	500	1,000	2,500	3,500	4,500	4,500
Multi-Master	500	1,000	1,900	2,700	3,500	4,500
Random Walk	500	1,000	1,500	2,000	2,650	3,250

Replication : LFU — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	100	225	375	440	475	500
Flooding	125	200	340	360	280	250
Centralized	150	300	600	660	760	860
Multi-Master	150	290	525	650	740	840
Random Walk	125	260	450	600	625	600

A.14 CMS - Mean Job Time Plot



A.15 CMS - Local File Access

Replication : No Replication — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	2,200	4,200	6,500	8,800	10,000	11,000
Flooding	2,200	4,600	7,000	9,000	11,000	12,200
Centralized	100	150	200	250	300	350
Multi-Master	1,000	2,000	3,000	4,000	5,000	6,000
Random Walk	1,800	3,500	5,000	6,900	8,500	9,900

Replication : No Replication — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	2,400	4,800	6,800	8,500	10,200	12,100
Flooding	2,400	4,800	6,800	8,500	10,200	12,100
Centralized	500	900	1,400	1,900	2,400	2,800
Multi-Master	1000	2000	2,900	3,900	4,800	5,800
Random Walk	1500	2900	4,200	5,800	7,200	8,800

Replication : No Replication — Access Pattern : Zipf

time(s)	5	10	15	20	25	30
Grass	2,400	4,900	6,900	8,500	10,100	11,900
Flooding	2,400	4,800	6,800	8,700	10,300	12,100
Centralized	500	1,000	1,400	1,900	2,400	2,800
Multi-Master	1,000	2,000	3,000	3,900	4,900	5,900
Random Walk	1,400	2,900	4,200	5,800	7,100	8,700

Replication : LRU — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	500	1,000	3,500	4,500	6,500	7,200
Flooding	1,100	2,800	7,900	10,100	11,800	11,900
Centralized	500	700	1,000	1,200	1,800	2,000
Multi-Master	500	1,100	3,800	5,500	6,000	6,600
Random Walk	700	1,800	3,500	5,200	7,500	9,000

Replication : LRU — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	500	1,000	3,500	4,000	6,000	7,200
Flooding	700	4,000	9,000	11,200	12,500	12,800
Centralized	240	480	960	1,440	1,920	2,400
Multi-Master	500	1,500	2,300	4,000	5,000	6,000
Random Walk	500	1,700	5,000	7,000	8,800	10,200

Replication : LRU — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	100	900	3,200	4,700	8,000	10,700
Flooding	100	500	2,000	4,200	8,500	12,500
Centralized	100	200	200	300	400	400
Multi-Master	100	400	1,100	3,000	4,700	6,800
Random Walk	100	600	1,100	4,200	6,750	10,250

Replication : LFU — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	500	2,000	7,500	8,500	10,200	11,100
Flooding	1,500	3,500	8,900	11,100	13,000	13,000
Centralized	500	2,200	3,500	4,300	4,500	5,000
Multi-Master	400	1,700	2,700	3,300	4,700	5,400
Random Walk	500	2,100	3,800	4,500	7,100	8,700

