Agent Based Efficient Consistency Control and Replica Management in Data Grids

A Thesis

submitted in partial fulfilment of requirements for the degree of

Doctor of Philosophy

Submitted by

Priyanka Vashisht

(951011001)

Under the supervision of

Dr. Rajesh Kumar
Professor ,CSED
TU, Patiala

Dr. Anju Sharma
Assistant Professor, CSA
MRSPTU,Bhatinda

Computer Science and Engineering Department
Thapar University, Patiala

July 2017

Certificate

I hereby certify that the work which is being presented in this thesis titled, "**Agent Based Efficient Consistency Control and Replica Management in Data Grids**", in partial fulfillment of the requirement for the award of degree of "Doctor of Philosophy" submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Rajesh Kumar and Dr. Anju Sharma, and refers other research works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any institution.

Priyanka

(Priyanka Vashisht)
Reg. No. 951011001

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Dr. Rajesh Kumar
Professor ,CSED
TU, Patiala

Anju Sharma
30/10/17

Dr. Anju Sharma
Assistant Professor, CSAD
MRSPTU, Bathinda

Acknowledgements

PhD is a once in a lifetime opportunity. It is an exhausting, and emotional struggling which demands regress hard work and discipline. I am in ecstasy that, with the grace of almighty God, I have had a chance to complete it. It would not be possible without the support of many people. I would like to take this opportunity to thank the people who inspired me during the ups and downs of this pleasant experience. First and foremost, I would like to express my sincere gratitude towards my PhD supervisors, Dr. Rajesh Kumar, Professor, Computer Science and Engineering Department, Thapar University and Dr. Anju Sharma, Assistant Professor, Maharaja Ranjit Singh Punjab Technical University, Bathinda for giving me this opportunity. Both of them gave me guidance, support, and encouragement throughout my PhD candidature. Their contribution to this thesis goes well beyond the role of an academic supervisor. They helped me even when I encountered personal issues, and supported me as a family. I am really indebted to them for providing me motivating and enthusiastic atmosphere to work in. I would like to express my gratitude to Dr. Maninder Singh, Head, Computer Science and Engineering Department, and Dr. O. P. Pandey, Senior Professor and Dean-RSP, for their constant motivation and encouragement. I would like to express my gratitude towards the doctoral committee members, Professor R. K. Sharma, Professor A. K. Lal, and Dr. Anil Kumar for their constructive comments and invaluable suggestions. Many thanks to all my friends Rajesh Saluja, Vibhor Sharma and Harjeet Singh for their cooperation and help they extended as and when required. I would like to give thanks to my parents-in-law, brother and sisters-in-law for their constant support, encouraging words, love, and support. I want to thank my mother Annpurna Sharma, father B. V. Vashisht,Late Porf. B. P. Singh, who encouraged me to pursue PhD. I would also like to thank all those who supported me at any stage in my work, from the inception to the completion of this thesis. I am also very thankful to the entire faculty and staff members of Computer Science and Engineering Department, Thapar University, Patiala for their direct and indirect help, cooperation, love and affection. I dedicate this PhD thesis to my lovely children, Aradhya and Saranaya who are the pride and joy of my life.

Lastly and most importantly, I would also like to dedicate this thesis to my beloved husband, Anuj Sharma, who has been making my life so incredible and prosperous each and every day. These few words cannot express my deepest appreciation for her selfless patience, unconditional love, and endless support during these past years. I cordially thank you for being so understanding and putting up with me through the toughest moments of my life. I thank God for enlightening my life with your presence.

Priyanka Vashisht

List of Notations

n_{ij}	i^th node of j^th region
Reg_i	i^th region
$avail$	availability of file
$Freq(f_i)$	average access frequency of a file
$Freq(f)$	average access frequency of all files
Num_{f_i}	number of replicas required for a f_i in data grid D
Num_{Reg_i}	number of replicas required for a f_i in a region Reg_i
P_{cost_i}	placement cost of replica i
C_{cost}	communication cost
$P_{network}$	propagation delay of network
Rep_{cost_i}	replication cost
P_{cap_i}	processing capacity of node n_i
S_f	size of file
D_{Rate}	transfer rate of storage media
$L_{storage}$	storage media latency
$C_{storage}$	access cost of storage media
$Aval_{str}$	available storage capacity
S_{reg}	total storage capacity
S_{usage}	storage space consumed by the node
N_{reg}	number of regions in the system
ENU	effective network usage
$MJET$	mean job execution time
SU	storage used
CU	Computing Usage
T_i	time to execute i^th job
W_i	waiting time of i^th job
$N_{remfile}$	number of times the computing element read file from storage element of different region
$N_{locfile}$	number of times the computing element read file from local storage
P_{cons}	percentage of accessing consistent data
Num_{con}	number of consistent data of file f_i

Num_{f_i}	total number of replica of a file f_i at time τ
UP_{trans}	number of write request executed for replica r_i
$N_{remfile}$	number of times the computing element read file from storage element of different re
A_{read}	access cost of read request
A_{write}	access cost of write request
r_k	number of read request
w_k	number of write request
Ret_{RR}	required request rate
R_{ret}	amount of reissued requests
$Total_{req}$	total number of request made by client
$Conf_{rate}$	conflict rate
$Conf_{num}$	number of conflict
W_{rep}	number of write request of replica over a period of time
$VV_{ij_{kp}}$	vector for storing version of replica p of file k located at node i of region j
$timespamp_{ij_{kp}}$	timestamp for storing version of replica p of file k located at node i of region j
Ref_{VV}	reference version vector
Ref_{time}	refernce time stamp
$node_{led}$	list of leader node
enu	counter for keeping track of read and write request

CHAPTER 1

Introduction

This chapter provides a general overview of data grids and a detailed discussion on the issues and challenges in data replications. Data replication is a well-known technique and has been realized and extensively used within the context of distributed data-intensive paradigms such as the World Wide Web, Peer-to-Peer file sharing networks, and distributed grid environment. This chapter discusses various replica management strategies, and address the similarities and difference between data grid replica management. This chapter ends with a discussion on various replica management issues and the organization of the thesis.

1.1 Overview

In today's scenario network application technologies and computing infrastructure need effective management of large data stored at distributed locations. Technology has evolved from monolithic to open and then to distributed systems [1]. Sharing of data in such environment leads to various design issues such as security, access permissions, consistencies etc. One of the effective measure for accessing the distributed data is replication. It is the most commonly used phenomena in distributed environment. According to S. goel and R. Buyya the architectural differences in distributed environment helps in classification of various data intensive system where the data is being stored [2]. The classification of data intensive system for existing distributed environment is shown in Figure 1.1. A distributed database (DDB) is a logically organized collection of data stored at different sites of a computer network. Each site has a degree of autonomy and thus is capable of executing local applications, and also participates in the execution of a global application (the DDB management system). In distributed database, central management policies for replica management are adopted as the replicas are in possession of single owner. Due to single point management, distributed database are tightly coupled. As the whole system is tightly coupled and integrated, the

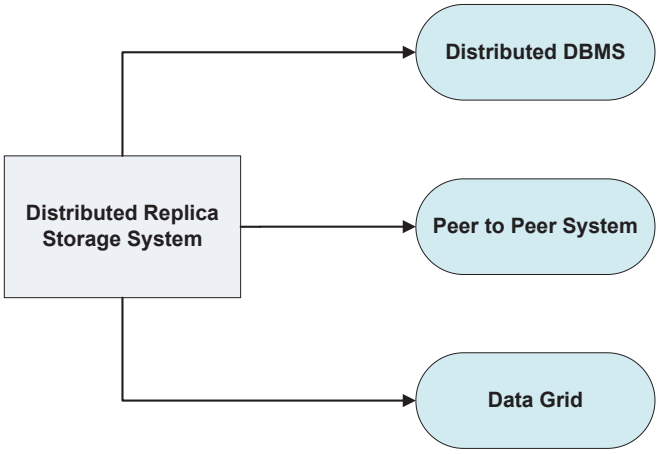

Figure 1.1: Classification of Distributed Data Storage

design choices get affected as they rely on one another. A well defined data access patterns and interfaces are provided in this environment. Additionally, the update mechanism in distributed database can be handled easily due to centralized control of the system.

Alternatively, in Peer-to-Peer (P2P) systems, no central policy is followed as the replicas are distributed at various nodes and is treated equally, unlike client server scenario. P2P systems are self-governing, as there is no dependence between distributed sites. The design choices of each site are autonomous and evolve without any intervention. The asynchronous behaviour and decentralized management of P2P systems causes monitoring issues such as update, create, delete etc. in such systems.

To increase access performance in the web, web documents can be cached on the clients, using either proxies or the servers directly Client side caching can be accomplished in two ways; either by using a browsers caching facility, or via a web proxy. Upon visiting a website, the clients browser retains a local copy of the page visited in its cache. The page can be loaded directly from the local browsers cache, if it is requested again.

A Content Delivery Network (CDN) consists of a collection of servers that attempt to speed up the delivery of web content and reduce the load on origin servers as well as the network generally. That is, within a CDN, client requests are satisfied from other servers distributed around the Internet (also called edge servers) that cache the content stored at the origin server. A client request is redirected from

the origin server to an available server close to the client that is likely to host the content required. If the selected server does not have the requested data then it is retrieved from the origin server or another edge server. The content is replicated either on-demand when users request it, or it can be replicated beforehand, by pushing the content to the content servers.

In grid environment, the replicas are formed at a central location and hierarchically distributed to various sites. A typical characteristic of grid system is that the sites always operate in a trusted environment even though they are independent of each other. In grid systems, applications are primarily dealing with read-only queries, as a result update performance is not well studied and understood. But, with progression in technology, applications will need to update stored data in grids as well. Grid systems, so far, have mainly focused on read-only queries, but the importance of write queries is also being realized and is attracting research interest.

1.2 Grid Computing

Recent trends in various scientific studies show a shift towards applications involving multi-administrative domains which are geographically dispersed. Scientific applications such as weather forecast, computation chemistry, multi-physics, tele-medicine and other scientific applications produce huge amount of data [3]. This data is then accessible to the scientific communities whose storage and computational devices are distributed all over the world. Grid technology seems to be a rational choice for all the data generated by these simulations and experiments. The term "Grid Computing" is a coalition of several computer resources from multiple organization put together to achieve a common goal. A typical grid system can be perceived as a distributed system with non-interactive workloads involving a large number of files. Grid computing is different from other existing high performance computing system such as cluster computing as it is heterogeneous, geographically distributed and is loosely coupled [4**?**].
According to Ian Foster and Kessleman, Grid Computing is a growing technology that facilitates the execution of large scale resource intensive application on geographically distributed computing resources. Grid provides an infrastructure that facilitates cooperative problem solving by collaborative use of heterogeneous resources in distributed environment [1]. Various characteristics that are defined

by Ian Foster and Kessleman to form a grid are:

- *Coordination of resources are not under centralized control:* The resources on a grid are owned and managed by different institutes or companies but the grid must manage access to them and deal with issues of security, payment, agreements with local policies etc.
- *Usage of standard, open, general purpose interfaces and protocols:* The protocols and interfaces on a grid must be standardised and accepted by all. The standards and protocols must be extensible, portable and interoperable in grid environment.
- *Delivery of significant quality of service (QoS):* The grid must ensure the required level of resource availability, security, throughput and response time for the users to maximize the utility of the whole system rather than part of system.

While these three characteristics can certainly define some grids, especially scientific grids, they can be a little too restrictive or too vague for those in industry wishing to exploit grid technologies [5]. For example, some grids may need some centralised control to deliver the desired quality of service and requiring open standards is a business model feature rather than a technological aspect of grid architecture. In summary, the definition of a grid is still not entirely agreed upon. Although, there is no overriding definition that fits all cases yet there are certain benefits that are provided by grid and are discussed in next section.

1.2.1 Benefits of Grid

- *Exploits under utilized resources:* There are at least two prerequisites for this scenario. First, the application must be executable remotely and without undue overhead. Second, the remote machine must meet any special hardware, software, or resource requirements imposed by the application. For example, a batch job that spends a significant amount of time processing a set of input data to produce an output data set is perhaps the most ideal and simple use case for a grid. If the quantities of input and output are large, more thought and planning might be required to efficiently use the grid for such a job. Often machines may have huge unused disk drive capacity, on the top of that all available processing resources may also be under utilized.

Grid computing (more specifically, data grid) can be used to aggregate this unused storage into a much larger virtual data storage thereby achieving improved performance and reliability over single machine.

- *Resource balancing:* For grid enabled applications, grid computing can offer a resource balancing effect by scheduling grid jobs on machines with low utilization. This feature can prove invaluable for handling occasional peak loads of activity in parts of a larger organization. This can happen in two ways:
  (i) An unexpected peak can be routed to relatively idle machines in the grid.
  (ii) If the grid is already fully utilized, the lowest priority task being performed on the grid can be temporarily suspended or even cancelled and performed again later to make room for the higher priority work.
- *Virtualize resources across an enterprise:* In case of multiple organizations with different policies, user of the grid can be organized dynamically into a number of virtual organizations, each with different policy requirements. These virtual organizations can share their resources collectively as a larger grid while still operating under their individual policy framework. A data grid can expand data capabilities in several ways thereby enabling data sharing in the form of files or databases. Files or databases can span many systems and thus have larger capacities than on any single system. Such spanning can improve data transfer rates through the use of striping techniques. Data can be replicated throughout the grid to serve as a backup. It can also be hosted on or near the machines most likely to need the data in conjunction with advanced scheduling techniques. It not only enables sharing of file but other resources such as services, software, specialized devices and so on. These resources are virtualized to give them a more uniform interoperability among heterogeneous grid participants.
- *Enable collaboration for virtual organization:* Collaboration is the key to effectiveness in the virtual organization. The collaboration is done for sharing resources among multi-dimensional virtual organizations having different architecture and policies. The collaboration is not limited to exchange of file only, rather access to computing resources are also provided. The resources shared among various virtual organizations includes hardware, special soft-

ware, data files, processing capacity, dissemination of information regarding other resources in grid. The collaboration among virtual resources helps in hiding the complexity of grid system from the user and provides virtulization of heterogeneous grid resources.

Grids are designed by considering the requirement of specified problems like data intensive, computation intensive etc. There exists various kind of grid topologies and architecture which offer solution to various user, business and scientific problems. Grids are categorized based on the services provided by them to the various users, scientific or business requirements.

1.3 Types of Grid

For the effective measures, different grids have been taken into considerations and are categorized as *Computational Grids, Service Grids, Network Grids, Collaboration Grids* and *Data Grids*.

- **Computational Grid:** Computation Grid is a federation of hardware and software infrastructure. It offers pervasive, consistent, dependable and inexpensive access to high-end computational capabilities [1]. It aims to provide a secure access to large number of shared processing power for compute intensive computing.
- **Service Grid:** A Service Grid provides an infrastructure to support the services provided by the grid suitable for the efficient execution of the request made by some application, resource or user. The services offered are special equipments, software, computation cycle and more. User has to enter their data and requirement to the machine offering service and receive the desired result.
- **Network Grid:** A high performance and fault tolerant communication service is provided by the Network Grid. Each participating node acts as a router among communicating nodes. Nodes also offers data caching facilities to speed up data communication between two active nodes.
- **Collaboration Grid:** Increasing usage on internet over a period of time leads for better collaboration among various entities. Grid facilitates such advance collaborations between entities. For instance, users from multi-organization domain can use different modules of a project without disclosing

components or technologies used.

- **Data Grid:** It is an infrastructure that support storage, discovery, handling, publication, manipulation etc. of large volumes of dataset located at various heterogeneous databases and file systems [6]. Distributed data give rise to many design issues like access permissions, consistency, robustness, security, maintainability, reliability etc. A Data Grid provides transparent secure access to data stored across geographically distributed heterogeneous storage resources whilst fulfilling the three characteristics as described earlier in section 1.2.1.

The next section discusses the data grid in more details.

