INITIAL FILE-PLACEMENT IN DATA GRID ENVIRONMENT USING GAME THEORY AND FICTITIOUS PLAY

ALOYSIUS INDRAYANTO

UNIVERSITI SAINS MALAYSIA

2008

INITIAL FILE-PLACEMENT IN DATA GRID ENVIRONMENT USING GAME THEORY AND FICTITIOUS PLAY

by

ALOYSIUS INDRAYANTO

Thesis submitted in fulfilment of the requirements
for the degree of
Master of Science

October 2008

ACKNOWLEDGEMENTS

This research is partially funded by the USM grant 1001/PPTM/817006: Grid Computing Cluster - The Development and Integration of Grid Services & Applications.

Thank you to my supervisor Assoc. Prof. Chan Huah Yong, PhD who has spent a lot of his time in supervising my research work and thesis writing.

Thank you to all my friends in School of Computer Sciences, especially Mr. Cheng Wai Khuen in Grid Computing Lab and Mr. Mozaherul Hoque Abul Hasanat in Computer Vision Lab.

Thank you to the School of Computer Sciences, Universiti Sains Malaysia for giving me a conducive working environment when I was working on my research and thesis.

Thank you to my father, mother, sister, grandmother, and grandfather for all their support and help in many things.

Thank you to my old friend Mr. May Ricky Soesanto and Mr. Choirul Muklis who often accompanied me by IRC during my research work and thesis writing.

TABLE OF CONTENTS

Acknowledgements ii

Table of Contents iii

List of Tables vi

List of Figures viii

List of Abbreviations x

List of Symbols xi

Abstrak xii

Abstract xiv

CHAPTER 1 – INTRODUCTION 1

1.1 Problem Statement and Research Objective 4

1.2 Research Scope 5

1.3 Main Contributions 5

1.4 Outline of the Thesis 6

CHAPTER 2 – LITERATURE REVIEW

2.1 Data Management in Data Grid Environment (DGE) 11

2.2 Some Work Related to Data Allocation 12

2.2.1 Work that are Based on Static Equations 12

2.2.2 Work that are Based on System Observations 14

2.2.3 Work that are Based on Decision-Making Algorithms 15

2.3 A Brief Introduction to Game Theory and Fictitious Play 18

2.3.1 Normal Form of the Game 19

2.3.2 Fictitious Play Applied in Game Theory 21

2.4 Summary of the Work and Methods 22

CHAPTER 3 – ALGORITHM DESIGN

3.1 Design Methodology 24

3.2 Model of the Proposed System 25

3.3 The Applied Concept in Agent: Game Theory and Fictitious Play 27

3.4 Payoff Calculation 29

3.4.1 Payoff Concerning the Network Topology 30

3.4.2 Payoff Concerning the Number of Files and Replicas 32

3.4.3 Payoff Concerning the Reliability of the Data Server 35

3.4.4 Payoff Concerning Local Interest and Disinterest 38

3.4.5 Total Payoff and Updated Payoff 38

3.4.5(a) Game Theory Adapted in the Algorithm 40

3.4.5(b) Fictitious Play Adapted in the Algorithm 41

3.4.5(c) Three Primary Sets and the Calculated Beliefs 41

3.4.5(d) The Discount Factors 43

3.4.5(e) The Updated (Final) Payoffs 44

3.4.5(f) The Execution Flow of the Algorithm 45

CHAPTER 4 – IMPLEMENTATION AND VALIDATION OF THE SIMULATOR

4.1 Design of the Simulator 48

4.2 Pseudo-Code of the Proposed Algorithm 52

4.3 Validation of the DGE Simulator Library 55

4.4 Real World Implementation Note 60

CHAPTER 5 – RESULT AND DISCUSSION

5.1 The First Network Topology 63

5.1.1 The Effect of Hop Delay 64

5.1.2 The Effect of Similar Files 67

5.1.3 The Effect of Simultaneous-File-Transfer Capacity (MST) 69

5.1.4 The Effect of Data Server Reliability (MTBF) 71
5.1.5 The Effect of Network Link Speed (Bandwidth) 72
5.1.6 The Effect of Other Parameters for the First Topology 73
5.2 The Second Network Topology 74
5.3 The Third Network Topology 76
5.4 The Fourth Network Topology 77
5.5 Chapter Summary 78
5.6 Chapter Conclusion 79

CHAPTER 6 – SUMMARY AND FUTURE WORK

6.1 Summary 81
6.2 Possible Improvement and Future Work 83
6.2.1 The Concept of File-Content Similarity 83
6.2.2 The Concept of Local Interest and Disinterest 84
6.2.3 The Calculation of the Beliefs 84
6.2.4 The Calculation of the Discount Factors 85
6.2.5 Other Parameters and Considerations 85

References 88

APPENDICES

APPENDIX A – DETAILED PLACEMENT AND QUERY TIME FOR THE VALIDATION OF THE SIMULATOR

APPENDIX B – SPECIFICATION DETAILS OF THE SIMULATION SETUP

B.1 Base Specification of the First Network Topology

B.2 Batch Specification of the Second Network Topology

B.3 Batch Specification of the Third Network Topology

B.4 Batch Specification of the Fourth Network Topology

List of Publications

LIST OF TABLES

		Page
Table 1.1	Overall Comparison between the Current Solutions and the Proposed Solution	**6**
Table 2.1	Comparison Between the Work	**22**
Table 2.2	Features of the Methods	**23**
Table 3.1	Example of Shortest Path Cost	**31**
Table 3.2	Example of Worst Path Cost	**31**
Table 3.3	Summary of How the Individual Parameters Affect the Final Payoff	**45**
Table 4.1	Configuration Entries for the Simulator's Entities	**49**
Table 4.2	Detailed Contents of the Definitions	**50**
Table 4.3	Placement Transfer Time of the First Scenario for Hop Delay 1 and 100	**56**
Table 4.4	Query Transfer Time of the First Scenario	**57**
Table 4.5	Placement Transfer Time of the Second Scenario for Hop Delay 1 and 100	**58**
Table 4.6	Query Transfer Time of the Second Scenario	**58**
Table 5.1	Summary of the Results on the Effect of Hop Delay	**66**
Table 5.2	Summary of the Results on the Effect of Similar Files	**69**
Table 5.3	Detailed Results on the Effect of Data Servers' *MST*	**70**
Table 5.4	Detailed Results on the Effect of Data Servers' MTBF	**71**
Table 5.5	Detailed Results on the Effect of Link Speed between Gates and Data Servers	**73**
Table 5.6	Summary of the Results on the Effect of Other Parameters for the First Topology	**74**
Table 5.7	Summary of the Results for the Second Topology	**75**
Table 5.8	Query Time of the Batch for the Third Network Topology	**76**
Table 5.9	Query Time of the Batch for the Fourth Network Topology	**78**