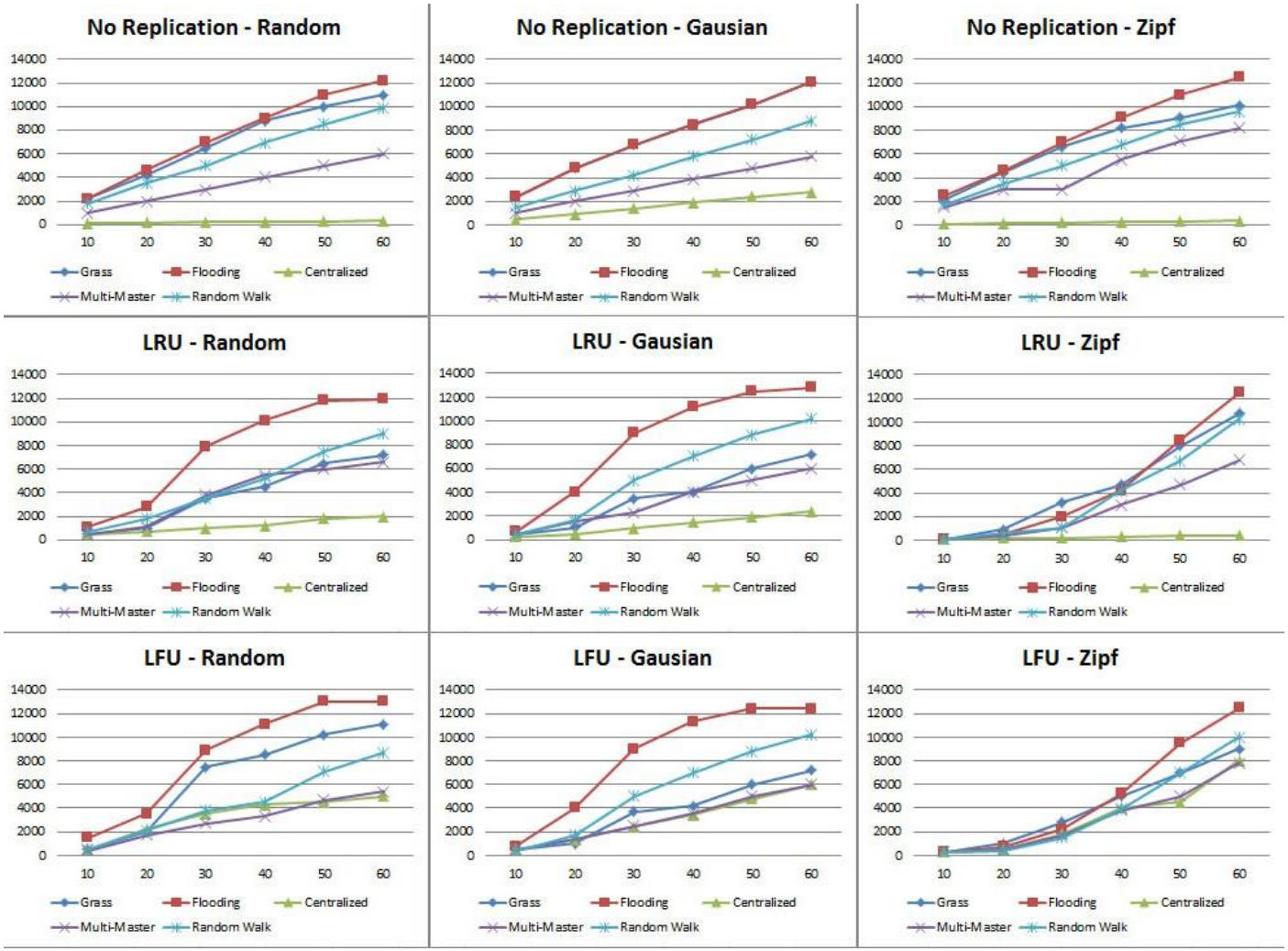
Replication : LFU — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	500	1000	3700	4200	6000	7200
Flooding	800	4000	9000	11300	12400	12400
Centralized	500	1300	2400	3400	4800	6000
Multi-Master	500	1400	2500	3500	5000	6000
Random Walk	400	1700	5000	7000	8800	10200

Replication : LFU — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	300	1,000	2,800	5,000	7,000	9,000
Flooding	300	700	2,200	5,200	9,500	12,500
Centralized	300	500	1,800	4,000	4,500	8,000
Multi-Master	300	500	1,700	3,800	5,000	7,800
Random Walk	300	400	1,500	3,900	7,000	10,000