1.4 Data Grid

Data grid is a specialization of a grid that is very commonly used within scientific computing, focused on processing large datasets. It offers a solution for collaboration and sharing of huge amount of data in distributed grid environment. Many scientific disciplines have started generating datasets from the scale of gigabytes to tera-bytes. The amount of data and the geographical distributed datasets have significantly increased causing a demand for new grid data management solutions. A. Chervenak *et al.* has suggested four principles for successful management of data in data grid [7]:

(*i*) **Mechanism Neutrality:** The data grid architecture should not be dependent on low level mechanism which is used for data transfer, storing data and metadata etc. Keeping data management architecture as neutral as possible of low-level mechanics such as data transfer, metadata storage and any other interfaces which deal with handling data on a lower level.

(*ii*) **Policy Neutrality:** In data grid, the design decisions with significant performance implication are exposed to user rather than encapsulated in black box implementation. Policy neutrality means that design decisions such as replication policies are kept open to users for substitution via application specific code. However, defaults policies should still be provided.

(*iii*) **Compatibility with Grid Infrastructure:** This simplifies implementation of strategies and provides compatibility of specialized data grid tools

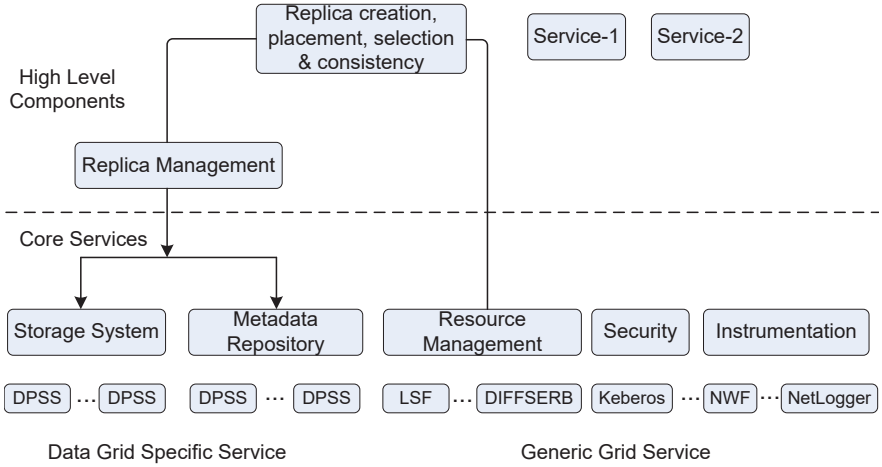

Figure 1.2: Major Components and Structure of Data Grid Architecture [7]

with lower level grid mechanism. Many tools are available for grid computing. The most well-known is the Globus Toolkit [8], used for authentication. A new data management system should use these tools to ease integration of new solutions into the existing system.

(*iv*) **Uniformity of Information:** Uniform and convenient access to information about resource structure and state is emphasized as a mean of enabling runtime adaptation to system condition. A new data management solution should use existing information systems to detect system and resource status.

The four principles lead in developing a layered architecture as shown in Figure 1.2. The lowest level provides high performance access without enforcing specific usage policies. The high level component provides the replica management and services with policies.

Data grid architecture focuses on the policy independent mechanism allowing unlimited implementation of a range of applications. It provides services that facilitate discovery, transfer, creation, management of large distributed data sets [9]. One of the principal goal of data grid is to provide easy access of globally distributed data. Due to the distributed nature of the grid, there are several issues that need to be addressed carefully in order to achieve this goal, the most important are:

- optimised access to data over the wide area to avoid large penalties on access performance
- the need for a solid, highly extensible and well-performing security
- access policy framework

Optimisation of data access can be achieved via data replication, whereby identical copies of data are generated and stored at various globally distributed sites. The overview of data replication is discussed in forthcoming section.

1.5 Data Replication

Grids have large amount of datasets which are distributed at various sites. The huge volume of data and calculations involved, create new problems regarding data access, processing and distribution [10]. Data grid provides an efficient solution for data-oriented applications that need to manage and process large data sets located at geographically distributed storage resources. It relies on replication to enhance the performance and the efficiency of the system. Replication is the process of creating and sharing multiple occurrences of particular file and places them at various locations. Multiple copies meet the requirement for better access of datasets, user collaboration, scalability and availability of data where failures are more likely to occur [11]. In case requested file is not available, the available replica of the file can be accessed to execute the request. Data replication can be optimized by two ways:(i) short term optimization and (ii) long term optimization [12].

- *Short term optimization* can be achieved through static replication. In static replication, management of replica such as creation, deletion, placement etc. are decided at the time of grid set-up, and never changes. Static replication is simple to implement and has advantage of less overhead. Despite the advantages of static replication the main drawback is that it does not adapt to the dynamic nature of grid [13].
- *Long term optimization* is attained by dynamic replication which aims at reducing average job execution time in data grids [14]. Dynamic replication is adaptive to the changing nature of the grids during the course of time to guarantee benefits of replication [15]. Dynamic replication takes advantage of file access patterns, making realistic storage assumptions, managing

available storage space in the nodes, and calculating costs of replication for providing better access[16]. Dynamic replication can adapt changes based on user requests, storage capacity and bandwidth [17]. Dynamic replication schemes are capable of making intelligent decisions to place data in the grid based on storage capacity and node availability. Multiple replicated copies of the data sets on distributed nodes need to be kept consistent when one or more copies are modified [18]. When dealing with read-only data sets, the problem of managing the replicated data is reduced as no modification has been done on the data sets. The situation is more complex when data set contents are required to be updated over time. In this case, the updates performed on one replicated copy must be propagated to other replicas.

If replica copies are not managed properly, the incoming updates applied on different replicas could result in inconsistency of data which could lead to wrong results for the requests made by clients. Maintenance of data consistency is the principal problem of replica management of updatable data sets. Consistency of data sets can be achieved by updating all the copies of data set. Based on the update technique dynamic replication is further classified as: (i) Synchronous and (ii) Asynchronous.

Synchronous Replication is also known as Eager replication which keeps all the replicas updated all the time. The operation of updating replica is not considered until both the primary and secondary site confirm the completion of operation [19]. The updating of all replicas is considered as one atomic transaction [20]. Synchronous replication guarantees data set consistency at all times. In synchronous replication, the replication technique increases transaction response time. It has the advantage of reduced overhead and high degree of consistency but at the cost of performance. Synchronous replications are not achievable when replicas are subject to failures, connected through unreliable links or having low latency links. The update propagation has to delay the update of a replica, and, even worse, could be blocked indefinitely or rejected in case of network partitions. Moreover, the performance of synchronous replication decreases when the number of replicas is high and updates are frequent [21]. Synchronous replication technique is also known as aggressive synchronization or pessimistic approach. Pessimistic approach is frequently used in replicated databases, in order to provide single-copy consistency [22]. Advantages of synchronous replication are (i) it guarantees

data set consistency all the time (ii) there is no data conflict. Disadvantages of synchronous replication are (i) this replication technique increases transaction response time, (ii) it is not suitable for mobile nodes and (iii) the primary system has to wait for all the other replicas to be updated.

Asynchronous replication does not rely on the acknowledgement from primary and secondary node. Synchronization takes place after small intervals. The duration of interval is dependent on the speed of network link and database server load. For the majority of tasks, short-term occurrence of stale data on distant servers is acceptable but access to replica is not blocked. Postponing the propagation of update phase benefits to speed up the write access equally when there are slow and unreliable links. For asynchronous replication, however, the total performance declines when all the replicas are writable, and write operations are frequent. The synonyms for this replication technique are lazy synchronization, optimistic or relaxed consistency [23]. Reliability and performance are the reasons why asynchronous approach is often preferred in most of the distributed applications.

Asynchronous dynamic replication has more overhead as compared to synchronous replication, as the replicas are updated with time and cost of replication is more than static replication strategy. Indeed, for any dynamic replication strategy, benefits of replication should always outweigh the overhead. Although the usefulness of replication in data grid systems is evident, it entails a number of maintenance issues such as replica creation, placement, resource discovery, selection of suitable replicas, the impact of replication on the performance of job scheduling, replica consistency maintenance, conflict resolution and so on. The issues of replication can be dealt with the help of replica management system (RMS). Appropriate replica management helps in optimizing data access in distributed grid environment. RMS deal with various replication issues throughout their life span i.e from the time they are created till they are deleted. Various issues that exists during data replication are presented in Figure 1.3 and are discussed in next section.

1.6 Challenges of Data Replication

Dynamic replication is an optimisation technique which aims to increase network bandwidth and availability of data and reduce total access time by considering different issues such as replica creation, placement, selection, consistency, conflicts etc. These issue are challenging and mostly considered for current research as

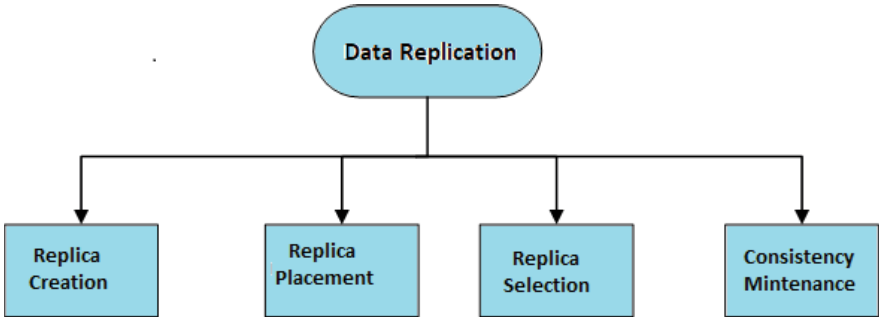

Figure 1.3: Taxonomy of Challenges in Data Replication

described below.

1.6.1 Replica Creation

Data replication is the key technique to manage large data in distributed environments such as grids. It is the process that deals with creating the identical copies or replicas of the file and store them at various distributed locations. The replicas are linked to their original file through some well defined mechanism[24]. The data replication technique allows the system in achieving its goals such as availability, access time, fault tolerance etc. Due to distribution of various files at multiple locations, a client request is routed to the replica in its vicinity, hence reducing the access time of the file. Moreover, the file requested by the client is available in the system as long as at least one replica is accessible. This increases the availability of the file and deals with the fault tolerance in grid environment. Replication also helps in load balancing and reducing the latency by creating multiple copies of same data [13].

Replication of data is bounded by two factors: (i) the size of storage available at various nodes and (ii) the bandwidth between these nodes [25]. Nodes have limited storage space and cannot accommodate replicas of every data file on the grid, while network have limited capacity for transferring them. A grid must therefore have a replica management system that manages the data files in grid environment with the aim to optimize the performance of the grid[24]. In order to obtain the maximum gain of replication, a strategic mechanism for replica management is desired. The mechanism must resolve the following problems of replication:

(i) Which metrics should be considered to take replication decision?

(*ii*) Which file to be replicated?

(*iii*) How many copies of file is required?

The selection of metrics, number of replicas and file for replication helps in optimization of replicas. As discussed earlier, the optimization of replicas are categorised as: short term optimization and long term optimization. The short term optimization is concerned with reducing the transfer time of data in distributed grid environment whereas, long-term optimization aims at reducing the network traffic in the grid by deciding which files should be replicated and where to place those replicas [26].
Though, replication is one of the key optimization techniques that supports high data availability, increased fault tolerance, improved scalability and low bandwidth consumption, the problem of replica placement has not been well studied for large-scale data grid environments.

1.6.2 Replica Placement

Although a substantial amount of work has been done on data replication in grid systems, most of it has focused on infrastructure for replication and mechanisms for creating and deleting replicas [27]. In grids, where operational control is decentralized and resources are managed by their local administrative domains, placing the replicas of a file is crucial. The placement of replica at appropriate location ensure improved scalability and reliability of the system. Replicating highly accessed files or replicas near to the clients can yield better results. However, to obtain the maximum benefit from replication, a strategic placement of replicas is done by considering factors such as quality of service (QoS), workload capacity, access frequency etc. The QoS requirements can be imposed by data requesters and grid infrastructure [28]. Grid computing infrastructure usually consists of various types of resources and the performance of these resources can be quite diverse. Moreover, different sites may have different service quality requirements as per the the system performance of the sites. Base quality of service requirements can be specified such as general distance metric or access time deadlines by users, and the system must ensure that there exists a replica within that quality range to answer the request. Another constraint is workload capacity of the servers that must be considered while placing a replica and is bounded by the bandwidth capacity

of a link. The average number of requests serviced by a replica server directly affects the average response time of the requesting sites due to queuing delays and network congestion. Thus, placement of replica near to the client reduce network delay, access time, response time, throughput etc.
Placement of replica is carried out either *statically* or *adaptively*. The static placement strategy of replicas is not suitable for dynamic environment like grids. Dynamic placement has several issues such as (i) load is dynamically changing on a node, (ii) the request for replica is quiet unpredictable [29]. Experiments have shown that dynamic placement responds better to changing data access patterns and provide better performance [30]. Replicas should be adjusted to the appropriate locations that are closer to the computing devices in order to adapt the current network environment, to reduce the access or response time as well as to maintain optimal performance of the grid environment. In addition to replica placement strategy another issue that is handled by replica management system is to schedule the request to the best replica which further enhances the system performance and provide better execution of job.

1.6.3 Replica Selection

Replication in data grids reduces access latency and bandwidth consumption. When different sites hold replica of datasets, there is a significant benefit realized by selecting the "best replica". Replicas are selected among the pool of available copies of file during replica selection process. By selecting the best replica, the access latency can be minimized. It provides an application with certain characteristics that optimize desired performance parameters, such as cost, speed, accessibility or security.
Replica selection mechanism is a challenging problem where multiple objectives play an important role during the selection process of a file. One of the major objective in optimizing the selection process in data grid is to reduce the response time of the system. In order to achieve the optimization, the available bandwidth between the storage elements that hold replicas and the requesting computing elements is examined. In this case, the node which require minimum transfer time to transport the requested site is selected. This can be challenging as bandwidth quality can be unpredictable due to dynamic nature of grids.
Although, network bandwidth plays a vital role during the selection of best replica,

other factors includes access time, access frequency, load balancing, job execution time, response time and stale data [31]. The access time and access frequency helps in predicting the best replica based on the historical logs that contain the information regarding the requested file. If replicas selected for the execution of request is not proper then the computational resources will be wasted. Replica selection is closely related to load balancing. It is concerned with all techniques allowing an evenly distribution of the workload among the available resources in a system [32]. The main objective of a load balancing is primarily to optimize the average response time of the currently executing applications, which can be achieved with suitable replica selection.
Moreover, to improve the overall performance of the system, replica selection policy should be such that it always select the consistent copy of the file. The distributed files and their replicas can be updated at any location in grid system which lead to the inconsistent copies at various location. Consistency maintenance is also a crucial problem in data grids and is discussed in next section.

