REPLICA PLACEMENT IN PEER-TO-PEER SYSTEMS

Thesis submitted for the degree of Doctor of Philosophy

By:

WAN SURYANI WAN AWANG

School of Computer Science & Informatics

Cardiff University

2016

ACKNOWLEDGEMENTS

First and foremost I would like to thank Allah (SWT), for the strength and enables me in completing this thesis. Warmest gratitude to my supervisor, Professor Omer F. Rana, for the support, constructive comments, guidance and patience throughout my study. I would like to extend my gratitude and acknowledgements to Dr Andrew C. Jones, Professor Stuart M. Allen, Professor David W. Walker, Dr Ioan Petri, Professor Mustafa Mat Deris, Associate Professor Dr Fadhilah Ahmad, Dr Mona Ali, Dr Asma Saidi, Dr Izan Jaafar, Dr Ahmad Nazari Mohd Rose and Dr. Zarina Mohamad for their insightful comments and constructive criticisms of the work. I am very thankful to Mrs Helen Williams for the administrative problems I have been putting her through.

This dissertation is dedicated to my husband Mohd Yusoff Mustafa and my daughters Yasmin Syauqina, Yasmin Syahirah, Yasmin Suraya, Yasmin Syafiqah and Maryam Aqilah for their patience and support during the time of hardship and for constantly inspiring me in the quest of knowledge. Without the support and security from my family members, it would be impossible for me to complete this work. This dissertation is also dedicated to the memory of my beloved parents. Their memories gave me the nourishment to fulfill my dreams.

I am indebted to my University, University Sultan Zainal Abidin (UniSZA) and my Faculty, Faculty of Informatics and Computing (FIK), for entrusting me with the opportunity to pursue this study. I thank the Malaysian government for funding my PhD in Cardiff University. Last but not the least, I thank everyone who has contributed in the completion of this thesis, directly or indirectly.

ABSTRACT

In today's distributed applications, replica placement is essential since moving the data in the vicinity of an application will provide many benefits. The increasing requirements of data for scientific applications and collaborative access to these data make data placement even more important. Until now, replication is one of the main mechanisms used in distributed data whereby identical copies of data are generated and stored at various distributed sites to improve data access performance and data availability. Most work considers file's popularity as one of the important parameters taken into consideration when designing replica placement strategies. However, this thesis argues that a combination of popularity and affinity files are the most important parameters which can be used in decision making whilst improving data access performance and data availability in distributed environments. A replica placement mechanism called Affinity Replica Placement Mechanism (ARPM) is proposed focusing on popular files and affinity files. The idea of ARPM is to improve data availability and accessibility in peer-to-peer (P2P) replica placement strategy. A P2P simulator, *PeerSim*, was used to evaluate the performance of this dynamic replica placement strategy. The simulation results demonstrated the effectiveness of ARPM hence provided a proof that ARPM has contributed towards a new dimension of replica placement strategy that incorporates the affinity and popularity of files replicas in P2P systems.

Table Of Contents

CHAPTER 6 CONCLUSION AND FUTURE RESEARCH

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This chapter presents an introduction of this thesis. It starts with the general area of the key concepts related to the research problem addressed. Then the fundamental motivation behind this research is stated and the proposed solutions to address the research challenges are briefly presented. The chapter ends with a discussion on the research contributions and the structure of the thesis.

1.1 Background Study

In distributed systems, data-intensive scientific computations have been quite common in many disciplines such as high energy particle physics, climate simulation, genomics, molecular docking, and bioinformatics (Chervenak et al., 2000; Ranganathan et al., 2002; Cohen and Shenker, 2002, Wang et al., 2013). The data in the distributed systems is organised as collections or datasets that are stored on mass storage systems or repositories. These datasets are accessed by users in

different locations who may create local copies or replicas of the datasets with the intention of reducing the latency involved in wide-area data transfer. A complete copy of the original dataset is referred as a replica.

Further on, the massive datasets in data-intensive scientific applications are been shared, generated, and accessed by a community of thousands of researchers located around the world. These researchers may need to transfer large subsets of the datasets to local sites or remote resources for processing. They may create local copies or replicas to reduce wide area network data transfer latencies. In most situations, the datasets requested by a user's job cannot be found at the local nodes. In this case, high latency is incurred since data has to be fetched from other nodes in the distributed systems. Until now, the data requirements in these applications continue to increase drastically every year. The increase of the scientific dataset has escalated from Terascale (10^{12}) to Petascale (10^{15}) and towards Exascale (10^{18}) systems in years to come (Reed et al., 2015; Parsons, 2013).

The problem is not only the massive needs of the input-output scientific data applications, but more importantly, the number of users, ranging from hundreds to thousands, who access and share the same datasets. Moreover, these users and the datasets are geographically distributed. Thus, there is an urgent need to obtain solutions to manage, distribute and access large sets of raw and processed data efficiently and effectively in the distributed environments (Deris et al., 2008).

An important technique to speed up access in data distributed systems is to replicate data at multiple locations, so that a user can access the data from a nearby site

(Venugopal et al., 2009; Abawajy and Mat Deris, 2014). One of the primary goals of data replication is to ensure data availability which is deemed essential in some applications such as in distributed systems, database, cloud networks and mobile systems (Goel and Buyya, 2013; Zhang et al., 2010). Creating additional copies at more than one site, not only cut down the probability of loss of all copies of data on a single site, but also brings down the bandwidth use and access latency (Chang and Chang, 2008). In addition, creating replicas can reroute client requests to the data with the closest proximity to the site where the request originated. Consequently, it will increase the system performance and provide higher access speed than a single server (Tang et al., 2005).

A replication mechanism suggested by (Chang and Chang, 2008; Zhao et al., 2008; Fadaie and Rahmani, 2012; Abawajy and Mat Deris, 2014) must always consider three important decisions pertaining to replica strategy. Firstly which file should be replicated, secondly when to replicate and thirdly where the new replicas should be placed. Then (Grace and Manimegalai, 2014) followed-up the discussion on the important decisions by identifying two important challenges in data replication. The first challenge in data replication technique is replica placement and the second challenge is replica selection.

Replica placement decides when to replicate and where the new replicas should be placed whilst replica selection decide which file needs to be replicated. Both replica placement and replica selection are equally important in proposing a dynamic replication strategy in distributed environments.

Several research works addressing data replica placement issues in distributed systems used the access pattern as guidelines in deciding the dynamic replica placement (Mansouri and Dastghaibbyfard, 2012; Rahman, 2006; Chen et al., 2002; Ranganathan and Foster, 2001). Most of these access frequency based solutions are assuming that files are independent of each other. In contrast, distributed systems such as peer-to-peer, files may be dependent or correlated to one another. Correlated or affine files refer to the files that are accessed by the same transaction or more than one transaction accessing the same files. For example, a client or a query accessing multiple queries accesses the same data. Figure 1.1 shows the correlated data accessed by the same transaction (C1) and two transactions (C1 and C2) accessing the same data.

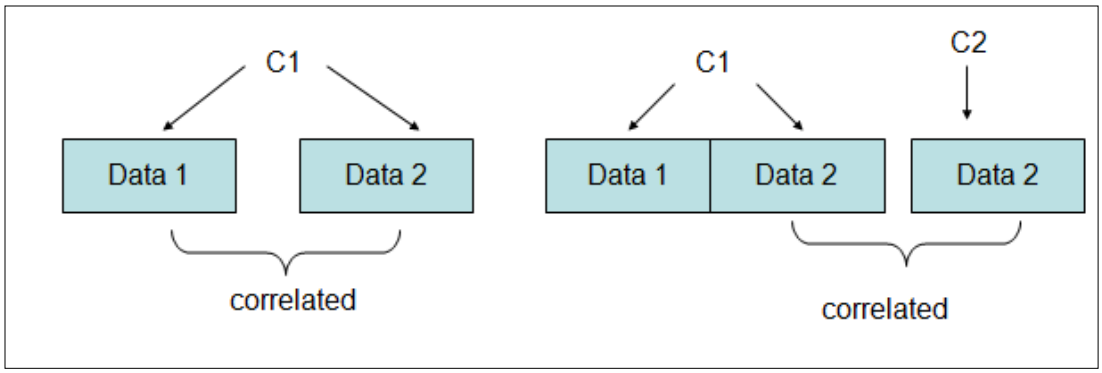

Figure 1.1: An example of correlated data for single and multi-transactions

As mentioned earlier, files that have correlated transactions are also known as affined files. In this thesis, the concept of affined files or simply affinity is used to make decisions to replicate the correlated files in solving the replica placement problems. Affinity not only refers to the relationship, but also refers to the linking between two or more people or elements. For example, a group of people sharing the same

hobbies, liking the same music, and graduated from the same university. This scenario creates affinity because the people have the same interest and work together. The interesting logic of an affinity concept inspires us to develop replica placement strategy in peer-to-peer systems that enable us to replicate files that have correlation with one another. Thus, in this thesis affinity is considered as one of the most important parameters in designing dynamic replica placement strategy.

Equally important, the second parameter that is taken into consideration in designing replica placement strategy is data popularity. Herein, both popularity and affinity are applied to address replica placement problems in peer-to- peer (P2P) systems. Our aim is to develop a technique to improve data access performance through minimizing the access time and to ensure data availability in P2P systems. The idea is that, given certain access pattern and affinity files, three important decisions can be made on which files to replicate, how many to replicate and where to place the replicas in P2P systems.

1.2 Problem Statement

Availability and efficient accesses are critical requirements in many data intensive applications. As discussed in the previous section of this thesis, the benefit of adopting affinity notion in replica placement is apparent when research collaboration among peers is required. The existing methods (Yang et. a., 2011; Madi et al., 2011) assumed that files are independent of each other. However, in fully-distributed systems such as peer-to-peer network, files may be dependent or correlated to one

another. These correlated files; referred as affinity files are required together by a query or a set of queries. Normally queries tend to access related files residing across multiple locations. Similarly, a file is often requested and accessed by multiple users. A set of files accessed by one user is also likely to be accessed together by other users.

In research collaboration environment, researchers in different regions with similar research interest may require data from other researchers. Suppose that in order to complete the research project, the researchers may request a set of files from servers at different locations. If the files belong to multiple owners in disperse locations, it requires a large amount of data and the task is time consuming. This is due to the need to find, access, analyse, and visualize data which will greatly affect the productivity of the researchers. Hence, data replication is strongly needed and further improvements on new algorithms, protocols, replication schemas, and placement strategies are critical. Despite this, file replicas must be managed intelligently and dynamically so that data is shared and replicated in the network with the objective of not just to merely fulfil the request but most importantly to have trusted transactions via the affinity relationship between the sender and the requester.

Currently, there are hardly any literature exploring the notion of affinity in creating and disseminating file replicas in file sharing distributed systems. There is a similar study (Abawajy, 2004) conducted on affinity replica location policy. However this policy only focused on the location of replicas to be replicated without considering affinity files. Some studies were conducted in other areas such as desktop grid, data

mining, self-immune systems, biology, and chemistry (Fedak et al., 2009; Bakhouya and Gaber, 2006; Gilson et al., 2016; Shi et al., 2015; Dallakyan and Olson, 2015). Therefore, this research aims to improve replica placement technique by exploring the aspect of affinity files. This is achieved through formulating the affinity degree of the related files in the systems.

This research focuses on replica placement issues. As identified by Grace and Manimegalai (2014), Rasool et al., (2009), Fadaie and Rahmani (2012), the overall replication problems evolve around these issues; (1) Which files should be replicated; (2) How many replicas should be created; (3) Where the replicas need to be placed in the system. The central point of the research is on the popularity and the notions of affinity to improve data availability and accessibility in peer-to-peer replica placement strategy.

1.3 Research Hypothesis and Questions

This thesis argues that a combination of popularity and affinity can be used to improve availability and accessibility in replica placements. The research hypothesis is been verified through simulation. Some keywords in the hypothesis are defined as below:

- **Availability** in replica placement context means that the placement of replicas can ensure the service continuity for the requested file by

guaranteeing the existence of a replica in another site when it is not available in a given site.

- **Accessibility** refers to the characteristics of being able to access when the data is required.
- **Popularity** in this hypothesis refers to how many times the data is requested by a client or the system site and it indicates the importance of the data.
- **Affinity** can be defined as correlated file, similarity, dependency, relationship, linking between two or more people or elements, and natural liking.

Reflecting upon the problem described in Section 1.2, the following research questions are formulated:

- Which files to replicate?
- How many replicas are required?
- Where these replicas should be placed in the system?

These decisions on P2P replica placement are very important in order to get the utmost benefit from the replication process.

1.4 Research Objectives

This thesis started with the subject of the common themes and differences in replica placement strategies in distributed systems. The concept of affinity in the setting of

file relationship and user access patterns was used to produce a simple model that supports the replication in P2P systems. A user can send a query to access the required file existing in any nodes in the network using an affinity placement mechanism. This research specifically aims to achieve the following objectives:

1. To propose a model for replica placement in peer-to-peer systems identifying the three research questions in section 1.3.
2. To propose an efficient strategy that incorporates affinity and the popularity of the files.
3. To measure the improvement of data access performance through simulation.

1.5 The Scope of the Research

This research focuses on combining the popularity and affinity files as two most important parameters in designing replica placement strategy in distributed systems. Given these two parameters, replica placement and replica selection for data replication can be constructed. The problem of file updates and synchronization are not addressed in this research with files are regarded as being read-only. This is due to the fact that certain characteristics of datasets are specific to the applications for which they are targeted. For example, in astrophysics or high energy physics experiments, the principal instrument such as a telescope or a particle accelerator is the single site of data generation.

This means that all data is written at a single site, and then replicated to other sites for read access only. Updates to the source are propagated to the replicas either by the replication mechanism or by a separate consistency management service (Shorfuzzaman, 2012).

1.6 Contributions of the Study

This thesis highlights several contributions towards improving the understanding of replication in distributed systems, focusing in the area of replica placement in peer-to-peer network.

There are four major contributions in the thesis as follows:

- ARPM has been proposed and it has successfully contributed to the improvement of data access performance through minimizing the access time.
- ARPM has successfully avoided the over replication of the unnecessary replicated files in the distributed system.
- The formula for access frequency is adapted mathematically to calculate the file popularity whilst the formula for affinity degree was established.
- The hybrid of the popularity and relatedness (affinity) of the files demanded by the clients in the network has been incorporated in our replica placement strategy

1.7 The Structure of the Thesis

This thesis is comprised of six chapters including this introductory chapter. The remainder of the thesis is organized as follows:

Chapter 2 describes the overview of replica placement in distributed systems. This includes the related works on replica placement strategy under different and similar topologies, and also exploring the concept of popularity and affinity. Finally this chapter discusses some important research in distributed systems which further focuses on the mechanism that we proposed in this thesis.

Chapter 3 discusses the methodology of the proposed model of Affinity Replica Placement Mechanism (ARPM). The notion of affinity in ARPM is defined as the relationship between two or more correlated files in peer-to-peer (P2P) systems. The replica placement strategy in ARPM considers popular files and affinity degree in deciding which file to replicate and when to replicate the files. The replica placement is presented and proved analytically. The objective of the proposed model is to minimize access latency and optimize availability by allowing files to be replicated based on their high popularity and strong affinity.

Chapter 4 describes the implementation of the proposed ARPM based on simulation. The performance of the proposed model presented here considers scenarios in single query and multiple queries from the source node that initiate the request to the destination node that hold the requested file.

Chapter 5 presents the evaluation and the experimental results of the proposed Affinity Replica Placement Mechanism (ARPM). Detailed discussions on the simulations results are presented. How queries in a fixed number of cycles and in a set of time intervals contribute to the replica placement performance is discussed in this chapter.

Chapter 6 summarizes the contributions of this thesis and discusses future direction of the research. The discussion allows further exploration of significant research areas which are closely related to the focus of this thesis.

CHAPTER 2

REPLICA PLACEMENT IN P2P SYSTEM

This chapter provides general overview of data replication in distributed systems, and presents a comparison of several replication strategies in peer-to-peer file sharing networks and data grids in distributed environment. Specifically, the emphasis is on the replica placement decisions for providing scientific communities with better availability and efficient access to massive data. Following the replica placement decisions, a broader discussion regarding the data access pattern and the affinity data for which ARPM would be used is also been discussed.

2.1 Introduction

In data scientific applications such as high energy particle physics, climate simulation, genomics, bio-medicals, and bioinformatics large datasets from simulations or experiments were generated (Abdullah et al., 2008; Mansouri et al., 2013). The amount of data in these scientific applications was in the order of a couple of hundred terabytes or petabytes per year. In addition, with the success of

generations of high performance computing (HPC) systems, the next generation of e-Science infrastructures predicts that HPC will generate data at a very high rate (terabytes) per year (Chen et al., 2014; Palaniswamy, 2010). The effect is that, by the year 2020, hundreds of exabytes distributed data are expected to be available through heterogeneous storage resources for access, analysis, post-processing and other scientific activities across several centres (Soosai et al., 2012). Figure 2.1 shows the scientific applications predicted by the year 2029. Adding up all the data from other scientific applications, the total amount of data to be processed is hard to estimate and is inapprehensible.

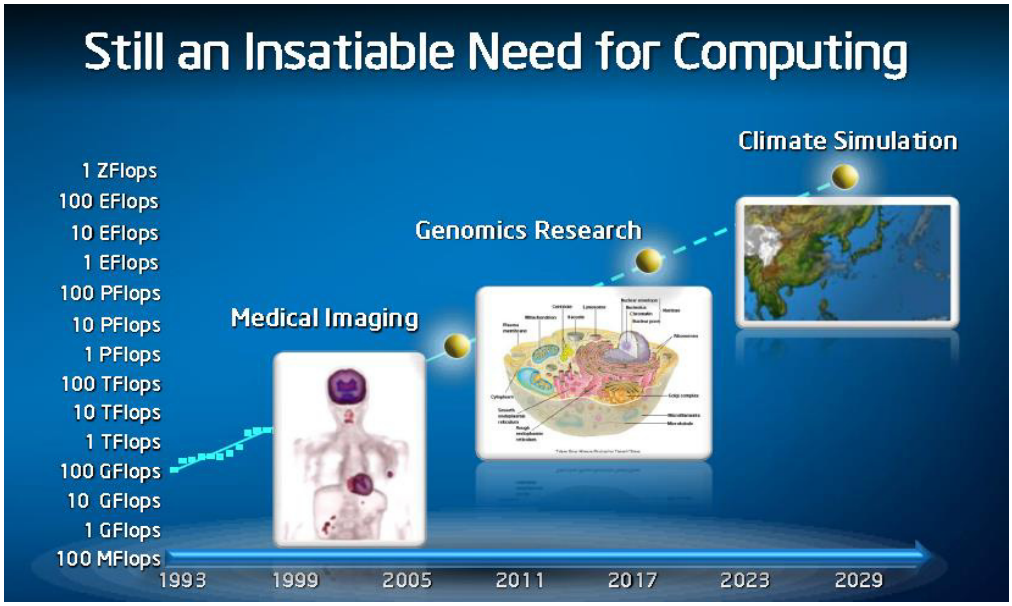

Figure 2.1: Data requirements for scientific applications (Palaniswamy, 2010)

The explosive growth of these large data will eventually impact many applications and collaborations in the research world. In data scientific applications, the placement of data can have significant impact on the performance of scientific computations and the availability of the datasets. In addition, the new generation of

applications like business intelligence, Web 2.0, social networking requires distributed processing of terabytes and even petabytes of data (Bakshi K., 2012). However, relational databases are found to be inadequate in distributed processing involving very large number of servers and handling Big Data applications. As a result, major web companies such as Amazon, Facebook and Google, developed their own, inherently distributed, lightweight solution to act as a database back-end for their services (Hecht and Jablonski, 2011). These developments spun interest in the open source world and numerous products appeared under the term NoSQL which means not only SQL (Dobos et al., 2014). NoSQL systems replicate data over many servers and support a large number of simple read/write operations per second.

Therefore, the need for efficient data management in distributed system is imperative. Foster et al., (2001) stated that the most critical requirements in many data scientific applications are availability and efficient access. Delayed accesses due to availability problems or non-responsiveness may cause undesired results. To effectively address these challenges, the need for data replication is apparent. In fact, data replication is a well-known technique and has been extensively used within the context of other distributed data intensive paradigms such as in the World Wide Web, peer-to-peer file sharing networks and mobile database (Shorfuzzaman, et al., 2014; Yang et al., 2011; Rasool et al., 2009).

Data replication is defined as the creation and maintenance of copies of data at multiple peers. Creating replicas at a suitable site based on data replication strategy can increase data availability and ensures efficient data access. Since the similar data can be found at multiple peers, availability of data is assured in case of peers' failure.

In addition, data replication can provide increased fault tolerance, improved scalability, reduced bandwidth consumption and improved response time (Devakirubai and Kannamal, 2013).