Table 5.10 Overall Performance of the Algorithm **79**

Table A.1 Placement-Query Scenario for the First Network Topology **92**

Table A.2 Placement-Query Scenario for the Second Network Topology **92**

Table B.1 Detailed Query Start Time of the Base Configuration for the First Network Topology **95**

Table B.2 Detailed Placement and Query Start Time for the Second Network Topology **97**

Table B.3 Detailed Query Start Time for the Third Network Topology **99**

Table B.4 Detailed Placement and Query Start Time for the Fourth Network Topology **101**

LIST OF FIGURES

		Page
Figure 1.1	Classification of Grids and Our Work Area	**3**
Figure 2.1	Component of a Data Grid (DG) in Relation with the Layered Grid Architecture	**10**
Figure 2.2	An Example of Game: Prisoner's Dilemma	**20**
Figure 3.1	An Example of a Simplified Model of the Proposed System	**26**
Figure 3.2	The Roles of an Agent	**28**
Figure 3.3	An Example Graph of a Simple Network Topology	**30**
Figure 3.4	An Example Plot of How the Total Cost (*cost_sum*) will Affect the Payoff (μ_{NT})	**32**
Figure 3.5	An Example Plot of How the Number of Stored Files will Affect the Payoff	**34**
Figure 3.6	An Example Plot of How the Number of Similar Files will Affect the Payoff	**35**
Figure 3.7	The Plot of Reliability (R) versus $MTBF$	**36**
Figure 3.8	An Example Plot of How Reliability (R) Affect the Payoff (μ_{MR})	**37**
Figure 3.9	An Example on How History of Transfers (H_r) Affects the Payoffs	**40**
Figure 3.10	An Example Plot of How Knowledge (h_r^i and b_r^i) Affect the Discount Factor (η_r^i)	**43**
Figure 4.1	Pseudo-Code for the Start of Round (Initialization Phase)	**52**
Figure 4.2	Pseudo-Code for the End of Round (Finalization Phase)	**53**
Figure 4.3	Pseudo-Code for the Placement Phase (Executed for Each File to be Placed)	**54**
Figure 4.4	The Definition of Cumulative Overlapped Transfer (COT)	**55**
Figure 4.5	The First Network Topology Used for the Validation	**56**
Figure 4.6	The Second Network Topology Used for the Validation	**57**
Figure 5.1	The First Network Topology Fed into the Simulator	**63**

Figure 5.2 The Effect of Hop Delay **65**

Figure 5.3 The Effect of Similar Files **68**

Figure 5.4 The Second Network Topology Fed into the Simulator **74**

Figure 5.5 The Third Network Topology Fed into the Simulator **76**

Figure 5.6 The Fourth Network Topology Fed into the Simulator **77**

LIST OF ABBREVIATIONS

API	Application Programming Interface
DBMS	Database Management System
DDBMS	Distributed Database Management System
DFS	Distributed File System
DG	Data Grid
DGE	Data Grid Environment
GSI	Grid Security Infrastructure
LAN	Local Area Network
MTBF	Mean Time Between Failure
OGSA	Open Grid Services Architecture
OS	Operating System
P2P	Peer-to-Peer
SRB	Storage Resource Broker
TCP	Transmission Control Protocol
VO	Virtual Organization

LIST OF SYMBOLS

μ_{NT}	Basic payoff concerning the network topology
μ_{FR}	Basic payoff concerning the number of files and replicas
μ_{MR}	Basic payoff concerning the reliability of a data server
μ_{LI}	Basic payoff concerning the existence of local interest and disinterest
μ_{Total}	Total payoff
μ_{Upd}	Updated payoff
Nr, n_r^i	The number of active ingoing-file-transfer to data servers
Hr, h_r^i	History of transfers
Br, b_r^i	Belief of others' behaviors
η_r^i	Discount factors

PENEMPATAN-FAIL PERMULAAN DALAM PERSEKITARAN GRID DATA DENGAN MENGGUNAKAN TEORI PERMAINAN DAN LAKONAN REKAAN

ABSTRAK

Peruntukan data merangkumi peletakan atau migrasi data. Ini merupakan salah satu pertimbangan utama dalam reka bentuk sistem perkongsian data teragih. Penggunaan persamaan-persamaan yang hanya bergantung kepada parameter-paremeter masukan statik kebanyakannya akan menyebabkan hanya pelayan-pelayan data terbaik terpilih. Ini akan membebankan pelayan-pelayan data yang terpilih.

Tesis ini mencadangkan satu algoritma untuk mengoptimumkan penempatan fail pertama dalam persekitaran grid data. Algoritma tersebut mempertimbangkan kandungan fail dan keadaan sistem sebagai parameter-parameter asas. Lakonan rekaan, sejenis peraturan pengajaran berasas kepercayaan, disesuaikan untuk membantu dalam penjangkaan keseluruhan perlakuan penempatan sistem. Teori permainan, sejenis teori untuk penyelesaian konflik, digunakan ke atas kepercayaan dan sejarah pemindahan fail sistem untuk mewujudkan pengimbangan beban.

Disebabkan oleh sifat semula jadi sistem grid dan kekangan pelantar pengujian kami, algoritma tersebut dicadang untuk diuji melalui simulasi. Oleh itu, satu pensimulasi berturutan (bersiri) yang berasaskan masa diskret telah dibangunkan. Merujuk kepada hasil simulasi, penggunaan kepercayaan dan sejarah umumnya mencapai prestasi pertanyaan yang lebih baik dalam kebanyakan kumpulan simulasi dengan 0.443 saat (0.137%) hingga 3.188 saat (2.067%)

lebih pantas berbanding tanpa penggunaan kepercayaan ataupun sejarah. Walau bagaimanapun, dengan hanya penggunaan sejarah dapat mencapai prestasi terbaik dengan 5.462 saat (3.161%) lebih pantas berbanding tanpa penggunaan kedua-duanya. Sementara itu, dengan hanya penggunaan kepercayaan mencapai prestasi yang paling buruk di dalam salah satu simulasi dengan 6.976 saat (4.344%) lebih lambat berbanding tanpa penggunaan kedua-duanya. Prestasi-prestasi ini adalah disebabkan daripada cara algoritma tersebut memperhatikan dan menduga perlakuan sistem keseluruhannya.