1.6.4 Replica Consistency and Conflict Resolution

Consistency and synchronization problems associated with replication in data grid systems are not well addressed in the existing research with files often being regarded as read-only. However, as grid solutions are increasingly used by a range of applications, requirements will arise for mechanisms that maintain the consistency of replicated data that can change over time. Replica consistency assures that two or more replicas have the same data value. Replica synchronisation is used for establishing consistency. Replica consistency is a traditional issue in distributed systems, but it introduces new problems in data grid systems. Traditional consistency maintenance techniques such as invalidation protocols, distributed locking mechanisms, atomic operations and two-phase commit protocols [21] are not necessarily suitable for data grid environments because of the long delays introduced by the use of a wide-area network and the high degree of autonomy of data grid resources [33]. For example, in a data grid, the replicas for a file may be distributed over different countries. So, if one node which holds a replica is not available when the update operation is underway, the whole update process could fail.
Replica consistency can be achieved using pessimistic or optimistic approaches in distributed environment like data grids[34]. In pessimistic replication approach,

consistency of data sets are guaranteed at all times. Pessimistic replication has high degree of consistency and less overhead but at the cost of performance. In optimistic replication approach consistency takes place with some delay. The duration of interval is dependent on the speed of network link and database server load. With increasing dimensions of data grids, if replica consistency policies of data files are inadequate, some replica updates may not be distributed timely and thus inconsistency may occur among the primary copies of a file and its replicas. This can have an evident consequence on the results of users and the problem will become worse as the magnitude of data grid grows.
In optimistic replication approach, consistency can be delayed by allowing execution of update operation on a single node before coordinating with other replicas. Temporary inconsistency leads to certain issues: (i) possibility of reading stale data and (ii) the risk of executing conflicting updates on different nodes or regions. Optimistic approach must follow certain mechanism so that it can detect and resolve the conflicts among the various copies of file. Conflicts can be detected using various techniques like logical timestamping, version vector and semantic conflict detection. Logical timestamping follows happen-before relation among various events of distributed grids. This relationship is used to detect the update conflicts using logical clock or Lamport clocks. The extension of logical timestamping is version vector, which helps in detecting the replica conflicts. Replica conflict can be detected by comparing the version vectors of two replicas of same file. If the version vector of any replica dominates the other, the related replica is strictly newer than or equally new as the other replica. If neither version vector dominates, a replica conflict has been detected. Semantic conflict detection is an application based detection mechanism, which can define and detect the conflicts among the replicas. These conflicts are detected on the basis of operation preconditions. For every operation that may potentially conflict with other replica, applications must specify a precondition that must evaluate the operation to avoid conflicts. Therefore, the key issues in data grids is how the file and its replicas can be used efficiently and effectively to reduce inconsistencies and resolving conflicts.

Moreover the data replication strategies which address various issue discussed in this section are influenced by several parameters like data access pattern, long term stability, network variations, workload size etc. Replication strategies predict

future data usage possibilities to perform improvement in data grids by analysing the data access behaviour of users. Predictions are performed based on the temporal locality, which is an assumption that an existing popular file has a higher probability to be used more than the unpopular files. When the patterns are analysed only few files have majority of access patterns as a result the number of replications can be controlled. Only the files with more access patterns can be replication. To check the long term stability of replica management system number of simulations need to be run continuously, with varying network load and resources. When the simulations are run for long time and the replication strategy will start giving the results with less variations, only then the system can achieve the long term stability. For achieving such stability system need to understand varying patterns of jobs that are provided in the system. In addition to long term stability, workload management of the system plays a vital role in enhancing the performance of grid system. Scheduling is closely related to workload balancing and resource allocation. It is concerned with all techniques allowing an evenly distribution of the workload among the available resources in a system [35]. The access time depends on where a job is scheduled for execution of file. Therefore, scheduling is important, which helps in assigning job to the node having replica. If the jobs are not scheduled suitably, then the computational resources will be wasted. This results in uneven distribution of resources in which some nodes are overloaded and other nodes are under loaded. Thus, effective scheduling measures help in reducing overall access time through load balancing across multiple nodes [35]. This section summarized the key issues of replica management in data grid environment and various parameters that affect several replica management strategies which handle these issues . The next section discusses about the agents and their role in data grids.

1.7 Grid Standards

Standard organizations provide some defined set of rules or framework around which developers can design their applications. The standard organization for grid community is Grid Global Forum (GGF) [36], which provide the technical specifications and guidelines for grid technologies. Open Grid Service Architecture/ Infrastructure (OGSA/OGSI)[37] is well known standard by GGF, with the idea of representing everything on the grid as a service. Organization which tries to drive

the adoption of industrial, management and e-business standards are Enterprise Grid Alliance, Distributed Management Task Force (DMTF) and Advancement of Structured Information Standards (OASIS) respectively. Web Services Resource Framework (WSRF) standard is published by OASIS, it provides a set of operations that can be used by web services to implement stateful communication with resource services and allow data to be stored and retrieved. Standards bodies which provide guidelines, developing specifications, tools and software for getting the most out of the web are World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF). Open Grid Forum (OGF), provides a standard namely, GridRPC in order for clients to access remote servers with simplicity. Grid Security Infrastructure (GSI) is another standard which is responsible for secure communication between grid environment and software.

1.8 Grid Projects

Large number of projects are working on various aspects of grid such as building physical resources, creating applications or middlewares to support the grid. In this section, various current and past grid projects are discussed. Globus is an open source toolkit to build large grid testbed of European DataGrid which assist resource discovery, monitoring, security and file management. Condor is another grid project which deals with workload management and take into account scheduling policies, job queue mechanism, priority schemes, resource management and monitoring of networked system. Further, Condor-G system provides the resource management by exploiting features of Condor and Globus. To form a geographically distributed virtual computers Legion provides a meta-system of high performance machines which are heterogeneous in nature. Grid offer solution to high energy particle physics project as it produces enormous amount of data. The requirement of high energy physics is met by connecting various projects such as Grid Particle Physics (GridPP) [38], Grid Physics Network (GriPhyN) [39], [40], Particle Physics Data Grid (PPDG)[38], Sequential Access to Metadata (SAM-GRID)[41], LHC Computing Grid (LCG)[42], European DataGrid (EDG). Information Power Grid (IPG) [43] project was developed by NASA for solving huge compute intensive problems. Association of various distributed resources and instruments with earthquake researchers is achieved by setting up the Network for Earthquake Engineering Simulation Grid (NEESgrid) project [**?**]. Moreover, the

Earth System Grid (ESG) [**?**] evaluates huge data produced by various climate researchers to predict weather conditions.

1.9 Agents and Data Grid

Proactive and autonomous sophisticated entities which can act on behalf of users are defined as agent. They help in resolving complex problems and conflicts across constantly changing grid environments [31]. An agent is a hardware or software component that has capacity of taking intelligent decision to accomplish the goals successfully on behalf of its user. Ferber proposed that an entity; physical or virtual can be called an agent if it follows certain characteristics [44]

- ✓ An agent is able to act in an environment.
- ✓ An agent can communicate with other agents.
- ✓ An agent is driven by a group of tendencies.
- ✓ An agent has its own resources.
- ✓ An agent is able to perceive its environment.
- ✓ An agent has a partial representation of its surrounding environment.
- ✓ An agent has skills and offers services.
- ✓ An agent may be able to reproduce.
- ✓ An agent acts to satisfy its objectives, by taking account of the resources and skills that it possesses. It perform according to its perception about the environment and the communications that it receives from other entities.

Management of data grid resources manually is not an easy task, as it involves large scale heterogeneous resource sharing and high performance computing. Based on above mentioned characteristics, an agent based approach helps in solving data grid challenges by providing following features

- *Scalability:* As the size of data grids increases, requirement for scalability also increases. This can be achieved by decentralizing the services provided by data grids. Agents help in achieving the decentralization of the services in data grid environment.
- *Efficiency:* By efficiency, means that the data grid system should be able to provide the output in less time.
- *Adaptability:* Availability of data resources distributed at various geograph-

ical location keeps on changing due to dynamic nature of data grids. The data grid system must adapt itself to the dynamic nature of grids. The autonomous entity such as agents provide the solution to the changing nature of grid. Using agents, the system is forced to perform independent monitoring task to look for abnormal behaviour.

- *Manageability:* Managing data grid include various areas such as resource management, replica management, performance analysis, fault tolerance etc. This can be achieved through intelligent entities working together to achieve a common goal but performing different management.

Agents facilitates various task in data grid environment for replica management. Each agent can execute a specific task assigned to it independently and communicate with other agents in the system to achieve a goal. Agents are assigned a role based on its characteristics. In this current work, various agents helps in achieving a consistent data located at appropriate locations.

1.10 Thesis Organization

The thesis is organized into seven chapters as discussed below:

Chapter 1: Introduction
The chapter starts with introduction to grid computing, and then describes how it evolves from the various key computing technologies. Further, it presents the essential characteristics, various types of grids. The chapter also introduces data grid and their architectural overview in existing environment. Further, it elucidates data replication challenges, objectives to be achieved, and thesis contributions. The chapter concludes with organization of research work in different chapters.

Chapter 2: Strategies for Replica Management and Consistency in Data Grid-A Comprehensive Survey [1]
This chapter presents literature survey on replica management and their consistency control in data grid. In this chapter, several asynchronous replica manage-

[1]Priyanka Vashisht, Rajesh Kumar, and Anju Sharma,"Strategies for replica consistency in data grid –a comprehensive survey." Concurrency and Computation: Practice and Experience, 26(4),2017. doi: 10.1002/cpe.3907

ment approaches are classified and analysed based on various strategies such as topology, level of abstraction, update propagation, and locality. Some other approaches are also discussed and analysed like adaptive consistency, quorum-based consistency, load balancing, and agent-based, economical efficient, check-pointing, fault tolerance, and conflict management. Parameters on which these strategies are analysed are: type of methodology used, replication classification, consistency, grid topology, environment, evaluation parameters, and performance.The chapter concludes with problem formulation on the basis of findings of the available literature on replica management and consistency maintenance.

Chapter 3: Agent Based Optimized Model for Replica Management
This chapter presents the proposed approach for replica management and consistency maintenance for efficient creation, placement, selection and maintaining consistency of file. The approach is layered in nature. In addition to a brief overview of all the four layers and their components, the chapter also presents comparison of the hierarchical and centralized policy with state-of-the-art grid model. The chapter also explains performance evaluation of the proposed approach on simulated environment.

Chapter 4: Agent based Replica Creation and Placement in Data Grids
This chapter starts with description of architecture of replica management for data grids, followed by derivation of mathematical expressions for replication cost and placement cost. The chapter also discusses how access frequencies helps in reducing execution time. Replica Creation and Placement (RCP) algorithm is employed at two levels. At first level, RCP is applied to create replicas in a region. The creation of replica is dependent on the popularity of the file. To make the system more efficient in terms of storage consumption the number of replicas of a particular file are also restricted for proposed grid environment. The replicas are distributed uniformly among the regions, also the number of replica in the region is limited which rely on the access frequencies of the file. At second level, RCP decides the placement of replicas over various nodes within the region. Various parameters such as processing capacity, latency, bandwidth etc. are considered while placing the replicas at appropriate location. The chapter concludes with performance evaluation of the proposed work.

Chapter 5: Agent Based Replica Selection in Data Grids [1]
In this chapter an algorithm, namely, EDRA using agents for data grid, has been proposed and implemented. EDRA consists of dynamic replication of hierarchical structure taken into account for the selection of best replica. Decision for selecting the best replica is based on scheduling parameters. The scheduling parameters are bandwidth, load gauge, and computing capacity of the node. The scheduling in data grid helps in reducing the data access time. The distribution of the load on the nodes of data grid is done evenly by considering scheduling parameters. The effective scheduling measures help in reducing overall access time through load balancing across multiple nodes. The main objective of a load balancing is primarily to optimize the average response time of the currently executing requests. The proposed approach is implemented in simulated environment and compared with existing state-of-the-art approaches.

Chapter 6: Agent based Replica Consistency and Conflict Resolution
The chapter presents an hybrid approach based consistency and conflict resolution mechanism to address the problem of inconsistency in multi master environment. The hybrid approach is used for maintaining the consistency at local and global level. The local level consistency are checked either periodically or when any write request is encountered. Additionally, a leader node is selected to propagate updates by a head node of a region to maintain consistency locally. Moreover, for global consistency master nodes are responsible for ensuring consistency among various regions and resolving the conflicts between them. The conflicts are classified as *replica conflict* and *update conflicts*. To resolve the problem of update conflicts, a logical time-stamping is used while the replica conflicts are taken care by using version vectors. The simulated results depict that proposed work is better than pessimistic and optimistic approaches when writeable requests are more likely to occur.

Chapter 7: Conclusion and Future Work

[1] Priyanka Vashisht, Rajesh Kumar, and Anju Sharma, Efficient Dynamic Replication Algorithm Using Agent for Data Grid, The Scientific World Journal, 2014(2014), Article ID 767016, 10 pages, 2014. doi:10.1155/2014/767016

The thesis concludes with a brief overview of the proposed replica management and consistency approach. This chapter also shed light on open issues and future perspectives in replica management and their consistency maintenance. An agent based efficient replica management and consistency maintenance approach is proposed. The details of various layers and its components are also given. The various components of different layers based on popularity are implemented for efficient replica management. Further, agent based replica creation and placement approach is proposed that is based on replication and placement cost, latency etc. Various factors are considered for reducing the execution time. The replacement policy is also considered for proper storage utilization. To extend the features of replica management an adaptive replica selection approach is also introduced. The proposed approach considers the job scheduling policy the request to the most appropriate location which improves the overall system performance. Further, the replica management approach is extended to support the consistency maintenance which has capability to deal with inconsistencies and conflicts. Finally, the chapter concludes by highlighting the various open research issues in the field of replica management and consistency resolution.

1.11 Objectives

The following objectives have been delineated to deal with the challenges associated with replica management in grid environment:

(i) To study and explore existing Replica Management System in Data Grids.

(ii) To analyse various Consistency Services and Replica selection scheme.

(iii To propose agent based Consistent Services and Replication method.

(iv) To validate the proposed method on a real or simulated Grid for variety of Grid Service Management scenarios

1.12 Thesis Contributions

The key contributions of this thesis are:

(i) A taxonomy and comparison of the existing approaches related to replica management and consistency management has been presented. An efficient

approach for replica management in data grid environment has been proposed, and its different components have been elaborated. .

(ii) The prototype implementation of resource management and its various components has been discussed.

(iii) A replica management and consistency component on Optorsim has been setup with 50 node to support the deployment and execution of heterogeneous machines.

(iv) An agent based strategy is applied for efficient creation and placement of replica. Placement cost, replication cost and access frequencies of the file are used while creating or placing the replica.

(v) Efficiency of the system is increased by considering the popularity and the storage usage of the file.

(vi) The proposed Replica Creation and Placement (RCP) approach is compared with the state of art replica creation and placement approaches in grid environment.

(vii) An adaptive replica selection approach is presented that can be tuned to balance load, bandwidth and access frequencies of the file.

(viii) Various scheduling parameters such as bandwidth, load gauge, and computing capacity of the node are used to select the best replica of the file. The scheduling in data grid helps in reducing the data access time.

ix) The result shows the efficiency of Efficient Dynamic Replication using agents (EDRA) in terms of mean job execution time, network usage, and storage usage of node as compared to another existing approaches.

(x) A novel two stage replica consistency and conflict resolution approach has been proposed that take care of writeable replicas in multi-master environment.

(xi) An agent based optimistic approach is followed for maintaining the consistency based on access frequencies of the file.

(xii) The conflicts are handled using agents at various regions in multi master

environment. Performance analysis of proposed work is carried out in simulated environment and is compared with existing strategies.

CHAPTER 2

Strategies for Replica Management and Consistency Control in Data Grid A Comprehensive Survey [1]

This chapter discusses various existing asynchronous replication strategies available in the literature. These strategies rationalize and investigate various parameters like bandwidth consumption, access cost, scalability, execution time, storage consumption, staleness and freshness of replicas. In this chapter, several asynchronous replica strategies are classified and analysed on the basis of various parameters such as topology, level of abstraction, update propagation and locality. Some other strategies are also discussed and analysed like adaptive, quorum based replica management, load balancing, and agent based economically efficient, checkpointing, fault tolerance and conflict management. Parameters on which these strategies are analysed are: methodology, replication classification, consistency, grid topology, environment, evaluation parameters and performance.

2.1 Introduction

Data grid is an architecture for analysis and administration of massive scientific datasets [7]. It offers facilities that enable detection, creation, migration, updation and handling of huge data sets [9]. The main aim of replication is to optimize data for ensuring robustness and efficient access so as to enhance the approachability of data and to confine cost of accessing data [45]. Data grid relies on data replication for better performance and to achieve reliability of the system [46]. The thumb rule for replica creation is: if $\frac{\texttt{(Number of read only query)}}{\texttt{(Number of Update query)}} \geq 1$, only then replication is advantageous, otherwise replication may cause problems. Despite advantages of

[1] Priyanka Vashisht, Rajesh Kumar, and Anju Sharma, *"Strategies for Replica Consistency in Data GridA Comprehensive Survey"*, Concurrency and Computation: Practice and Experience, 26(4), doi: 10.1002/cpe.3907

replication in data grid, the main challenge is replica management and consistency control of data. The Replica Management System (RMS) provides the ability to access and manage data and data resources in the grid. It allows the users to create, register, place, access and update the data sets.