Currently many replication strategies have been proposed in distributed environments (Hamdeni et al., 2016; Luo et al, 2015; Chettaoui and Charrada, 2014; Sivakumar et. al, 2013; Amjad et al., 2012; Mansouri and Dastghaibbyfard, 2012; Sashi and Thanamani, 2011; Liu et al., 2006a, 2006b; Rahman et al., 2006; Tang et al., 2005; Ranganathan and Foster, 2001). Replication strategies can either be in the form of centralised or distributed. In centralised replication, the replica placement decisions will be taken by a centralised node. Whilst in distributed replication, all the nodes in the system participate in decision making (Shen et al., 2010).

As the demand for data increases, these centralised replication strategies are liable to a single point of failure and become a bottleneck when dealing with huge amount of data trying to access the same data simultaneously. However, the single point of failure problem has been solved with the deployment of decentralised replication strategies (Spaho et al., 2015; Amjad et al., 2012; Mat Deris et al., 2007; Weil et al., 2006; Wan Awang et al., 2004). According to Grace and Manimegalai (2014) when developing a data replication protocol, the selection of which files should be replicated, the number of replicas to be used and the sites where the replicas will be hosted are the three important decisions to be made such that the aims of data replications are achieved. These decisions led to the proposed of dynamic replica placement strategies in major topologies used in a data grid environment (Rahman et al., 2006; Ranganathan et al., 2002).

2.2 Grid Topologies

A data grid topology represents the manner in which data sources are organised in the grid. Numerous topologies are available for data grid operations (Venugopal et al., 2009; Rasool et al., 2007, 2008; Tang et al., 2005). Figure 2.2 to Figure 2.5 present the most common topologies established in data grid environments namely: hierarchical, peer-to-peer and hybrid. In this thesis, the term hierarchical, tree, and multi-tier refer to the same topology.

2.2.1 The Hierarchical Topology

The replica placement in hierarchical topology has been studied intensively by (Ranganathan et al., 2002; Abawajy et al., 2004; Tang et.al., 2005; Lin et al., 2006; Liu et. al, 2006a, 2006b). A hierarchical topology is used when there is a single source for data and the data has to be distributed among collaborations worldwide. The architecture of hierarchical or multi-tier data grid is shown in Figure 2.2. As an example, the LHC (The Large Hadron Collider) application, a project in CERN (European Organization for Nuclear Research) is hierarchical and is organised in tier. Each tier refers to a different region namely, local nodes, regional nodes, national nodes, and international nodes (Chang and Chang, 2008; Goel and Buyya 2013).

All of the leaf nodes represent the clients, and each client can only access the replicas from its ancestor. Tier-0 represents the main node located at CERN, where all data are produced. The data is distributed to regional centres at Tier-1. Regional centres

further distribute the data to Tier-2 centres which in turn distributed data to processing institutes at Tier-3. The data is then finally distributed to the end users at Tier-4. The rate of data transfer from Tier-0 to Tier-1, Tier-1 to Tier-2 and Tier-2 to Tier-3 is ≈ 622 Mb/sec. Data transfer between Tier-3 and Tier-4 ranges from 10-100 Mb/sec (Goel and Buyya, 2013).

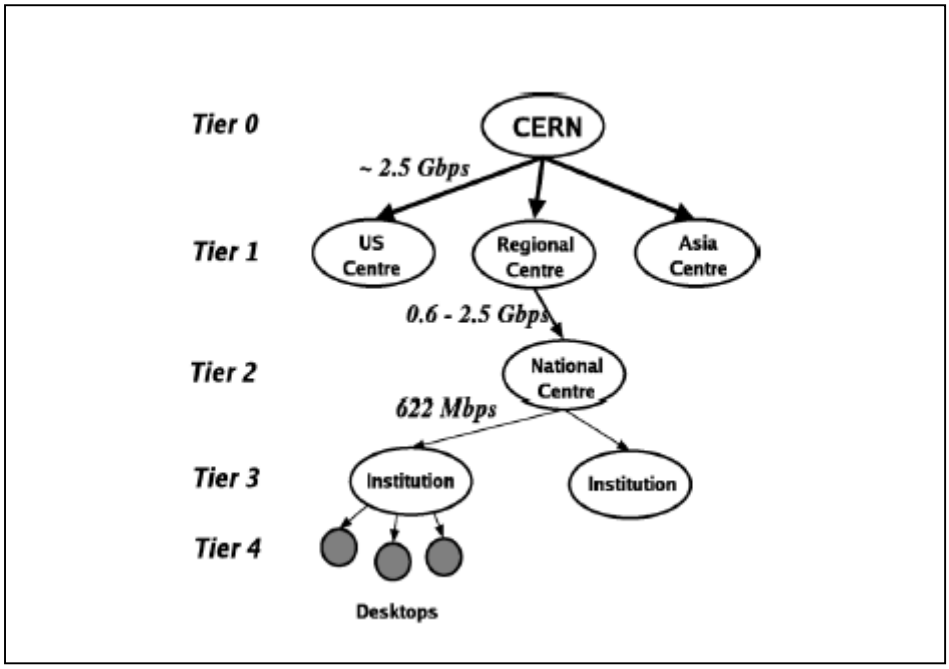

Figure 2.2: A hierarchical model (Venugopal et al., 2006)

A hierarchy model in data grid allows scientific community to access the resources in a common and efficient way. More importantly, the massive amounts of data resided in the sites motivate the need for robust data distribution mechanism (Venugopal et al., 2009). However, achieving this target is difficult especially when new nodes connect to the hierarchy and performance of the systems becomes degraded. This situation occurs because the hierarchical model cannot transfer data among sibling nodes or nodes situated on the same tier. Nevertheless, in a

hierarchical model, it is easier to maintain data consistency as all data are kept in a single source (root or Tier-0).

2.2.2 The Peer-to-peer Topology

A peer-to-peer system has managed to overcome the limitations of hierarchical and centralised server approaches from network congestion, scalability and fault tolerance limitations. The term "peer-to-peer" (P2P) refers to a class of systems and applications that employs distributed resources to perform a function in a decentralized manner. Figure 2.3 shows the pure decentralised peer-to-peer topology. The pure decentralized P2P topology allows more complex dependencies between computing resources in a fully distributed behaviour. Some of the benefits of a P2P approach include: improving scalability by avoiding dependency on centralized points; eliminating the need for costly infrastructure by enabling direct communication among clients; and enabling resource aggregation.

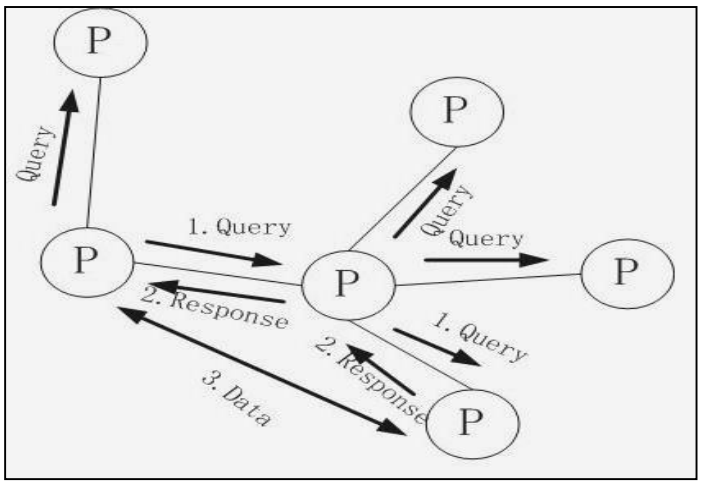

Figure 2.3: Peer-to-peer topology (Steinmetz and Wehrle, 2005)

Figure 2.4 shows P2P model for a scientific data grid that will support scientific collaboration proposed by Abdullah et.al, (2009). This model is specified as unstructured P2P model, where peers could be any network devices such as PCs, servers and even supercomputers. The analogy of the proposed model is similar to electrical power grid. The users or scientists (peers) can access their required datasets without knowing which peers deliver the datasets. This means that the users can execute their applications, obtain the remote datasets and then wait for the results. This will be done by the discovery service. Each peer operates independently and asynchronously from all other peers and it can be self-organized into a peer group. Peer group contains peers that have agreed upon a common set of services, and through this peer group, peer can discover each other on the network.

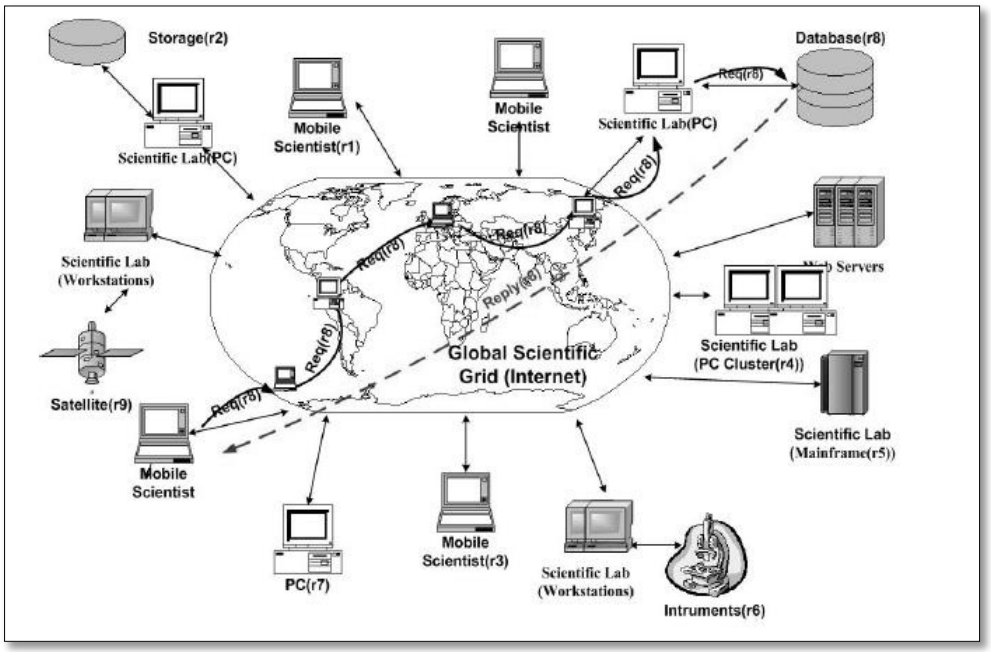

Figure 2.4: P2P model for scientific data grid (Abdullah et al., 2009)

Once a peer joins a group it uses all the services provided by the group. Peers can join or leave the group anytime that they want. In this model, once a peer joins the

group, all the datasets that are shared by other peers in the group will be available to him. Peers can also share any of their own datasets with other peers within the group. A peer may belong to more than one group simultaneously. The focus of the research in Abdullah et.al, (2009) is to propose a decentralized discovery strategy for Scientific Data Grid that addresses the scalability problem and also reliability problem.

The attractive features of P2P systems are the high availability and reduced query latency towards user request (Karun and Jayasudha, 2013). These are achieved because of the inherent redundancy in the system through replication where peers replicate each other‘s data so that when one peer is offline the other can serve the request. Many studies in P2P networks consider the replica placement problem i.e. how to place replicas in proper locations so that the overall performance of the system is improved.

2.2.3 The Hybrid Topology

The hybrid topology is a new emerging topology as data grids mature and widely used in industries (Garmehi et. al, 2014; Lahemahedi et al., 2002). This topology combines all the centralised, hierarchical, and P2P topologies. Figure 2.5 shows a hybrid topology of a hierarchical data grid and peer linkages at the edges. Another hybrid topology model in Figure 2.6 shows the peers and super-peers connections. A Super-peer is responsible for returning results to the queries posed by their neighbouring leaf-nodes.

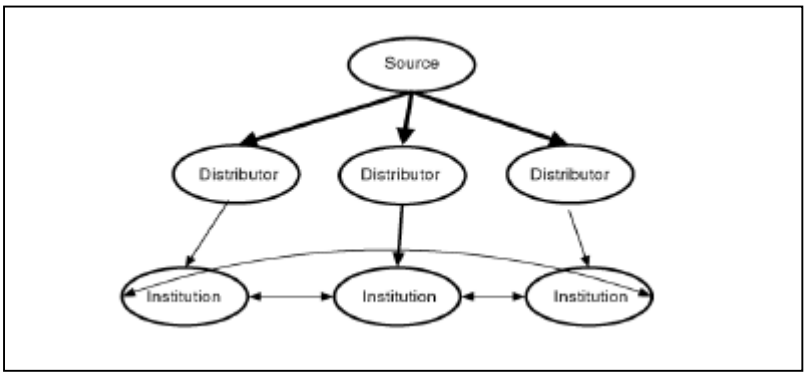

Figure 2.5: A hybrid model (Venugopal et al., 2006)

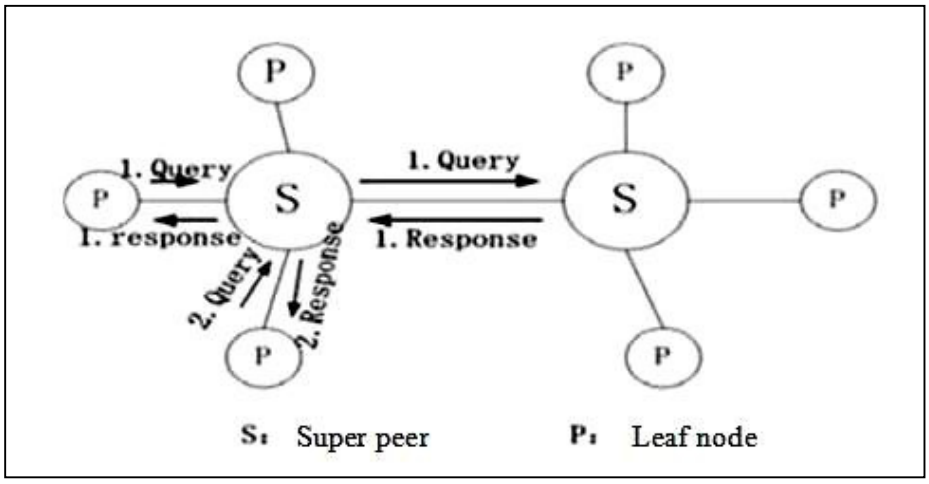

Figure 2.6: A hybrid topology (Steinmetz and Wehrle, 2005)

The overall file replication problem consist of making the following decisions: which files should be replicated; how many replicas should be created; and where the replicas should be placed. Depending on those answers, various different replication strategies are proposed (Hamdeni et al., 2016; Rasool et al., 2011; Abawajy et al., 2008; Weil et al., 2006)

2.3 Data Replication Strategies

This section provides several recent studies on data replication focused on ensuring data availability, improving fault tolerance and reducing file access time (Garmehi and Mansouri 2007; Amjad et al., 2012). Data replication as one of the best known strategies used to achieve high level availability and fault tolerance as well as minimizing the access times in distributed systems are emphasized by Garmehi and Mansouri (2007). They proposed an algorithm to find optimal placement of *k* replicas of an object over data grid systems such that the overall cost of storage and read is minimised.

Three types of user access patterns (random access, temporal locality, geographical and temporal locality) were identified by Ranganathan et al., (2001) and they suggested six replica strategies which include: No Replica, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replica and Fast Spread. The simulation results show that matching replica strategy with suitable access pattern would save bandwidth and reduce access latency. In the geographical locality pattern, Cascading can have best performance in response time.

The problem of placing a new replica in proper place by considering its priority list was addressed by Lin et al., (2008). Herein, the proposed replica placement algorithm find out the minimum number of replicas when the maximum workload capacity of each replica is given. The authors extended the tree architecture to the Sibling Tree model. In this model, if the requested data is not present in sibling ring then the parent ring is searched. Furthermore, the logical connection between the siblings and

all connections from one sibling to another physically involves the parent at most two hops. This means that the actual time taken to serve a request is infected more than it is presented, as this logical connection is assumed physical and already the time complexity is too high. The drawback in this proposed replica placement algorithm is the problem of network congestion or bandwidth consumption which is not considered.

In 2004, Abawajy proposed a heuristic algorithm called Proportional Share Replica (PSR) Policy to improve on the cascading technique proposed by Ranganathan et al., (2001). PSR puts the data replica at the best site in which the numbers of sites and total replicas to be distributed are already known. This technique starts with calculating the distribution ideal load. Subsequently, replicas will then be placed at a candidate site that has the ability to serve a request for replica at better rate or equal to the calculated ideal load. Ideal load is calculated using the following formula:

$$Load = Totalrequest / (Originalcopy + Replicas) \quad (2.1)$$

TWR (Two Way Replication strategy) was proposed in 2009 by Rasool et al. The strategy is an updated version to the multi-tier sibling tree architecture presented by Ranganathan et al., (2002) and Lin et al., (2006, 2008). In TWR, the most popular data is identified and placed to its proper host in a bottom up manner in which they are closer to the clients. In a top down manner the less popular files are identified and are placed to one tier below the root node, so that it is closer to the root. In this approach, replica selection is done by using the closest policy that tries to provide

the data from the nearest sites. The drawback of the research is that it only considers the homogenous data grid nodes and cannot be applied to heterogeneous nodes.

Abdullah et al., (2008) proposed a P2P model for higher availability, reliability and scalability. They developed their own data grid simulator to test the replication strategy, taking response time, number of hops and average bandwidth consumption as basic parameters for evaluation. In this research, four replication strategies have been studied. Two of the existing strategies: requester node placement strategy and path node placement strategy and two new replication strategies are proposed: path and requester node placement strategy and N-hop distance node placement. In the requester node placement strategy, the required file is placed only if the file is found. Whilst in the path requester node, the requested file is copied to all the nodes on the path from the requester node to the provider node.

The new proposed strategy path and requester node placement strategy actually is the combination of the two existing strategies. In N-hop distance node placement, a file is replicated to all providers' node neighbours within N-hop distance. The result shows that the new strategies have shown better performance than the existing one in terms of performances, success rates and response time. However, the proposed strategies use more bandwidth than the existing strategies. Besides, the storage loads of replica servers are not considered in their strategies. This is due to the file being replicated to all the nodes on the path from the requester node to the provider node.

A modified form of Bandwidth Hierarchy Replication (BHR) has been presented by Sashi et al., (2011) as a way in overcoming the limitations of the replication strategy

proposed by Abdullah et al., (2008). In the modified BHR model, a network region is defined as a network topological space where sites are located closely. Whenever the required replica is present in the same region, the job completion will be fast. BHR model is based on tree structure which is not really suitable in a real data grid environment.

Figure 2.7 shows the replication strategy taxonomy which determines when and where to create a replica of the data. These strategies are guided by factors such as the number of user requests, network conditions and cost transfer. Method is the first classification that is based on whether the strategies are static or dynamic. In static strategy, the replica remains in the system waiting to be removed by the user or if it reaches its expiration limit. The static replication strategies are simple to implement but not frequently used because they do not support replication during job execution.

In comparison, dynamic strategies can adapt changes based on user requests, storage capacity and bandwidth. Dynamic strategies are capable of making decisions to place data in P2P systems based on storage and node availability. In addition, dynamic replication automatically builds and removes replicas according to the changes of access patterns. This is to ensure the benefits of replication continue regardless of users' behaviour changes to form popular data (Lamehamedi et al., 2002; 2003, Kawasaki et al, 2006). The second classification is the granularity which relates to the level of subdivision of data that the strategy works with. Replication strategies that deal with multiple files at the same time work at the granularity of datasets. The next level of granularity is the individual files while there are some strategies that deal with smaller subdivisions of files such as objects or fragments.

Objective Function is the third classification deals with the objective function of the replication strategy. Possible objectives of a replication strategy are to maximise the locality or move data to the point of computation. By exploiting the popularity file or most requested datasets, the update costs can be minimised. A taxonomy of file modes is shown in Figure 2.8 (Ma et al., 2013). The file taxonomy considers file types, file access pattern, file access permissions and file origin.