INITIAL FILE-PLACEMENT IN DATA GRID ENVIRONMENT USING GAME THEORY AND FICTITIOUS PLAY

ABSTRACT

Data allocation comprises data placement or migration. It is one of the main considerations in the design of a distributed data sharing system. The use of equations which only depends on mostly static input parameters would cause only the best data servers to be selected. It would eventually overload the selected data servers.

This thesis proposes an algorithm for optimizing initial file-placement in Data Grid Environment (DGE). The proposed algorithm considers the content of the files and the condition of the system as its basic parameters. Fictitious play, a belief-based learning rule, is adapted to help predicting the overall placement behavior of the system. Game theory, a theory of conflict resolution, is applied on top of the belief and the system's file-transfer history to provide load balancing.

Due to the nature of the Grid system and the limitation of our testbed, it was decided to test the proposed algorithm using simulation. Therefore, a sequential (serial), discrete-time based simulator was developed. From the simulation results, the use of both belief and history generally achieved better query performance in most of the simulation groups with 0.443 seconds (0.137%) to 3.188 seconds (2.067%) faster than using none. However, the use of only history was able to achieve the best performance with 5.462 seconds (3.161%) faster than using none. Meanwhile, the use of only belief achieved the worst performance in one of the simula-

tions with 6.976 seconds (4.344%) slower than using none. These results are due to how the algorithm observes and predicts the overall system behavior.

CHAPTER 1

INTRODUCTION

Data management in a distributed data sharing system is one of the active research topics nowadays. *"A distributed system is a collection of independent computers that appears to its users as a single coherent system"* (Tanenbaum and van Steen, 2002). The machines (computers) in a distributed system can be utilized as a pool of resources. A user may subscribe for some resources from the resource pool and use virtualization to deploy the OSes and environments needed by the user's applications. Virtualization allows a machine to run many different Operating Systems (OSes) and environments (called guests) on top of the existing OS (called host) (Vallée et al., 2008). Virtualization isolates the guests from the host (Vallée et al., 2008; Uhlig et al., 2005). A distributed data sharing system can isolates the users from the real, physical, storage implementation. Hence, the system provides the users with storage virtualization.

Two examples of distributed system intended for data sharing and management are Distributed Database Management System (DDBMS) and Data Grid Environment (DGE). The user of such systems does not need to know in which machines the requested data is actually stored (the systems provide data transparency). Such kind of system also distributes the stored data among the available machines. Hence, it balances the load over the machines. Two of the key features of storage virtualization are storage allocation and the mapping of the stored object (data) (Milligan and Selkirk, 2002). Therefore, DDBMS and DGE can be grouped as a system which provides virtualization of data management.

"A Grid is an infrastructure that allows for coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" (Foster et al., 2001). The vision of Grid is to provide an environment in which the users can easily gain access to resources such as computing power, storage, and applications without the need to know how those resources were deployed. The key concept of Grid is in analogy with the concept of current power grid in which users can subscribe for the resources they need (Foster and Kesselman, 1999) (some people call it as "Utility Computing"). Grid provides the users with resources which are not available for them locally (Jacob et al., 2005). Negotiation between the users (consumers) and the owner of the resources (providers) can be arranged to determine what resources the users can use.

Grids can be classified according to their primary purposes (Mambretti, 2006; Foster et al., 2001). A Sensor Grid is a Grid which is a collection of sensors. A Computational Grid is a Grid which provides the users with high computational power. A Data Grid (DG) is a Grid which provides the users with storage. Please refer to Figure 1.1 for more details on the area of classification and some work which have been done on the area.

A DG is a specialization and extension of the Grid system which emphasizes on sharing a large amount of data between its users (Chervenak et al., 1999). A DGE provides its users with an integrated view of the storage machines as if they were a single large storage (Chervenak et al., 1999; Jacob et al., 2005). Hence, the actual location of the machines are transparent to the users (Minoli, 2005). In a DGE, the machines can be owned by many different organizations and individuals. Due to the different policies of the organizations and individuals, a DGE would be more complex to manage than a DDBMS.

Data allocation is the process of allocating one or more *data fragments* into the machines in the system (Hababeh, 2005). The term *data fragment* may refer to an individual file or to a more