As the number of request for accessing a file increases with time, file located at single server leads to bottleneck in the system resulting in low throughput and response time. Creating multiple replicas at multiple locations facilitate the execution of the request in parallel with each replica providing better data access. Replicas of a file located on geographically distributed locations speed up the access of data which results in reduced execution time. Additionally, replicas also save bandwidth between the sites thus avoiding the network congestion when a sudden demand of frequently used data is made [47]. The RMS is guided by some replica creation strategies that consider some parameters such as locality of request, current and future demand of the file and storage capacity of the node elements for better performance. The degree of replication is also handled by RMS to provide the optimal number of replicas without reducing the performance of the system. Although creating replicas increase the availably and performance of the system but to maximize the gains of replication, strategic placement of replicas are required.

The advantage of the replications can be increased manifolds by placing the replicas at the suitable location. When a user generates a request for a file, large amounts of bandwidth could be consumed to transfer the file from the server to the user. Furthermore, the latency involved could be significant considering the size of the files involved and the geographic distance between user and server. Placing the file close to the user reduces the distance between user and file over the grid, resulting lower response time and high performance. RMS should consider certain placement strategies that are derived using parameters such as placement cost, access frequency of the file, bandwidth, latency etc. to optimize the performance of data grid.

Additionally, the replica selection strategy handled by RMS, should identify an appropriate replica among the pool of replicas, that best matches the user requirement. The selection of replica is done for two purposes:(i) Selection of replica for executing the job and (ii) selection of replica for deletion, during replica replacement process. The selection of replica for executing a job is based on certain

criteria such as bandwidth, popularity, latency, workload etc. Network bandwidth plays a crucial role in selection process of replicas. Slow network access restricts the effectiveness of system during file transfer irrespective of client-server implementation. Selection of best replica from different physical locations is optimized by investigating the available bandwidth between the requesting user and various storage nodes that hold replicas. The replica selected would be the one which takes minimum transfer time for transferring the replica to the requested site. At the time of replacement policy the selection of replica is done on the basis of certain criteria such as Least Recently Used (LRU), Least Frequently used (LFU) strategies, Exponential based Replica Replacement Strategy (ERRS) etc. [**?**].
The main challenge of RMS system in data replication is consistency maintenance or synchronization of data. The consistency is a process which deals in keeping replicas of a file up-to-date, i.e. consistent [48]. In case the applications are allowed to modify the data in an uncontrolled manner, the other replicas become stale. In order to re-establish consistency among replicas, there is need to propagate the modification done on the first replica to other copies of file, through a process that is usually called update propagation.
According to G. Belalem, consistency is a relation between the copies of distributed replica which defines the degree of similarity among them [49]. There exist many models for consistency in the literature [50] such as synchronous replication and asynchronous replication models [51, 52]. Synchronous replication require sophisticated resources in terms of hardware and software. This model cannot tolerate discrepancies among replicas at any instance of time. In asynchronous replication, degree of consistency can be compromised for a particular time interval. This boosts the replica access rates but discrepancies must be checked as the complexity of maintaining the consistency among the replicas increases with increase in number of update requests.
The main focus of this chapter is to provide a comprehensive survey for asynchronous replica management and consistency control in data grid, which has been classified and analysed, on the basis of various existing strategies. The next section will discuss the advantages and disadvantages of asynchronous replication.

2.2 Advantages and Disadvantages of Asynchronous Replication

Asynchronous replication offer several benefits over more costly synchronous replication. The main advantages of asynchronous replication is discussed below.

2.2.1 Advantages of Asynchronous Replication

Asynchronous replication scores on synchronous replication and are widely used strategy for grid environment. Main advantage of asynchronous replication is that the application is unaware of any updates or replication taking place at some remote location. Therefore, there is very less effect on overall performance of the system.

When updates are propagated, the replicas of a file in data grid are allowed to be accessible by the users. In case of failure of master replica other replicas are made available in the system. Asynchronous replication ensures the availability of the replicas all the time in spite of single server failure. This helps in stabilizing the system.

The high availability of the data items helps in enhancing performance of the system. When the request is made by any user, it will be fulfilled immediately due to availability of the replica. This reduces the access time of the replica for execution and the total throughput of the system will increase. The scalability of the system increases manifold in case of asynchronous replication, which also defers the process of updating the replicas to some extent. When data is updated to a site, these updates are propagated at scheduled intervals to all other replicas. This method can work over longer distances as compared to synchronous replication, thus increasing the scalability of the system. Asynchronous replication also known as "store and forward" technique [53]. It buffers the updated data for an interval before forwarding it to destined replica. The buffering process in asynchronous replication significantly reduces bandwidth consumption and yields lower cost.

The key benefit of asynchronous replication is easy recovery from faults [31]. During update propagation, if the node where update is being sent fails, then it gets stored in defer queue. When the failed node is alive again, the defer queue is drained to update the failed node replica.

Despite of various advantages, asynchronous replication has certain disadvantages.

2.2.2 Disadvantage of Asynchronous Replication

With the increasing number of computation intensive applications, there is a requirement for increasing the accessibility of updated data in the distributed environment. During data replication, duplicate copies of data are kept at suitable nodes and can be selected during job execution [31]. The placement and selection of replicas on these nodes play an essential role while getting update using asynchronous strategy. Asynchronous replication , cannot provide any assurance of consistent data all the time. It can guarantee that if no further updates are made to a particular file, all the updates will be propagated eventually. The delay between time when updates are propagated to all replicas and the changes are made is known as replication latency [54].

Replication latency leads to the problem of data loss during node failure. Asynchronous replication, rules out the possibility of node failure in the system. Rather, it is expected that at some point of time all the updates might propagate to all the replicas in future. These updates never make it to the destined node and results in the loss of transaction.

Another issue of asynchronous replication is data conflict in multi-master systems. Concurrent writes on different masters of the same data item located at different replicas leads to data conflicts [55]. Concurrent writes need to be carefully managed and possible conflicts must be resolved.

One more problem with asynchronous replication is determination of push interval while propagating the updates. If the push intervals are short it subsequently leads to excessive use of grid resources, resulting more overhead and weak performance. If the push intervals are long, it will take long to propagate the updates which cannot rule out the possibility of inconsistency. For optimal results push interval must be determined optimally.

2.3 Asynchronous Replica Management Strategies

The grid environment is dynamic and distributed in nature. For the reasons, such as scalability, performance, efficiency etc. it is always preferred to keep data in

distributed manner. As the replica occurs over long distance in grid environment, asynchronous replication is the best choice. There exists various asynchronous replica management strategies; in this article the classification of the strategies are based on following questions.

- How the structure based asynchronous replica management is achieved? The structured based management can be attained by using different topologies in grid environment. The existing topologies are graph, free-scale, multi-tier, tree, peer-to-peer and hybrid.
- What are the methods used for asynchronous replica management? This is explained by describing the level of abstraction, by identifying the part of the file for propagating the updates. The updates can be disseminated either by sending the whole file or the fragment of the file which was modified.
- How the updates are propagated using asynchronous replica management? The critical decision must be made while propagating the update. The decisions are made based on, whether specific application is client based or server based.
- How to ensure the consistency considering the location of replicas? By determining the location of the replicas of a file, the updates can be propagated efficiently. Researchers have classified locality as geographical, temporal and spatial.
- What are properties or parameters considered while developing asynchronous replica management? There are properties like adaptive, quorum based, conflict management etc. and parameters like Quality of Service (QoS), cost, space etc. being considered by various researchers. Quality of service (QoS) for replica management is further achieved from factors such as number of replicas, storage capacity, processor capacity, replica locality, cost, network bandwidth and number of hops towards root of tree [9, 56].

The detailed answers to the following questions are explained beneath.

2.3.1 Grid Topologies

The topology defines the basic layout of the data grid and relationship between various entities. A data grid consists of number of geographically distributed nodes which requires some arrangement for management of data grid nodes. The

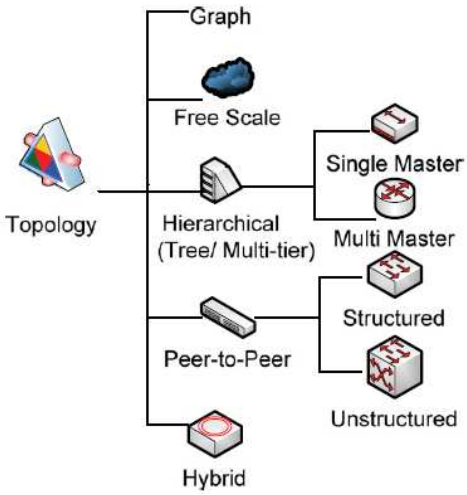

Figure 2.1: Classification of Replication Topology in Data Grid

management specifies how these nodes interact with each other. The performance of replica management and its consistency depends on various topologies that are supported by the grid environment. Grid can have any of the following topologies: *free scale* [57, 58], *graph* [59, 60], *hierarchical* [61, 62], *peer to peer* [63–65] and *hybrid* [66] as illustrated in Figure 2.1. These topologies can employ either a *centralized*or *decentralized* approach. Centralized approaches, are employed in [67, 68] whereas, decentralized approaches are employed in [69].

2.3.1.1 Strategies Based on Graph Topology

X.F. Meng *et al.* proposed a concept of spanning graph, in which a node can share the update route information by using clone, variation and crossover processes for the update routes so as to decrease the redundant propagations of the update messages [70]. The time complexity of proposed algorithm is $O(d)$, where d represents average number of parent nodes for each child. However, this strategy needs a delayed period to conduct a crossover operation, and it is difficult to predict the length of a delayed period in a specific P2P network.

M. Tu *et al.* proposed two layered Peer-to-Peer grid topology for analysing the problem of allocating sensitive data objects [59]. For accomplishing enhanced data access, survivability and security, data partitioning scheme is pooled with dynamic data replication. The proposed method assumes a two-fold layered topology. The topmost layer represents networks topology using general graph, whereas, the topology within each cluster is characterized using Directed Acyclic Graph (DAG).

For achieving enhanced scalability, solution to the replica allocation is divided into two sub parts: Optimal Inter cluster Resident Set Problem (OIRSP) decides which cluster needs to hold the shared replicas. Optimal Intra-cluster Share Allocation Problem (OISAP) decides the necessary amount of shared replicas required within a cluster and their locations. The OIRSP can be discovered using Minimal Spanning Tree (MST). Marginal communication cost can be achieved by determining the total number of replicas to be positioned in a node. Experimental results illustrate that the suggested heuristic algorithm achieved improved performance by reducing communication cost and the results are in proximity to optimal solution. The placement issue can be improved by authors using multiple objectives like security, availability and performance.

2.3.1.2 Strategies Based on Free-Scale Topology

OptorSim is a grid simulator used in European Union (EU) data grid project, provides replication policies such as placement of replica, scheduling, consistency maintenance and selection [57, 58]. A new Replica Creation (RC) strategy has been introduced which reduces the storage usage as compared with EcoModel, EcoModel-Zipf and AlwaysReplicate. The RC effectively reduces storage consumption and data access ovearhead. But, it is very difficult to write configuration files and modify other parameters used in the simulation using Optorsim.
D. Chen *et al.* proposed a node selection model which is constructed on the degree of distribution of freescale complex networks, rather than simple network [71]. Author defined two candidate replica nodes: a frequency-based candidate pool and a degree-based candidate pool. In view of the free-scale network, the data is replicated in the node with least possible local cost. The high degree and high frequency nodes are placed in degree-based-candidate-pool and the frequency-based-candidate-pool, respectively. The selection of a node for replica creation is based on the availability of node in both the pools stated above. If there is no suitable node found in the pool then the least cost of the node is calculated by summating the storage and file transfer cost. Authors have implemented Dynamic Multi-Replica Creation (DMRC) algorithm for creation of replicas on the selected node. Simulation results determine that the suggested strategy has reduced the data storage consumption and make span. The algorithm can be further investigated using consistency so that the effect of conflict can be studied among replicas

in data grid. The computational complexity for DMRC is $O(n^2)$, where n is the number of nodes.

2.3.1.3 Strategies Based on Hierarchical Topology

In this topology, node in the data grid are connected in parent/child or client/server fashion. There exist multiple levels in the data grid. The node at the top is treated as parent or server which is responsible for management of all the nodes beneath the root node. The advantage of this hierarchical structure is efficient usage of bandwidth, effective workload management and scalable management of datasets and user [72]. The problem with such topology is that there is direct communication only between the child and its immediate parent. This topology does not support any changes made in the grid.

The hierarchical topology can further be divided as; *Single master* and *Multi-master*. In single master hierarchical topology, single replica server is maintained where all the replica updates are posted [23]. Here, updates are made only on the server node located on the root of the tree, whereas all the leaf nodes have read only replicas. One central entity manages and propagates updates to all client nodes at lower level. This topology is simple to implement but has a disadvantage of single point failure [73].

In multi-master topology, a client server system can have one or more server nodes. In extreme cases, all the nodes are treated as server node [68]. In such system, updates can be done on any node hence it is very difficult to maintain the consistency in all the replicas, though it is possible to manage multiple versions of a file. This topology can withstand the single point failure but with an overhead of maintaining multiple replicas.

2.3.1.4 Strategies Using Multi-tier Topology

C. Yang *et al.* [74] have proposed a technique called One-way Replica Consistency using synchronous and asynchronous approach. In this method there are three types of nodes: super node (SN), master node (MN), and a child node (CN). SN can hold original data only, which can be replicated to MN called master replica. The CN can hold replicas based on two factors: access frequency of file and storage capacity. The files are automatically updated from SN to MN and from MN to CN. Updates can be made only on original data as the files in MN and CN are read only. ORCS considered two issues (i) check the storage capacity of the replicas and

(ii) find the access frequency of the replicas. The result shows remarkable balance between the performance of the system and maintaining consistency between the files.

In addition to ORCS, authors provide a Dynamic Maintenance Service (DMS) for maintaining replicas [18]. The DMS dynamically replicates, migrates and deletes file from the node according to variant parameters. Files are replicated if value on access frequency exceeds access rate of a file and sufficient storage space is available on the node. To avoid generation of excessive replicas, the files are migrated from one place to another depending on the requirement of the file. Finally, file is deleted if the access rate of file is less than minimum access rate and other replica of file exist on another node. The results are evaluated with other techniques such as Least frequently Used (LFU), Least Recently Used (LRU) and Bandwidth Hierarchy Replication (BHR). The ORCS and DMS performed more efficiently than other existing strategies and make efficient usage of storage element.

G. Belalem *et al.* [75] have presented hybrid approach for consistency management based on hierarchical model with three levels. At *layer 0*, the storage elements are kept which give physical support to replica. At upper *layer 1*, each node gathers the elementary element of layer 0. Each node is responsible for intra node consistency and co-operates with other node for inter node consistency at *layer 2*. The upper most layers has intelligent module, which takes decision for resolution of conflicts not resolved at *layer 1*. This approach uses multi-master hierarchical approach which combines optimistic and pessimistic approach for replica consistency.

2.3.1.5 Strategies Using Tree Topology

H. E. Ahangaran and A. M. Rahmani [45], addresses the issue of coherence while maintaining the replicas. A tree-Based clustering was used to characterize the communication capability of replicas, and reduce average response time. The clustering not only categorizes replicas having resemblance in "demand number" within clusters, but also tries to choose the shortest link. Results demonstrate that the proposed model can efficiently moderate the total transfer rate of data over data grid. Hence the data transferred in contrast to One-way method for write requests of final users is improved due to controlled solitary editable replica.

Wolfson and Milo have developed a novel method for completely connected, tree and ring networks. The processor constructs a minimum spanning tree for write

operation and propagates data to its node in a balanced way. For the determination of optimal residence set in completely connected and ring network, authors suggested constant time algorithm. In the tree network, determination of optimal residence is done with linear time algorithm. The optimal residence set determined by the algorithm improves the communication cost by using multicast algorithm. Computation complexity for the cost of residence set is *O(n)*, where *n* is number of nodes of residence set. The proposed algorithm can be further investigated on other existing network topologies. Moreover, bounded error approximations for general graphs can be analysed.