A.16 CMS - Local File Access Plot



A.17 CMS - Effective Network Usage

Replication : No Replication — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : No Replication — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : No Replication — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	1.00	1.00	1.00	1.00	1.00	1.00
Flooding	1.00	1.00	1.00	1.00	1.00	1.00
Centralized	1.00	1.00	1.00	1.00	1.00	1.00
Multi-Master	1.00	1.00	1.00	1.00	1.00	1.00
Random Walk	1.00	1.00	1.00	1.00	1.00	1.00

Replication : LRU — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	0.75	0.30	0.23	0.17	0.17	0.18
Flooding	0.60	0.30	0.23	0.17	0.20	0.23
Centralized	0.90	0.35	0.33	0.34	0.35	0.35
Multi-Master	0.75	0.23	0.15	0.16	0.17	0.18
Random Walk	1.00	0.40	0.25	0.23	0.20	0.20

Replication : LRU — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	0.27	0.27	0.17	0.20	0.19	0.19
Flooding	0.35	0.15	0.15	0.17	0.20	0.21
Centralized	0.40	0.28	0.26	0.25	0.25	0.25
Multi-Master	0.40	0.20	0.23	0.23	0.23	0.22
Random Walk	0.43	0.25	0.15	0.17	0.18	0.19

Replication : LRU — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	0.25	0.10	0.06	0.06	0.04	0.04
Flooding	0.35	0.27	0.15	0.11	0.07	0.06
Centralized	0.35	0.30	0.20	0.20	0.20	0.20
Multi-Master	0.25	0.16	0.13	0.08	0.05	0.05
Random Walk	0.30	0.15	0.12	0.06	0.05	0.05

Replication : LFU — Access Pattern : Random

time(s)	10	20	30	40	50	60
Grass	0.48	0.25	0.12	0.14	0.15	0.17
Flooding	0.33	0.25	0.20	0.23	0.25	0.25
Centralized	0.33	0.23	0.23	0.23	0.23	0.23
Multi-Master	0.27	0.21	0.23	0.24	0.24	0.25
Random Walk	0.43	0.20	0.23	0.23	0.21	0.23