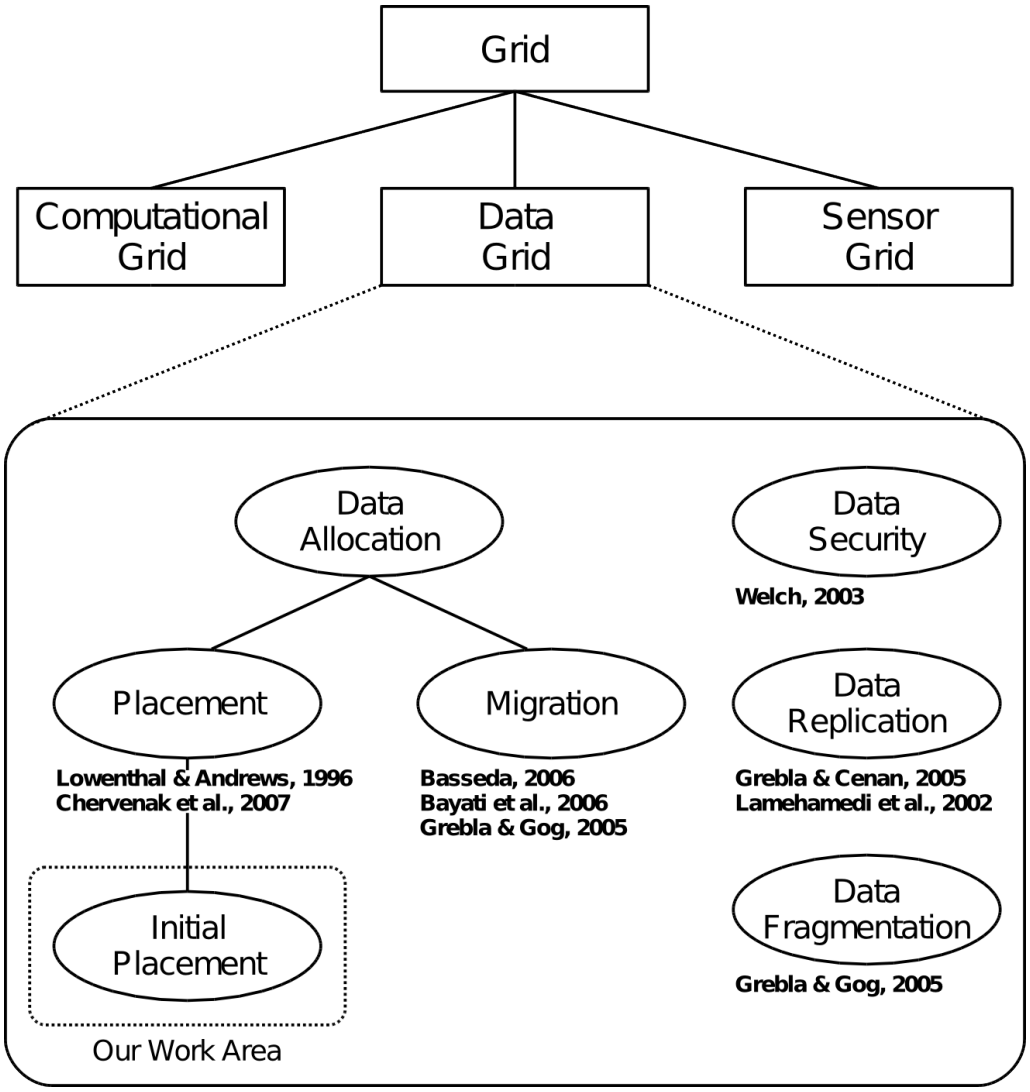

Figure 1.1: Classification of Grids and Our Work Area

complex structure. For example, in the case of horizontal, vertical, or hybrid fragmentation in a relational database, a data fragment will be the rows or the columns of a table. The main purpose of data allocation is to allocate the data fragments optimally so that the overall performance and reliability of the system are maximized while the implementation cost is minimized (Özsu and Valduriez, 1999). Data allocation consist of the placement or migration of data fragments into and between the machines (Indrayanto and Chan, 2008).

Some current solutions in data-related problem are implemented mainly based on static equations (equations of which outputs depend only on mostly static input parameters). Examples of such algorithms are the Optimal algorithm by L.C. John, NNA (Near Neighborhood

Allocation) (Basseda et al., 2006), and the BGBR (Bayati et al., 2006). Some other algorithms embed artificial intelligence (decision-making algorithms) and learning. The EFA (Evolutionary Fragmentation and Allocation Algorithm) (Grebla and Gog, 2005) adapts genetic algorithm. The FNA (Fuzzy Fragment Allocator) (Basseda, 2006) utilizes fuzzy logic. Meanwhile, the work by Grebla & Cenan (Grebla and Cenan, 2005) and Khan & Ahmad (Khan and Ahmad, 2005) adapt game theory. The work by Chervenak et al. (Chervenak et al., 2007) finds optimal locations for placing data by examining the workflow that will be applied to the data. However, the work was specialized for scientific applications. Finally, the work by Lowenthal & Andrews (Lowenthal and Andrews, 1996) implements an adaptive data placement for distributed-memory programming.

1.1 Problem Statement and Research Objective

The fore-mentioned algorithms work on already stored data or on specialized area. They try to improve the overall access performance by migrating or replicating the data fragments. The algorithms consider the access patterns of the data fragments and the properties of the destination locations (data servers). However, most of them do not consider the internal properties of the data fragments such as the types and the contents of the fragment. The algorithms also do not consider the possible existence of local interest and/or disinterest (organizational policy) to certain types of data fragments. Other factors such as the overall placement behaviors and dependencies are also not considered.

The fore-mentioned algorithms also do not explicitly specify how newly created data fragments would be initially allocated (the first-time placement). If the data fragments were placed in better locations from the start, the average future work to migrate them can be made less. Possible occurrence of ping-pong effect would become fewer. Hence, the network bandwidth wasted for re-migrating the fragments can be reduced.

The main objective of the research would be: to find better starting locations for first-time (initial) placement of files in DGE via dynamic load balancing. The research aims to distribute the newly created files to the data servers in the system while still considering the properties of the system and data as well as the possible existence of organizational policy.

1.2 Research Scope

The target domain of the research is on a DGE. Therefore, the research proposed and algorithm that work on a collection of individual files. The proposed algorithm aims to find better location for initial file-placement in DGE. Query performance (speed or time) would be the main output parameter that will be considered to evaluate the algorithm.

The proposed algorithm considers parameters such as the properties of the files, the properties of the network and machines, the possible existence of local interest and disinterest, and the possible existence of replicas of the files. The proposed algorithm would utilize game theory (Turocy and von Stengel, 2001; Ozdaglar, Spring 2005; Brandenburger, 2002; Slantchev, 2004-2007) and fictitious play (Ozdaglar, Spring 2005). The work by Marden (Marden et al., 2005) is an example of work which combines game theory and fictitious play.

1.3 Main Contributions

The main contributions of the thesis are as follows:

- an algorithm for finding better initial locations for initial file-placement in DGE;
- an application of game theory and fictitious play for solving the initial file-placement problem; and
- a DGE simulator for simulating and testing the proposed algorithm.

Table 1.1 shows the overall comparison between current solutions and the proposed solution. The main difference is that the proposed solution would apply decision making algorithms on top of a more broad range of parameters to provide a dynamic load balancing for initial file-placement.

Table 1.1: Overall Comparison between the Current Solutions and the Proposed Solution

Current Solutions	Our Proposed Solution
most work on already stored data	work on initial placement of data
access pattern is the main consideration	considers the internal properties of the data (file) and the properties of the destination location
most do not consider organizational policy	considers organizational policy (via local interest and disinterest)
do not consider the overall placement behaviors and dependencies	considers the overall placement behaviors and dependencies
utilizes static equations, artificial intelligence or learning	utilizes static equations, decision making algorithms, and learning
some can provide load balancing	provides adaptive load balancing
some are specialized for certain areas	could be adapted to a more broad area

1.4 Outline of the Thesis

Chapter 1 describes briefly about the area of the research, some current solutions and their problems, research scope, description of the proposed solution, main contributions, and the outline of the theses.

Chapter 2 will explain about DGE and work related to data allocation in more detail. Some possible solutions to the problem and the basic of game theory and fictitious play will be also explained briefly.

Chapter 3 will explain about the design of the proposed algorithm. All the concept, equations, and flows will be explained in detail.