Perez *et al.* have offered a Branch Replication Scheme (BRS) which enhances the performance, scalability and fault tolerance of the system [76]. The topology is constructed by producing sub-replicas, which are placed such that no two sub-replicas can overlap each other. The consistency of the replicas and their sub-replicas can be achieved by using parallel propagation of update message. The update request is processed only at master node and then is propagated to other nodes. BRS reduces the accessing time of a file in read and write operations which, in turns, help in enhancing the overall performance of system compared with Hierarchical Replication Scheme (HRS) and Server-Directed Replication Scheme (SDRS).

2.3.1.6 Strategies Based on Peer to Peer Topology

This topology is an alternative to client server topology, which allows two nodes to maintain and update the replicas on their own, without the help of any special node like parent node. In P2P topology when the request is made for a particular file, any peer in the network can fulfil the request that has the replica of the file. In P2P topology, the requests do not rely on central server to get fulfilled. P2P topology is further classified as: *Unstructured* and *Structured*. Unstructured peer-to-peer topology is designed by nodes that unsystematically form networks with each other [77]. Unstructured P2P is easy to build as there is no formal structure imposed while making the data grid. The primary drawback of unstructured P2P is that if request for a file or a replica is made, messages will be flooded which creates congestion in the network. Gnutella [78], Kazaa [79] are examples of unstructured P2P protocols.

In structured peer-to-peer network topology nodes are arranged in a systematic

manner. In case, any request is made for a file or replica the search is done very efficiently. The commonly used structured P2P networks implement a Distributed Hash Table [80]. Structured peer-to-peer topology, such as Chord [68] and is fabricated based on certain rules. There exists an index node that has good grasp of the locations of a file's replicas, which helps in easy maintenance of replica consistency in structured P2P networks. SCOPE [65] obtains location of each replica associated with a file by building the replica partition trees with the aid of hashing function. Other example of structured P2P is WOW [81].
X. Meng and C. Zhang [82] proposed a replica maintenance strategy employing an ant colony model in unstructured P2P topology. Their main focus is to make the best use of the ant colony model for resolving the replica consistency maintenance problem. In unstructured P2P topology several definitions, such as the structure of an ant, the action of an ant and the state of an ant are provided. These definitions contribute to the design of the replica consistency maintenance strategy. The algorithm for ant creation, sending, processing and returning is based on pheromone update which ensures complete replica consistency maintenance strategy. Since, churn is a basic and inherent problem in P2P topology; this strategy takes churn problem into account to design the ant returning process so as to reduce the impact of churn on the update of file's pheromone. Simulation results showed that this strategy maintained a higher success rate with high speed and less message propagation while updating file.

2.3.1.7 Strategies Using Hybrid Topology

Grid system has complex organization that is not controlled by any simple topology. In real world various topologies are combined with each other into one system making a hybrid topology. In present scenarios, most of the times, hierarchical and peer to peer topologies are combined to get the benefits of both. By combining the topologies, researchers are able to optimize the desired results.
Choi *et al.* [66] proposed Dynamic Hybrid protocol for replication in data grid. The physical structure meritoriously combines tree and grid topology, which flexibly adjust parameters like: tree height, number of child nodes and grid depth. The values of parameters depend on the target performance measure. Improved read availability is achieved by reducing number of child nodes and height of tree while the depth of grid needs to be increased. Enhanced write availability is achieved by

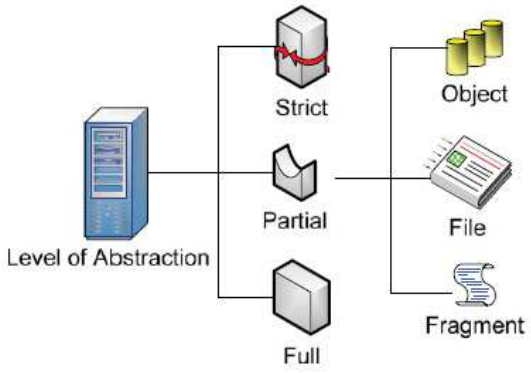

Figure 2.2: Level of Abstraction in Data Replication

reducing the depth of grid and height of tree, whereas the number of child node needs to be increased. This protocol exploits the benefits of tree and grid configuration to ensure minimum cost for operations and provide high accessibility. The throughput of protocol declines due to increase in queue length when the arrival rate for read operation becomes high. The protocol can be further investigated with some load balancing mechanism for optimizing the results.

2.3.2 Level of Abstraction

Level of abstraction defines the degrees of replication and also if any update is made on the file, which part of the file will be sent to propagate the updates. The highest level signifies that all the files are replicated on all the available nodes and for update propagation whole file is sent to the destination. The next level would be partial, where file are replicated on few nodes and only updated fragment of file is directed to the destined replica. Strict or no replication is defined as lowest level entity, which does not allow any replication. The file is stored on a single source for the execution of job. The classification for different level of abstraction is illustrated in Figure 2.2.

2.3.2.1 Strategies Using Strict Replication

In this strategy there exists one primary copy of a file and no replica. The primary copy is the only source of information that exists for a particular file. This strategy is also known as No Replication. Dirk Dullmann *et al.* [48] proposed a high-level replica consistency service, called Grid Consistency Service (GCS). The study offered some levels of consistency ranging from tightly synchronized to loosely

synchronized data based on the knowledge about data and use cases. Several consistency levels have been offered to grid user with different level of guarantee for consistency and their impact on implementation. Grid users can dynamically select any consistency services by regulating replica consistency level. Level-0 is not used for database controlled files as they provide a snapshot view of the system. Level-1 is not used for most applications. Level-2 is managed at local level, here remote level consistency is not provided. Level-3 also called "replicated federation" and can be used for distributed databases.

2.3.2.2 Strategies Using Partial Replication

The partial replications means that the whole file is not replicated rather a part of the file is replicated. The partial replication can be done at two levels: (i) Object and (ii) Fragment level. Objects are any piece of data that is being shared. Logical connection between two objects of a file gets lost by replicating individual sets of objects in a file [49]. Consistency between replicas of an object can be maintained by optimistic replication algorithms, but not between objects [83]. Fragment level replication means any part of file can be replicated as per the requirement. In this type of replication only the necessary fragment of a file is replicated and is transferred for modification. Spigot [84] framework is one example of fragment level consistency.

J.M. Perez and F.G. Carballeira [76] have addressed the problem of scalability and replica alteration in data grid by studying the use of a novel data replication method, called Branch Replication Scheme (BRS). In BRS, replicas are composed as a set of sub-replicas, structured as hierarchical tree topology. These sub-replicas do not overlap and the unification of the set of sub-replicas is complete replica. This model is suitable for applying parallel I/O techniques and offers an effective approach for updating data replicas, which results in replication and update of the replication tree efficiently. The update on the replica is made at leaf node only and is propagated immediately to the upper node. Only the part of the file is sent for the update. The results of the model illustrate that BRS enhances the data access performance for read and write processes with different file size.

2.3.2.3 Strategies Using Full Replication

In full replication, the whole file or metadata is replicated. Most of the researchers have considered the full replication of file. Main disadvantage of full replication as compared to partial replication is the overhead of transferring whole file, which consumes lot of bandwidth while transferring the file.

A.S. Syed *et al.* [85] have developed an algorithm which collectively use data partitioning schemes with dynamic replication. There are four modules involved in this algorithm:

(i) Network Module, which provides client server architecture
(ii) Dynamic randomization process ensures delivery of packet to next hop
(iii) Secure data share which provide optimal and secure allocation of sensitive data
(iv) Replication data grid offers sharing and coordination between various grid resources.

The algorithm developed assigns correlated data shares in large-scale peer-to-peer data grid by partitioning them at fragment level and encoding the fragment while transferring to attain data survivability, security, and access performance in data grid. Replicating data share may enhance the performance of the system but security is compromised. The study can be elaborated by considering optimal placement of data share using multiple factors.

P. Xu *et al.* [86], proposed metadata management algorithms which ensure the retrieval of up to date dataset by users. The approach added two components, namely, Operation Broker and Mapping Keeper in the original data access framework. The communication is divided into three phases: Phase one is requesting phase where a client interacts with the metadata server to acquire a lease lock for desired operation and select a nearby data server containing updated replica for a requested transaction. Then in the second phase, the client directly communicates with the selected data server to carry out the transaction. When the transaction is accomplished, the data server sends a close report to the metadata server to release the lock in third phase. This strategy ensures metadata level consistency. The simulated results have revealed the proposed scheme achieve performance gain up to 3.3 times for write intensive jobs compared with WriteMaster scheme.

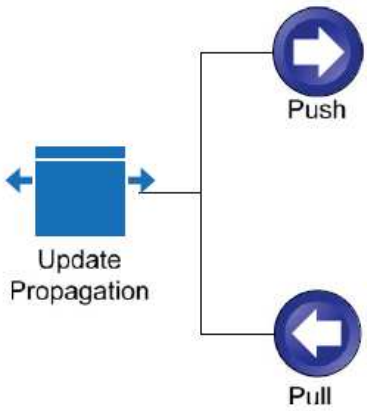

Figure 2.3: Classification of Distributed Data Storage and Replica management System

2.3.3 Update Propagation Mechanism

Whenever, write operation is performed on any copy of file or data, it must be propagated to all existing replicas. This helps in ensuring the consistent data on the data grid. In update propagation, primary node contains all the files where updates can be done and communicates with other nodes so as to make two replicas consistent by transferring the changes quickly [87]. One of the desirable properties here is epidemic propagation [88]. Any node can transfer its own updates and can receive it from other replicas also. This helps in attaining eventual consistency irrespective of the network topology conditions, including poor or incomplete network connectivity, as in Usenet [89]. It is robust when communication links change or when nodes fail. To propagate the updates across all the replicas of grid, two approaches are used i.e. push and pull as illustrated in Figure 2.3.

2.3.3.1 Strategies Using Push Mechanism

Push based approach is used when all the updates are propagated from the server to the clients. Servers are participating very actively to keep all the replicas identical [87]. This approach is useful when there is a large number of reads and updates of the file. In push based approach, if updates are being pushed then response time for client will be negligible. Main disadvantage of Push mechanism is that no updates will be performed if the server is down.
In Aggressive Update Propagation (AUP) approach [83], also mentioned as Push protocols, updates are propagated to all replicas even when they are not requested

by the nodes. Aggressive protocol is functional when replicas need to sustain a fairly high degree of consistency. The read to update ratio of AUP is relatively higher than pull replication. They need to keep all the replicas identical.
Flooding is a breadth-first replica consistency maintenance approach that forwards an update message to all the nodes in the network [90]. It ensures the replica consistency of a file, but pushes large number of update messages and thus the network bandwidth is wasted. Also, a great number of redundant update messages are produced in the network; node may receive multiple duplicated update messages.

2.3.3.2 Strategies Using Pull Mechanism

The pull based approach is client based scenario which means the client will initiate the changes by pulling the data from the server [87**?**]. When there are less number of updates as compared to the reads of file by client, pull based approach can be used. The updates are made on the stable storage [88].

2.3.3.3 Strategies Using Hybrid Mechanism

In hybrid approach both pull and push mechanism are combined together.
Gossip [91] based replica consistency maintenance is the push and pull combined algorithm, namely, Balanced Consistency Maintenance (BCoM), in which an update message is pushed to the replica nodes actively by the node which creates the update message and in pull approach replica node sends query messages to obtain the current updated replica. Though the push approach ensures the update message to be received by the replica nodes, it cannot make the update message arrive at all the replica nodes in the P2P topology with high churn rates. Also, the pull approach may result into an update message not being received by all the replica nodes in time. BCoM enhances the availability by lowering the discard rate of files from almost 100% to 5% but at the cost of latency.
K. Xie et al. [92] makes an improvement on the gossip based replica consistency maintenance approach, which records the information of nodes and update message passed through, i.e. the so-called trace label is stored in the header of the update message to avoid forwarding the update message to the same node multiple times. Moreover, it adopts Bloom filter algorithm to denote the trace label in an update message to reduce the consumption of network bandwidth. However, since this approach could only make the trace label in an update message shared

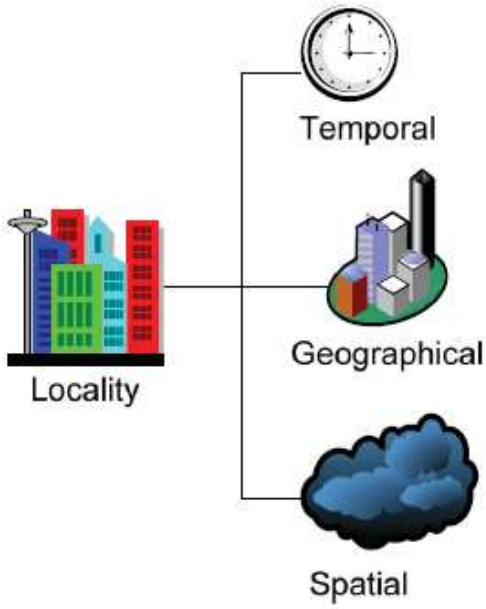

Figure 2.4: Localities in Data Grid

and transferred in a single propagation path, its ability to decrease the redundant propagations of the update messages is greatly limited.
IAUP is the modified version of traditional AUP [83]. IAUP has two steps, one at the master node where master pushes the message to replicas if update is done at any node. At second step, the secondary replica nodes, where message is received by the replica, check the timestamp of received message with the local replica time-stamp. If the difference between the two values is 1, the node will send the message to pull the information from the server. The simulation result shows that this technique improves the reliability with high degree of consistency without affecting performance of the system.

2.3.4 Locality

The locations where the replicas are placed affect the efficiency of replica management strategy. Locality plays vital role during replication, which helps in improving the overall access time and availability of update data in grid environment [34]. In data grid environment the locality is primarily divided into three parts i.e. geographical, spatial and temporal locality as illustrated in Figure 2.4.

2.3.4.1 Strategies Using Temporal Locality

Temporal constraint of consistency is specified in terms of time intervals. Temporal constraints can be classified as events and time interval [56]. The event indicates a method to check if replica management or update is required to maintain the consistency. The time interval indicates the duration when creation, placement, selection or update is initiated by the system.

J. Chen and Y. Yang [93] have investigated well-defined conventional consistency and inconsistency conditions and the run-time ambiguity of task completion period in work-flow systems of grid. They have presented four situations to a fixed-time constraint i.e. weak consistency, strong consistency, weak inconsistency and strong inconsistency. A temporal verification is set periodically that is conducted on built time, at the time of initialization as well as at execution time to identify any violation, and change any strong and weak inconsistencies to strong and weak consistencies, respectively. If during the verification process, the previous fixed time constraint is of strong consistency then no modification is required. Moreover, for weak consistency, they have established a technique which uses mean activity time redundancy and temporal dependency between fixed-time constraints consistently.

OptorSim [57, 94] uses a prediction function based on temporal and spatial locality irrespective of the cost of accessing data on grid. The prediction function helps in maintaining the replicas based on the popularity of the file. The files are created or deleted depending on the access frequency of the file. Moreover, the job is scheduled based on various scheduling strategies such as Random, Sequential, Zipf etc.

2.3.4.2 Strategies Using Geographical Locality

Ranganathan *et al.* [95] have proposed a hierarchical framework in which different replication approaches are assessed using three different access patterns. Access patterns for various files can exhibit different localities i.e. temporal, geographical and spatial locality. Temporal locality specifies that prospects for accessing a recently accessed file are more likely. Geographical locality is formed on the perception that a file accessed by a node is expected to be accessed by its neighbouring node. The spatial locality specifies the files in the vicinity of recently accessed file have high probability to be accessed. The simulation was done using

six strategies, namely, best client, cascading replication, caching, plain caching, caching plus cascading replication and fast spread with three access patterns i.e. P-random, P1,P2. Fast spread saves 25% bandwidth consumption compared with caching. On the other hand, the storage utilization of the caching is reduced up to 50%. Results show that if access pattern contains smaller degree of geographical locality then average response time and bandwidth consumption can be reduced. Also, results indicate that depending on various requirement of grid scenario the different replication strategy is selected.