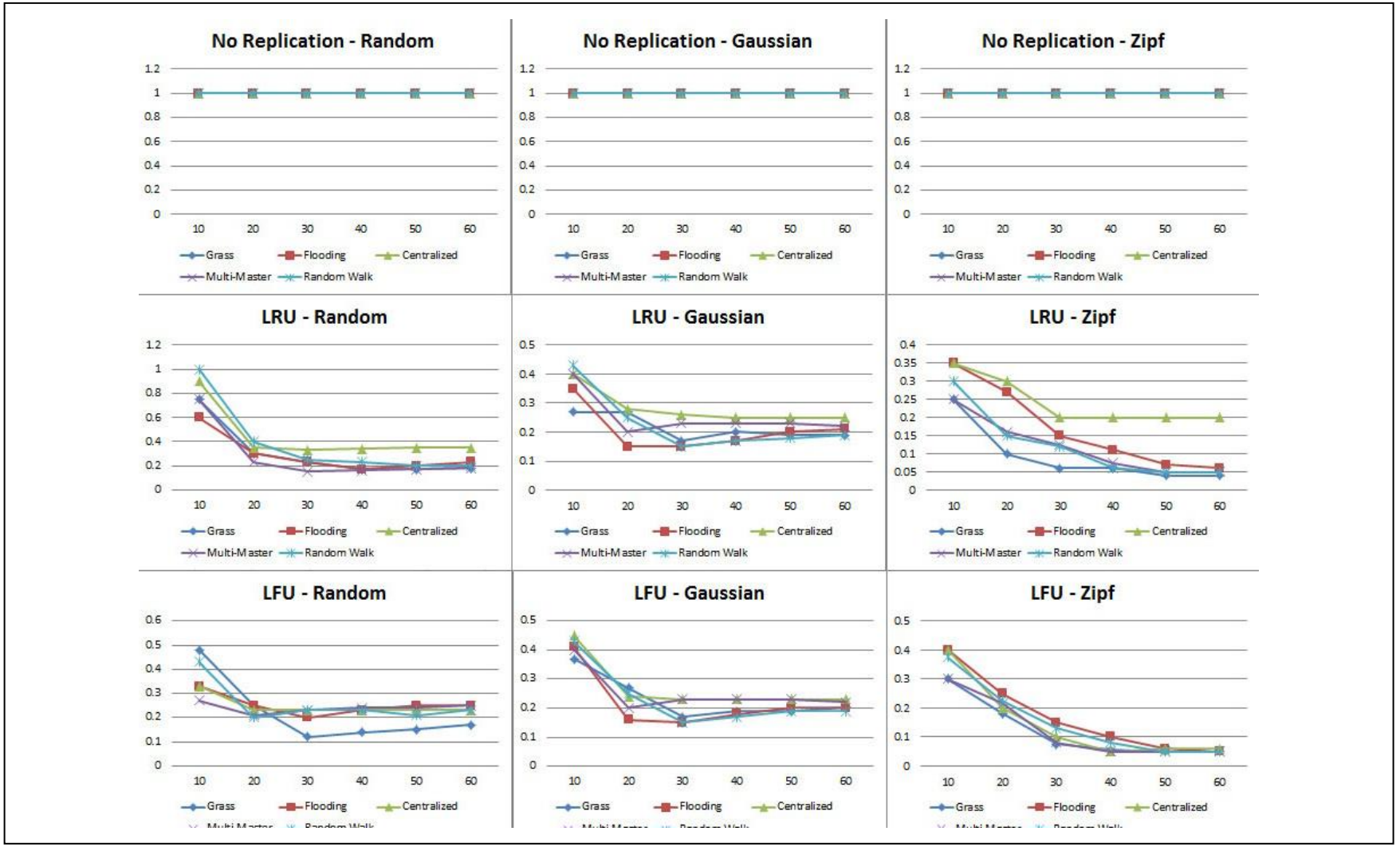
Replication : LFU — Access Pattern : Gausian

time(s)	10	20	30	40	50	60
Grass	0.37	0.27	0.17	0.19	0.19	0.20
Flooding	0.41	0.16	0.15	0.18	0.20	0.20
Centralized	0.45	0.24	0.23	0.23	0.23	0.23
Multi-Master	0.40	0.20	0.23	0.23	0.23	0.22
Random Walk	0.43	0.25	0.15	0.17	0.19	0.19

Replication : LFU — Access Pattern : Zipf

time(s)	10	20	30	40	50	60
Grass	0.30	0.18	0.08	0.06	0.05	0.05
Flooding	0.40	0.25	0.15	0.10	0.06	0.05
Centralized	0.40	0.20	0.10	0.05	0.06	0.06
Multi-Master	0.30	0.22	0.08	0.05	0.05	0.05
Random Walk	0.38	0.23	0.13	0.08	0.05	0.05

A.18 CMS - Effective Network Usage Plot



Appendix B

Conference Paper - 2013 International Conference on Computer, Network and Communication Engineering (ICCNCE 2013) ISBN: 978-90-78677-67-3 p.446-449

International Conference on Computer, Networks and Communication Engineering (ICCNCE 2013)

Distributed Positioning of Replica using Grass Growing Structure

Worawit Fankam-ai and Peraphon Sophatsathit

Advanced Virtual and Intelligent Computing (AVIC) Center, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok, Thailand

fworawit@gmail.com, peraphon.s@chula.ac.th

Abstract - This paper proposes a distributed data replication scheme based on Grass Growing Structure to reduce bandwidth consumption and access latency in distributed systems. The replication will be multicast to the nearest nodes using Depth Limit Search algorithm within a predefined limiting distance. Performance is measured by means of effective network usage and mean job time. The EU Data Grid Testbed is employed as the benchmarking assessment to compare the proposed approach with conventional Centralized and Flooding algorithms. The results yield less access latency, good scalability, and reliability than those comparable approaches.