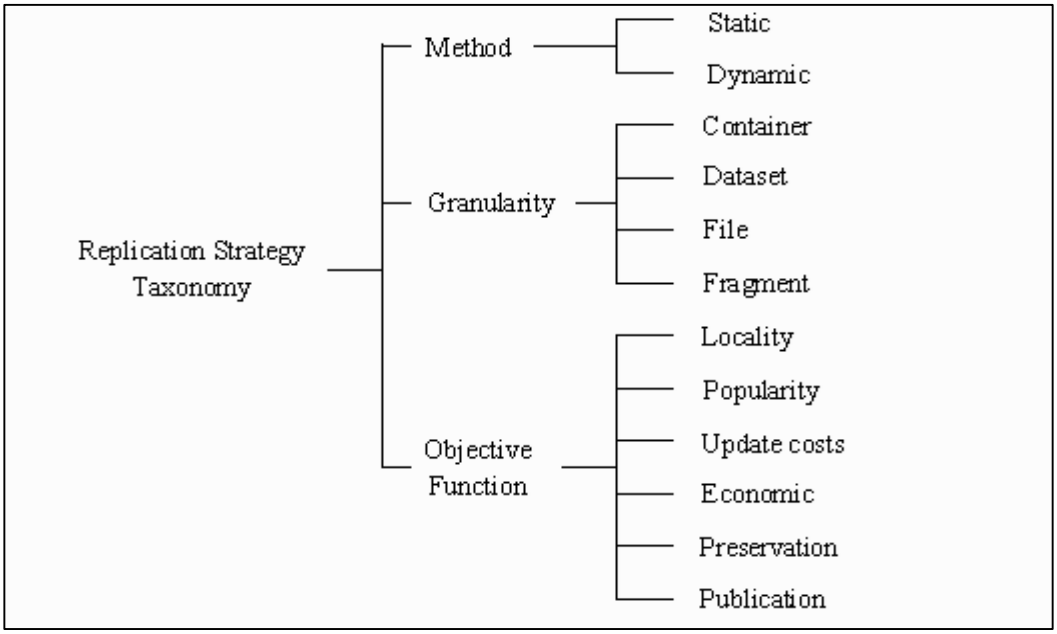

Figure 2.7 Replication strategy taxonomy (Venugopal et al., 2006)

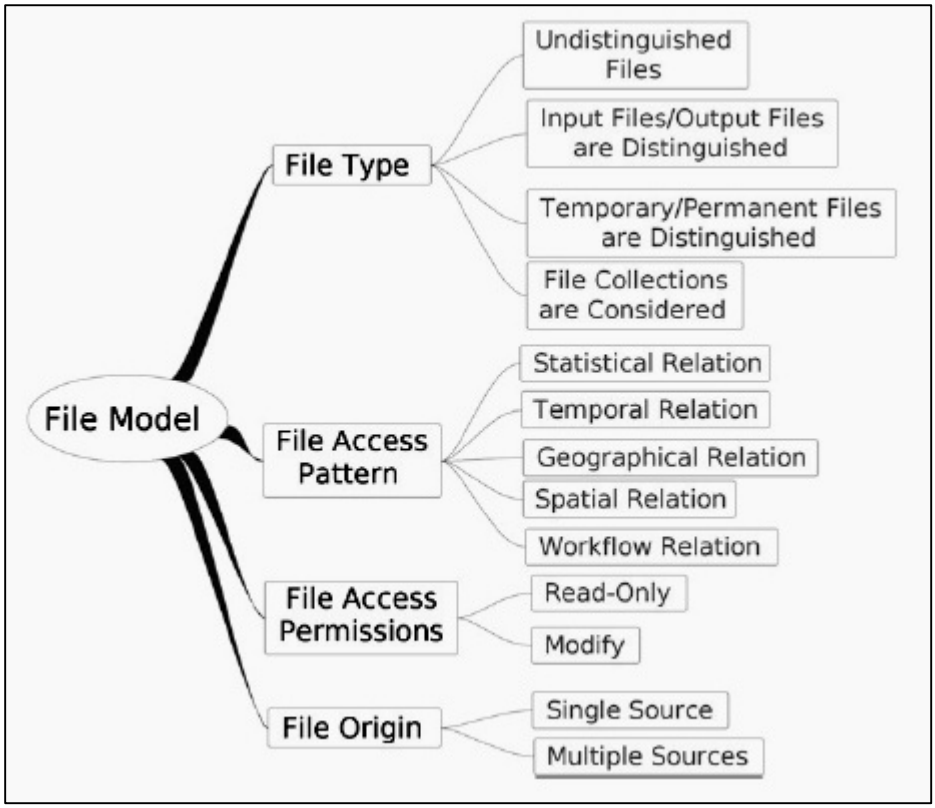

Figure 2.8: The taxonomy of the file models (Ma et al., 2013)

As stated by (Abawajy, 2004), a strategic replica placement is the key to get the maximum benefit out of replication. Each strategy aims at different goals and optimizes various aspects of the system. Furthermore, replica placement as an approach for making replication decisions has the advantages of improving the efficiency of data access and the capability of fault tolerance (Zhao et al., 2008). More importantly, replica placement is one of the important factors to improve performance in scientific research collaboration distributed systems. Therefore, the next section in this chapter discusses the different replica placement strategies in peer-to-peer (P2P) network systems.

2.4 Issues related to Data Replication in Distributed Systems

Although the necessity of replication in distributed systems is evident, its implementation involves issues such as replica placement, resource discovery and management, selecting suitable replicas, the impact of replication on the performance of job scheduling, and replica consistency maintenance. However this thesis focused more on the replica placement issue. The following fundamental issues in replica placement are identified:

a) Replica Selection: Identification of which files to replicate. The strategic placement of selecting replicas is very important to obtain maximum gains from replication based on the objectives of applications.

b) Replica Allocation: The degree of replication must be selected to use the minimum possible number of replicas without excessively reducing the performance of applications and user request.

c) Replica Placement: The component of a distributed system architecture that decides where the file replicas should be placed in the system.

In general, replication strategies depend on when, where, and how replicas are created and destroyed. A detailed discussion of existing work in replica placement focusing on peer-to-peer systems is presented in the next section of this chapter.

2.5 Replica Placement in Peer-to-Peer Systems

Replica placement strategies in unstructured P2P systems can be classified using two criteria techniques related with site selection and techniques related with replica

distribution. Table 2.1 shows some of the replica placement techniques that belong to these two groups.

Table 2.1 Replica placement strategies in P2P (Spaho et al., 2014)

Strategy	Site Selection/ Replica Distribution	Advantage	Limitation
Owner	Site Selection	No storage Consumption	Large amount of time needed
Path	Site Selection	Good search performance	Large time for recovery
Random	Site Selection	Small search delay	Hard to implement
Uniform	Replica Distribution	Reduce search traffic	Replica placement where peers do not access the files
Proportional	Replica Distribution	Reduce search traffic	Difficult to find not popular data
Square Root	Replica Distribution	Reduce the number of hops to find an object	Requires knowledge of the query rate for each item

2.5.1 Owner Replication

Owner replication and Path replication were evaluated by Lv et al., (2002), whereby in the owner replication, a file or an object is replicated at the requester node only whenever a search is successful. As a result, the number of replicas will increase in proportion to the number of requests for the service. Since the number of replicas generated in P2P is limited to one replica at each data exchange, the time taken to propagate replicas over the P2P network is increased. Consequently, the search performance for the requested data is slightly decreased. Owner replication is an example of non-active replication.

2.5.2 Path Replication

Path replication is an active replication whereby the file is replicated to all the nodes along the path between the source and the destination nodes (Lin et al., 2008). In this technique, the peer with a high degree of neighbours forward much more data than the peer with a low degree. The number of replicas produced per query is proportional to the number of search. If the system fails due to overload, recovery will take longer time. Nevertheless, Path Replication has been used in many distributed systems due to its good search performance and ease of implementation.

2.5.3 Random Replication

This technique distributes the replicas in a random order. Random replication is the most effective approach for achieving both smaller search delays and smaller deviations in searching. Random replication is harder to implement, but the performance difference between the random and the path replication highlights the topological impact of path replication.

2.5.4 Uniform Replication

Uniform replication strategy replicates everything equally. The replicas in this technique are distributed uniformly through the network. For each data object, the same number of replicas is created. While this controls the overhead of replication, replicas may be found in places where peers do not access the files.

2.5.5 Proportional Replication and Square-root replication

In the proportional replication, the number of replicas is proportional to their popularity. This replication is used for reducing search traffic. In Square-root replication, the number of replicas of a file is proportional to the square root of query distribution. This technique reduces the number of hops needed for finding an object.

The major features of replication algorithms for P2P systems are the criteria for the selection of suitable objects for replication and selection of suitable sites for hosting new replica. These two important aspects have a direct impact on the performance of the system. If a node decides to replicate all the objects present in its shared directory to other nodes, it will increase the overhead in the network. The replica should be maintained in sites which are close to the source nodes to increase the search performance. The site selection policy of a replication technique decides where the replica should be stored. The number of sites may vary based on the replication scheme being employed. For example, if popular files are not replicated appropriately, overwhelming requests from peers can cause network congestions and slow download speed.

2.6 Replica Selection

A system that includes replicas also requires a mechanism for selecting the right files based on the data access patterns. Choosing and accessing appropriate replicas are very important to optimize the use of P2P resources. Replica selection criteria might include access time as well as the source node that initiate the request, and the number

of accesses. Slow network access hinders the efficiency of data transfer regardless of client and server implementation.

Correspondingly, an optimization technique to select the best replica from different physical locations is by examining the available bandwidth between the requesting nodes and the node that hold the replicas. To transport the replica to the requested site would be the one that has the least transfer time required. Although network bandwidth plays a vital role in selecting the best replica, additional characteristics of data transfer (most notably, latency), replica host load, and disk I/O performance are other important factors as well (Shorfuzzaman, 2010).

2.7 The Affinity Concept

The word affinity in general refers to the close similarity, likeness, relationship or correspondence. In peer-to-peer systems, we defined an affinity as the correlated files, similarity, dependency or the linking between two or more files. Whereas, in Chinese culture, the word affinity means "luck" by which people are brought together. An affinity also means a meeting between friends with the same hobbies, various relationships with people such as friend to friend, parent to offspring, employee to boss and so on. These are some examples in relationship and social behaviour of an affinity (Larbani and Chen, 2009; Chen et al., 2006; 2009). Inspired by the ancient social systems and human behaviour, Larbani and Chen (2009) explore the concept of affinity further in fuzzy and rough set framework, data mining and other applications.

Different affinities according to various relationships with people can be defined and developed. For example a group of people sharing the same hobbies, liking the same music, or an institution that created affinity because they work together.

Affinity does not only refers to the relationship, but more importantly refers to the linking between two or more people or elements. In chemistry, for example, the elements of molecules can be a set because of similar affinities that bind them together. Depending on how affinity is defined, it can be used to examine, describe and predict the behaviour of access pattern or data similarity in placing replica in distributed organizations. Different measurement systems lead to various affinity degrees and more importantly may lead to the dynamic decision or strategy in replica placement environment. Therefore, in this thesis the concept of affinity has been explored further as a mechanism to select the popular files for data copies and to assess to what extent the affinity components can improve the access performance and availability of data replicas in peer-to-peer systems. In the next sub sections, affinity in different applications is discussed further.

2.7.1 Affinity implemented in Desktop Grids

Data-intensive applications require secure and coordinated access to large datasets, wide-area transfers and broad distribution of TeraBytes of data while keeping track of multiple data replicas. This data grid aims at providing an infrastructure and services to enable data intensive applications. In comparison, desktop grids is a specific class of grid that use computing, network and storage resources of idle desktop PCs distributed over multiple LANs or the Internet. The aim of desktop

grids is to compute a large variety of resource demanding distributed application. BitDew proposed by (Fedak et al., 2009) is a programmable environment for automatic and transparent data management on computational Desktop Grids. It is a subsystem which could be easily integrated into Desktop Grid systems. Bitdew offers programmers (or an automated agent that works on behalf of the user) a simple API for creating, accessing, storing and moving data with ease, even on highly dynamic and volatile environments.

Afinity is used in BitDew as the placement dependency between data and it indicates that data should be scheduled on a node where other data have been previously sent. Affinity drives movement of data according to the dependency rules (Fedak et al., 2008a, 2008b, 2009). In BitDew, the programmer can specify a replication level of an object. (E.g. 5 copies) leaving the run-time system to determine the placement of the file replicas. One of the disadvantages of Bitdew is that a programmer has no possible basis for choosing five replicas, since availability does not vary proportionally with the number of replicas. The placement fails to take into account the reliability, the performance or failure interdependences of the nodes on which the replicas are placed. Furthermore, the reliability of individual nodes and their failure interdependencies are parameters that cannot be controlled and must be monitored so that their effects can be accounted for in replica placement strategy.

Figure 2.9 shows the three-tier schema adopted by BitDew as its software architecture. The uppermost level is the API which offers a programmer a simplified view of the system. The programmer or user is allowed to create data and manage their repartition and distribution over the network of nodes. The intermediate level is

the service layers which implements the API: data storage and transfers, replicas and volatility management. The lowermost level is composed of a suite of backend.

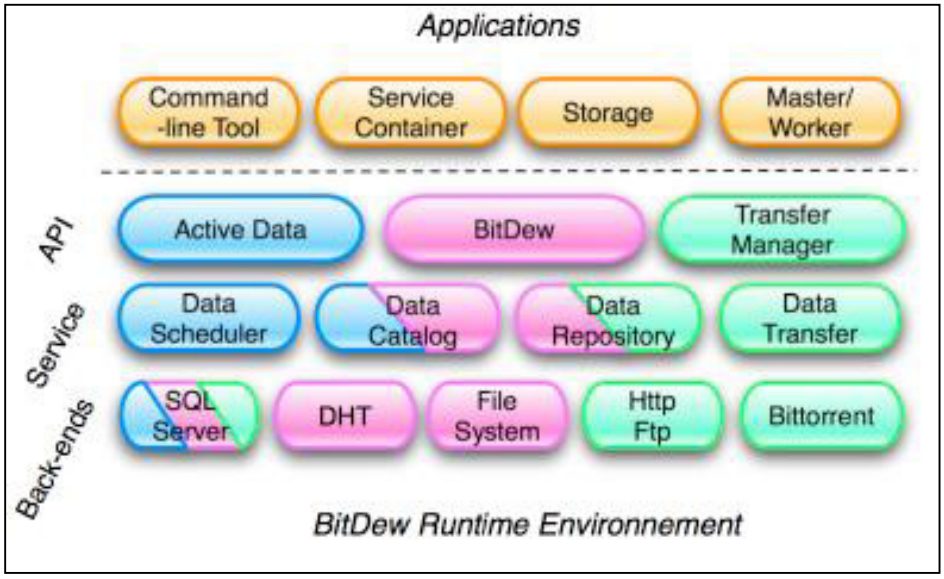

Figure 2.9: The BitDew software architecture (Fedak et al., 2009)

2.7.2 Affinity in Self-Immune Systems

In many pervasive applications such as information sharing, a user is much more likely to communicate with other users having similar interests (Bakhouya and Gaber, 2006). Thus based on this concept of communities composed of users with similar preferences and interests, an approach called Propitiate Multi Agent System (PMAS) is proposed by Bakhouya and Gaber (2006). The aim of PMAS is to reinforce the learning based approach by imitating the human immune systems behaviour.

Agents together with their affinity relationship as a whole form a propitiate multi agent system (PMAS) based on the self-immune system behaviour. In PMAS, affinity corresponds to the adequacy with which two services could bind to share common interest attributes. The affinities are adjusted by users' satisfaction regarding their interaction and dynamic work condition changes. User interests or services are represented by agents that establish a relationship based on affinities to create a spontaneous PMAS. The concept of affinity based on self-immune system is perhaps can be investigated further as an alternative or an extended approach to apply affinity in P2P data placement in future. Figure 2.10 shows Propitiate Multi Agent System (PMAS) describing affinity network between six agents based on keyword similarity.

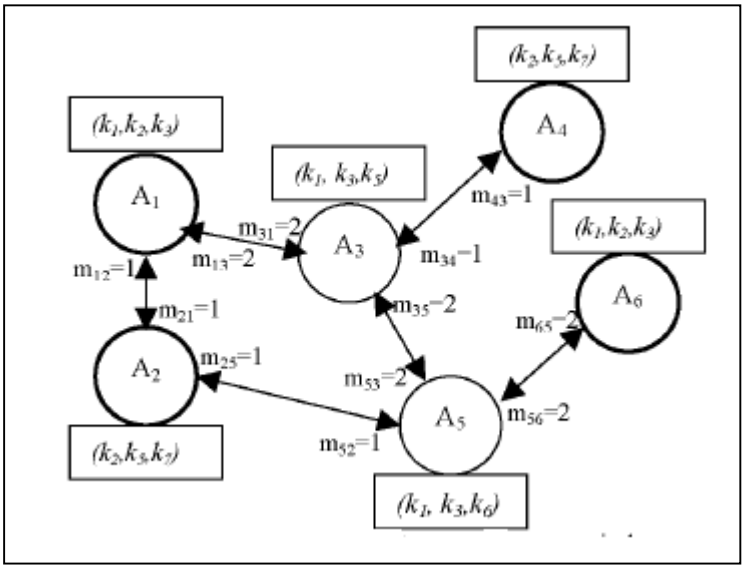

Figure 2.10: Propitiate Multi Agent System (PMAS) created between six agents based on keyword similarity (Bakhouya and Gaber, 2006)

2.7.3 Affinity Replica Location Policy

The affinity replica placement algorithm replicates data near the user nodes where the file is accessed most (Abawajy, 2004). A file is copied and placed near to the user that generates access traffic the most. The algorithm is similar to the cascading replica placement algorithm discussed in Ranganathan et al., (2001).

2.8 Popularity Driven Replica Placement Algorithm

In the real world, some files may be popular than others and data access pattern may change over time. The popularity of a file is determined by its recent access rate. Therefore, any dynamic replica placement strategies must keep track of file access histories to decide on when, what and where to replicate. The dynamic replication algorithm proposed by (Tang et al., 2006; Chang and Chang, 2008; Shorfuzzaman, 2010; Madi et al., 2011) determines the popularity of a file by analysing data access history. Figure 2.11 is an example of access history for two files, *X* and *Y* and the node relation. Nodes N_1, N_2, and N_3 are siblings and their parent is P_1. In the figure, the records indicate the state that *N1* and *N3* have accessed file *X* 5 and 9 times whilst node *N2* has accessed file *Y* 10 times. If the threshold is assumed to be 10, file *Y* will be replicated at node P_1 because the number of request exceeds the threshold value according to Simple Bottom Up (SBU) algorithm (Ranganathan et al., 2001, 2002). However, file *X* is accessed 14 times by node N_1 and N_3 and thus is more popular than file *Y*. The better solution is to replicate file *X* to P_1 first then replicate file *Y* to P_1.

NodeId	FileId	Number of Accesses
N_1	X	5
N_2	Y	10
N_3	X	9

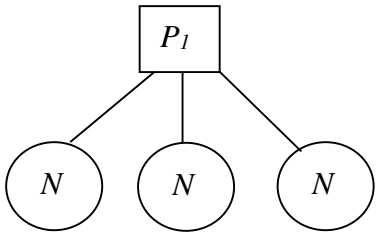

Figure 2.11: An example of access history and node relation

The replication strategy based on file popularity based on the calculation of the number of accessed has been proposed by a number of researchers (Tang et al., 2006, Chang et al., 2008, Shorfuzzaman, 2010, Madi et al., 2011). The Latest-Access-Largest Weight (LALW) proposed by Chang (2008) calculates the Access Frequency (AF) to represent the importance of access histories in different time intervals.

The AF for file *X* is calculated as:

$$Access\ Frequency = \ AF_{Nt}(f) = \sum \left(a^t f \times 2^{-(N_t - t)} \right), \quad \forall f \in F_t = 1 \tag{2.2}$$

Assume N_T is the number of time interval passed, *F* is the set of files that have been requested and a_f^t indicates the number of accesses for file *f* at time interval *t*. The above formula is calculated in the first phase after collecting all access records from the cluster headers. In the second phase, the average of AF per time interval for the popular file (assuming p represent popular file) and all files in F are calculated as:

$$AF_{avg}(p) = \frac{AF(p)}{N_T}) \text{ and} \tag{2.3}$$

$$AF_{avg}(f) = \frac{(AF(f))sum}{N_F x N_T} \tag{2.4}$$

where $AF(p)$ is the AF for the popular file p, N_T is the number of time intervals passed, $N_F = |F|$ is the number of different files that have been queried, and $(AF(f))sum$ indicates the sum of AF for all file queries.

In the third phase, the number of replicas needed for the popular file to ensure a better network performance and to achieve a load balance is calculated. The number of replicas is calculated as follows:

$$R\_number\ (p) = \left[\frac{AF_{avg}(p)}{AF_{avg}(f)} \right] = \left[\frac{AF_{avg}(p)(N_F)}{(AF_{avg}(f))_{sum}} \right] \tag{2.5}$$

The approach in LALW proposed Chang, (2008) has been studied by other researchers (Shofurzaman; 2010, Madi et al., 2011, Ming et al., 2012)

2.9 *PeerSim* Simulator

PeerSim is partly developed within the BISON project and is under the General Public Licence (GPL) open source license (Jelasity et al., 2004, 2009; Jamal et al., 2014). BISON project is a three-year Shared-Cost Project (IST-2001-38923) funded by the Future & Emerging Technologies initiative of the Information Society

Technologies Programme of the European Commission. The project runs from January 2003 until April 2006. BISON explores the use of ideas derived from complex adaptive systems (CAS) to enable the construction of robust and self-organizing information systems for deployment in highly dynamic network environments. Consequently, a network simulator, *PeerSim* is developed within the BISON project. *PeerSim* is written in Java language and has been designed to be both dynamic and scalable.

The scalable simulation environment are the contributing factors to the important features in P2P: scalability and dynamism. In *PeerSim,* interaction protocols between peers may either be implemented using a predefined *PeerSim* API or they can be embedded into a real implementation (Jelasity, 2009). *PeerSim* provides a number of pre-developed modules that can be combined in different ways and provides the flexibility to support a variety of different system configurations. The P2P network is modelled as a collection of nodes, where each node has a list of associated protocols. The overall simulation is regulated through initialisers and controllers that allow either events to be introduced into the simulation or to enable a particular capability to be added at predefined simulation time points (Petri et al., 2012, 2014). The component architecture of *PeerSim* is shown in Figure 2.12.