H. Shen *et al.* [96] introduced a poll-based distributed file consistency maintenance method called Geographically Aware Wave (GeWave). GeWave ensures reliability of data consistency with high scalability and low overhead. In GeWave nodes are organized in a hierarchical manner based on their time to refresh (TTR) value and location. Relying on the adaptive decentralized polling method, where replica node elects another node with an updated file for ensuring the consistency of its replica. Polling is conducted within the nodes which have close proximity that dramatically enhances the efficiency of consistence maintenance. Experimental results show that GeWave performs better than other consistency maintenance schemes with respect to effectiveness, scalability and churn resilience. The experiment is performed on real world PlanetLab testbed and proved the effectiveness of GeWave has increased 44.5% to 53.5% compared with SCOPE, UMPT and Push/Pull method. The major disadvantage is that authors do not consider any failure recovery mechanism during node failure.

2.3.4.3 Strategies Using Spatial Locality

The spatial constraints of consistency describes coherence predicate [91] that indicates the degree of replica consistency to the primary copy. Consistent coherent predicate means that replica should have full degree of consistency with primary copy. The weak consistency means that a certain degree of inconsistency with primary copy can be tolerated. No requirement predicate means that multiple versions of data can exist at a time.

P. Zhang *et al.* [97] used a decentralized swarm intelligence based approach for replication using agents in data grid. Agent find the location of the node and also compute “degree of heat” to create a new replica. The access information along with the neighbour's information of the node helps in determining the workload of

the node. The workload will be migrated to other nodes with the help of agents. Agent also helps to find the fittest node where the file should be migrated or replicated. Simulation results validates that the approach outperforms No replication and Economic model in terms of execution time but the storage space utilization is marginal with respect to the economic model.

In view of spatial locality, Predictive Hierarchical Fast Spread (PHFS) [15] attempts to forecast forthcoming requirements and pre-replicates data in hierarchical structure to enhance area of vicinity in accesses. The algorithm collects the data of accesses from whole system and generates access log files by means of data mining techniques; it finds the association between files for forthcoming forecasts. The nodes at higher levels have supplementary computational power and storage capacity, and consequently, the algorithm replicates more replicas at higher levels and only replicas with high priority are replicated at lower levels. The simulated results show that PHFS showed improvement of 22% compared with Common Fast Spread (CFS) in terms of performance and latency. The results can be optimized more by considering the storage space for evaluating the priorities.

2.4 Other Strategies Used For Replica Management and Consistency Control

2.4.1 Adaptive Replica Management and Consistency Control Strategies

Due to the dynamic nature of data grid environment number of users varies over a duration of time. The approaches for replication management should be adaptive in nature to the changing size of the grid to provide optimized results. IDEA [98] is a self-managed middleware that adaptively satisfies system's requirements, and it is based on the continuous consistency model provided by H. Yu and A. Vahdat [98].

R. Chang and J.S. Chang [99] proposed Adaptable Replica Consistency Service (ARCS), which uses access weight in order to accomplish effective load balancing and replica consistency in data grids. The access weight indicates the average rate of recurrence to access secondary replica within a node. Partial consistency service

is applied on secondary replica with infrequent access for saving network bandwidth. Whereas, full consistency service is well-designed for a secondary replica having high access probability in order to decrease the delay while accessing the replica. The experimental simulation leads to the conclusion that ARCS outperforms Lazy and Aggressive protocols in terms of total job time and average file access delay.

J.S. Chang and R.S. Chang [100] have advocated a novel Replica Consistency Decision Model with Naive Bayesian Classifier (RCDM-NBC) which addresses the replica consistency decision problem. Here, replicas are categorized as: a master replica and numerous secondary replicas. They have considered the system as multi-master model. All RRAs (Regional Replica Agents) know the whereabouts of additional master replicas in order to deliver strong consistency service over all master replicas. The modification of data at one master replica is propagated immediately to other master replicas. This ensures that a master replica in a region constantly hold up-to-date data. When an update is requested by secondary replica, it first points to intra-region communication, if the region has a master replica. Otherwise, a master replica in the upstream for the secondary replica is there and is named as a source master replica. RRA ensures the consistency of the replicas within a region. RRA performs two processes, i.e., a training and a periodic decision process. The training process evaluates the likelihood from the training samples and accurate priori probability, so that RRA acquires the most specific statistics about the average arrival rate of request for accessing the replicas. The Naive Bayesian Classifier in decision process helps in determining the category of consistency service for each secondary replica. The simulated results demonstrate that RCDM-NBC is more compliant to a dynamic nature of the grid by taking into account flexibility and adaptability as compared with other models.

H. E. Chihoub [101] proposed three fold method for managing cost effective consistency in cloud computing. First, a self-adaptive approach namely, Harmony, dynamically adjusts the level of consistency at runtime to enhance the performance of system and availability of replicated data. Second, a consistency-cost efficiency metric is introduced which reduces the overall cost based on relative calculations of expected cost and probabilistic estimation of consistency, without violating the consistency requirement of the user. Third, a behaviour modelling method is introduced that dynamically evaluates the application and obtains its

consistency requirements. The overall system performance has improved by maintaining desired consistency while reducing monetary cost for user. The result shows that Harmony reduces the read of stale data by almost 80% and throughput is increased by 45% as compared with weak and strong consistency.

T. Kraska *et al.* [102] suggested a consistency management that dynamically guarantees the consistency by rationalizing data, which helps in optimizing the cost of system. The data can be divided into three data categories i.e. *A, B, C*. The *A* category represents the data having high level of consistency or strong consistency. Category *C* contains the data with weak consistency, whereas, *B* category is the mixes of strong and weak consistency. B data dynamically chooses the consistency level according to the calculated threshold value. The threshold value is dependent on the probability of number of update conflicts happens during transaction and the monetary cost of pending operations in update queue. This approach ignores the staleness of data in eventual consistency model as well as the storage cost of the replicas.

X. Wang *et al.* [103] proposed a centralized structure based adaptive consistency mechanism. Various applications require different level of consistency. In this paper, the consistency is divided into four categories "C" and is associated with numeric value. *C=1* is considered when number of read operation are more than number of update operations. Strong consistency protocols are applied to this category. *C=2*, here, number of read and write operations are high. The application can choose between consistency and its cost, to maximize the benefit of both. *C=3* means number of read operations are less than number of write operations. Weak consistency model is preferred, otherwise the overhead of updating all the replicas immediately, will result in high cost. *C=4* represents the situation where number of reads and writes are low. It follows the mixes of second and third consistency category. The result depicts number of transaction decreases significantly while the consistency requirement is fulfilled.

J. Liao *et al.* [104] proposed a file system namely Gmei, which provisions lazy and adaptive replica synchronization mechanism between storage servers (SSs) in geographically distributed environment to enhance the I/O performance without the intervention from the metadata server (MDS). It updates only the chunk list data structure and delays update of all appropriate chunk replicas which benefits in reducing write latency. The chunk replicas are updated only when storage servers

have to handle huge amount of concurrent requests for a specific replica. Experimental results show that the proposed mechanism has enhanced I/O performance by reducing the synchronization overhead in particular application context. Authors can enhance the performance of their algorithm by reducing synchronization cost and also robustness of chunk list should be improved.

2.4.2 Strategies Using Quorum Based Consistency Control

Quorum or voting based protocols deals with the failures of components in distributed systems. Every replica is assigned a non-negative vote (quorum) and the threshold value for read and write operations [105, 106]. When a read operation is to be performed, the replica system must collect the votes from other participants. The operation is executed if the collected votes exceed the defined read threshold, "R". Similarly, the threshold for write operation "W" must be reached when an update transaction is invoked. The obligatory conditions for the quorum system are as follows [107]

- The sum of read and write threshold for each replica is larger than the sum of all the votes of participants i.e. $R + W >$ sum of votes.
- The write threshold W must satisfy that $2W$ is larger than the overall number of participant votes.

J. Abawajay and M. M. Derris [108] proposed a new quorum based data replication protocol which aims to, providing high availability, reduce update cost and maintain consistency of data. In this paper, a replica placement policy had determined how many replicas to create and where these replicas would be placed. Replication maintains equilibrium between the load of data and requests inside the system, both on the host and network levels. This was achieved by involving smaller number of nodes in the execution of both read and write requests which have low response time followed by a replica consistency control policy, which determines the level of consistency among replicas. The load balancing helps in enhancing the system reliability as well. The proposed approach was compared with read-one-write-all (ROWA) and grid structured protocol (GS) by means of data availability, response time, communication costs and data consistency.
S. Goel *et al.* [109] discusses that the synchronization protocols used in homo-

geneous systems which cannot be implemented in a heterogeneous grid environment and proposed a synchronization protocol for heterogeneous grid architecture based on a quorum system, able to handle multiple networks partitioning using time stamp. There are many other quorum based protocols mainly for database systems which are discussed by various researchers [110–116].

2.4.3 Strategies Using Load Balancing for Replica Management and Consistency Control

The main objective of a load balancing is primarily to optimize the average response time of the currently executing applications [14]. When combined with consistency it increases the reliability of the system.

I. Frain *et al.* [117] proposed a dynamic consistency based protocol called elementary protocol. Nodes are arranged in a tree based coterie structure. A permutation operation is applied to achieve a new coterie with lesser load. This can be done by swapping of two selected nodes (parent and its son) in a tree, where load of son is always lesser than father's load. The load of quorum "Q" is defined as maximum number of loads of all the processors "P" of nodes, in the route from root to leaf. The overall load of the coterie is calculated by summating overall loads of all the quorums. The simulation results show a significant improvement when number of processors are limited with high number of request results in better throughput. The system throughput is increased between 20%-50%, in best case the efficiency is increased to 50%. Due to elementary permutation large number reconfigurations of coterie happens which results in poor scalability.

A quorum based consistency protocol has been proposed which uses load balancing for logical organization of grid nodes, using structured tree (coterie) [118]. Grid nodes of tree have three characteristics (N, S, V), where N represents the node which modifies or create this version, S symbolizes stamp which represent time instance for creating or modify the version of replica and V is the value of replica. A node is defined in n versions and can acquire three states i.e. free, occupied and blocked. Free states means that all the available version of replicas of node are released, Occupied state guarantees that at least one version of replica is released and the Blocked state represents blockage of all versions of replicas The quorum is constructed from the root to a leaf node of the tree using load balancing strat-

egy. Access frequency is the key parameter to calculate load on the node. The load balancing strategy has three states of load which are associated with numeric value. These are defined as: (i) *Under loaded* if the node is in passive state,(ii) *Medium* when the load of node is midway (iii) *Up loaded* when nodes are over-loaded. Reconfiguring the coterie provides better workload balance in order to increase access performance . The result depicts that the consistency of replicas is dynamically balanced by following the state of individual node of the coterie. The communication cost of read and write operations as well as load of coteries are reduced by reconfiguring the coterie. Some of load balancing solutions have been proposed in the literature [19, 58].

2.4.4 Strategies Using Agent Based Replica Management and Consistency Control

Author uses multi-agent based strategy for accomplishing replica management consistency control goals between the replicas of file by efficiently propagating the newest updates judiciously [119]. The proposed agent based replication strategy with fault tolerance comprises two modules: (i) replication architecture and (ii) agent based replication method. Replication architecture provides an arrangement for refining data availability with the help of three types of agents, namely, base station agent (BS agent), node agent (N-agent) and update agent (U-agent). BS agent keep the system stable by performing various function like expansion, contraction, fault tolerance, whereas, N-agent is accountable to mobile agents and U-agent is responsible for maintaining consistency between nodes. Replication system is separated into two levels: base station and mobile node. Replication method propagates the updates to various replicas stored on nodes of replication architecture. Hybrid approach for replication used by the authors lead to data conflicts, which can be detected and resolved by using proposed multi-agent based dynamic replication strategy. The optimal number of node selection during replica placement enhances the overall proficiency of the system in terms of storage cost, consistency maintenance, update propagation and communication cost.

H.M. Lee and C.H. Yang [120] proposed agent based consistency service. The agent assists Grid Consistency Service Module (GCSM) and Grid Transaction Module (GTM). GCSM keep track of storage space of the node and on-line availability of the replicas during execution whereas, GTM take care of consistency and

integrity of the replicas. GCSM and GTM coordinate with each other to make consistency and backup of a node more convenient and secure.

2.4.5 Strategies Using Economically Efficient Replica Management and Consistency Control

G. Belalem [121] proposed model for replica consistency management in data grid, which combines both pessimistic and optimistic approach together in economic model. Pessimistic approach helps in supporting quality of service in the model. The performance is improved in the consistency model by optimistic approach. At level 0 the local consistency is controlled and managed whereas at level 1 there exists virtual nodes which are used to represent nodes at level 0. This is an economic model that uses Dutch and English auctions to resolve the conflicts in the replicas. Economic model is compared with hybrid and optimistic approach and outperforms them in terms of number of divergence and average response time.
I. Fetai and S. Heiko [122] introduced a dynamic and adaptive cost based concurrency approach (C3) which uses both weak and strong consistency according to different application scenarios in cloud environment. The model uses multi-master approach. The C3 approach is three-fold. First, it dynamically evaluates the cost based on weak and strong consistency strategy; second, the set of rules are provided for best possible consistency level with minimum cost; third, it provides a procedure that assures the adaptive adjustment to the level of consistency at runtime in the transaction mixes supported by C3. The C3 protocol is independent of organization of a specific service provider. The evaluation of C3 demonstrates that transactions in adaptive manner outperform the transactions which have static mode in terms of cost and performance.

2.4.6 Strategies Using Check Pointing Technique for Consistency Control

Check pointing is the process of saving the state of replica consistency at a particular instance of time. In grid environment if any replica has failed during transaction, the request can be migrated to other replica where the transaction can be executed from a checkpoint. For ensuring the consistent recovery of replica failure an efficient and scalable check pointing mechanism for concurrent and inter

communicating process are required whose states must be synchronized. Synchronization is needed as the concurrent process while passing the message can change their internal state. Three check pointing strategies were described for concurrent processes [123]

- Co-ordinated check-pointing: The checkpoints are always synchronized by the inter-communicating processes to ensure the consistency. While initiating the check-pointing operation, the transit of all the messages take into account by the processes. During synchronization the Co-ordinated check-pointing blocks the processes, which degrades the system performance in the grid environment.
- Uncoordinated check-pointing: The checkpoint operation in this scenario is initiated at any time without considering the message transit. One major problem with Uncoordinated check-pointing is domino's effect. The domino effect can cause repeated and unnecessary roll-backs by the process. The Uncoordinated check-pointing is combined with message logging to avoid the domino's effect and overcome the drawback of Co-ordinated check pointing. In message logging the information of message is stored at secondary media. During recovery phase, the message is accessed from the storage media rather than initiating roll-back to know the previous consistent state [124].
- Communication-induced check-pointing: Some selected checkpoints are considered for synchronization. For large-scale environments like distributed grid environment this check-pointing is not appropriate [125].

Large cluster environments using check pointing were stated in the Legion [126], Cactus [127, 128] and in Condor [129]

2.4.7 Strategies Using Fault Tolerance with Replica Management and Consistency

In grid environment maintaining replicas with regular asynchronous insertions is a critical problem that involves judicious management and balance between availability and consistency. Several efficient approaches to maintain replica consistency have been proposed over a period of time.
Lau and Madden [130], proposed a methodology to restore a failure node by using

data replication in case of machine failure in a distributed environment. Their approach called HARBOR, uses timestamps to determine the recovery process by copying or updating record required by the process. On three node distribution database the HARBOR is evaluated on runtime with two other approaches and offers an efficient recover performance. HARBOR provides high availability of data and recovery from crash in an optimal way. It reduces the complexity cost by the factor of 2, compared with traditional on-disk log.