Index Terms - Distributed Positioning, Replica location, Grass Growing Structure.

1. Introduction

Replication is a common method used to improve the performance of data access in distributed systems. It improves not only data access efficiency, but also data availability and fault tolerance. In order to achieve higher replication performance, there must be an efficient replica scheme to manage the replication process. Replica scheme mainly includes replication strategy and replica selection strategy to find the best-fit replica, replication consistency, and replica positioning mechanisms. Replication strategy determines when and where to create a replica, taking into account of the factors such as number of data requests, network condition, and storage availability of each replica site.

In this paper, we propose a replication positioning algorithm based on Grass Growing Structure. The focus on replica positioning mechanisms is to determine where the new replica should be located so that multicast traffic will be minimal. The algorithm is inspired by the growing of grass with adequate irrigation which will flourish more than those depleting of water. Grass areas that receive water represent distribution of data replicas, while grass trunk represents network topology. Thus, data replicas spread around starting from the initial source node. Data are replicated only via the designated routes. No superfluous distribution that occupies limited bandwidth to be wasted. Thus, performance of this algorithm will be compared with Flooding algorithm [1][5][11] and Centralized Location algorithm [1][5][11] measured by Mean Job Time and Effective Network Usage [2][3][4][5].

2. Related Work

Some principal definitions of replication location model and two related replica location algorithms [1][5][11] are described as a basis for the development of the proposed algorithm.

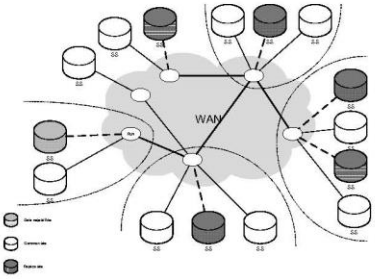


Fig. 1 Replica Location Server Model

A. Terminology

A logical file name (LFN) is a unique logical identifier for desired data content. The replica location service must identify one or more physical copies (replicas) of the logical file. Each physical copy is identified by a physical file name (PFN), which specifies its location on a storage site.

A number of storage sites (SS) collaborate to share their storage capabilities to all users. A replica location node (RLN) aggregates LFN to PFN mappings from one or more SSs and collaborates with other RLNs to build a distributed catalog of LFN mappings.

RLNs offer both a query interface to clients and a registration interface that SSs can enlist PFN to LFN mapping for files stored locally. RLNs also organize into a search network to allow remote searches. Nodes in this network distribute compressed information on the set of LFN mappings stored locally in the form of node digests.

Depth Limit Search (DLS) [8], like the normal depth-first search, is an uninformed search. It works exactly like depth-first search, but avoids the completeness drawbacks by imposing a maximum limit on the depth of the search. Even if

the search could still expand a vertex beyond that depth, DLS will not do so and thereby will not follow infinitely deep paths or get stuck in cycles. Therefore depth limited search will find a solution if it is within the depth limit, which guarantees at least completeness on all graphs.