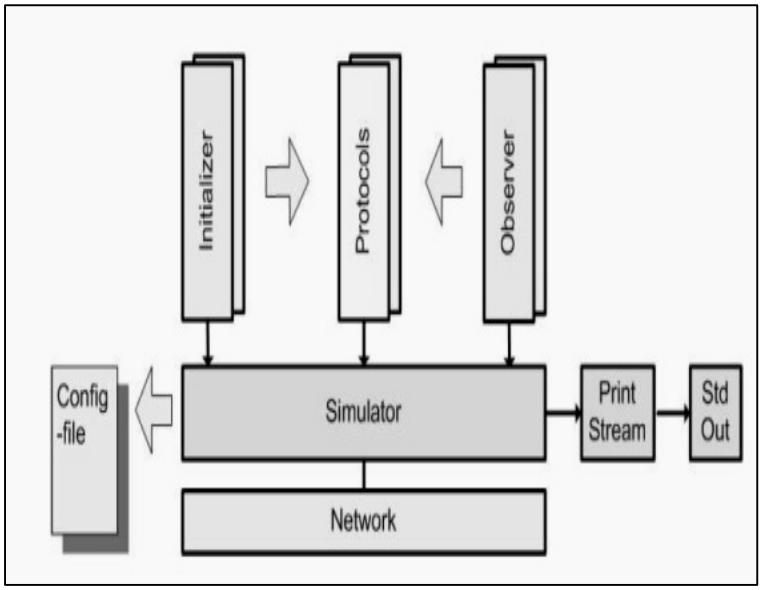

Figure 2.12: *PeerSim* architecture

PeerSim works in two different modes: cycle-based or event-based. The cycle based engine is built on a very simple time scheduling algorithm and is very efficient and scalable. However, it has some limitations. *PeerSim* can achieve a network consisting of 10^6 nodes using the cycle-based engine. As an example it does not model the layer. The event-based engine is based on a more complex but more realistic approach. However, the event based mode is not well documented and its performances are quite unknown.

Further on, the key features of peer-to-peer (P2P) systems are scalability and dynamism. Often the evaluation of a P2P protocol in complex scenarios cannot feasibly be carried out using realistic environments due to issues of scale, cost and availability. It is also difficult to do performance evaluation in a repeatable and control manner due to the dynamic nature of P2P environments. However, *PeerSim* as one of the P2P simulators could provide an extremely scalable simulation

environment that supports dynamic scenarios. Protocols need to be specifically implemented for the *PeerSim* Java API, but with a reasonable effort the protocols can be evolved into a real implementation. Then again, *PeerSim* provides a number of pre-developed modules that can be combined in different ways and provides the flexibility to support a variety of different system configurations. In this thesis, *PeerSim* is chosen as the P2P simulator to evaluate the performance of replica placement of ARPM algorithms. The basic architecture in *PeerSim* has been discussed in chapter 2. In the next section, the detailed configurations set up in *PeerSim* is explained.

2.10 Summary

Data replication is very important in data intensive distributed applications. A number of replica placement strategies are proposed for distributed environments. Most of the work done in the literature discussed in this chapter aimed at increasing the availability and improving data access performance which are the most important factors for replica placement in distributed systems. Replication strategies can be centralised or distributed. In centralised systems, the replica placement decisions are done in a centralised node whilst in the decentralised replication; all the nodes participate in taking decision. The replica placement algorithms may assume different topologies for placement environment. However, in grid, most replica placement algorithm assumed a tree topology whereby the requests can only be forwarded upwards towards the root node. In this chapter, the replica placement algorithms can be popularity based whereby the highest popularity will be selected to be replicated. Some algorithms are threshold based where the files with the access rate higher than the threshold value is considered as popular and will be replicated. Most importantly, the notion of affinity is discussed in this chapter is to highlight the importance of the affinity concept in decision making and replica placement strategy.

The next chapter proposes a method for replica placement mechanism in peer-to-peer distributed system. The proposed Affinity Replica Placement Mechanism (ARPM) aimed to improve data access performance through minimizing the access time and to ensure data availability in P2P distributed systems. Two dominant factors namely affinity and access frequency are formulated in this thesis and as part of the thesis contributions.

CHAPTER 3

AFFINITY REPLICA PLACEMENT

This chapter presents a model for P2P replica placement called Affinity Replica Placement Mechanism (ARPM). The ARPM selects popular files and affinity files for replication, calculates sufficient number of copies and place the replicas on the source node. The objective of this ARPM is to improve data access performance through minimizing the access time and to ensure data availability in P2P systems. In this thesis, the access time is minimised by replicating the popular and affinity files to the requesting node(s). Likewise, to ensure data availability in the P2P network system, sufficient number of replicas is maintained in the system.

3. 1 Introduction

A replication mechanism has three important decisions that affect strongly the performance of the replication strategies. The decisions include which file should be replicated, how many to replicate and where to replicate. In this thesis, the first decision, which file should be replicated, is referred as replica selection phase. The

second decision is referred as replica allocation phase whilst the third decision is referred as replica placement or replica location phase.

In the first phase, replica selection is the problem of selecting files to be replicated. Most of the current dynamic approaches in designing replica placement strategies, focus more on the popularity of the files (Chang et al., 2008; Rasool et al., 2007; shorfuzzaman, 2014). Undeniably, data popularity is considered as a key feature at several levels, namely replication decision strategies (Bsoul et al., 2012), selection strategies (Thampi and Sekaran, 2009), placement strategies (Rasool et al., 2007), replacement strategies (Soosai et al., 2012), load balancing strategies (Senhadji et al., 2013), and update propagation strategies (Wantanabe et al., 2009). In the real world, some files will be more popular than others (e.g. current or "hot" areas of experimentation in ATLAS or CMS). It is worth noting that data in distributed systems may be an object of file, a file or a set of files. It may be also an object of a database table, a database table or a database. Herein, data is also referred as the term dataset.

The second phase refers to the allocation of how many replicas should exist in the P2P distributed systems. The number of replicas should be sufficient enough to ensure data availability in the systems. If the number of replicas selected is too small, data availability decreases. However, if there are too many replicas in the system, data may be overloaded with unnecessary files. This is particularly undesirable toward the beginning of the network lifetime when most nodes are very reliable. Therefore, a good balancing of replicas is required in the P2P distributed systems.

The third phase refers to the placement of replicas. To maximize the potential gain from file replication, a replica placement strategy is also important. A replica placement phase decides where a new file replica should be placed in the system. In this thesis the new replicas will be copied from the destination node (the node that has the requested file) to the source node (the node that initiates the query).

Figure 3.1 and Figure 3.2 show a Venn diagram of the possible scenarios in replica selection phase proposed in this thesis. Selecting files to be replicated can be done by choosing the affinity files out of the popular files in the system or choosing the popular files after finding the affinity files in the system. In this thesis, the affinity files were chosen out of the popular files as shown in Figure 3.1. If affinity is chosen first followed by popularity, the set of files may be the same but the order in which the files would be considered would be different. In many cases, files popularity can change over time. If we just take popularity as a measure a system may over replicate. In addition, there will be lots of replicas which may not be needed. Thus, taking affinity into consideration reduces the number of replicas. One of the primary goals in this thesis is to reduce over replication.

The popular and affinity files were the two dominant factors proposed in ARPM whereby both dominant factors are calculated and discussed in the next section. The access frequency determines the popularity of the access files whilst the affinity degree determines the binding feature between two nodes.

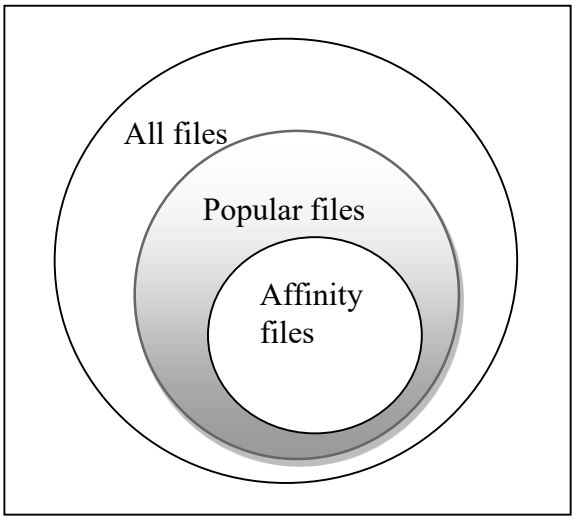

Figure 3.1: The Venn diagram of selecting affinity files from the popular files in the system.

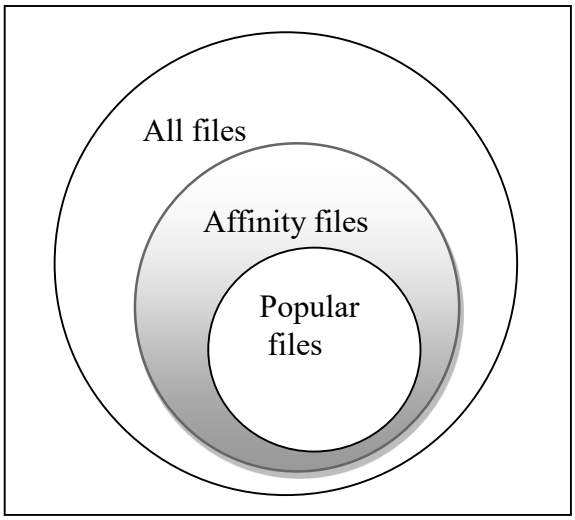

Figure 3.2: The Venn diagram of selecting popular files from the affinity files in the systems.

We present Affinity Replica Placement Mechanism (ARPM) in P2P systems to decide which files to replicate, how many replicas needed to ensure availability of the systems and where to place the new replicas. Herein, replicas are defined as files

instead of objects. The focus is on the file replica placement strategies regardless of what the files contain.

In this thesis, it is assumed that the recent popular files will tend to be accessed more frequently than others in the near future. Thus, an average access frequency threshold on access counts was calculated to determine popularity. If some files have access counts greater than or equal to the threshold, they will be considered to be popular. Next, an affinity degree is proposed in the replica selection phase. Herein, the notion of affinity represents the degree that the files are intersecting with one another.

Normally, a set of files accessed by one user is also likely to be accessed together by other users. This set of files has common features that bind or stick them together. The binding feature, or we defined it as affinity is explicitly exploited in this thesis. An Affinity Replica Placement Mechanism (ARPM) was proposed to highlight the importance of affinity relationship to improve file access performance and assist replica placement decisions. In this thesis a single query and multi queries scenarios were considered. The files in the P2P system were randomly broadcasted.

3.2 The Affinity

The proposed data affinity in this thesis is defined as the similarity between two or more correlated data. The affinity set is a set of any data that creates an affinity between files. Thus, the affinity between sets *A* and *B* is the set consisting of the intersection of elements between *A* and *B* plus the requested file in the destination node, and is not a null set.

Definition 1: Let $A = \{ f_{a1}, f_{a2}, ...f_{an} \}$ and $B = \{ f_{b1}, f_{b2}, ...f_{bn} \}$, f_{jk} is a file from the source node *j* to destination node *k*. The sets *A* and *B* are said to have affinity denoted by aff_{AB}:

$$aff_{AB} = \{ x | x \in (A \cap B + f_{qid}(B)) \neq \emptyset \} \tag{3.1}$$

where $f_{qid}(B)$ is the requested file in *B*.

Definition 2: The affinity degree between *A* and *B* with respect to *A*, aff_{AB}^{A}, is defined as

$$aff_{AB}^{A} = \frac{| aff_{AB} | + f_{qid}(B)}{| A | + f_{qid}(B)} \tag{3.2}$$

where the symbol $| aff_{AB} |$ is the cardinality of affinity set *A* and *B* over *A* including $f_{qid}(B)$ which refers to the number of requested files in *B*.

The value of aff_{AB}^{A} as shown in Equation 3.2, expressing the degree of affinity between the dataset *A* and the affinity sets *AB* with respect to *A*. The affinity function is defined as the cardinality of the affinity dataset between *A* and *B* over the cardinality *A*. Likewise the degree of affinity between *B* and *A* with respect to *B* is defined as the cardinality of the affinity set *A* and *B* over *B*.

Example 1 below shows how the proposed affinity degree is calculated.

Example1:

let $A = \{ f_{11}, f_{12}, f_{13}, f_{14}, f_{15} \}$ and $B = \{ f_{21}, f_{22}, f_{23}, f_{13}, f_{14}, f_{15}, f_{26}, f_{27}, f_{28} \}$, and the requested fileId is f_{28}. Therefore the affinity degree over A

$$= \frac{|\{f_{13}, f_{14}, f_{15}\}| + f_{28}}{|\{f_{11}, f_{12}, f_{13}, f_{14}, f_{15}\}| + f_{28}}$$

$$= 4/6$$

$$= 0.67 \text{ (moderate)}$$

Table 3.1 shows a categorisation of affinity correlation adapted from Dancey and Reidy (2004). The correlation of an affinity degree indicates that not every correlation deserves to investigate and some filtering mechanisms can be adopted to remove those files with weak correlation. In general, the higher the absolute value of affinity correlation coefficient, the stronger the relationship between the two nodes in the P2P network. For example, in Table 3.1, if the value of the aff_{AB}^{A} is equal to 0.49 or below, this indicates that the degree of the affinity files is weak and thus can be ignored. In this case, the file has weak affinity and will not be replicated.

Table 3.1: The affinity degree indicator (Adapted from Dancey and Reidy, 2004)

Value of the aff_{AB}^{A}	**The Degree of the affinity files**
$0.9 \leq x < 1.0$	Very Strong
$0.7 \leq x < 0.9$	Strong
$0.5 \leq x < 0.7$	Moderate
$0.1 \leq x < 0.5$	Weak
$x < 0.1$	Zero

Likewise, if the value of the affinity degree is either moderate, strong or very strong, then the file will be replicated. The explanation is detailed in the next paragraph. The representations of the affinity files are as follows:

Table 3.2: Example of affinity degree

A	B	f_{qid}	$(A \cap B)+ f_{qid}$	$aff_{AB}^{A} = \frac{\lvert aff_{AB} \rvert + f_{qid}(B)}{\lvert A \rvert + f_{qid}(B)}$	*Affinity Indicator*
{1,2,3,4}	{1,2,3,4,5,6}	6	5	5/5 = 1.0	Very strong
{1,2,3,9}	{1,2,3,4,5,6,7, 8}	5	4	4/5 = 0.8	Strong
{1,2,3,4,7, 9,10, }	{1,2,3,4,5}	5	5	5/8 = 0.61	Moderate
{1,2,3,4,5,6,7,8, 9,10,11,12}	{1,13}	13	2	2/13 = 0.15	Weak
#500	#1000	#20	300	300/520=0.58	Moderate
#1000	#5000	#50	300	300/1050 = 0.29	Weak

is the number of files

If the value of aff_{AB}^{A} is near to 1, we can say that the affinity set between files is very strong whilst if the value of aff_{AB}^{A} is near to zero, we can say that the degree of affinity set between files is very weak or zero affinity. Through the affinity indicators, we can predict on how strong or high and how weak or low the affinity set between files in the nodes. This means that if the strength of similarity files is high, and if the average frequency of the access number of the file requested is also high, ARPM will choose the file to be replicated. This answers the issue of which file to replicate in replica placement problems. Despite this, if the degree of the affinity set is weak or zero, ARPM will NOT consider the file to be replicated regardless of how high the value of the file access frequency. The decision of replica placement depends on the affinity degree and the average number of access frequency. In the next section, the access frequency as another criteria for replica selection is discussed.

3.3 Access Frequency

ARPM only consider affinity and popular files to replicate (deciding which file to replicate). An access frequency, *AF* is calculated to represent the importance of access histories in different cycle number. Assume N_t is the cycle number passed, *F* is the set of files that have been requested and a_f^t indicates the number of accessed files in each cycle. Then *AF* is adapted from the calculation of *AF* in (Chang and Chang, 2008):

$$Access\ Frequency = \ AF_{Nt}(f) = \sum \left(a^t f \times 2^{-(N_t - t)} \right), \quad \forall f \in F_t = 1 \qquad (3.3)$$

For example, if an affinity file has been accessed 7 times and 10 times in the first cycle and second cycle, respectively, then $AF(f)$ is $(7 \times 2^{-1}) + (10 \times 2^{0})$. AF assigns different weights to access files for a different cycle number. The highest or largest AF is chosen as the popular files. Next we compare the average AF per cycle number of the popular files. The average AF is calculated as:

$$\text{Average Access Frequency} = AF_{N_c}^{average}(f) = \sum AF_{N_c}(f)/N_c \quad , \quad \forall f \in F \qquad (3.4)$$

$N_F = |F|$ is the number of different files that have been requested by any nodes. The threshold value of access frequency is considered as the average of access frequencies in the systems. If the access frequency is above or equal to the average access frequency, then we categorise it as "high" or "popular". Likewise, if the access frequency is below than the average frequency, then we categorise it as "low" or "unpopular". Table 3.3 shows which file to replicate based on the two dominant factors proposed in this thesis.

Table 3.3 Dominant factors which file to replicate

Affinity Indicator	#Average Access Frequency	Replicate	Not Replicate
Very Strong	High	1	
	Low		0
Strong	High	1	
	Low		0
Moderate	High	1	
	Low		0
Weak	High		0
	Low		0
Zero	High		0
	Low		0

Note: 1 = Yes 0 = No

The primary goal of the algorithm is to increase data access performance from the perspective of the clients by dynamically creating replicas for "popular" files. In the real world, some files will be more popular than others and data access patterns may change over time, so any dynamic replication strategy must keep track of file access histories to decide on when, what and where to replicate. The "popularity" of a file is determined by its recent access rate by the clients. Identifying popular files is thus one of the dominant factors of ARPM. In ARPM, popular data files are identified by analysing file access histories. The replica placement algorithm is invoked at regular intervals and it processes the access histories and affinity degree to determine new replica locations based on file popularity and affinity.

3.4 Replica Selection Decisions

This section focuses on the decisions in replica selection phase. In this section, the affinity property from Table 3.3 has been transformed into Table 3.4 in Boolean-valued data. In Boolean-valued data, the dominant factor is holding either a value 0 or 1. In this Boolean representation, the aim is to qualify the different importance of linguistic terms of vague terms of affinity factors which include very strong, strong, moderate, weak and zero.

Table 3.4 Dominant factors which file to replicate in Boolean representation

Affinity Indicator	#Average Access Frequency	Replicate	Not Replicate
1	**1**	**1**	
	0		0
1	**1**	**1**	
	0		0
1	**1**	**1**	
	0		0
0	1		0
	0		0
0	1		0
	0		0

Definition 3: Let affinity and average access frequency be two dominant factors for replica placement. The replica placement occurs when both dominant factors are equal to 1 respectively.

The Boolean representations in Table 3.4 are used as indicators to decide whether to replicate or not. The replica placement occurs when both dominant factors are equal to 1. Indeed, if the affinity degree is high and the access frequency exceeds the

threshold value of the average number of accesses, or if both values are equal to1, then the decision to replicate is made.

3.5 Access Frequency as Dominant Factor

This section describes four cases considered in this thesis in selecting popular data files and calculating the files affinity degree. Case-1*: Single-Query to Single-File,* Case-2*: Single-Query to Multiple-Files*, Case-3*: Multiple-query to Single-File* and Case-4*: Multiple-query to Multiple-Files*. Based on these various queries, both dominant factors play important roles in influencing the decision of replica placement. Table 3.5 shows the single and multi-queries scenarios between the requestor/source node(s) and the query file (s). During experimentation, the number of cycles and files are increased whilst the number of nodes simulated is up to 10000 nodes.

Table 3.5: The single query and multiple query scenarios

Cycle Number	Requestor NodeId	FileId	
0	3	28	
1	39	23	Case-1
2	92	31	
3	67	25	
4	97	15	
5	63	6	
6	42	19	
7	69	25	Case- 2
8	31	3	
9	1	29	
10	97	17	
11	50	21	
12	54	8	
13	32	12	Case- 3
14	46	12	
15	71	3	
16	25	22	
17	31	6	
18	14, 15, 37	9, 27,33	Case- 4
19	91	30	
20	28	19	

3.5.1 Case 1: Single-Query to Single-File

In Table 3.4 during *cycle1*, a *NodeId 39* requests for a *FileId23*. This is a case of a *Single-Query to Single-File* request whereby only one client node is requesting for one file in the systems during a period of time. This refers to the cycle number between *cycle0* to *cycle20*. This is the case of no replication.

3.5.2 Case 2: Single-Query to Multiple- Files

In *cycle4* and *cycle10*, the same *NodeId 97* is requesting two different files, *FileId15* and *FileId17*. This is the case of the same client node requesting two files in the systems during a period of cycles. Table 3.6 and Table 3.7 show an example of historical records of the *NodeId97* during the first and the second time interval respectively. Assume N_T is the number of time interval passed, *F* is the set of files that have been requested and a_f^t indicates the number of accesses for file *f* at time interval *t*.