Chapter 4 will explain about the implementation of the algorithm. Due to the complexity of the system, it would be complicated to test the algorithm in a real-world system. Hence, the algorithm was tested using simulations. The design of the simulator will be explained in detail as well as the simulation processes.

Chapter 5 will discuss about the result of the simulations. Graphics and tables generated from the simulation results will be presented. Summary of the results will be explained in detail.

Chapter 6 will summarize the whole thesis. Possible improvements and future work regarding the proposed algorithm will be also outlined in the chapter.

CHAPTER 2

LITERATURE REVIEW

Data Grid (DG) is implemented mainly to support data-intensive computing. Two core services in a Data Grid Environment (DGE) are *data access* and *metadata access*. Higher level services are implemented on top of those two core services.

The *data access* service provides the mechanism needed to access, manage, and transfer the data stored in low-level storage system (Chervenak et al., 1999). The low-level storage system may be implemented using many different hardware, operating system, and file system. The data access service provides Grid applications with a uniform Application Programming Interface (API). Hence, the applications need not to be aware of how the underlying storage system is working.

The *metadata access* service provides the mechanism needed to access and manage informations about the stored data (Chervenak et al., 1999). The metadata may describe the data itself, such as the size, the type, and the content of the stored data. The metadata may also contains information about the fragmentation and replication scheme applied to the data. An application can query the metadata catalog to find the needed data.

Due to the fact that a DG is a Grid emphasizing on data sharing, a DGE inherits many of the Grid properties. A DGE has the following main properties :

- the machines belong to many different individuals and organizations (Laszewski and Wagstrom, 2004);

- aggregates storages into a larger virtual storage (Minoli, 2005);
- provides metadata service (Chervenak et al., 1999);
- location transparency is an important part of the implementation (Minoli, 2005);
- security is provided by Grid Security Infrastructure (GSI) (Jacob et al., 2005; Minoli, 2005; Laszewski and Wagstrom, 2004);
- enforce the policy implemented by the Virtual Organization (VO), example: via the Open Grid Services Architecture (OGSA) standard (Minoli, 2005).

Figure 2.1 depicts the components of a DG in relation with the layered Grid architecture. In the figure, a DG aggregates the storages provided by Distributed Database Management System (DDBMS) and Distributed File System (DFS) to form a larger virtual storage. The storage nodes (data servers) in a DGE can be owned by many different organizations and individuals. Centralized administration is not mandatory in a DGE. However, a specialized node such as Storage Resource Broker (SRB) can be provided to ease the system management. The communication between the nodes in the system is done by using Peer-to-Peer (P2P) technology. A DGE would offer the users with an acceptable level of security due to the use of GSI certificate authentication. A DG is an extension of the Grid, hence, the VO policies are automatically enforced by the underlaying Grid middleware and toolkit. Extending a DG would be similar with extending a Grid. Adding a new storage nodes would involve standard certificate exchange and user mapping. Data transparency, availability, and reliability can be also provided by a DGE due to the fact that a DGE also allows for automated resource discovery, catalog management, and data replication.

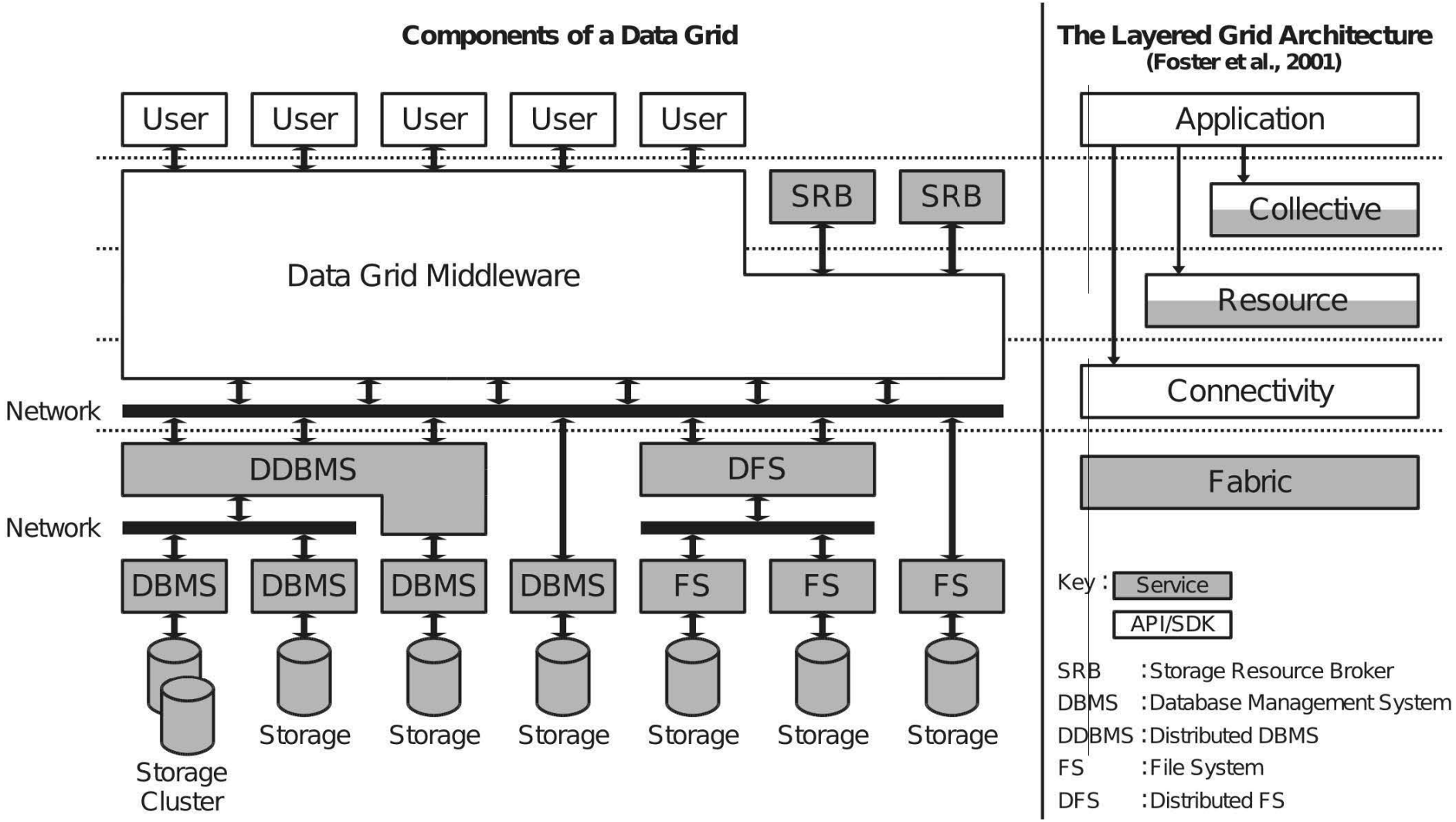

Figure 2.1: Component of a Data Grid (DG) in Relation with the Layered Grid Architecture

2.1 Data Management in DGE

The most common data-related operations in a DGE are:

- data security;
- data replication;
- data fragmentation; and
- data allocation.