B. Definitions

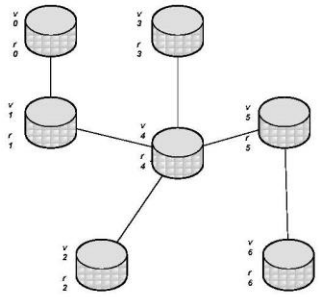


Fig 2. Centralized Location

- V is the aggregation of sites in the data grid system
- V_{SS} is the set of nodes that storage desired data(files) and their copies
- l_{vi} is the location of the node v_i
- V_{rln} is the set of sites that aggregate information about LFN to PFN mapping or some information about nodes
- Flooding algorithm. Each progress of distribution location starts from the node $v_0 \in V_{SS}$ that stores the information about the location of V_{SS}, but does not include the corresponding relationship between files and V_{SS}. That is to say, the algorithm does not know the location of V_{SS} before locating one replica r. In all the information for any node $v_i \in V_{SS}$ stored in the node v_0, the corresponding location L_i is unknown. Thus, this algorithm can be costly in terms of wasted bandwidth while a message may only have one destination to be sent. Moreover, messages can duplicate in the network which increase the load on the network bandwidth. Worse yet, duplicate packets may circulate forever unless certain precautions are taken [9][10].
- Centralized Location Algorithm [1]. In Fig. 2, V_4 is only one V_{rln} which contains all the information about location of V_{SS} and LFN to PFN mappings.

3. Distributed Positioning of Replica using Grass Growing Structure

The proposed approach to position the replica will follow grass growing pattern, aka Grass Growing Structure. The area of growing grass with adequate irrigation will flourish better than the one depleting of water. Grass areas that receive water represent the position of data replicas, having grass trunks as the replica links that form the distribution topology. Fig. 3 shows an example of grass growing structure. The left figure represents grass trunk and right one is link distribution topology. If grass receives water at node 4, then node 6,7,8 will flourish better than other nodes. So the nearest nodes to the initial source node 4 are 2,6,7,8

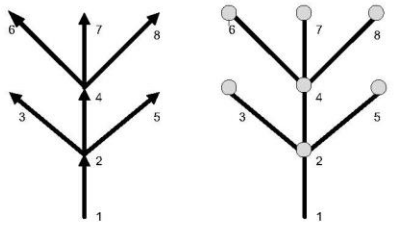


Fig. 3 Grass Growing Structure

The first stage positioning of replicas is to set up a random initial source node. Selection of neighboring nodes uses DLS algorithm [8] to find the path between source node and neighboring nodes. The procedure starts stepping each node with a vector to all directly attached nodes and advertising its current vector to all neighboring nodes. In the process, it finds the shortest distance and updates the distance cost. The nearest neighboring nodes just selected will be used as the second stage positioning of replica source nodes. This process repeats until one of the nearest neighbour nodes is the destination node. At which point, data replication commences. Thus, considerable network traffic is reduced compared with conventional flooding algorithm. However, we imposed a stopping criterion on the Grass Growing Structure algorithm by limiting the DLS depth to 2 to prevent indefinite depth search. The *GrassGrowing* algorithm is shown below.

```
Function GrassGrowing
Begin
    Location replica r starts from initial node v0;
    // Select a node vi using DLS algorithm
    If DLS limit != 2
      Replicate data on node vi
    Else repeat GrassGrowing
End;
```

4. Simulation and evaluation

To measure the performance of the Grass Growing Structure algorithm, we employed a simulation using OptorSim [2][3][4] with network topology from EU Data Grid Testbed [6] as shown in Fig. 4. The results were compared with Flooding and Centralized Location algorithms which performed on the same testbed.

Performance measurement is carried out by means of mean job execution time [2][3][4] and effective network usage (ENU) [2][3][4]. Details are described below.

A. Grid Configuration

This research used OptorSim [2][3][4] as the data grid simulator to simulate real data grid environment. This simulator is developed in Java under the funding of the European Data Grid project (EU Data Grid).

The OptorSim simulation ran on Intel Xeon 2.1GHz. There were three input configuration files. They are:

1. Network topology file that described the links between different sites, the available network bandwidth, and size of disk storage on each site.
2. Data file that contained the number of replicas and information on how they distribute.
3. Optorsim configuration file is the set number of tests and data request randomization procedure.

B. Network Topology Testbed

Fig. 4 shows the EU Data Grid Testbed [6] as the network simulation topology. Site S0 is the CERN (European Organization for Nuclear Research) location. The star denotes a router and the circle denotes a site. Each link shows the available bandwidth between two connecting sites. In this experiment, each Testbed site, excluding CERN, was assigned a computing and storage element. The CERN was allocated a Storage Element to hold all the master files but was not assigned any Computing Elements (CEs). A CE ran jobs that used data files stored on Storage Elements (SEs). Nodes without Computing or Storage Elements acted as network nodes or routers.