In the first time interval, *t* = 1, *FileId15* have been requested by *NodeId4* times and 10 times during the second time interval, *t* =2. Then The *Access Frequency (AF)* for each file can be calculated as:

$$Access\ Frequency = \ AF_{Nt}(f) = \sum \left(a^t f \times 2^{-(N_t - t)} \right), \quad \forall f \in F_t = 1$$

Thus for *FileId15*, *Access Frequency* $= AF_{N_t}(f) = \left(4 \times 2^{-(1-1)}\right) + \left(10 \times 2^{-(2-1)}\right)$

$= 12$

Table 3.6: An example of access frequency for *Single-Query to Multiple-Files* at time interval *t=1*

a_f^t	Requestor *NodeId*	*FileId*	Number of Access Frequency
4	97	15	4
10	97	17	10
2	97	21	2

Table 3.7: An example of access frequency for *Single-Query to Multiple-File*s at time interval *t=2*

a_f^t	Requestor NodeId	FileId	Number of *Access Frequency*
10	97	15	5
5	97	17	2.5
3	97	21	1.5

Based on equation 3.3, number of access frequency for *File15*, *FileId17*, and *FileId21* were 5, 2.5, and 1.5 respectively. Therefore, the threshold of the average access frequency in the period of cycle can be calculated as in 3.4. The average *threshold* is 4.17. Therefore two files with *FileId15* and *FileId17* are above the *threshold* value that is considered as popular files. These files will be selected to be replicated if the affinity degree for these files is moderate, strong, or very strong.

3.5.3 Case 3: Multiple-Query to Single-File

In *cycle13* and *cycle14*, two different node ids, *NodeId32* and *NodeId46* request the same *FileId12*. This is the case of different client nodes requesting the same file in the systems during a period of time. Table 3.8 until 3.13 below show an example of the *Multiple-Query to Single-File* case whereby many nodes request a single file.

Table 3.8: An example of access frequency for *Multiple-Query to Single-File* at time interval *t=1*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency* (popular)
5	32	17	5
8	46	17	8
10	25	17	10

Table 3.9: An example of access frequency for *Multiple-Query to Single-File* at time interval *t=2*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
10	32	17	5
12	46	17	6
5	25	17	2.5

Aggregate *AccessFrequency* for *FileId17* = 18.25

Table 3.10: An example of access frequency for *Multiple-Query to Single-File* at time interval *t=2*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
10	32	15	4
8	46	15	4
10	25	15	5

Table 3.11: An example of access frequency for *Multiple-Query to Single-File* at time interval *t=2*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
5	32	15	2.5
12	46	15	6
5	25	15	2.5

Table 3.12: An example of access frequency for *Multiple-Query to Single-File* at time interval *t=2*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
2	32	21	2
8	46	21	4
2	25	21	2

Table 3.13: An example of access frequency for *Multiple-Query to Single-File* at time interval *t=2*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
3	32	21	1.5
12	46	21	6
5	25	21	2.5

The average Threshold for access frequency calculated is 12.5. Thus, the popular file in this case is *FileId17*. The file is selected to replicate in the next phase. Another two files, *FileId15* and *FileId21* below the threshold average access frequency and these files are less popular.

3.5.4 Case 4: Multiple-Query to Multiple-Files

In *cycle18*, different *NodeIds 14, NodeId15* and *NodeId37* are requesting multi different files (*FileId9, FileId27, FileId33*) in the systems. This is the case of multiple queries requesting multiple files in the same cycle or at that point of time. Table 3.14 until Table 3.17 show an example of Multiple-Query to Multiple-Files case.

Table 3.14: An example of access frequency for *Multiple-Query to Multiple-Files* at time interval *t=1*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
5	32	15	5
8	46	17	8
10	25	21	10

Table 3.15: An example of access frequency for *Multiple-Query to Multiple-Files* at time interval *t=2*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
4	14	15	2
10	15	17	5
2	37	21	6

Table 3.16: An example of access frequency for *Multiple-Query to Multiple-Files* at time interval *t=3*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
10	14	15	2.5
5	15	17	1.5
5	37	21	1.5

Table 3.17: An example of access frequency for *Multiple-Query to Multiple-Files* at time interval *t=4*

a_f^t	Requestor *NodeId*	*FileId*	Number of *Access Frequency*
10	14	15	1.3
10	15	17	1.3
7	37	21	0.9

(a) The first time interval

$t_1(2^0)$ - Aggregated Records

FileId	*Average Number AF*
15	30
17	27
21	15

Phase 1

$AF\,(15) = 30 \text{ x } 2^0 = 30$

$AF\,(17) = 27 \text{ x } 2^0 = 27$

$AF\,(21) = 15 \text{ x } 2^0 = 15$

Phase 2: (Popular is *FileId15*)

$AF_{avg}(p) = 30/1 = 30$

$AF_{avg}(all) = (30 + 27 + 15)/(3*1) = 72/3 = 24$

(b) Case involving Two Time Intervals

$t_1(2^0)$ - Aggregated Records

FileId	*Average Number AF*
15	30
17	27
21	15

$t_2(2^{-1})$ - Aggregated Records

FileId	*Average Number AF*
15	10
17	15
21	12

Phase 1

$AF\,(15) = 30 \text{ x } 2^0 + 10 \text{ x } 2^{-1} = 35$

$AF\,(17) = 27 \text{ x } 2^0 + 15 \text{ x } 2^{-1} = 34.5$

$AF\,(21) = 15 \text{ x } 2^0 + 12 \text{ x } 2^{-1} = 21$

Phase 2: (Popular is *FileId15*)

$AF_{avg}(p) = 35/2 = 17.5$

$AF_{avg}(all) = (35 + 34.5 + 21)/(3*2) = 90.5/6 = 15.08$

3.6 Number of Replicas

In order to have better network performance, the number of replicas (adapted from Chang, 2008) needed for the popular file is been calculated. The number of replicas is calculated as follows:

$$Number\ Of\ Replicas\ (p) = ceiling\left[\frac{AF_{avg}(p)}{AF_{avg}(f)}\right]$$

$$= ceiling\left[\frac{AF_{avg}(p)(N_F)}{(AF_{avg}(f))_{sum}}\right] \quad (3.5)$$

where $AF_{avg}(p)$ is the average access frequency of the popular file *p*, and $AF_{avg}(f)$ is the average access frequency of other files in the system. The number of replicas acts as a threshold checker to determine sufficient replicas exist in the system.

As an example, from section 3.5.4 above, at the first time interval, *FileId15* is the popular file. Thus, from the formula in 3.5, the number of replicas is calculated as follows:

$$Number\ Of\ Replicas\ (p) = ceiling\left[\frac{AF_{avg}(p)}{AF_{avg}(f)}\right] = ceiling\left[\frac{AF_{avg}(p)(N_F)}{(AF_{avg}(f))_{sum}}\right]$$

$$= ceiling(30/\ 24) \quad = 2$$

At the second time interval, the number of replicas is calculated as follows:

$$Number\ of\ Replicas\ (p) = ceiling\left[\frac{AF_{avg}(p)}{AF_{avg}(f)}\right] = ceiling\left[\frac{AF_{avg}(p)(N_F)}{(AF_{avg}(f))_{sum}}\right]$$

$$= ceiling(17.5/15.08) \quad = 2$$

This indicates that two replicas of popular *FileId15* need to be created in the system at both time intervals. The next phase of our proposed ARPM is finding an affinity degree of the correlated files.

3.7 Affinity Degree as Dominant Factor

The second dominant factor will be calculated based on the affinity degree between the source node and the destination node. Table 3.18 shows the *NodeId* and the *FileId* whilst Table 3.19 shows the discovery layer where the file requested by the source node is found in the destination node. This also refers to the success hit whenever a query file is found in the destination node.

Table 3.18: An example of *NodeId* and *FileId*

NodeId	***FileId***
40	23, 6, 34, 36, 17, 30, 15, 29, 19, 22
26	29, 39, 42, 27, 23, 21, 6, 5
39	10, 44, 43, 40, 21, 48
25	10, 44, 43, 40, 18, 3, 6
46	42, 1, 41, 14, 3, 31, 13
27	31, 26, 25, 4, 28, 37
11	6, 43, 38, 24, 19, 23, 7, 32
24	19, 12, 15, 28, 2, 25, 37, 27
97	30, 48, 25, 7, 22, 19
14	23, 17, 36, 34, 40, 29, 51
32	40, 10, 44, 48, 43, 31,13

Table 3.19: An Example of Success Hit

SourceNode	*FileId*	*DestinationNode*
14	15	24,40
40	1	46
18	1	46
32	21	39
16	23	11,26
10	3	25
97	17	40
25	21	26,39
46	21	26,39
97	15	24,40
18	21	26,39

In section 3.2, the definition of affinity and how to calculate the affinity degree has been discussed in detail. In this section, the affinity degree is calculated based on the formula from 3.1 and 3.2. The affinity degree as the second denominator will be calculated using similar four cases as in section 3.5.

3.7.1 Case 1: Single-Query to Single-File

In a case of a *Single-Query to Single-File* request, only one client node is requesting one file in the system during a period of time. There is no replication and thus affinity degree is will not be calculated in this case.

3.7.2 Case 2: Single-Query to Multiple-Files

In this case, the same node is requesting two or more files in a fixed time interval. Prior to this, an average access frequency has been calculated in section 3.5 and the popular files were found. As calculated in section 3.5, only *FileId15* and *FileId17*

are popular whereas *FileId21* is below average frequency threshold and therefore is considered as less popular. Next, the affinity degree is calculated between the source node, *NodeId97* and the destination node, *NodeId40*, as shown in Table 3.9. The affinity degree is calculated as below:

Example 1:

Let source/Query node be S_{97} and the destination node be D_{40}. The query file is *FileId17*.

S_{97} = *{30, 48, 25, 7, 22, 19}* and

D_{40} = *{23, 6, 34, 36, 17, 30, 15, 29, 19, 22}*

The affinity is

$$aff^{S_{97}}_{S_{97}D_{40}} = S_{97} \cap D_{40} + \text{Requested File in } D_{40}$$

$$= \{22, 30, 19, 17\} = 4$$

From equation in 3.2, the affinity degree over S_{97},

$$= \frac{|\, aff_{S_{97}D_{40}} \,| + f_{qid}(D_{40})}{|\, S_{97} + f_{qid}(D_{40}) \,|}$$

$$= 4 / 7$$

$$= 0.57 \text{ (Moderate affinity)}$$

Example 2:

Let source node be S_{97} and the destination nodes be D_{24} and D_{40}. The query file is *FileId15*.

S_{97} = *{30, 48, 25, 27, 22, 19}* and

D_{24} = *{19, 12, 15, 28, 2, 25, 37, 27}*

$$aff^{S_{97}}_{S_{97}D_{24}} = S_{97} \cap D_{24} + \text{Requested File in } D_{24}$$

$$= \{15, 19, 27, 25\} = 4$$

From equation in 3.2, the affinity degree is

$$= 4/ 7$$

$$= 0.57 \text{ (Moderate Affinity)}$$

By calculating the affinity degree of the files between the source nodes and the destination nodes using the proposed affinity formula, the affinity degree in example 1 indicates that the relation is strong. Therefore we can conclude that, *FileId17* is a popular file and the nodes (the source node and the destination node) has strong relation. Not only *FileId 17* will be replicated but also all the intersection files that represent the affinity data, will be replicated as well to the source node. However, in example 2, the affinity degree calculated indicates "weak affinity". The *FileId15* will not be replicated since the affinity degree is "low" despite of the file is popular.

The rationale is that, when a user generates a request for a file, large amount of bandwidth could be consumed to transfer the file from the server to the client. Furthermore, the popular files tend to be accessed more frequently than less popular files in the near future. Therefore to select a popular file in the replica placement strategy is very important. In the real world most of the files have affinity with one another. A user searching for one song from "The Beatles", may search for another song from the same music group. A researcher from a university may need more than one related journal or research file from other university. These two examples of searching and accessing files need to be done repeatedly. As a consequence, not only the total access cost is increased but also the total communication cost in

accessing the files. However, the increase of both costs can be reduced if related files are copied instead of just one file per request from the client.

Therefore, the idea behind ARPM is to create a set of replicas where affinity and popularity are equally important and significant criteria in replica placement strategy. Besides, ARPM place the new replicas as close as possible to those clients that frequently request the corresponding files, subject to storage availability. The effectiveness of this ARPM algorithms also depend on the number of accesses threshold value and the proximity threshold value that were used herein to determine the placement of replicas in the P2P systems.

3.7.3 Case 3: Multiple-Query to Single-File

This is the case of different client nodes requesting the same file in the systems during a period of time. Table 3.7 and 3.8 show example of *Multiple-Query to Single-File* case whereby many nodes request a single file. As mentioned in 3.5.3, only one file is requested by multiple source nodes. In the example in 3.5.3, the affinity degree for popular *FileId21* can be calculated as below:

Example 1:

Let source node be S_{32} and the destination node be D_{39}. The query file is *FileId21*.

S_{32} = *{40, 10, 44, 48, 43, 31, 51, 13}* and
D_{39} = *{10, 44, 43, 40, 48, 31, 34, 54, 21}*
$S_{32} \cap D_{39} + f_{qid}(D_{39})$ = *{40, 10, 44, 43, 48, 31, 21}* = 7

From equation in 3.2, the affinity degree

= 7/ 8
= 0.86 (Strong Affinity)

Example 2:

Let source node be S_{46} and the destination node be D_{39}. The query file is *FileId 21*.

S_{46} = {42, 1, 41, 14, 3, 31, 13} and
D_{39} = {10, 44, 43, 40, 48, 21}
$S_{46} \cap D_{39} + f_{qid}(D_{39})$ = {21} = 1

From equation in 3.2, the affinity degree

= 1 / 8
= 0.13 (Weak Affinity)

Let source node be S_{25} and the destination node be D_{39}. The query file is fileId 21.

S_{25} = {10, 44, 43, 40, 18, 3, 6} and
D_{39} = {10, 44, 43, 40, 48, 21}
$S_{32} \cap D_{39} + f_{qid}(D_{39})$ = {10, 44, 43, 40, 21} = 5

From equation in 3.2, the affinity degree

= 5/ 8
= 0.63 (Moderate Affinity)

In the above examples, the requested file(s) from the destination node D_{39} will be replicated to the source node *NodeId32* and *NodeId25*. The relatedness of these source nodes with the destination nodes are "high" as indicated by their affinity degree of the files between the source nodes and the destination nodes. In contrast, example 2 indicates that the relation is “weak”. Therefore, the requested *File21*,

although it is popular, but the file will not be replicated to the source node due to its *weak* or "low" affinity degree.

3.7.4 Case 4: Multiple-Query to Multiple-Files

This is the case of multi queries from many source nodes request many files in the system in during a certain time intervals. The affinity degree can be calculated for each source nodes that request popular files. For example, *FileId15* is the popular file requested by the source *NodeId14*, thus the affinity degree can be calculated as below:

Example 1:

Let source node be S_{14} and the destination node be D_{40}. The query file is *FileId15*.

$$S_{14} = \{23, 17, 36, 34, 40, 29, 51\} \text{ and}$$
$$D_{40} = \{23, 6, 34, 36, 17, 30, 15, 29, 19, 22\}$$
$$S_{14} \cap D_{40} + f_{15}(D_{40}) = \{23, 17, 36, 34, 15\} = 6$$

From equation in 3.2, the affinity degree

$$= 6 / 8$$
$$= 0.75 \text{ (Strong Affinity)}$$

The affinity degree in example 1 above indicates that the relation is strong. Therefore we can conclude that, *FileId15* is a popular file and the nodes (the source node and the destination node) has strong relation. Therefore, all the intersection files will be replicated to the source node.

In the *Multiple-Query to Multiple-Files* case, for each source node that request a file, if the file is popular and the affinity degree calculated is "high", then all the affinity files will be replicated to the source nodes, subject to the storage availability.

3.8 ARPM System Model

The P2P network system model considered in this thesis consists of a set of *N* storage nodes or simply called nodes. Herein, the nodes are interconnected with one another and each node at most is linked to three neighbouring *nodes (k).* On these nodes replicas of *files (r)* are stored representing data aggregates such as documents, web directories, or research materials.

In the network systems, users generate read accesses to the files located on the servers. Herein, it is assumed that at least one file exist. Until now, in any replica placement strategies, three important decisions which affect strongly the performance of the replication strategies proposed are:

i. Which file to replicate? Replica selection. Selecting target replicas depends on the popularity and importance of the relatedness of the files or their affinity degree. This can be gained by tracing users' access history and finding the affinity degree of the queried files. This thesis focus on read-only access as the file access type and consistency issue is not considered in this thesis.

ii. How many to replicate? - Replica allocation. In addition to the popularity and the affinity degree, the access bandwidth of peers affect strongly the decision of the number of replicas. In this thesis, the number of replicas threshold is calculated after the threshold of the average number of access frequency is calculated in each cycle numbers. The number of replicas threshold ensures that sufficient replicas exist in the systems.

iii. Where to place the replicas? - Replica location. The location of replicas in the ARPM services tier decides where the created replicas should be placed. Herein, the threshold proximity is set in the configuration text file. It is assumed that if the number of hop or distant between the source node and the destination node exceeds the proximity threshold, the new replicas will be replicated to the nearest neighbours and not the source node. The proximity threshold is an indicator whether the distant of the source node is closer or far from the destination node. This indicator is very important to place replicas in appropriate locations so as to reduce access latency.

Figure 3.3 shows the Affinity Replica Placement Mechanism (ARPM) for 3 tier architecture. The concept of affinity is used to model the framework to make decisions of selecting which files to replicate based on the correlated files receive from a source node. In ARPM model, there are three important components namely: Affinity, Placement and Replication. Replication is executed after the selection of popular and affinity files in the affinity component and after deciding where should the new replicas be placed in the placement component.

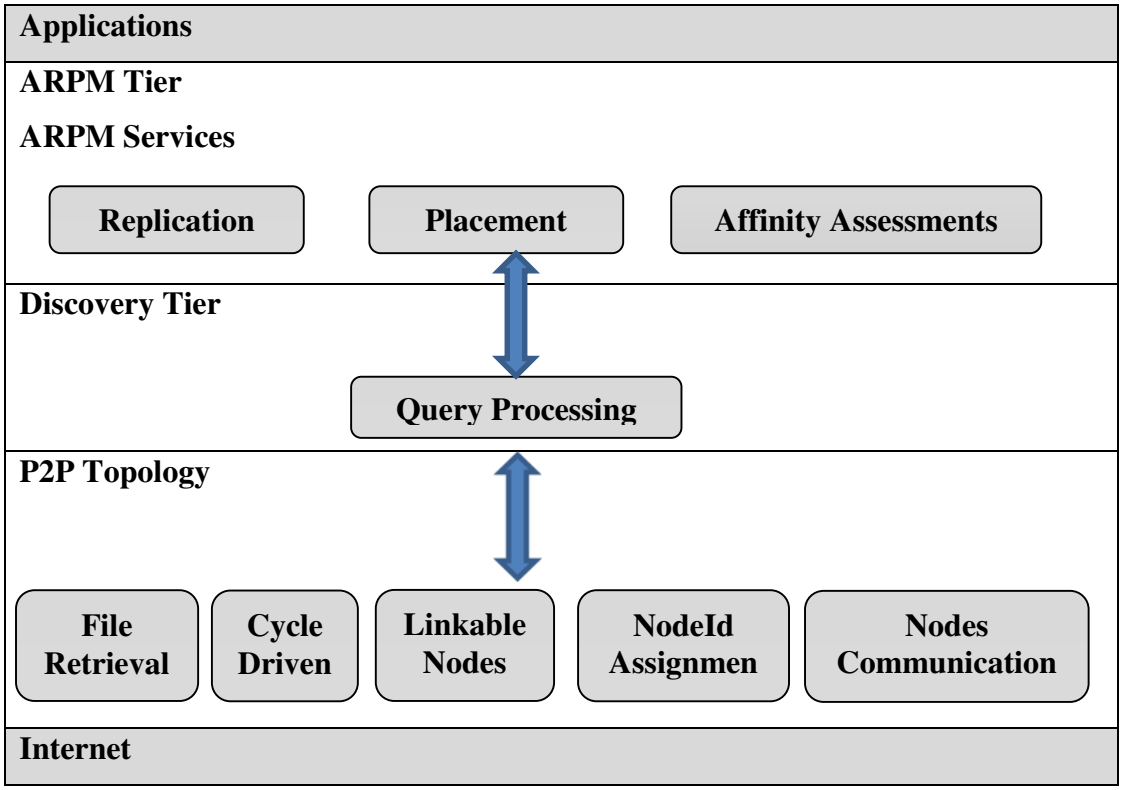

Figure 3.3: Affinity Replica Placement Mechanism (ARPM) 3 tier architecture

3.9 ARPM Algorithms

Under the ARPM algorithm, file(s) will be replicated if the affinity indicator value is moderate, strong, or very strong and only if the number of access frequency during that cycle is high. These affinity indicator values and the selected value of popular files show that only qualitative files are chosen to be replicated instead of quantitative files in a large dataset. Further on, the demand of large scale of data in P2P networks is likely to increase in future. However, the large scale of data makes it impractical to replicate all the data on every node that request the files. Hence, for efficient access, a sustainable mechanism to decide which files to replicate according to its

common interest or relatedness (affinity) and the file's popularity (number of access frequency) is highly recommended. The results of better access performance has been shown in chapter 5.