J. Gracia and C. Ordones [131], proposed new algorithms which efficiently repair and measure replica consistency in distributed databases. The algorithm is divided into two parts: first it is suggested when number of update operations is less; it can continuously update the large number of records on multiple sites. In second case, when number of operations increases; to optimize the results large sets of data records are not updated immediately but few foreign keys are responsible for update. In former case large sets of records are reconciled with data itself, whereas, the latter case suggests the reconciliation of summarized set of records. This approach is applicable when symmetric size difference of set of records is small as well as network speed is slow. The results are compared with pf-comparator which is based on checksum, and depicts the effectiveness of the approach. These algorithms also decrease the prerequisite of tracking database insertions and deletions with temporary tables or explicit indexes.

H. Yu and A. Vahdat [132, 133] proposed numerous metrics to calculate the excellence of replicated services. A middleware layer manages consistency restraints between replicas and permitting applications to adaptively employ synchronization for better results based on certain parameters like network, request characteristics and current service. The protocol applies consistency and the failure characteristics of the original network to evaluate the metrics. The evaluation parameters are only the access of data rather than the quality of replication.

2.4.8 Strategies Using Conflict Management in Replica Consistency

As discussed in one of the previous sections, multi-master systems can improve the performance of both read and write operations. However, they must be carefully planned in order to avoid conflicts between concurrent writes on the same data

item done at different replicas. An update conflict happens when two users update the same data item at different master replicas. Conflict management is an application specific topic, and can be dealt in different ways: (i) conflict avoidance, (ii) conflict detecting and (iii) conflict resolving [52].

Conflict avoidance can be obtained with pessimistic and optimistic replication. Pessimistic replication approach is similar to synchronous locking techniques. In optimistic replication, when locking is not used, conflicts may happen and need to be detected and resolved. Different solutions that are used to avoid the conflict are straightforward solution, procedure codes etc. [134].

Conflict detection is obtained with the method named vector clocks [52]. A vector clock is a data structure that is used to propagate synchronized information in a distributed environment. It is usually implemented as an array of "M" elements, where "M" is the number of master replicas. The array contains timestamp values; when node "i" has the timestamp value "a" in the "j^{th}" element of its vector clock. It means that, it has received all the updates from node "j" with timestamp up to "a". The size of this data structure is the main concern when vector clocks are used in environments with many replicas, and can impose a limit on the system scalability.

After detection, conflict must be resolved. Different mechanisms that exist for the resolution are manual and automatic. Manual conflict resolution detects the conflicts and it provides two versions of the data to the user. The conflict can be resolved either by merging the content of the new versions or discarding one of them by the user. Automatic conflict resolution is performed by specific procedure that is applications specific. In replicated file systems different procedures are used to resolve conflicts depending on the type of file that experienced the conflict timestamp. For example, when the file is a compiled file, the conflict can be resolved by recompiling it again from its source. After resolving the conflicts, the RMS will decide which particular replica should be selected for modification.

G. Belalem and B. Yagoubi [55] proposed model for consistency management of replicas in data grid, which combines both pessimistic and optimistic approach together. Pessimistic approach helps in supporting quality of service in the model. The performance is improved in the consistency model by optimistic approach. Authors have implemented their approach in Globus project which aims to resolve conflicts between the replicas by using collaborative negotiation between

representatives of nodes in data grid. Here data grid consists of two levels. At level 0 the replica is managed with the help of replica manager using uni-master and multi-master as well. At level 1 there exist virtual nodes which are used to represent nodes at level 0. This is an economic model used to resolve the conflicts in the replicas. Load balancing factor is also included in the model which helps in reducing the number of divergence and conflicts between replicas.

2.5 Conclusion

This study presents a survey on replica management and consistency strategies for dynamic data grid environment. Replica consistency is summarized based on certain parameters, namely, (i) Replication Classification,(ii) Consistency, (iii) Grid Topology, (iv) Simulator Used, (v) Evaluation Parameter and (vi)Performance and is tabulated in Table **??**. Various asynchronous approaches have been considered by researchers to accomplish goals efficiently even when replicas are subject to failures or connected through unreliable or low latency links. The proposed methods available in the literature address one or more of the following issues: reliability, scalability, fault tolerance and load balancing etc. These strategies try to improve certain parameters like: freshness, make-span, communication cost, update cost, response time, bandwidth consumption, access latency, load balancing, maintenance cost, job execution time, fault tolerance and strategic replica placement. Majority of replica consistency or synchronization strategies aim to achieve the correctness and quality of service (QoS) of replica. Future scope of asynchronous replication is to include higher flexibility of swapping and choosing replica strategies to optimally adapt the dynamic nature of grid environment. Various strategies have been proposed to manage the large number of replica and their consistency in existing grid environment. Most of the proposed strategies are based on simulation and not on real environment. Employing these replication strategies in real environment, testing and evaluating the actual efficiency are still an open area and are of high interest. Moreover, the strategies or techniques have not proven to be consistent enough to take care of heterogeneous and dynamic nature of grids. Most of the techniques do not include multi-master approach which leads to a problem of data conflict. Conflict resolution is another critical issue which is augmented by a few replication techniques. Consistency and conflict management in multi-master environment is an open research question for

writeable data. Storage issue is another research area where existing strategies need some attention as most of the existing strategy assumes unlimited storage space. Though, most of the previous work uses same metrics for all applications but there is need to develop application specific asynchronous strategy, where each application has different degree of requirement.

Table 2.1 Replica Management Strategies Analysis

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
QoS aware data replication based on data importance [9]	Centralized	N	Hierarchical Client-server	Not mentioned	Number of nodes Selected, Mean Position, Cost	outperforms Greedy Algorithm.
Consistency Using Tree-Based Clustering [45]	Distributed	Y	Hierarchical Multi- master	Optorsim	Average response time, total data transferred	outperforms One-way and Simple approach.
Replica Consistency Service [46]	Centralized	Y	Flat Single master	Optorsim	Conflict/ retired request rate	Various consistency models are presented.
Efficient Dynamic Replica Algorithm [14]	Decentralized	N	Hierarchical	Optorsim	Execution time, storage and bandwidth	Outperforms LRU and BHR.
Replica Consistency Service [48]	Distributed	Y	Hierarchical	Not required	Consistency	Old replica consistency model is enhanced theoretically.
Load Balancing for achieving Consistency [49]	Distributed	Y	Hierarchical	Optorsim	Load balancing, response time	Advantages of both synchronous and asynchronous consistency.
Relaxed Currency and Consistency [51]	Centralized	Y	Mid-tier database	DBMS and backend server	Timeline consistency , finer-granularity consistency	Not mentioned.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Replica Consistency Service [21]	Centralized	Y	Hierarchical Single master	Real Environment	Database Schema, Initial Database Population, Number of logical Database	Not mentioned.
Constanza [23]	Centralized	Y	Hierarchical Single master	Real world use case like LCG and EGEE	Load Balancing, Fault tolerance	Not mentioned.
Replica Placement Strategies [57]	Distributed	N	Free-scale	Optorsim	Processing time, storage consumption	Replica creation algorithm based on social ability is made.
Optorsim [58]	Distributed	Y	Free-scale	Optorsim	Job time, Simulation time, Job access pattern	Economic model shows improved market performance.
Secure Data Object Replication [59]	Distributed	N	Graph	Optorsim	Communication cost	Outperforms K replication, No replication, complete replication.
Replica creation in complex network [71]	Distributed	N	Free-scale	Optorsim	Makespan, storage consumption	Outperforms Alwaysreplicate, EcoModel and EcoModel-zif.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Distributed concurrency control [60]	Distributed	Y	Wait-for-graphs	real	Message overhead	Better than serialization graphs approach.
DUP [61]	Distributed	Y	Peer-to-Peer	Not mentioned	Query latency, query cost	Outperforms PCX and CUP.
Consistent Hashing and Random Tree [62]	Distributed	Y	Tree	Theoretic ally proven	Latency, Swamping, Storage	Caching protocol.
Updates in unreliable system [63]	Distributed	Y	Peer-to-Peer	Analytically proven	Message overhead, scalability	Robust algorithm and can be applied to unreliable networks.
CUP [64]	Distributed	Y	Peer-to-Peer	Stanford Narses Simulator	Latency and cost	Outperforms PCX.
Content-Addressable network [67]	Distributed	N	Hierarchical	ns simulator	Scalability, robustness, latency	Multiple dimensions are introduced.
Chord [68]	Distributed	Y	Peer-to-Peer	Iterative met	Scalability, communication cost	Not mentioned.
Hybrid replication Grid topology [66]	Centralized	Y	Hybrid	SimJava	Throughput, response time, cost of communication for read/write operation	Outperforms Logarithmic and Tree protocol

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Decentralized file replication [69]	Distributed	N	Peer-to-Peer	Own Simulation environment	Storage, file availability	HAF, Random and Group algorithms are compared.
An optimized stratergy for update path selection [70]	Distributed	Y	Peer-to-Peer	Own simulator	Number of redundant message	Hybrid Push-Pull achieves high fidelity.
Automatic data placement and replication [72]	Distributed	N	Free-scale	Own modular simulator	Load balancing, response time, network bandwidth	Better performance than Random data placement and Threshold based algorithm.
Decentralized replication Strategies for P2P based scientific Data Grid [**?**]	Distributed	N	Peer-to-Peer	Own data grid	Response time, number of hops, bandwidth consumption	Path and requester and N hop distance placement strategies are introduced.
One-way File Replication [74]	Centralized	Y	Hierarchical	OptorSim	Number of reads / writes, file size, threshold, replication frequency	Better than LFU,LRU and BHR.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Consistency management approach [75]	Distributed	Y	Hierarchical	Optorsim	Communication cost, number of divergence, response time, number of conflicts	Better than economical approach.
Hybrid approach for consistency management [75]	Distributed	Y	Hierarchical	Not mentioned	Communication cost, adaptively, consistency	Not mentioned.
Branch Replication [76]	Distributed	Y	Hierarchical	Omnet++	Bandwidth Consumption,	"Improved data access performance than hierarchical (HRS) and server-directed (SDRS)."
Structuring Unstructured Peer to Peer network [77]	Distributed	N	Peer-to-Peer	Custella	Search efficiency, message size	Better than Gnutella-like strategy and random walk strategy.
Efficient load balance with partial knowledge of system [80]	Distributed	N	Peer-to-Peer	Not Mentioned	Load balancing	Not mentioned.
WOW [81]	Distributed	N	Peer-to-Peer	WOW	Latency, throughput, connectivity	Provides platform for various applications.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
SCOPE [65]	Distributed	Y	Peer-to-Peer	Not mentioned	Fault tolerance, maintenance cost, operation effectiveness, scalability	Not mentioned.
Ant colony model based replica consistency maintenance strategy [82]	Distributed	Y	Peer-to-Peer	Peersim	Higher update success rate with high update speed, less update messages	Resolves the problem of churn.
Improved Aggressive Update Propagation Technique in Cloud Data Storage [83]	Distributed	Y	Hierarchical	Discrete event simulator has been developed using a C++	Number of inconsistent reads, average response time	Outperforms LUP and AUP.
Spigot [84]	Centralized	Y	Client-Server	NIST Net v2.0.12	Turnaround time, storage space, bandwidth	Not mentioned
Data Grid concepts for data security in distributed computing [85]	Distributed	Y	Peer-to-Peer	Not mentioned	Data security, data availability, response time	Not mentioned.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Optimizing write operation on replica in Data Grid [86]	Centralized	Y	Client-Server	OptorSim	Latency, throughput, performance gain, read ratio	Better performance than Write master.
Epidemic algorithms for replicated databases [88]	Centralized	Y	Hierarchical	Not mentioned	Response time, commit ratio	Outperforms ROWA.
Analysis of distributed multi-periodic systems to achieve consistent data matching [90]	Distributed	Y	Graph based on architecture of the application	Theoretical analysis	Data matching, Queue size, consistency	Not mentioned.
A balanced consistency maintenance protocol for structured P2P systems [91]	Distributed	Y	Peer-to-Peer	P2PSim	Availability, latency, scalability	Outperforms SCOPE from almost 100% to 5%
A Trace Label based consistency maintenance algorithm in unstructured P2P systems [92]	Distributed	Y	Peer to Peer	Gnutella	Network bandwidth, reduced message	Better than flooding, can be applied to ad-hoc and sensor networks.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Optimal placement of replicas with locality assurance [**?**]	Centralized	N	Hierarchical	Not mentioned	Workload balance, number of replicas	Theoretically proved better than QoS aware data placement.
Identifying Dynamic replication strategies [95]	Distributed	N	Multi-tier	PARSE Tool	Bandwidth consumption, Response time	For getting faster response time Cascading is best.
A replication strategy based on Swarm intelligence in spatial Data Grid [97]	Distributed	N	Hierarchical	Optorsim	Storage space, number of jobs	Outperforms economic model.
PHFS [15]	Centralized	N	Hierarchical	Not mentioned	Latency	Better performance than CFS.
Multiple states based Temporal consistency for dynamic verification of Fixed-Time constraints in Grid workflow systems [93]	Not mentioned	Y	Not mentioned	Not mentioned	Cost	conventional work is outperformed by mean WC number N and mean WI number M.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Optorsim [5]	Distributed	Y	Hierarchical	Optorsim	Access time, throughput, bandwidth, latency	Based on JavaSim.
Adaptive consistency guarantee for large-scale replicated service [135]	Distributed	Y	Hierarchical	Planet-Lab	Delay, communication overhead	Evaluated the adaptability and performance of IDEA.
Adaptive replica consistency service (ACRS) [99]	Distributed	Y	Master-Secondary replicas	Optorsim	Load balancing, network bandwidth, access delay	Better than Aggressive and Lazy protocols.
Replica consistency model in Data grids [100]	Distributed	Y	Hierarchical Multi-master	Data Grid simulator an extended module based on NDHU Grid client	execution time of jobs, flexibility, adaptability	Outperforms Lazy, Aggressive and ARCS protocol.
Harmony [101]	Specific to application	Y	Specific to application	Amazon Elastic Cloud Compute (EC2) , and the French cloud and grid test bed Grid5000	Cost of consistency, number of stale reads	Performed better by almost 80% than weak consistency and by 45% than strong consistency model in Cassandra.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Consistency rationing in the Cloud [102]	Distributed	Y	Hierarchical	TPCW benchmark	Response time, cost, threshold	Not mentioned.
Adaptive replica consistency for Cloud storage [103]	Centralized	Y	Hierarchical	OptorSim	Amount of operation , consistency	Better than strong and eventual consistency.
Adaptive replica synchronization for distributed file systems [104]	Distributed	Y	Peer-to-Peer	Gmei file system in C, Linux environment	Write latency, Throughput	Improves I/O data bandwidth dynamically with minimum synchronization overhead.
Weighted voting for replicated data [107]	Distributed	Y	Hierarchical	Violet	Number of read and write, round trip delay	Not mentioned.
Quorum-based data replication protocol with data consistency [112]	Distributed	Y	Peer-to-Peer	GridSim	Response time, data consistency, data availability communication costs	Outperforms ROWA and GS protocol.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
An efficient replicated data access approach for large-scale distributed systems [110]	Distributed	Y	Box shaped grid	Proved by theorems	High availability, low communication cost, fault tolerance	Compared with grid configuration and tree quorum.
Managing data using neighbour replication on a Triangular-Grid structure [111]	Distributed	N	Hierarchical	Not mentioned	Storage capacity, high availability, fault tolerance	Not mentioned.
Group protocol for Quorum-Based replication [113]	Dependent on application	Y	Dependent over application	Proven by theorems	Communication cost	Not mentioned.
Dual-Quorum replication system for Edge services [114]	Distributed	Y	Peer-to-Peer	Emulab	Availability, response time, and consistency	Outperforms ROWA and ROWA-A.
Hybrid quorum protocol for Byzantine fault tolerance [115]	Distributed	Y	Client server	Emulab	Communication cost, access time	Better than BFT and Q/U analytically and experimentally.
Dynamic reconfiguration of a coterie [117]	Centralized	Y	Hierarchical	Neko	Throughput, scalability	Not mentioned.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Replica consistency maintenance using load balancing strategy [118]	Centralized	Y	Hierarchical	Gridsim	Load balancing, cost of communication, consistency	Not mentioned.
Multi-agent based data replication [119]	Distributed	Y	Graph/Tree		Update propagation, consistency, storage and communication cost	Fault tolerance is incorporated.
Distributed Backup Agent Based on Grid Computing Architecture [120]	Distributed	Y	Free-scale		Security, available, storage space	The agent make accessibility more convenient and secure.
Economic model for consistency management [121]	Distributed	Y	Hierarchical	Optorsim	Number of divergence, response time	Outperforms optimistic and pessimistic approach.
Capability and capability computing in Legion [126]	Centralized	Y	Modular	Ethernet network	Latency, Fault frequency	Outperforms Pessimistic, Optimistic and Message logging protocol.