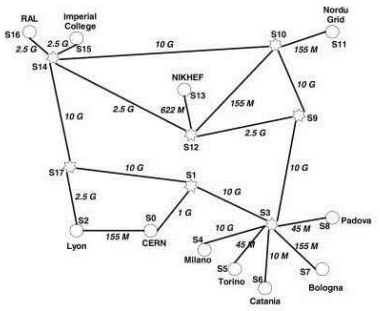


Fig. 4 EU Data Grid Testbed sites and their associated network topology. The numbers indicate bandwidth between the two ending sites in Mbit/s(M) or Gbit/s(G). Stars denote routers and circular nodes denote replica sites.

The mean job execution time is defined as the total time to execute all the grid jobs divided by the number of jobs. ENU (r_{ENU}) is defined as network usage after executing all the grid jobs as follows:

$$r_{ENU} = \frac{N_{remote\_file\_accesses} + N_{file\_replications}}{N_{local\_file\_accesses}}$$

where $N_{remote\_file\_accesses}$ is the number of times the CE reads a file from different SE sites. $N_{local\_file\_accesses}$ is the number of times a CE reads a file from an SE on the same site. For a given network topology, a low value of r_{ENU} indicates that replication is a better optimization strategy than locating another site.

Assuming S0 is the starting point, the Centralized Algorithm places all the data in S0. All sites must retrieve the desired data from S0. Flooding Algorithm starts distributing data through routers and individual site (enclosed by parenthesis) in the following order: (S0), (S2), S1, S17, S3, S14, (S4), (S5), (S6), (S7), (S8), S9, (S16), (S15), S10, S12, (S13), and (S11). The proposed Grass Growing Structure Algorithm replicates the data in the following order: (S0), (S2), S17, S14, S1, S3, (S4), (S5), (S6), (S7), (S8), and (S9).

C. Simulation Results and Discussion

Based on randomization procedures, data requests were issued on EU data grid testbed to measure the performance of the proposed replication method. Figure 5 shows ENU comparison of Grass Growing Structure Algorithm, Flooding, and Centralized Location algorithms. From the outset, the Centralized Location algorithm performs the best, having lowest r_{ENU} while the simulation is still in transient state. As more runs elapse, the graph steadily increases and levels at 0.8 after 500 runs. In the meantime, both Grass Growing Structure Algorithm and Flooding algorithms start with high r_{ENU} but gradually drop to 0.2 after 500 runs.

Fig. 6 shows the mean job time of the three algorithms. The Centralized Location algorithm expends the highest ratio among all algorithms. However, after 100 runs, the ratio begins to level off and reaches a steady state at 500 runs.

Table 1 illustrates comparative Percentage of Storage Filled/Available [11] that is calculated from SE usage multiplied by available SE storage. Notice that the Centralized Algorithm has the highest percentage because data are stored (filled) at only one (S0) available location.

One noteworthy benefit precipitates from this work is reliability and robustness of the underlying network. The proposed Grass Growing Structure Algorithm positions the replicas at appropriate sites not only to reduce the traffic in comparison with the other two algorithms, but also increase reliability and robustness to the system. The distribution can thus be more widely dispersed and reachable by all clients in the network.