Without loss of generality, lets assume there are a set of files in each node. Lets further assume that at least there is one R replica in the network. Figure 3.4 to Figure 3.6 show the algorithm of the proposed ARPM. The algorithm takes the *data file (* f_i *),* the query id of *a node q (nodeid, fileid)* that holds the node that request a file(s) in the network systems as the input.

Algorithm 3.1 : *CalculateAccessFreq*

Function *accessfreq (NodeId, FileId, NumberOfAccess)* {
a. *Calculate accessfrequency*
b. *Calculate average accessfrqeuency //threshold*
c. **If** *accessfrequency >= threshold*
calculate number of replicas
Tabulate popular file
Call Function *CalculateAffinityDegree()*
else
file will not be selected
} **End function**

Figure 3.4: Algorithm for access frequency

Algorithm 3.2 : *CalculateAffinityDegree*

Function *CalculateAffinityDegree* {

a. *compare the file in sourcenode and destination node*
b. *Calculate affinitydegree*

if *the affinitydegree == 1*

callPathAffinity

else

do not replicate //(Number of replicas are sufficient)

} **End function**

Figure 3.5: Algorithm for affinity degree

Algorithm 3.3 : *PathAffinity*

Function *PathAffinity(HopCount)*{

a. *Calculate the distance from the source node (requestor node) to the destination node (hopCount)*
b. *Replicate the file to the requester node*

} **End Function**

Figure 3.6: Algorithm for path affinity

3.10 Summary

The model of ARPM is proposed for the replica placement in P2P systems. The primary objective of ARPM is to improve data access performance through minimizing the access time and to ensure data availability in P2P systems. In this thesis, the access time is minimised by replicating the popular and affinity files to the requesting node(s). Likewise, to ensure data availability in the P2P network system, sufficient number of replicas is maintained in the system. The replica placement approaches addressed the fundamental issues in replica placement: which file to replicate, how many to replicate and where to replicate. The access frequency and the affinity degree were proposed as the two dominant factors and formulated in ARPM. On the contrary, most of the replica placement algorithms are based on the popularity of the files to replicate data.

In the next chapter, the implementation of the proposed ARPM based on simulation will be discussed. The performance of the proposed model presented here is not limited to single query but also to multiple queries request from source node to the destination nodes.

CHAPTER 4

SIMULATION BASED AFFINITY

This chapter presents the implementation of replica placement in the proposed Affinity Replica Placement Mechanism (ARPM) for P2P systems. In this thesis, simulation is chosen to assess the effectiveness of ARPM replica placement algorithms. The performance of ARPM was validated and evaluated through experimentation in *PeerSim*, a peer-to-peer simulator.

4.1 Introduction

When a user generates a request for a file, large amounts of bandwidth could be consumed to locate the appropriate node that has the file and to transfer that file to the requester node. In general, one request may lead to another file request that is correlated to the file that has been requested earlier. A set of files accessed by one user is also likely to be accessed together by other users. Similarly, a set of related files is often requested and accessed by multiple users.

The degree that these set of files are referencing together in multiple queries scenario is computed through affinity algorithm proposed in this thesis. The proposed ARPM highlighted the importance of popular files and its affinity relationship to improve file access performance and assist replica placement decisions. Moreover, the performance of the proposed model presented here is not limited to a single query but also to multiple query from requester or source nodes to the destination nodes. In this thesis, to simplify the requirements, we assumed that file is read only. How these queries in a fixed number of cycles contribute to the replica placement performance is also discussed in this chapter.

4.2 System Parameters

In this thesis, *PeerSim* is chosen as the P2P simulator to evaluate the performance of replica placement of ARPM algorithms. The basic architecture in *PeerSim* has been discussed in Section 2.7, Chapter 2. Table 4.1 shows the simulation parameters. In this simulation, a range of P2P random topology composed of 20, 50, and 1000 nodes were tested. The number of cycles was maintained to 50 cycles.

The next section describes the implementation of the simulation environment of the ARPM model.

4.3 Simulation Parameters

Replica placement in ARPM is done in cycle-based mode that runs in a sequential order. In each cycle, each protocol can run its behaviour. ARPM three tier architecture as explained in chapter 3, section 3.8 is simplified in Figure 4.1. Figure

4.1 shows ARPM tiers which were divided into three layers namely: the topology layer, the discovery layer, and the ARPM Replica Placement layer. The next section describes the implementation of the simulation environment of the ARPM model.

Table 4.1 Simulation parameters

Parameter	Value	Description
Simulation Cycles	50	Number of cycles in the simulation
Network Size	20, 50, 1000	Number of peers in the network
Access Frequency	Depends on the access frequency calculated for each time interval	The value calculated in the formula indicates the access frequency whether the file is popular or less popular.
Affinity Degree	Depends on the affinity degree calculated for each time interval	(1) is the value that indicates the dataset of files has moderate to strong affinity degree and (0) is the value that indicated the affinity degree between two datasets is weak.
k	3	Number of neighbours is initialised to 3
TTL (Time-to-Live)	7	Time to Live for forwarding query
Replica Threshold	Depends on the value calculated in the formula	The threshold as a checker to a number of replicas allowed in the network.
Proximity Threshold	3	The threshold as a checker to the distance between source node to the destination node in hop count.

4.4 ARPM Implementation

Replica placement in ARPM is done in cycle-based mode that runs in a sequential order. In each cycle, each protocol can run its behaviour. ARPM three tiers architecture is as explained in chapter 3, section 3.7 is simplified in Figure 4.1. The Figure 4.1 shows ARPM tiers which were divided into three layers namely: the topology layer, the discovery layer, and the ARPM Replica Placement layer.

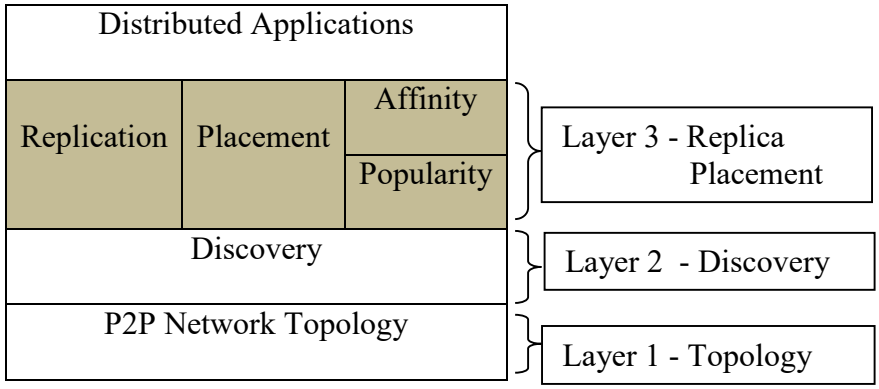

Figure 4.1 ARPM 3-tier architecture

4.4.1 The Topology Layer

In the topology layer, each node has a list of associated protocols. The overall simulation is regulated through initializers and controllers. The topology layer defines the configuration of P2P overlay network and provides an interface to the discovery layer. This layer involves building a topology of the P2P network including the number of nodes, how they are connected from one node to another, and the distribution of files across the nodes in the network.

PeerSim.init.WireKOut is used as the first initializer to perform the wiring of the static overlay network having the specified *degree (k)* parameter which is set to 3. The parameter *k* represents the maximum number of neighbouring nodes. In this simulation, each node has 3 neighbouring nodes linked to the node.

4.4.2 The Discovery Layer

The files in the discovery layer were disseminated among nodes randomly in the peer-to-peer networks. The storage for all nodes is assumed unlimited and all the files were considered to have the same size. In this simulation, each node is connected to 3 neighbours and the order in which the files are requested follow random walk *Gaussian distribution*. In the discovery layer, a node sends a query to its neighbouring nodes to find the queried data file. If the neighbour has the data file, it responds to the source node. Otherwise, the source node sends the query message to the neighbours until the queried data file is found or the *TTL (Time-To-Live)* is expired. The *TTL* is used to control the number of hops propagated. In this thesis, a single query and multi queries were considered. In a single query, a node can request a single file and a node can also request a number of files in one cycle. Whereas, in the multi queries, many nodes can either request a single file or request for multiple files in a number of cycles.

Table 4.2 is similar to Table 3.6 in chapter 3 which shows an example of a simple single query and multiple query scenarios for 20 nodes. For example, in *cycle0*, *NodeId3* requests a *FileId28 (Single query*). In *cycle4* and *cycle10, NodeId97*

requests a *FileId15* and *FileId7* respectively (Single-Query to Multiple-Files). An example of *Multiple-Query to Single-File* is in *cycle5* and *cycle17*, where two different nodes request the same file. Lastly, in the case of *Multiple-Query to Multiple-Files*, different nodes requesting different files in the system. This reflects the real scenarios in the P2P collaborating research group, but with larger picture whereby multi queries node or clients requesting multi files in the network.

4.4.3 ARPM Replica Placement Layer

This layer is divided into three phases: Replica Selection, Replica Allocation and Replica Placement. The configuration file is read at the beginning of the simulation. Each node maintains an access record of the files. The record is in the format <*Cycle number, NodeId, FileId*>. Files shared here were assumed to be read only. The number of requests for file should not exceed the maximum query (*maxQueries*) which acts as an upper bound threshold in a period of cycle, this is to control a maximum number of queries that node can emit. Both minimum query (*minQueries*) and maximum query in this experiment were set in the *DataInitializer*. The proposed ARPM solution for replication is based on popularity of files and the affinity degree. If both dominant factors are high, then a set of intersections files are copied to the source node, subject to storage availability. Figure 4.5 below shows a screen shot of ARPM random queries in *PeerSim*.

Table 4.2: The single query and multiple query scenarios

Cycle Number	*Requestor NodeId*	*FileId*	
0	3	28	
1	39	23	*Case-1*
2	92	31	
3	67	25	
4	97	15	
5	63	6	
6	42	19	
7	69	25	*Case- 2*
8	31	3	
9	1	29	
10	97	17	
11	50	21	
12	54	8	
13	32	12	*Case- 3*
14	46	12	
15	71	3	
16	25	22	
17	31	6	
18	14, 15, 37	9, 27,33	*Case- 4*
19	91	30	
20	28	19	

```
Simulator: loading configuration
ConfigProperties: File example/config-rpproject5.txt loaded.
Simulator: starting experiment 0 invoking peersim.cdsim.CDSimulator
Random seed: 1234567890



CDSimulator: resetting
Network: no node defined, using GeneralNode
CDSimulator: running initializers
- Running initializer init.rnd: class peersim.dynamics.WireKOut
- Running initializer init.datainit: class rpproject5.DataInitializer
time: 0 cycle  init.datainit starts running....
- Running initializer init.requestinit: class rpproject5.RequestInitializer
time: 0 cycle  init.requestinit starts running....
--------------------- Table 1 - create random queries-------------------
  Cycle : 0     nodeId= 14      FileId = 19
  Cycle : 1     nodeId= 14      FileId = 28
  Cycle : 2     nodeId= 10      FileId = 23
  Cycle : 3     nodeId= 2       FileId = 31
  Cycle : 4     nodeId= 15      FileId = 25
  Cycle : 5     nodeId= 14      FileId = 15
  Cycle : 6     nodeId= 7       FileId = 25
  Cycle : 7     nodeId= 14      FileId = 6
  Cycle : 8     nodeId= 16      FileId = 19
  Cycle : 9     nodeId= 6       FileId = 25
  Cycle : 10    nodeId= 18      FileId = 3
  Cycle : 11    nodeId= 18      FileId = 29
  Cycle : 12    nodeId= 9       FileId = 17
  Cycle : 13    nodeId= 14      FileId = 21
  Cycle : 14    nodeId= 16      FileId = 8
  Cycle : 15    nodeId= 6       FileId = 12
  Cycle : 16    nodeId= 2       FileId = 12
```

Figure 4.3: ARPM screenshot for random queries in *PeerSim*

4.5 Fundamental Decisions in Replica Placement

As mentioned in chapter 3, the important decisions for the replication models to get the upmost benefits are:

1) What to replicate? - The decision refers to the replica selection phase. Selecting target replicas depends on the popularity and the affinity degree of files, which can be gained by tracing users' access history and calculating the affinity degree between the files in the source and destination nodes. A precise metric to determine popular file for replication is used by calculating the

number of access frequency *(AF)* for the file at time interval *t*. File with maximum *AF* is a popular file. Next, the average access frequency for the popular files is calculated and compared with all other requested files. Then an affinity degree, *AD*, is calculated. If both *AF* and *AD* values are "high", *a* set of files will be replicated.

2) How many to replicate? - This decision refers to the replica allocation. The threshold is based on the number of average accesses frequency calculated in each case scenarios. The threshold acts as a checker to ensure that sufficient replicas exist in the system. The formula of this threshold and few examples were explained in chapter 3, section 3.5. The number of replicas needed for the popular files is calculated as the average access frequency of the popular file divided by the average access frequency for all other files. Table 4.3 shows some examples of the output.

3) Where to place the replicas? - This refers the replica placement phase to determine replica location. If the decision is not to replicate, the file will be read remotely. This refers to *Case 1 – Single-Query to Single-File* as mentioned in chapter 3, section 3.1. No replication indicates that the access file was less popular. In other cases, the access frequency and the affinity degree were calculated and if the value for both dominant factors are high. In this experiment, the replica is copied to the requester node from the destination node. A proximity threshold was set to a certain value which acts as a checker in hop count. If the hop count from a source node to the destination node exceed the proximity threshold, then we assumed that the

distant between the nodes are far. On the other hand, if the hop count from the source node to the destination node is less than the proximity threshold, we assumed that the distant between the nodes are closer.

4.6 System Testing

The objective of system testing is to ensure that the developed system performs as specified by the requirements. The output results from the *PeerSim* simulator were shown in Figure 4.3 to Figure 4.5. The output demonstrates the creation of random query table running in *Cycle Driven (CD)* mode. The table consists of the number cycles, the source nodes that request the files and the queried files. These results show that the simulations for system testing have been successfully executed. The simulations were tested on the P2P random topology composed of 50 nodes and 100 nodes and 1000 nodes in the network.

4.7 Experiments of Affinity Replica Placement Mechanism (ARPM)

In this section, the implementation and results of ARPM simulation in *PeerSim* will be discussed. The first step in the simulation is to read the configuration file which include all the simulation parameters objects in the experiment. Then the simulator sets up the network initializing the nodes and the protocols in the system.

In this experiment, there are four initializers namely *DataInit*, *RequestInit*, *DataNeiInit*, and *RequestRoutingIndicesInit.* These initializers set up the initial state

of each protocol in this experiment. Six protocols were implemented including *DataProtocol*, *QueryFileProtocol*, *QueryProtocol*,*RequestProtocol*, *PlacementProtocol* and *PlacementFileProtocol*.

4.7.1 Single-Query to Single-File

In *Single-Query to Single-File*, only one node request a single file. This case is considered as a basic case with no file replication.

4.7.2 Single-Query to Multiple-Files

In *Single-Query to Multiple-Files* case, the same node request for many different files. Herein, average access frequency and affinity degree were calculated. The selected file and its correlated files were replicated to the requester node. Table 4.2 shows example of *Single-Query to Multiple-Files* from the experiment.

4.7.3 Multiple-Query to Single-File

In *Multiple-Query to Single-File*, different nodes request for the same file. Herein, average access frequency and affinity degree were calculated. The selected file and its correlated files were replicated to the requester node. Table 4.2 shows example of *Multiple-Query to Single-File* from the experiment.

4.7.4 Multiple-Query to Multiple-Files

In *Multiple-Query to Multiple-File*, different nodes request for many different files. The average access frequency and affinity degree were calculated. The selected file and its correlated files were replicated to the requester node. Table 4.2 shows an example of *Multiple-Query to Multiple-File* from the experiment.

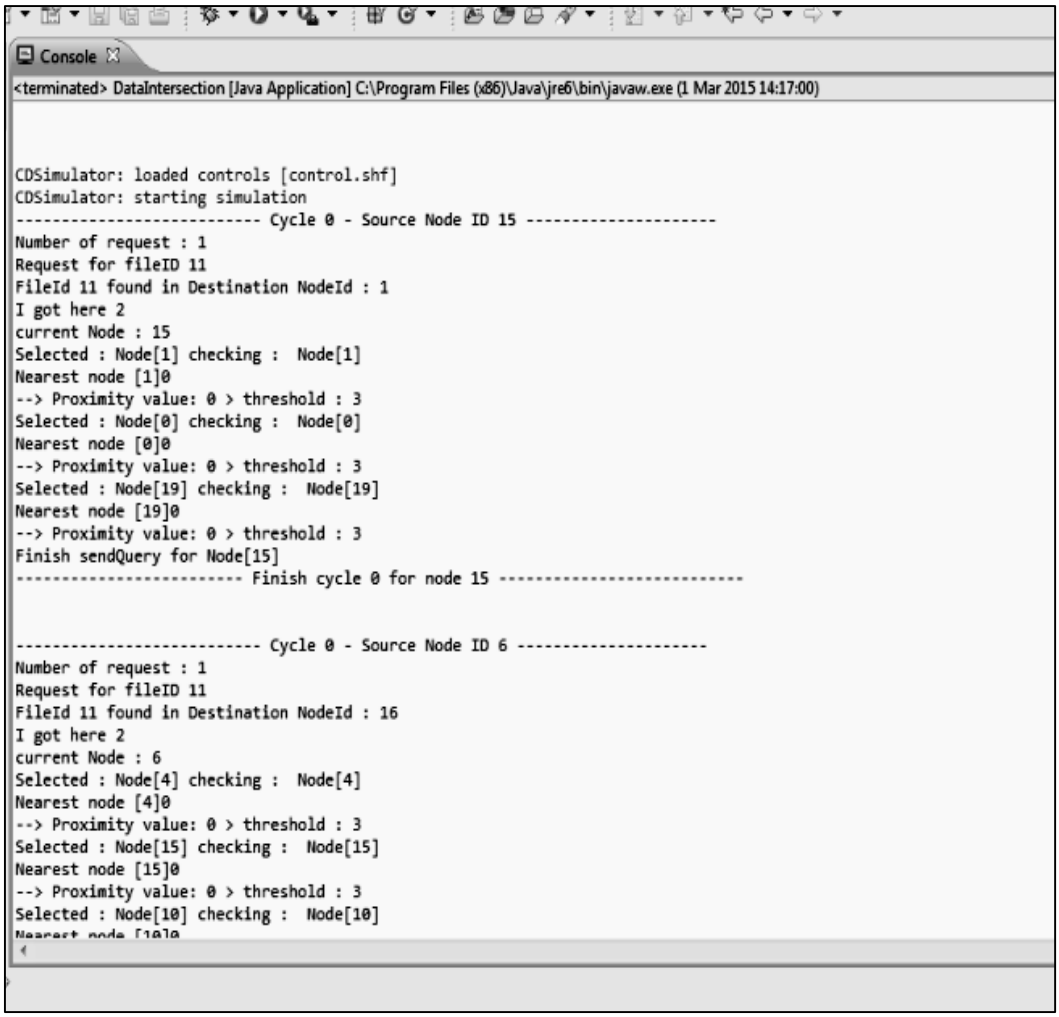

Figure 4.4: The ARPM screenshot simulated in *PeerSim*

4.8 Summary

In ARPM, the threshold value varies depends on the number of file queries and the time intervals. *PeerSim* is a peer-to-peer simulator to evaluate the algorithm of replica placement mechanism. In this thesis four cases have been discussed and successfully tested in ARPM. In each case, the number of access frequency and affinity degree were calculated and tabulated. The results act as the finding towards evaluating the performance metrics to find the popular and affinity files.

CHAPTER 5

EVALUATION AND EXPERIMENTAL RESULTS

In this chapter, the performance results of ARPM are presented and discussed. The studied performance metrics include Access Frequency *(AF),* Affinity Degree *(AD)*, and the number of replicas created.

5.1 Experimental and Simulation Platforms

PeerSim simulator is used as the simulation platform in this research to measure the replica placement performance of the proposed system and to validate the affinity notion introduced as mechanism in supporting data placement. All simulations were implemented in cycle driven mode. In this cycle driven mode, it is assumed that communications and processing delays can be neglected. In general, the fundamental concept of ARPM is to place replicas based on Access Frequency (*AF*) which indicates that the queried files are popular. Popularity is a very important factor to avoid unnecessary replication in the P2P networks.

Another important factor and a core of the approach proposed in this thesis is an affinity degree between dataset of files. The affinity degree reflects the real scenarios in collaborating research environments. A dataset of files may have affinity with another dataset of files in dispersed locations. Therefore ARPM is proposed to place affinity files together to improve data access performance through minimizing the access time and to ensure data availability of files in P2P replica placement decision.

5.2 Simulation Results

In this section, the performance results of ARPM algorithms are presented and discussed. The studied performance metrics include access frequency (popularity), affinity degree (relatedness), and the number of replicas created.