Data security concerns with the fact that only authorized users may access the data. In a DGE, data security is based on the GSI (Welch et al., 2003). Data replication concerns about how to manage replicas. Data fragmentation concerns about how to fragment large data. Data allocation concerns about how to allocate the data into the storage nodes.

Both data replication and data fragmentation are related to data allocation. After a data fragment is replicated, the replica will need to be allocated. After a large data fragment is fragmented, the fragments will also need to be allocated. Sometimes, it maybe also necessary to move an allocated data to another storage node to improve performance. Hence, data allocation comprises data placement or data migration (Indrayanto and Chan, 2008).

The work explained in this thesis is about initial data-placement. The focus of this thesis is about where to place a newly created data. The target domain of the application area is on a DGE. Hence, the algorithm described in this thesis works on a collection of individual files. However, it would be possible to adapt the algorithm for other types of distributed data sharing system.

2.2 Some Work Related to Data Allocation

Several methods have been used in the current solutions for data allocation problem. They are:

- static equation;
- system observation;
- artificial intelligence (decision-making algorithms):
  - fuzzy logic;
  - genetic algorithm;
  - game theory; and
  - fictitious play.

2.2.1 Work that are Based on Static Equations

The Optimal algorithm by L.C. John, the Near Neighborhood Allocation (NNA) algorithm (Basseda et al., 2006), and the BGBR (Bayati et al., 2006) are algorithms which are based on static equations.

The Optimal associates a data fragment with counters. One counter for each node (data server). Each time a node request for a data fragment, the counter that corresponds to the requesting node is incremented. If the total number of remote accesses to a certain data fragment is larger than the total number of local accesses to the same data fragment, the fragment is migrated to the corresponding remote node. However, according to the author, if the access counter of a data fragment changes rapidly, the data fragment will be migrated over and over again (called *ping-pong* or *yo-yo* effect). This would increase the overall response time and delay as the system is busy migrating the fragment.

The NNA tries to overcome the weakness of the Optimal algorithm. Instead of moving the data fragment directly to the final destination node, the NNA moves the fragment step by step. The NNA considers the network topology to determine the possible routes from the originating node to the final destination node. Hence, the NNA would prevent over-migration of the data fragment. According to the authors, the data fragment will be finally stored in a node which has the average access cost. However, the NNA may still cause wasted movements as it converges slowly by taking many hops.

The BGBR, of which name came from the abbreviation of the authors' names, tries to further improve the NNA algorithm. The BGBR would converge faster because it is aware of the complete network topology. The BGBR models the network topology by using graph. A matrix which defines the shortest paths from and to every node in the system is precalculated by the BGBR. Using access counters and shortest path matrix, BGBR finds a node which is close to all the other nodes that accessed the data fragment. The combination between the number of accesses to a data fragment and the locations of the nodes (users) who accessed the fragment is defined as *access pattern*.

Static equations are generally used as the basis for the more advanced methods. Many parameters in a DGE can be considered as static. The capacity and reliability of a storage node (data server), the bandwith of the communication network, and the VO policy would not change frequently. Hence, using static equations would be a good basis for solving data placement problem in a DGE.

Depending on what are considered as the input parameters, static equation might be able to capture the dynamic nature of a system in a limited way. Static equation may not be able to capture the dynamic interaction between the many nodes in a system. For example, a static equation for selecting a data server based on the storage capacity might always select the data

server with the largest storage until the remaining free space falls below the second best data server. This selection would happen not only from one source node, but from all nodes that use the same equation. This would cause many data stored in the selected data server. In the future, many queries to the data server might cause network congestion. Embedding some kind of intelligence would be necessary to implement an algorithm which is able to capture the dynamic interaction between the participants (the nodes which use the same algorithm).

2.2.2 Work that are Based on System Observations

Chervenak et al. (Chervenak et al., 2007) suggests that workflow management services together with the VOs can provide some hints to data placement services. A data placement service analyzes which data fragments (such as files) that will be used by a given workflow. Based on the analysis, the service requests the system to place (move or replicate) the fragments to the storage system associated with the workflow execution system. According to the authors, this would allow the system to asynchronously prestaging the data before the workflow is executed. Hence, the execution performance of the workflow can be improved.

The work by Lowenthal & Andrews (Lowenthal and Andrews, 1996) implements an algorithm (named Adapt) which gathers information regarding communication and computation performance in distributed-memory machines. Using the gathered information, Adapt performs some calculations to determine in which nodes (machines) the data needs to be placed for the next program iteration (hence, migrating or replicating the data). According to the authors, the Adapt is able to perform adaptive run-time data placement during program execution to ensure that the needed data is accessible by the program with less overhead. However, Adapt itself would also incur additional calculation overhead.

The two algorithms above observe how the system runs or performs. They collect and analyze information from the system. The analysis are static because the same algorithm and equations are used. However, the results are dynamic (depend on the condition of the system). Hence, such kind of algorithms can be said as adaptive or capable of learning.

A DGE is a dynamic environment. An individual or organization can join and leave the system at any time. This would imply that at one time, many new data may need to be placed. However, at another time, maybe only few new data need to be placed. The types and contents of the data may also vary greatly. The selected storage nodes (destination data servers) would also vary greatly due to the types and contents of the data. Therefore, system observation would be an important method for solving data placement problem in a DGE.

2.2.3 Work that are Based on Decision-Making Algorithms

Heavily accessed data fragments may have their access patterns changing frequently. This would cause the fragment to be migrated over and over again (oscillation). Due to a migrating data fragment is locked and thus cannot be accessed by users, oscillation may degrade the system performance. The Fuzzy Fragment Allocator (FNA) algorithm (Basseda, 2006) is an algorithm which adapts fuzzy logic to detect possible oscillation. The FNA is a supplementary algorithm. When an algorithm decided that a data fragment should be moved, the FNA will check if the movement would cause oscillation. The FNA considers the access patterns, differentiated access patterns, and distance vectors of the data fragments to detect oscillation. If an oscillation is likely to occur, the affected data fragment would not be moved.