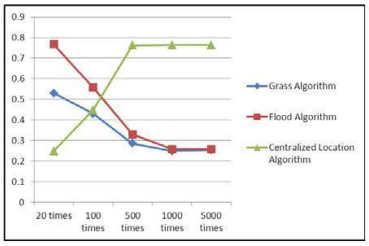


Fig. 5 Effective Network Usage (τ_{ENU})

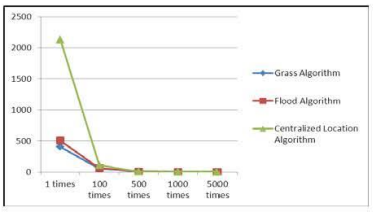


Fig. 6 Mean Job Time

Table 1 Summary of Storage Filled/Available percentage of all replicas by each algorithm

Algorithm	percentage
Grass Growing Structure Algorithm	0.378084
Flood Algorithm	0.480568
Centralized Location Algorithm	0.823892

5. Conclusion and Future Work

In this paper, we propose a replica positioning algorithm called Grass Growing Structure as a means for data replication in a distributed environment. The proposed approach is based on natural grass growing process depending on the amount of water irrigation, whereby data are replicated at the designated location accordingly. This introduces a simple yet effective selection and replication of data over the network. Performance of the proposed algorithm is assessed in comparison with conventional Centralized Location and Flooding algorithms. The results were proved to be satisfactory in terms of ENU and mean job time.

In our future work, we will extend our simulation to incorporate wider network topology testbeds to assess the performance of the Grass Growing Structure.

References

[1] Runqun Xiong, Junzhou Luo, and Aibo Song, "An Effective Replica Location Algorithm Based on Routing-Forward in Data Grid", in *Proceedings of The fifth Annual China Grid Conference*, pp. 31-36, July 16-18, 2010.

[2] David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, and Floriano Zini, "Evaluating Scheduling and Replica Optimisation Strategies in OptorSim", in *the 4th International Workshop on Grid Computing (Grid2003)*, IEEE Computer Society Press, pp. 52-59, November 17, 2003.

[3] William H. Bell , David G. Cameron , Luigi Capozza, A. Paul Millar , Kurt Stockinger, Floriano Zini, "OptorSim - A Grid Simulator for studying dynamic data replication strategies", *International Journal of High Performance Computing Application*, vol. 17, no. 4, pp. 403-416, February, 2003.

[4] David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, and Floriano Zini, "UK Grid Simulation with OptorSim", in *e-Science All-Hands Meeting*, Nottingham, UK, September 2003.

[5] Coles, J., "The evolving grid deployment and operations model within EGEE, LCG and GridPP", in *Proceedings of the First International e-Science and Grid Computing, 2005*, pp. 97-115, July 11, 2005.

[6] Ian Foster, C. Kesselman, "Globus : A meta-computing infrastructure toolkit", *International Journal of Supercomputer Application*, pp. 115-128, 1997.

[7] A. Chervenak, I. Foster, A. Iamnitchi, C. Kesselman, W. Hoschek, P. Kunszt, M. Ripeanu, H. Stockinger, K. Stockinger, and B. Tierney, "Giggle: A Framework for construction Scalable Replica Location Services", *Global Grid Forum*, pp. 1-17, 2001.

[8] Russel, Stuart J., Norvig, Peter (2003), "Artificial Intelligence: A Modern Approach (2nd ed.)", Prentice-Hall, pp. 88.

[9] Wikipedia contributors. (2013, March 20). Flooding Computer Network.Available: http://en.wikipedia.org/wiki/Flooding_computer_networking.

[10] A. Tanenbaum, D. Wetherall, Prentice Hall, "Computer Networks, 5th Edition", pp. 368-370.

[11] David G. Cameron, A. Paul Millar, Caitriana Nicholson, Rubén Carvajal-Schiaffino, Kurt Stockinger, Floriano Zini, "Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using OptorSim", *Journal of Grid Computing*, vol. 2, pp. 57-69, 2004.

VITA

Mr. Worawit Fankam-ai graduated in Bachelor's Degree in Computer Engineering from the Faculty of Engineer , Chiangmai University in 2003.

After graduated from University,He worked on many projects and he has experience on JAVA Programing, PHP Programing, MySQL, Oracle, Geographic Information System(GIS), Mobile Payment Gateway System, Game Online System, Game Online Payment Gateway System, Internet TV System, Windows and Linux System Administrator.