5.2.1 Access Frequency

One of the main objectives of the algorithm is to increase data access performance from the perspective of the clients by dynamically creating replicas for "popular" files. As mentioned in chapter 3 section 3.1, in the real world, some files will be more popular than others and data access patterns may change over time, so any dynamic replication strategy must keep track of file access histories to decide on when, what and where to replicate. The "popularity" of a file is determined by its *Access Frequency (AF)* from the clients or the requester node.

5.2.2 Affinity Degree

The metric of *Affinity Degree (AD*), denotes the relatedness between files that were requested by the nodes in the system. The formula of *AD* was discussed in chapter 3 section 3.2. The calculation of *AD* reflects the category of affinity between files whether the files have zero affinity, weak, moderate, strong or very strong. Only categories for moderate, strong and very strong were chosen to be replicated, provided that the files were popular as calculated in section 3.1. Chapter 4 is the continuity from chapter 3 whereby the tables in chapter 4 were the outputs from the experiment simulated in *PeerSim*.

5.2.3 Number of Replicas

The metric of number of replicas represents the total number of replicas created for all data accesses requested by the clients in a simulation session. An increased number of replicas implies a higher replication frequency which is the value of how many replications occur per data access. Therefore, the frequency of replication operations must be controlled to avoid heavy network and server load. In this thesis, not all file queries will be replicated. The decision whether to replicate depends on the calculation of the access frequency and the affinity degree.

5.3 Discussions

Table 5.1 shows the access frequency of the queried files as calculated in the proposed formula explained earlier in 3.3, chapter 3. Duration of 10 times intervals

are chosen to tabulate access frequency that indicates the number of files that have been queried. Then the average access frequency were calculated to find a threshold value for the establishment of the popular files. Figure 5.1 illustrates the popular files from time interval 1 to time interval 6 whilst the data from time interval 7 to time interval 10 from Table 5.1 were filtered since the data indicates the unpopular files. The graph is decreasing towards the end of the intervals. The result indicates that the access frequency that pass the average access frequency threshold were between interval *T1* to *T6*, where from interval *T5* onwards, the files queried were less popular. This result illustrates that the files over the time intervals were decreased and the files became less popular. In real scenarios, this reflects that the popularity of files increased in the first dissemination and became less popular after a period of time.

Further on, the results reflect the dynamic replication which takes into consideration changes in the peer-to-peer environments and creates new replicas for referenced data files or moves the replicas to other sites as needed to improve performance. When a request is found in any node, the node will become further reference for file access. Thus no new replication is needed. The usefulness of this replication strategy is evident and can be seen in the new technology communication products or in fashions trend. The communication technology depreciate rapidly whenever new technology coming in. Similarly Fashion nowadays become trendy in the current time situation and will depreciate over a period of time and thus become less popular.

Table 5.1: An example of calculated *Access Frequency (AF)*

ACCESS FREQUENCY	TIME INTERVAL									
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
	2.0000	2.0000	0.7500	0.2500	0.5000	0.1880	0.0469	0.0469	0.0117	0.0039
	3.0000	3.0000	0.5000	0.2500	0.2500	0.0100	0.0469	0.0313	0.0117	0.0059
	3.0000	1.0000	1.0000	0.7500	0.1250	0.0040	0.0625	0.0469	0.0313	0.0059
	4.0000	2.5000	0.7500	0.1250	0.1875	0.0060	0.0469	0.0781	0.0156	0.0059
	7.0000	0.5000	1.2500	0.6250	0.2500	0.0120	0.0313	0.0781	0.0273	0.0059
	3.0000	2.5000	1.2500	0.6250	0.4375	0.0060	0.0313	0.0625	0.0156	0.0039
	3.0000	1.5000	1.0000	0.3750	0.0625	0.0040	0.0781	0.0469	0.0078	0.0098
	3.0000	2.5000	0.5000	0.6250	0.2500	0.0060	0.0313	0.0313	0.0039	0.0039
	2.0000	0.5000	0.2500	0.2500	0.3125	0.0080	0.0156	0.0156	0.0117	0.0059
	5.0000	2.5000	1.0000	0.6250	0.3750	0.0060	0.0469	0.0313	0.0156	0.0078
	6.0000	1.0000	0.7500	0.7500	0.1875	0.0040	0.0781	0.0469	0.0195	0.0059
	1.0000	1.5000	1.0000	0.8750	0.1875	0.0060	0.0625	0.0469	0.0078	0.0059
	4.0000	2.0000	1.2500	0.1250	0.1875	0.0040	0.0781	0.0938	0.0039	0.0059
	3.0000	1.0000	0.5000	0.3750	0.2500	0.0020	0.0781	0.0625	0.0078	0.0117
	5.0000	1.5000	1.2500	0.3750	0.2500	0.0060	0.0781	0.1094	0.0117	0.0098
	2.0000	0.5000	1.2500	0.5000	0.0625	0.0100	0.0625	0.0469	0.0117	0.0059
	3.0000	1.0000	0.2500	0.3750	0.1875	0.0040	0.0469	0.0156	0.0117	0.0059
	3.0000	2.5000	0.5000	0.2500	0.2500	0.0060	0.0625	0.0469	0.0078	0.0039
	3.0000	2.5000	1.0000	0.2500	0.1875	0.0100	0.0156	0.0313	0.0039	0.0039
	4.0000	2.0000	0.7500	0.3750	0.1875	0.0060	0.0469	0.0313	0.0117	0.0078
	4.0000	3.0000	0.5000	0.1250	0.1250	0.0060	0.0313	0.0156	0.0078	0.0059
	1.0000	0.5000	1.2500	0.1250	0.2500	0.0040	0.0469	0.0313	0.0078	0.0039
	3.0000	1.0000	1.2500	0.6250	0.1250	0.0100	0.0313	0.0313	0.0156	0.0039
	3.0000	1.0000	1.2500	0.3750	0.1250	0.0040	0.0156	0.0625	0.0234	0.0059
	1.0000	1.0000	0.5000	0.1250	0.0625	0.0020	0.0469	0.0625	0.0039	0.0020
	2.0000	1.5000	0.5000	0.2500	0.1250	0.0060	0.0156	0.0781	0.0039	0.0020
	2.0000	1.0000	0.2500	0.2500	0.0625	0.0080	0.0469	0.0313	0.0234	0.0020

	2.0000	1.5000	1.0000	0.2500	0.2500	0.0040	0.0156	0.0156	0.0039	0.0039
	2.0000	1.0000	0.5000	0.1250	0.1250	0.0040	0.0156	0.0469	0.0078	0.0059
	2.0000	1.5000	0.5000	0.2500	0.0625	0.0060	0.0313	0.0625	0.0117	0.0098
	1.0000	1.0000	0.2500	0.5000	0.0625	0.0040	0.0156	0.0625	0.0039	0.0020
	3.0000	1.0000	0.2500	0.1250	0.0625	0.0040	0.0156	0.0156	0.0078	0.0039

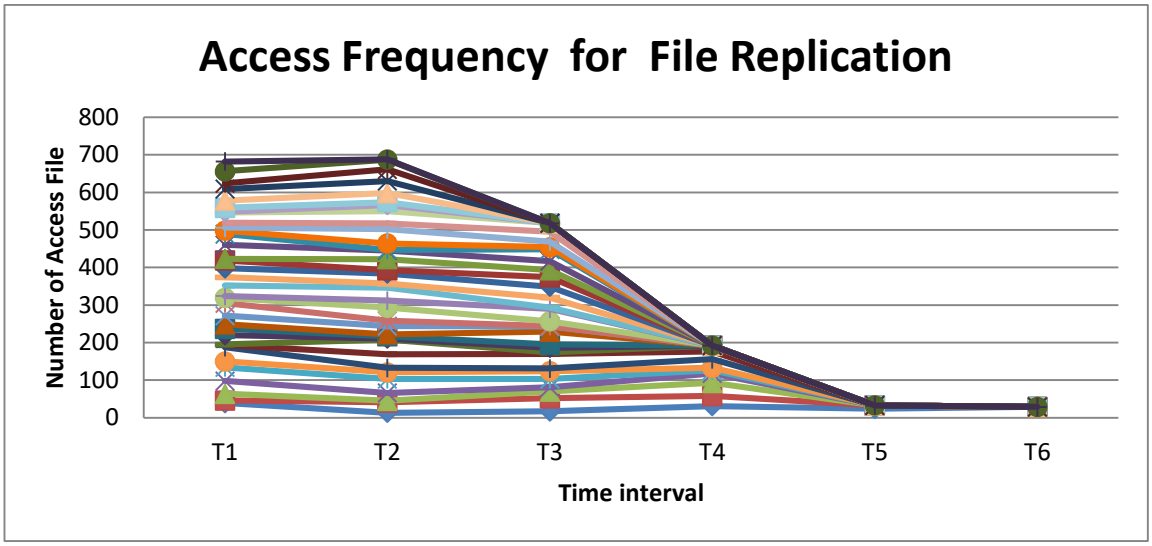

Figure 5.1: The relationship between *Access Frequency (AF)* and *time interval (T)*

Table 5.2 shows the calculated *Access Frequency (AF)* and *Affinity Degree (AD)* as proposed in this thesis. The excerpted data in Table 5.2 shows that, only popular files were considered to be replicated in the P2P networks. The *Affinity Degree (AD*) were calculated based on these popular files to find the files relatedness or affinity. The shaded region in the table indicates the unpopular files, and hence were filtered from the table.

Table 5.2: An example of calculated *Access Frequency (AF)* and *Affinity Degree (AD)*

AFFINITY DEGREE	TIME INTERVAL											
	T1		*T2*		*T3*		*T4*		*T5*		*T6*	
	AF	*AD*	*AF*	*AD*	*AF*	*AD*	*AF*	*AD*	*AF*	*AD*	*AF*	*AD*
	2.0000	0.3300	2.0000	0.1818	0.7500	0.5789	0.2500	0.5714	0.5000	0.3750	0.1880	0.6471
	3.0000	0.2700	3.0000	0.6000	0.5000	0.3125	0.2500	0.3636	0.2500	0.5454	0.0100	0.5385
	3.0000	0.2800	1.0000	0.4444	1.0000	0.1818	0.7500	0.4375	0.1250	0.3181	0.0040	0.3000
	4.0000	0.7500	2.5000	0.5556	0.7500	0.2500	0.1250	0.6363	0.1875	1.0000	0.0060	0.6842
	7.0000	0.4737	0.5000	0.4231	1.2500	0.4444	0.6250	0.4444	0.2500	0.3500	0.0120	0.5000
	3.0000	0.5556	2.5000	0.2727	1.2500	0.3333	0.6250	0.5714	0.4375	0.4000	0.0060	0.4000
	3.0000	0.5000	1.5000	0.1250	1.0000	0.3571	0.3750	0.4286	0.0625	0.4500	0.0040	
	3.0000	0.5556	2.5000	0.2727	0.5000	0.4286	0.6250	0.4000	0.2500	0.3571	0.0060	
	2.0000	0.5000	0.5000	0.4000	0.2500	0.3043	0.2500	0.5385	0.3125	0.5333	0.0080	
	5.0000	0.5333	2.5000	0.4118	1.0000	0.4667	0.6250	0.5000	0.3750	0.5000	0.0060	
	6.0000	0.3330	1.0000	0.5000	0.7500	0.4118	0.7500	0.7143	0.1875		0.0040	
	1.0000	0.5000	1.5000	0.2353	1.0000	0.6667	0.8750	0.2857	0.1875		0.0060	
	4.0000	0.7857	2.0000	0.4545	1.2500	0.4286	0.1250	0.6250	0.1875		0.0040	
	3.0000	0.6363	1.0000	0.6250	0.5000	0.4706	0.3750	0.2857	0.2500		0.0020	
	5.0000	0.3889	1.5000	0.5455	1.2500	0.4167	0.3750	0.5000	0.2500		0.0060	
	2.0000	0.0769	0.5000	0.5000	1.2500	0.5789	0.5000	0.5385	0.0625		0.0100	
	3.0000	0.5333	1.0000	0.5789	0.2500	0.3571	0.3750		0.1875		0.0040	
	3.0000	0.5714	2.5000	0.4167	0.5000	0.3333	0.2500		0.2500		0.0060	
	3.0000	0.2500	2.5000	0.5000	1.0000	0.3846	0.2500		0.1875		0.0100	
	4.0000	0.2941	2.0000	0.3478	0.7500	0.4667	0.3750		0.1875		0.0060	
	4.0000	0.4545	3.0000	0.5385	0.5000	0.3333	0.1250		0.1250		0.0060	
	1.0000	0.6111	0.5000	0.5556	1.2500	0.3846	0.1250		0.2500		0.0040	
	3.0000	0.3077	1.0000	0.2000	1.2500	0.6000	0.6250		0.1250		0.0100	
	3.0000	0.5882	1.0000	0.4167	1.2500	0.4118	0.3750		0.1250		0.0040	
	1.0000	0.5556	1.0000	0.5455	0.5000	0.1538	0.1250		0.0625		0.0020	
	2.0000	0.2000	1.5000	0.5000	0.5000	0.3571	0.2500		0.1250		0.0060	
	2.0000	0.4615	1.0000	0.2000	0.2500	0.3750	0.2500		0.0625		0.0080	
	2.0000	0.5882	1.5000	0.5000	1.0000	0.2500	0.2500		0.2500		0.0040	
	2.0000	0.5556	1.0000	0.1429	0.5000	0.3333	0.1250		0.1250		0.0040	
	2.0000	0.3750	1.5000	0.5714	0.5000	0.7500	0.2500		0.0625		0.0060	
	1.0000	0.5625	1.0000	0.3750	0.2500	0.4737	0.5000		0.0625		0.0040	

	3.0000	0.4615	1.0000	0.6667	0.2500	0.1818	0.1250		0.0625		0.0040	
	2.0000	0.4347	0.5000	0.3125		0.6000	0.1250		0.0625		0.0060	
	1.0000	0.4285	0.5000	0.5000		0.3125			0.0625			

The result from Table 5.2 were excerpted into Table 5.3. In Table 5.3, the affinity degree were calculated based on the popularity files which exceed or equal ~~to~~ the threshold of average access frequency calculated earlier. Only access frequency and affinity degree files that complied with the rules proposed in this thesis will be selected to be replicated. The shaded region indicates the data that has been filtered from Table 5.2. This suggests that the relatedness of the files were weak in comparison with the unshaded region that shows the moderate, strong and very strong affinity degree in Table 5.3.

Table 5.3: An example of calculated affinity degree

	TIME INTERVAL					
	T1	T2	T3	T4	T5	T6
AFFINITY DEGREE	0.7500	0.6000	0.5789	0.5714	1.0000	0.6471
	0.5556	0.5556	0.6667	0.6363	0.5454	0.5385
	0.5000	0.5000	0.5789	0.5714	0.5333	0.6842
	0.5556	0.6250	0.6000	0.5385	0.5000	0.5000
	0.5000	0.5455	0.7500	0.5000		
	0.5333	0.5000		0.7143		
	0.5000	0.5789		0.6250		
	0.7857	0.5000		0.5000		
	0.6363	0.5385		0.5385		
	0.0769	0.5556				
	0.5333	0.5455				
	0.5714	0.5000				
	0.6111	0.5000				
	0.5882	0.5714				
	0.5556	0.6667				

	0.5882	0.5000				
	0.5556					
	0.5625					
No. of files	18	16	5	9	4	4
Referring to the Affinity Degree Formula 3.2 in chapter 3						

Figure 5.2 Illustrates the affinity degree calculated based on the files that exceed or equal to the access frequency threshold and the affinity degree that have strong files relatedness. In time interval 4 (*T4*), there was a slight increase in the number of the replicated files. The replicated files over a period of time in *T3* were decreased but gained back in *T4* before the pattern is repeated. The graph in Figure 5.2 verified that there is a certain access pattern and relatedness of the requested files by the clients in the network.

The demand for the popular and correlated files are high during the first dissemination and then decreased after certain period. Consequently it will gain popularity and correlativity before it decreases hence this pattern will be repeated. In real scenario, in research collaboration for example, a new found technology or research will initially expected to be highly demanded and therefore the number of replicas is increased and copied to the trusted or affine clients. However, this data will decrease over a certain period of time and whenever newer technology is found, the pattern will be repeated.

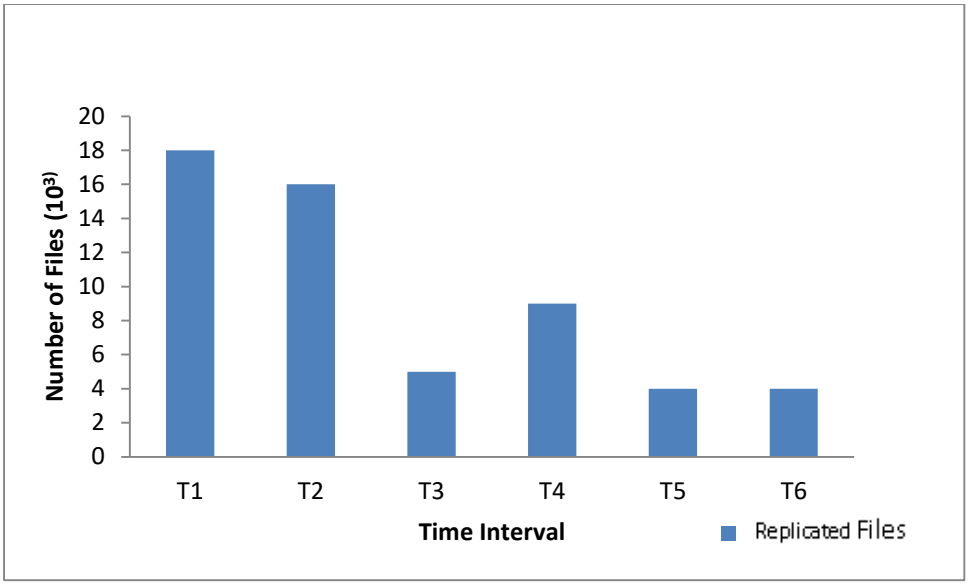

Figure 5.2: Files replication based on degree of affinity

5.4 Summary

In order to evaluate the performance of the proposed replica placement strategy, an affinity and access frequency were chosen as the two dominant factors in this thesis. The results were explained and illustrated in this chapter. At constant time intervals, the dynamic Affinity Replica Placement Mechanism (ARPM) calculates the files access frequency that denotes the popularity of the file queries and calculate the affinity degree that reflects the relatedness or the dependency of the files between the source node and the destination node. More importantly, by calculating the affinity degree, more precise metrics are found to indicate the affinity between files in the nodes. The files that complies with the two dominant factors were replicated to the source nodes that initiate the request. Thus, the network performance is increased since more than one replicas exist in the systems. Additionally, the number of file replicas calculated is to ensure availability by having sufficient numbers of replicas in the network.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

This chapter concludes the thesis followed by the discussion of contributions and future directions. In this thesis, we proposed Affinity Replica Placement Mechanism (ARPM) which is encapsulated in three layers: (1) Peer-to-peer topology layer, (2) a discovery layer and (3) an ARPM services. We have addressed the problem of replica placement in peer-to-peer (P2P) systems to improve the performance of data availability and accessibility.

6.1 Conclusion

Would you open your front door to a stranger? Would you share a copy of your document or files with an unknown person who is neither your relative nor your friends or colleague simply because they request a copy of the files? Why molecules bind with certain molecules? Is it because of its strong chemistry or perhaps affinity? These are some of the analogies that were the turning point in this thesis, a study of the affinity notion in replica placement strategies.

Popularity and affinity are two most important parameters in replica placement strategy which aims to improve the performance of data availability and accessibility. The idea behind ARPM as proposed in this study is to create replicas based on files' popularity and affinity. Popularity refers to how many times the file is required by a client or a system site and it indicates the importance of the file. The performance of replication strategies in distributed systems closely depend on the popularity parameter precision. Indeed, for a given datasets, the closer is the popularity prediction to the reality; the better is the satisfaction of the client needs. Consequently, the data popularity parameter is one of the key factors to decide which files have to be accessed, replicated or even deleted.

Equally important parameter to address replica placement problems is affinity. Affinity refers to the correlated files, similarity, dependency or the linking between two or more files in the P2P systems. The affinity parameter will replicate file only to trusted sites and therefore replicating sufficient quality research file. Generally, other replica placement strategies deal with the quantity data dissemination. However, ARPM in this thesis deals with the quality over quantity data replication strategy. If we just take popularity as a measure, a system may over replicate. Moreover, in many cases, popularity does not continue. There will be lots of replicas which may not be needed. Therefore, taking affinity into consideration as another measure is very significant to reduce the number of replicas in the P2P network systems.

Combining both popularity and affinity parameters in replica placement will finally strengthen our primary goals to improve data availability and accessibility whilst

reduce over replication. Up until now, there are not many literatures combining popularity and affinity in designing dynamic replica placement strategies in distributed systems. Furthermore, the adapted access frequency and proposed affinity mathematical formulations have been established and taken into account for both single query and multiple queries scenarios of requesting file(s) from any nodes in the peer-peer systems. Then again, different replication policies under different topologies are complicated to compare due to the diverse nature of assumptions made regarding the topology, data access patterns and policy objectives. Thus, the decisions of replica placement are very important to improve the performance of any replication scheme effectively.