Fuzzy logic is a decision-making algorithm. Fuzzy logic maps input variables to output variables similar to the way humans think. Fuzzy logic uses linguistic rules rather than numerical rules (Kulkarni, 2001). For example: instead of only two conditions, *empty* and *full* (binary

0 and 1), fuzzy logic can also represent *almost empty*, *moderate*, and *almost full*. Hence, fuzzy logic would be able to make "smarter" decisions.

Fuzzy logic uses *membership functions* to map between the measured numerical values to their linguistic values. Calculations are done based on the linguistic values rather than the real numerical values. The final results are then mapped back to their numerical values. The membership functions can be made static or dynamic. Information from the current system condition can be applied to generate dynamic membership functions. Hence, the fuzzy system would become capable of learning (Khuen et al., 2005).

While fuzzy logic is now widely adopted in many fields, it is not specifically designed for solving conflicts. Data placement is competition (and conflict) to find the best locations. Therefore, it may be better to use other method which is designed more towards conflict resolution.

Selecting the best location (node) is similar with the evolution of nature known as the *"Survival of the Fittest"* (*The Complete Work of Charles Darwin Online*, 2007) because the available nodes are limited. Based on this idea, the Evolutionary Fragmentation and Allocation Algorithm (EFA) (Grebla and Gog, 2005) adapts genetic algorithm to find the best node. The EFA also models the network topology using graph. In EFA, the population is defined as the set of tuples (data fragments) to be distributed to the nodes. A chromosome (a potential solution) is a string of constant length of which genes indicate to which nodes the tuples belong to. The *fitness function* is a function which minimize the cost sum between the number of requests from particular nodes and the corresponding edge cost. The EFA performs recombination and mutation to obtain the best chromosome (solution).

Genetic algorithm transforms solutions into better solutions (Poli et al., 2008). The process is done by *crossover* and *mutation*. *Crossover* generates a child solution by combining

random parts from two parent solutions. *Mutation* generates a child solution by changing parts randomly from one parent solution. The parent solutions are selected from the collection of solutions based on fitness values. The selection is done by applying a function to score the fitness values of the solutions. The two best solutions are selected to become the parents. Hence, genetic algorithm is a learning algorithm which mimics the evolution of nature in which only the best individuals (solutions) will prevail.

While genetic algorithm is already adapted in many fields, it may not be suitable for every problem. Genetic algorithm needs many iteration before reaching the final solution. This would imply that many calculations need to be done. The randomness factor in genetic algorithm may also cause the solution to be non-deterministic. Given the same input parameters, the final result may not be the same between calculations. Hence, it may be not an ideal method for solving data placement problem.

Selecting the best location (node) is also a competition because the resource of the destination nodes are limited. Hence, the data fragments are "competing" to get the best node. Game theory is one of the formal studies of conflict resolution (Turocy and von Stengel, 2001). Grebla and Cenan (Grebla and Cenan, 2005) adapts game theory to solve the problem of *"if there are N data fragments which are spread into some nodes, how many nodes (P) of which data fragments need to be replicated into another M nodes?"*. According to the authors, no replication and full replication result in poor performance. Partial replication of P nodes in M nodes in which $P,M < N$ provides good performance. There would be some pairs of P and M that make the system perform very well. This problem is similar to the "Tragedy of the Commons" (known also as the "Santa Fe Bar Problem"). The "Tragedy of the Commons" is a problem of sharing resources of limited capacity which satisfies a non-cooperative, repeated game. The authors' main idea is: an agent will replicate the data fragment if a node is under crowded (not yet reached its optimal capacity) and not to replicate if the node is over crowded. The authors

also stated that the agents would become more adaptive (hence, making better decisions) when fictitious play are applied together with game theory.

Game theory is the theory of conflict and cooperation. It is one of the formal studies of decision making. In game theory a player will try to maximize its payoff by executing its best strategy according the current condition (Ozdaglar, Spring 2005; Slantchev, 2004-2007; Turocy and von Stengel, 2001). Game theory allows both cooperation and non-cooperation between the participating players.

Data placement is related to competition and conflict. In data placement, both cooperation and non-cooperation may be needed to obtain the best result with an acceptable amount of overhead. Hence, game theory can be a good method for solving data placement problem in a DGE.

The key concept of fictitious play is the use of beliefs to model the opponents' behaviors (Ozdaglar, Spring 2005). The beliefs are updated after each round (iteration). Fictitious play can be applied to implement learning in game theory (Lambert-III et al., 2005; Marden et al., 2005).

Data placement in a DGE can involve many, dynamically-joining nodes. Examining the actions of all the nodes would add many calculation and communication overhead to the algorithm. Therefore, the use of fictitious play to predict the actions (behaviors) of other nodes can be a good method for solving data placement problem in a DGE.

2.3 A Brief Introduction to Game Theory and Fictitious Play

The concepts of game theory apply whenever the actions of the players are interdependent (Turocy and von Stengel, 2001). The main components which form a game are:

- a finite set of players;
- available pure strategies (actions or moves) for the players; and
- payoff functions.

In game theory, all players are rational and will not deviate from their best strategies (Turocy and von Stengel, 2001).

A cooperative game is a game between coalitions (groups) of players. In a cooperative game, the main concerns are about how to form a coalition and how to apply its strategies in order to maximize the mutual payoff. The emphasize of a cooperative game is on the outcomes when some players meet together in many different combinations (Brandenburger, 2002).

A non-cooperative game is a game between players. In a non-cooperative game, players apply their own strategies based on their preferences in order to maximize their personal payoffs. The detail in a non-cooperative game is in all the available strategies that can be taken by the players (Brandenburger, 2002).

2.3.1 Normal Form of the Game

In the *normal form* of the game (also known as the *strategic form* of the game), a matrix is used to represent the game. The payoffs are put in the matrix. All participants (players or groups of players) move simultaneously without any prior knowledge about the opponents' moves (Ozdaglar, Spring 2005).

Another form of the game is called the *extensive form*. In this form, the game is represented using a tree. In this kind of game, participants make their moves sequentially (Ozdaglar, Spring 2005). Due to more than one nodes in a DGE may perform actions simultaneously, the extensive form of the game would not be suitable.