6.2 Contributions

Based on the results of this study the following contributions can be drawn:

1) ARPM has been proposed and it has successfully contributed to the improvement of data access performance through minimizing the access time.
2) ARPM has successfully avoided the overflow of the unnecessary replicated files in the distributed system.
3) The formula for access frequency is adapted mathematically to calculate the files popularity whilst the mathematical formula for affinity degree was established.
4) The hybrid of the popularity and relatedness of the files demanded by the clients in the network has been incorporated in our replica placement strategy.

6.3 Practical Applications

This section discusses the applicability of real situations where ARPM can be applied namely: media affinity and file sharing. By its nature, audio and video streaming in media affinity has a linkage to be recorded or played back at its real time rate. Both media can be written into the system or copied with certain chunk sizes or in a sequence of related frames. Since frames are delivered in sequence in video/audio streaming, a request for frame one will likely require frames two … n. Later when the data is accessed in the file system, it has already been optimised for the way the system will be read or written based upon the data's affinity for real time use.

Another practical application is file sharing. In distributed file sharing such as the aspects of temporal/spatial locality, files in a common repository often will be requested together. Hence, request for one file potentially also leads to other files in the repository. For example, a High Energy Physics (HEP) device called the Large Hadron Collider (LHC), at CERN will produce roughly 15 Petabytes (15 million Gigabytes) of data annually, which thousands of scientists around the world will access (Shorfuzzaman, 2012). The data distribution model for the CERN (LHC) experiments where datasets were first generated and stored at CERN, and later copied to different distribution and regional centres. From these centres the data is then distributed to different labs worldwide to give access to scientists from around the world.

6.4 Issues and Limitations

There are several issues with consistency, storage and file deletion which are not highlighted in this study. The issues of consistency deals with concurrent updates made to multiple replicas of a file. When one file is updated, all other replicas have the same contents and thus provide a consistent view. Consistency and synchronization problems associated with replication in P2P systems are not addressed in this research with files are regarded as being read-only. Next is the storage issue which is one of the limitations in this thesis. A better replica placement strategy would distribute replicas over many storage peers in the system and balance the access load among the peers. Another limitation is file deletion in P2P environments. Due to the dynamic nature in P2P environment, the replication strategy should be more adaptive in deleting replicas which is least important or not popular anymore.

In the simulations, the connection between sites is assumed to be reliable throughout the simulations. As future research, ARPM can be extended further to include sites that can join or quit the P2P network besides ARPM capable of handling fault tolerance issue.

6.5 Future Directions

The combination of popularity and affinity files in replica placement strategies have open up to other significant contributions in distributed systems and other applications. The possibility of combining affinity files, affinity path and affinity

nodes can bring the performance of replication strategies to another level. By understanding the patterns of interactions in the peer-to-peer network and who is connected to whom, shed up the new dimensions in replica placement strategies, social networks, and E-science and non E-science applications in big data driven environment.

This thesis has investigated the properties that are unique to peer-to-peer environment. However, the research can also be applied to other environment such as hierarchical data grids, federated data grid, and hybrid distributed systems. Currently, scientific collaborations that need to manage volumes of shared data. Some of the tools developed within distributed environments may find applicability to areas outside of scientific computing such as in enterprises with similar requirements for resource sharing and data access. This would require taking into account more strict reliability and security requirements. Another challenge would be to extend existing techniques to work with technologies within enterprises such as NoSQL databases. In addition, ARPM copies data across multiple servers, so each bit of data can be found in multiple places. Besides read operation, ARPM should allow writes operation as well to any node and synchronize their copies of the data.

Another extension would be to modify ARPM algorithms to determine replica placement in the hybrid topology instead of pure P2P topology. This would broaden the scope of applicability of ARPM algorithms across various grid and distributed environments that require both non-hierarchical and hierarchical network structures.

This thesis has reached the initial fixed goals; however that much still needs to be investigated. The work done in this thesis contributes to some understanding of replica placement in peer-to-peer environments and advances the state-of-the-art through its contributions. The thesis finishes with the idea of popularity and affinity in replica placement as the base to pursue with various opened directions in the future.

REFERENCES

ABAWAJY, J., and Mat Deris, M., 2014. Data Replication Approach with Consistency Guarantee for Data Grid, *IEEE Transactions on Computers*, vol.63, no. 12, pp. 2975-2987.

ABAWAJY, J., 2004. Placement of File Replicas in Data Grid Environments. Springer Verlag, Berlin Heidelberg, 3038, 66 – 73.

ABDULLAH, A., OTHMAN, M., IBRAHIM, H. SULAIMAN, M.N. and OTHMAN, A.T. 2008. Decentralized replication strategies for P2P based scientific data grid. Information Technology, ITSim 2008, International Symposium, 3, 1 – 8.

AMJAD, T., SHER, M., DAUD, A. 2012. A survey of dynamic replication strategies for improving data availability in data grids. Future Generation Computer Systems, Volume 28, 337 – 349.

BAKHOUYA, M., GABER, J. 2006. Self-organizing Approach for Emergent Multi-agent Structures", copyright 2006 ACM 1-59593-186-4/06/0007.

BAKSHI, K. (2012, March). Considerations for big data: Architecture and approach. In Aerospace Conference, IEEE (pp. 1-7).

BARREFORS, B., 2015. Dynamic Data Management In A Data Grid Environment. Master thesis: The University of Nebraska.

BSOUL, M., Al-KHAWSANEH, A., KILANI, Y. and OBEIDAT, I., 2012. A threshold-based dynamic data replication strategy. *The Journal of Supercomputing*, *60*(3), pp.301-310.

CHANG, R.S. and CHANG, H.P. 2008. A dynamic data replication strategy using access-weights in data grids. *J Supercomput,* 45, 277 – 295.

CHEN, M., MAO, S. and LIU, Y. 2014. Big data:a survey. Mobile Networks and Applications, Volume 19(2), 171-209.

CHEN, Y.W. and LARBANI, M. 2006. Developing the Affinity Set (Guangxi Set) Theory and Its Applications In ICPADS, 465-474, IEEE Computer Society.

CHEN, Y. W., LARBANI, M., HSIEH, C. Y., & CHEN, C. W. (2009). Introduction of affinity set and its application in data-mining example of delayed diagnosis.*Expert Systems with Applications*, *36*(8), 10883-10889.

CHEN, Y., KATZ, R. H., & KUBIATOWICZ, J. D. (2002). Dynamic replica placement for scalable content delivery In Peer-to-peer systems 306-318. Springer Berlin Heidelberg.

CHERVENAK, A., FOSTER, I., KESSELMAN, C., SALISBURY, C. and TUECKE, S. 2000. The Data Grid: Towards and architecture for the distributed management and analyss of large scientific datasets. Journal of Network and Computer Applications, 23, (3), 187.

CHETTAOUI, H. AND CHARRADA, F.B., 2014. A new decentralized periodic replication strategy for dynamic data grids. *Scalable Computing: Practice and Experience*, *15*(1).

COHEN, E. and SHENKER, S. 2002. Replication strategies in unstructured peer-to-peer networks. In: Proceedings of the conference on Applications, technologies, architectures and protocols for computer communications. ACM Press: New York, 2002, 177 – 190.

DALLAKYAN, S. AND OLSON, A.J., 2015. Small-Molecule Library Screening by Docking with PyRx. *Chemical Biology: Methods and Protocols*, pp.243-250.

DANCY, C.P. AND REIDY, J., 2004. Statistics without maths for psychology.*Harlow: Pearson Education Limited*.

DERIS, M.M., ABAWAJY, J.H. AND MAMAT, A., 2008. An efficient replicated data access approach for large-scale distributed systems. *Future Generation Computer Systems*, *24*(1), pp.1-9.

DEVAKIRUBAI, N. AND KANNAMMAL, A., 2013. Optimal replica placement in graph based data grids. *The International Journal of Engineering and Science (IJES)*, *2*(3), pp.95-103.

DOBOS, L., PINCZEL, B., KISS, A., RÁCZ, G., & EILER, 2014. A Comparative Evaluation Of NoSQL Database Systems. *Annales Univ. Sci. Budapest., Sect. Comp. 42,* pp 173–198.

FADAIE, Z. and RAHMANI, A.M. 2012. A new Replica Placement in Data Grid", International Journal of Computer Science Issues (IJCSI), 9 (2).

FEDAK, G, et al.,. 2009. "BitDew: A data management and distribution service with multi-protocol file transfer and metadata abstraction. J.Network Computer Applications, doi:10.1016/j.jnca.2009.04.002

FEDAK, G., HE, H., CAPELLO, F. 2008. BitDew: A programmable Environment for Large-Scale Data Management and Distribution. Grand-Large/INRIA-Saclay Laboratorie de Recherche en Informatique : France.

FEDAK, G., He, H. and CAPELLO, F. 2008. A File Transfer Service with Client/Server, P2P and Wide Area Storage Protocols, LNCS, 1-11, Springer.

FOSTER, I., KESSELMAN, C., and TUECKE, S. 2001. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications, 15 (3), 200 – 222.

GARMEHI, M., ANALOUI, M., PATHAN, M. AND BUYYA, R., 2014. An economic replica placement mechanism for streaming content distribution in Hybrid CDN-P2P networks. *Computer Communications*, *52*, pp.60-70.

GARMEHI, M. AND MANSOURI, Y., 2007, December. Optimal placement replication on data grid environments. In *Information Technology,(ICIT 2007). 10th International Conference on* (pp. 190-195). IEEE.

GILSON, M.K., LIU, T., BAITALUK, M., NICOLA, G., HWANG, L. AND CHONG, J., 2016. BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic acids research*, *44*(D1), pp.D1045-D1053.

GOEL, S. and BUYYA, R. 2013. Data replication strategies in wide area distributed systems. Business and Information. Bali, July 7 – 9.

GRACE, R.K. and MANIMEGALAI, R. 2014. Dynamic replica placement and selection strategies in data grids – A comprehensive survey. J. Parallel Distrib. Computing, 74(2), 2099-2108.

HAMDENI, C., HAMROUNI, T., & CHARRADA, F. B. 2016. Data popularity measurements in distributed systems: Survey and design directions. *Journal of Network and Computer Applications*.

HECHT, R and JABLONSKI, S. (2011). Nosql evaluation. In *International conference on cloud and service computing* (pp. 336-41). IEEE.

JAMAL, A. A., AWANG, W. S. W., KADIR, M. F. A., AZIZ, A. A., & TEAHAN, W. J. 2014. Implementation of Resource Discovery Mechanisms onto PeerSim, *The Third International Conference on Informatics and Conference (ICIA 2014)* Malaysia.

JELASITY, M., MONTRESSOR A., JESI G., and VOULGARIS S. 2009. PeerSim: A scalable P2P simulator, *In. Proc.of the 9th Conference on Peer-to-peer (P2P'09),*99-100, Seattle, WA, Sept.

JELASITY, M., MONTRESSOR, A., JESI, G., and S.Voulgaris. 2004. PeerSim: A peer-to Peer Simulator. http://PeerSim.sourceforge.net

KAWASAKI, Y., MATSUMOTO, N., & YOSHIDA, N. (2006). Popularity-based content replication in peer-to-peer networks. In *Computational Science–ICCS 2006* . pp. 436-443. Springer Berlin Heidelberg.

LARBANI, M., CHEN, Y. W. (2009). A Fuzzy Set Based Framework for Concept of Affinity. Applied Mathematical Sciences, *3*(7), 317-332.

LAMEHAMEDI, H., SHENTU, Z., SZYMANSKI, B. 2003. Simulation of dynamic data replication strategies in data grids. In Proceedings of the International Parallel and Distributed Processing Symposium. Washington, D.C. : IEEE Computer Society.

LAMEHAMEDI, H., SZYMANSKI, B.K., SHENTU, B., DEELMAN, Z. 2002. Data Replication Strategies in Grid Environments In: proceedings of 5th Int'l Conference on Algorithms and Architecture for Parallel Processing, IEEE Computer Society Press, 378-383.

LIN Y.F., LIU P. and WU, J.J. 2008. Optimal replica placement in hierarchical data grids with locality assurance. Journal of Parallel and Distributed Computing, 12, 1517 – 1538.

LIN Y.F., LIU P., WU, J.J. 2006a. Optimal Placement of Replicas in Data Grid Environments with Locality Assurance. In ICPADS, 465-474, IEEE Computer Society.

LIU, P., LIN, YI-F.and Wu, J.J. 2006b. Optimal Replica Placement Strategy For Hierarchical Data Grid Systems. International Symposium on Cluster Computing and the Grid -CCGrid'06.

LUO, X., XIN, G., WANG, Y., ZHANG, Z. AND WANG, H., 2015. Superset: a non-uniform replica placement strategy towards perfect load balance and fine-grained power proportionality. *Cluster Computing*, pp.1-14.

MA, J., LIU, W., & GLATARD, T. (2013). A classification of file placement and replication methods on grids. *Future Generation Computer Systems*, *29*(6), 1395-1406.

MADI, M., YUSOF, Y., HASSAN, S. AND ALMOMANI, O., 2011. A Novel Replica Replacement Strategy for Data Grid Environment. In *Software Engineering and Computer Systems* (pp. 717-727). Springer Berlin Heidelberg.

MANSOURI, N., DASTGHAIBYFARD, G.H. AND MANSOURI, E., 2013. Combination of data replication and scheduling algorithm for improving data availability in Data Grids. *Journal of Network and Computer Applications*, *36*(2), pp.711-722.

MANSOURI, N., & DASTGHAIBYFARD, G. H. (2012). A dynamic replica management strategy in data grid. *Journal of network and computer applications*, *35*(4), 1297-1303.

MOKADEM, R. AND HAMEURLAIN, A., 2014. Data replication strategies with performance objective in data grid systems: a survey. *International Journal of Grid and Utility Computing*, *6*(1), pp.30-46.

PALANISWAMY, A. Accelerating HPC. Symposium on Application Accelerators in High-Performance Computing (SAAHPC'10), July 13-15, 2010 University of Tennessee Conference Center Knoxville, Tennessee

PARSONS, M. 2013. Petascale to Exascale: The Hardware and Software Challenge", HPC Finance Conference, May 13, 2013 Tampere University of Technology, Finland.

PETRI, I., RANA, O. F., REZGUI, Y., & SILAGHI, G. C. (2012). Risk assessment in service provider communities. In *Economics of Grids, Clouds, Systems, and Services* (pp. 135-147). Springer Berlin Heidelberg.

PETRI, I., RANA, O. F., SILAGHI, G. C., & REZGUI, Y. (2014). Risk assessment in service provider communities. *Future Generation Computer Systems*, *41*, 32-43.

RAHMAN, M.R., BARKER K. and ALHAJJ R. 2006. Replica Placement Design With Static Optimality and Dynamic maintainability. International Symposium on Cluster Computing and the Grid -CCGrid'06.

RANGANATHAN, K., LAMNITCHI, A. and FOSTER, I. 2002. Improving data availability through model-driven replication for large peer-to-peer communities. In Proceedings of Global and Peer-to-peer Computing on Large-Scale Distributed Systems Workshop, Berlin, Germany. IEEE.

RANGANATHAN, K., and Foster, I. (2001). Identifying dynamic replication strategies for a high-performance data grid. In Grid Computing—GRID 2001 Springer Berlin Heidelberg. 75-86.

RASOOL, Q., LI, J., ZHANG, S. 2009. Replica placement in multi-tier data grid. Eighth IEEE International Conference on Dependable, Automatic and Secure Computing, 103 – 108.

RASOOL, Q., JIANZHONG, L., GEORGE, S.K. and EHSAN, U.M. 2007. A comparative study of replica placement strategies in data grids. Springer-Verlag Berlin Heidelberg, LNCS, 4537, 135 – 143.

RASOOL, Q., LI, J., ZHANG, S. 2008. On P2P and Hybrid Approaches for Replica Placement in Grid Environment. Information Technology Journal 7 (4):590-598.

REED, D.A. AND DONGARRA, J., 2015. Exascale computing and big data. *Communications of the ACM*, *58*(7), pp.56-68

SASHI, K., & THANAMANI, A. S. (2011). Dynamic replication in a data grid using a modified BHR region based algorithm. *Future Generation Computer Systems*, *27*(2), 202-210.

SENHADJI, S., KATEB, A. and BELBACHIR, H., 2013. Increasing Replica Consistency Performances with Load Balancing Strategy in Data Grid Systems. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, *7*(1), pp.153-158.

SHI, C., LIN, Q. AND DENG, C., 2015. Preparation of on-plate immobilized metal ion affinity chromatography platform via dopamine chemistry for highly selective isolation of phosphopeptides with matrix assisted laser desorption/ionization mass spectrometry analysis. *Talanta*, *135*, pp.81-86.

SHORFUZZAMAN, M., 2014, DECEMBER. Access-Efficient QoS-Aware Data Replication to Maximize User Satisfaction in Cloud Computing Environments. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2014 15th International Conference on* (pp. 13-20). IEEE.

SHORFUZZAMAN, M. 2010. Placement of Replicas in Large-Scale Data Grid Environments. Phd thesis: The University of Manitoba.

SHEN, H. 2010. An efficient and adaptive decentralized file replication algorithm in P2P file sharing systems. *IEEE Transactions on parallel and distribution systems,* 21, 6, 827 – 840.

SIVAKUMAR, A., RAO, S. AND TAWARMALANI, M., 2013. D-tunes: self tuning datastores for geo-distributed interactive applications. *ACM SIGCOMM Computer Communication Review*, *43*(4), pp.483-484.

SOOSAI, A.M., ABDULLAH, A. and OTHMAN, M., LATIP, R., SULAIMAN, M. N., IBRAHIM, H. 2012. Dynamic replica replacement strategy in data grid. *In Computing Technology and Information Management (ICCM), 2012 8th International Conference on* Vol. 2, pp. 578-584. IEEE.

SPAHO, E., BAROLLI, A., XHAFA, F. AND BAROLLI, L., 2015. P2P Data Replication: Techniques and Applications. In *Modeling and Processing for Next-Generation Big-Data Technologies* (pp. 145-166). Springer International Publishing.

SPAHO, E., BAROLLI, L. AND XHAFA, F., 2014, September. Data Replication Strategies in P2P Systems: A Survey. In *Network-Based Information Systems (NBiS), 2014 17th International Conference on* (pp. 302-309). IEEE.

STEINMETZ, R., WEHRLE, K. 2005. Peer-to-peer Systems and Applications. , Springer, Library of Congress Control Number: 2005932758, Springer-Verlag Berlin Heidelberg 2005.

THAMPI, S.M. and SEKARAN, K.C., 2009. Review of replication schemes for unstructured P2P networks. *arXiv preprint arXiv:0903.1734*.

TANG, M., LEE, B.S., YEO, C.K., TANG, X. 2005. Dynamic Replication Algorithms for the Multi-tier Data Grid, Future Generation Computer System, Elsevier , 775-790.

TANG, M., LEE B.S., YAO, C.K. and TANG X.Y. 2005. Dynamic replication algorithm for performance data grid. In proceedings of the International Computing Workshop. Denver, Colorado, USA, 2001.

TU, M., MA, H., XIAO, L., YEN, I.L, BASTANI, F. and XU, D. 2013. Data placement in P2P data grids considering the availability, security, access performance and load balancing. *J Grid Computing,* 11, 103 – 127.

VENUGOPAL, S., BUYYA, R. and RAMAMOHANARAO, K. 2009. A Taxonomy of data grids for distributed data sharing, management and processing. ACM Computing Surveys, Vol. 38, March 2006, Article 3.

WAN AWANG, W.S., MAT DERIS, M., HITAM, M.S., MOHAMMAD, Z., and ZAKARIA, A. 2004. Data Replication Scheme based on Neighbour Replica Distribution Technique for Web Server Cluster. WSEAS Transactions on Systems, 3, 4,1779-1785.

WANG, L., TAO, J., RANJAN, R., MARTEN, H., STREIT, A., CHEN, J., & CHEN, D. 2013. G-Hadoop: MapReduce across distributed data centers for data-intensive computing. Future Generation Computer Systems, 29(3), 739-750.

WATANABE, T., KANZAKI, A., HARA, T. and NISHIO, S., 2009, April. Update Propagation Strategies Considering Degree of Data Update in Peer-to-peer Networks. In *International Conference on Database Systems for Advanced Applications* (pp. 328-333). Springer Berlin Heidelberg.

WEIL, S.A., Brandt, S.A. and Miller, E.L. 2006. CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data,SC ’06 Proceedings of the 2006 ACM/IEEE Conference on Supercomputing.

YANG, Z., TIAN, J., ZHAO, Y., CHEN, W. and DAI, Y. 2011. Protector: A Probablistic Failure Detector for Cost-effective Peer-to-peer Storage, IEEE Transactions on Parallel and Distributed Systems, 22, 9, 1514 – 1527.

ZHANG, Q., CHENG, L. AND BOUTABA, R., 2010. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*,*1*(1), pp.7-18.

ZHAO, W., XU, X., XIONG, N., and WANG, Z. 2008. A Weight-Based Dynamic Replica Replacement Strategy in Data Grids, Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conferences, Taiwan.