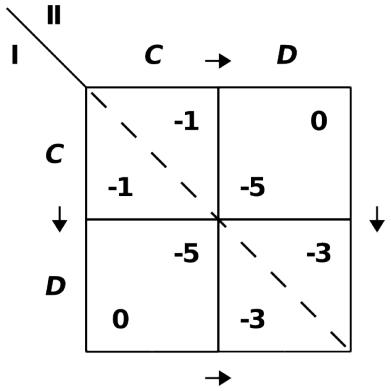

Figure 2.2: An Example of Game: Prisoner's Dilemma

Figure 2.2 (Turocy and von Stengel, 2001) displays an example of a game in its normal form. The game is called the "Prisoner's Dilemma". In the figure, **I** and **II** are the players, **C** and **D** are the strategies, the numbers (**0**, **-1**, **-3**, and **-5**) are the time (in years) the prisoners (players) will spend their live in prison if they choose the particular strategy. In an implementation, instead of static, the numbers in the matrix may actually calculated using equations (Khan and Ahmad, 2004, 2005; Ozdaglar, Spring 2005).

In the storyboard of the above game, two persons had committed crime and caught by police. However, the police does not have enough visible evidence of their crime. Hence, the police interrogate them in two separated rooms. In the process of interrogation, finally the two suspects are given two choices:

- comply (**C**), which means keep silent and cover (protect) each other;
- defect (**D**), which means become the witness of others' crime.

The police states that if they want to be witness of others' crime, they will get reduction in their penalties (less time spent in prison).

The properties of the above game are:

- the strategy **D** dominates the strategy **C** (please refer to the arrows in the figure);

- the game is symmetric: the game stays the same even if the players are exchanged corresponding to the reflection line (shown as dashed line in the figure);
- the game has one equilibrium point which is **(D, D)**.

The strategy **(D, D)** is an equilibrium point because no rational player will choose a dominated strategy (Turocy and von Stengel, 2001). In the example game, the players do not trust each other. Hence, they choose the strategies that will give less worse outcome.

2.3.2 Fictitious Play Applied in Game Theory

Fictitious play is one of the earliest and simplest learning rules. Fictitious play relies on the history of previously played game to form the beliefs. Fictitious play assumes that there is more than one best strategies (all give the same payoffs) that can be chosen by a player and the opponents. Hence, the player and the opponents may alternately choose between the available best strategies (Ozdaglar, Spring 2005).

Below is a simple example of how fictitious play may be applied together with game theory:

- there are one player and one opponent;
- currently, despite the other dominated strategies, there are two best strategies (both give the same payoffs), **A** and **B**;
- from the perspective of the player, the opponent may choose either **A** or **B**;
- choosing the same strategy with the opponent will lower the payoff.

In the first game, both **A** and **B** have the same probability (50%) to be chosen by the opponent. Hence, the player will just choose either **A** or **B** randomly. Assume that the opponent then chooses **A**. In the second game, the player will have the history of the previous game stating that the opponent has chosen **A**. This will update the belief of the player. In the perspective

of the player, the opponent will now most probably also choose **A**. Therefore, the player now should choose **B**.

2.4 Summary of the Work and Methods

Table 2.1 summarizes the comparison between the fore-mentioned work. One part which seems to be missing is the part which explicitly determines how a newly created data would be allocated (initial data-placement).

Table 2.1: Comparison Between the Work

Algorithm	Area	Purpose	Method
Optimal	Data migration	Finding better locations for the data fragments	Static equations
NNA	Data migration	Finding better locations for the data fragments	Static equations
BGBR	Data migration	Finding better locations for the data fragments	Static equations
FNA	Supplementary	Preventing oscillation of movements	Fuzzy logic
EFA	Data migration	Finding better locations for the data fragments	Genetic algorithm
Adaptive Agent	Data replication	Finding whether it will be beneficial to replicate or not	Game theory and fictitious play
Asynchronous Prestaging	Data placement (but more towards migration)	Finding better locations for the data fragments	Scientific-application workflow analysis
Adapt	Data placement (but more towards migration)	Finding better locations for the data fragments	Distributed-memory machines communication and computation performance analysis

Table 2.2 summarizes the features of the methods used by the work. Data placement is a competition for finding the best location with the need of predicting the overall dynamic-interaction between the participants. Therefore, using static equation and system observation as the base, adopting game theory and fictitious play would be a good method for solving data placement problem in a DGE.

In game theory all participants (players) are assumed to be rational (they will not deviate from their strategies and targets). A DGE is an example of a system which is rational. Today's data servers would also have enough memory. Hence, the disadvantages of game theory and fictitious play would not be major problems.

Table 2.2: Features of the Methods

Method	Advantages	Disadvantages
Static equation	Simple, directly predictable result	Not able to capture the dynamic interaction of the participants
System observation	Able to capture the dynamic properties of the data and system condition	May still not able to capture the dynamic interaction of the participants
Fuzzy logic	Capable of learning and making a more find-grained decision	Not specifically designed for conflict resolution
Genetic algorithm	Mimics the evolution of nature to generate better solutions	Needs many iterations, the randomness factor may produce non-deterministic results
Game theory	It is the theory of conflict and cooperation	All participants (players) are assumed to be rational
Fictitious play	Able to model the opponents behaviors and improving the model (learning), applicable with game theory	Need to maintain (store) history in order to calculate the beliefs

The next chapter will explain how static equations, system observation, game theory, and fictitious play are adapted in the algorithm. The design of the algorithm will be described in detail.

CHAPTER 3

ALGORITHM DESIGN

This chapter summarizes the design methodology of the proposed algorithm. The model of the proposed system and the applied concept will be described briefly. Payoff calculations and the adaptation of game theory and fictitious play will be explained in detail, followed by the execution flow of the algorithm.

3.1 Design Methodology

A simplified model of the system is needed because it would be complicated to perform calculation and measurement on a real, complete system. The model was created based on the structure of a real Data Grid Environment (DGE). In a DGE, there are specialized nodes with limited centralizing roles such as proxy server and resource broker. The model was prototyped and verified by using simulation.

The static and dynamic (observed) system's properties are transformed into basic payoffs. There are four basic payoffs. Each payoff is related to a specific property of the system. The first payoff represents the condition of the network (topology and bandwidth). The second payoff represents the storage and processing capacity of the storage nodes (data servers) as well as the existence of replicas. The third payoff represents the reliability of the data servers. The fourth payoff represents the Virtual Organization (VO) policy via the existence of local interests and/or disinterests. These four basic payoffs are combined into one total payoff.