Table 2.1 – continued from previous page

Methodology	Classification	Consistency	Topology	Simulator	Parameter	Performance
Efficiently Repairing and measuring replica consistency in distributed database [131]	Distributed	Y	Relational database	Syentic and real database	Access of data	Faster than previous state of algorithm.
Design and evaluation [132]	Application Specific	Y	Application Specific	TACT	Numerical error, order error and Staleness	A new layer is introduced in existing consistency middleware.
Cost and limit of availability for replicated services [133]	Distributed	Y	Hierarchical	Event-Driven simulator	Availability, fault-load	Outperforms voting and aggressive protocol.
HARBOR [130]	Distributed	Y	Not mentioned	Javas TCP socket, SQL	Runtime overhead, recovery	Outperforms ARIES and 2PC.
Conflict resolution using collaborative negotiations [55]	Distributed	Y	Hierarchical	Simulator in Java	Load balancing, QoS	Not mentioned.

CHAPTER 3

Agent Based Efficient Model for Replica Management

This chapter outlines an agent based efficient Replica Management (RMM) Model to address the problems of replica management in large-scale data grids. Management of replica in distributed grid environment helps in improving the performance of data grid system while ensuring efficient use of both computational and storage resources. This chapter describes various strategies that support RMM model for managing distributed data in grid environment. An overview of layered approach of the model along with use cases of proposed strategies: Replica Creation and Placement (RCP), Efficient Dynamic Replication (EDRA) and Replica Consistency and Conflict Resolution (RCCR) are also discussed. A simulation is done to justify the choice of topology for RMM model based on two parameters: workload and impact of topology on availability. The experimental results shows that choosing a hierarchical topology helps in balancing the system load and increases the availability of file.

3.1 Introduction

In today's scenario the implementation of actual data grid which can be used to their complete degree of utilization as a part of day to day scientific work is very difficult and expensive. Moreover, the possibility to evaluate various grid scenarios, optimization strategies, user patterns and topologies in repeated and controlled manner is nearly nil. In order to test the effectiveness of grid, simulation provides a cost effective solution for modelling data grid environment. The simulation environment provides an opportunity to test various strategies and implementation paradigms as well as their effectiveness before deploying on actual network. Simulation has the ability to execute multiple tests and models with varying system

parameters and experimental set-up with little overhead. These features of simulation attracted various researchers in simulating data grid environment including data replication protocols and techniques to support development of data grid middleware.

Often, the feasibility for evaluating the complex scenarios using real grid environment is not possible with various issues like scalability, availability, dynamic nature and cost. Because of dynamic nature of the grid it is hard to do the performance evaluation in controlled and repeatable manner. As a result, grid testbeds are limited, moreover, constructing an adequate sized testbed is time consuming and as well as expensive. Additionally, the testbeds require handling different policies at various resources. The extensive use of simulation is done for evaluation and modelling of real world environment, for factory assembly line, business process to computer system design. Subsequently, the simulation and modelling has emerged as an important discipline.

Consequently, in complex environment like grids many application specific technologies, tools and standards have been introduced for handling large volume of data. These include application specific simulation like network simulator ($NS2$) [136] simulation environment such as Parsec [137], simulation languages like Simscript and simulation environment such as SimJava [138]. There exists number of tools for simulation for various applications, but very few are well-maintained tools that are available for simulation in data grid environment. GangSim [139], developed at the University of Chicago, is a set of Perl modules which is built on top of Ganglia distributed framework. GangSim was developed for studying different scheduling strategies in grid environments, while focus is on exploring the communication between local and various virtual organization having different resource allocation policies. The SimGrid toolkit [140] is developed in C language at the University of California at San Diego (UCSD) for the simulating application scheduling. The resources are modelled that are time-shared and the network load is introduced for real traces of grid environment. SimGrid is a powerful system that allows creation of tasks in terms of their execution time and resources, with respect to a standard machine capability. GridSim [141] was developed at the University of Melbourne, which supports simulation of various types of grids and application model scheduling.

A data grid simulator namely, OptorSim [58] has been developed and is written in

Java. OptorSim was developed as part of European DataGrid project to check the credibility of optimization strategies used in replica optimisation services. OptorSim simulates grid topologies, set of jobs, resources and data movement around the grid using various parameters to evaluate the performance of optimization strategy. Optimized RMM model is developed on the top of Optorsim. The domain analysis of RMM model is provided in next section.

3.2 Domain Analysis for RMM Model

RMM model is designed to perform two-stage efficiency through replica management and consistency control. Replica management takes the replica placement at distinct grid nodes while considering actual network load. Moreover, this stage of optimization uses replica creation and deletion process to reflect changing conditions of dynamic grid environment. Additionally, an effective replica selection strategy has proposed which selects the best replica on the basis of current status of the network. During the second stage, the consistency among the replicas of a file is maintained for writable replicas. Consistency mechanism in RMM model helps in providing the up-to-date results. The RMM model provides transparent access to the grid system and various strategies that support the replica management in data grid. This model has full control on all the files that exists in the system, which allow it to manage the files efficiently. The management of the replicas are done either by creating, deleting, placing or updating the files. This requires a close interaction among various components of the proposed system. There are certain parameters such as access frequency, processing capacity, storage capacity etc. that affects various proposed strategies, namely, Replica Creation and Placement (RCP), Efficient Dynamic Replication (EDRA) and Replica Consistency and Conflict Resolution (RCCR). Different components of RMM model from user prospective is shown in Figure 3.1

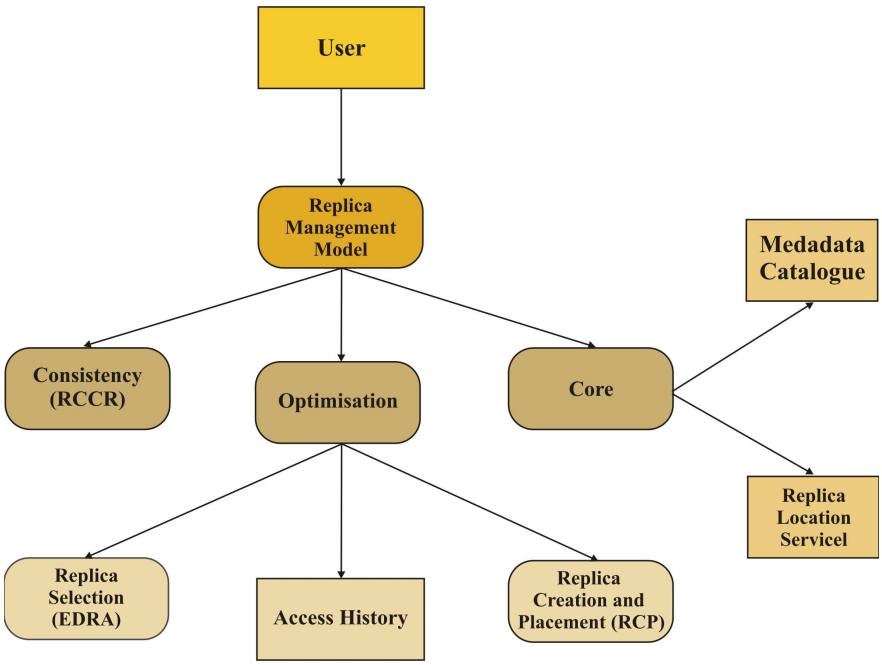

Figure 3.1: Components of Replica Management Model from User Prospective

The main functionality of RMM model is replica management, by creation, placing and cataloguing the replicas, which is supported by RCP strategy and is discussed in Chapter 4. RCP is an adaptive strategy which takes the decisions for replication and placement based on the access frequency of the file.
When user submit any request for execution to RMM model, the EDRA helps in selecting the best replica such that the execution time is reduced and network resources are used effectively. EDRA strategy is elaborated in Chapter 5. When ever a write request is submitted by the user, it modify a file or its replica which leads all the copies of file in inconsistent state. The consistency of the file and its replica is maintained by proposed RCCR strategy and is discussed in Chapter 6.
A good replica creation strategy will offer faster access to data files with minimum access cost and maximum availability. An efficient scheduling strategy leads to best replica selection with less job completion time. Strategically placed replica helps in optimizing the overall performance of the system. At the end, the consistency ensures availability of up to date file at all the time.
A high level view of RMM model and interaction of its various system components are shown in Figure 3.2. The components of RMM model are elaborated in the forthcoming sections.

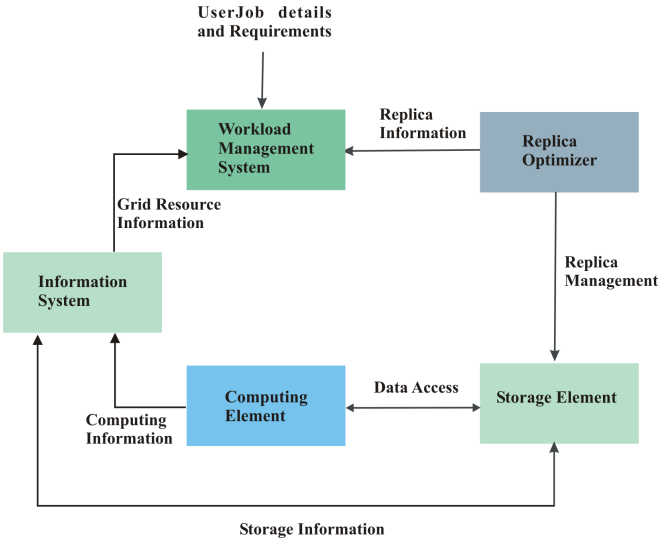

Figure 3.2: High Level View of RMM Model and its Interaction with Various System Components

3.3 System Components of Efficient RMM Model

Components of proposed RMM Model built on top of OptorSim is discussed below.

3.3.1 Grid User

An application or a broker that submit jobs on to grid resources is considered as a grid user. This component is able to query/request dataset transfers, submit jobs and register for events. Within OptorSim, it has been implemented by creating a specific data grid user object for a particular application or scenario.

3.3.2 Workload Management System (WMS)

Workload Management System (WMS) has the responsibility of mapping replicas resources to jobs in such a way that the resources can be used efficiently and usage limits cannot be exceeded. WMS also submits and monitors jobs via interactions with the CE and maintains a persistent database of job information to enable recovery in the case of job failure. WMS interacts with several other grid middleware components such as the Information System, the Data Management Services and the CE as shown in Figure 3.2. For a user, however this is all transparent process.

They supply the description of the job and its requirements in Job Description Language (JDL) to the user interface component of the WMS and the interactions between other services deal with the WMS itself.

3.3.3 Resource Broker(RB)

Resource Broker (RB) act as an agent and is the main component of Workload Management System (WMS). The WMS communicate with various components of grid such as Information System, Data Management Service, CE [5]. Information System provides information regarding the currently available grid nodes and their characteristics. Data Management Service gathers information regarding location of the data or file that is needed for the completion of the submitted job. The Computing Element (CE) monitors the authentication and authorisation of the submitted job. The RB component is the intelligent agent of the WMS, which is responsible for the optimization of the resources. The resources are matched with the submitted job based on the ranking provided by the RB. The ranking of the resources are obtained on certain factors such as access frequency, current load on the node, scheduling strategy etc. After the mapping of resources and jobs, jobs are forwarded to the chosen CE.

3.3.4 Grid Node

The grid environment consist of number of nodes. Node is the lowest level entity in data grids. Each node has three main components i.e.

- Computing Element (CE)
- Storage Element(SE)
- Replica Manager (RM)

A node can have zero or more computing elements and storage elements. Some special nodes can have one Replica Manager (RM). The pictorial representation of the grid node is depicted in Figure 3.3.

3.3.4.1 Replica Manager (RM)

RM is the single point logical entry for the user to submit a request in data grids. The RM component manages communication between various components of the

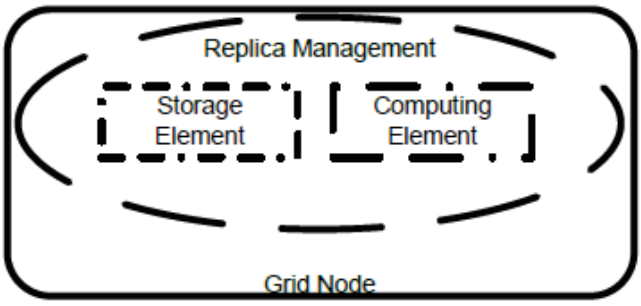

Figure 3.3: The Grid Node

system. RM interacts with other services through intelligent agent (RB) to obtain the complete information about the resources like location of file, access frequency of a file, optimization strategy etc. The choice of various service components are done by specifying them in configuration files of Optorsim. A particular component is made available at the time of execution by exploiting Java dynamic class loading feature. Replica Manager can interact with external services in addition to obtain the location of a file. Replica manager uses GridFTP protocol for the transportation of the file.

3.3.4.2 Computing Elenemt(CE)

The computing element serves as a computational resource for the execution of the jobs by using data which is kept on storage resources. CE has two main components:

(i) *Gatekeeper:* It provides a channel to establish communication between grid and non-grid users.

(ii) *Job Scheduler:* It is an interface between the request made by the user and Replica Manager (RM).

The structure of CE is shown in Figure 3.4. At local level RM take cares of the request submission and also handles the monitoring information during the lifetime of the request. The request submitted to the Gatekeeper is checked for authorization using Local Centre Authorisation Service (LCAS). If the submitted request has privilege to use the resource according to predefined policy, then the request is mapped to the local resource using Local Credentials Mapping Service (LCMAPS).

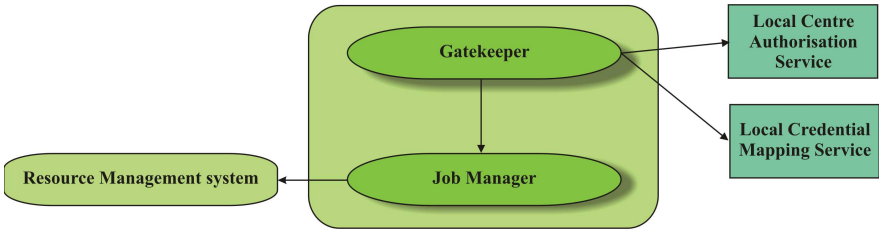

Figure 3.4: The Computing Element and its Components

3.3.4.3 Storage Element (SE)

The storage element offers storage resources for storing data for fulfilling the requirement of the submitted jobs. The Storage Resource Manager (SRM) [142] helps in providing the transparent access to the stored data at various distributed storage systems using uniform interface [142]. The SE is a layer between the storage and grid middleware service, but cannot store file itself [5]. When a request is made for a file or its replica, SE returns the Transport URL used for accessing the file based on specific transport protocol such as GridFTP. In addition, SE also caches the requested file. SE can add or access files from mass storage and also support third party transfer of files between two SEs.

3.3.5 Metadata

This provides the basic information and properties of all other data. In proposed replica management model various kind of data need to be dealt with.

- *Filename meta-data:* It provides physical characteristics of a file as accounted by the Storage Element.
- *Logical filename meta-data:* It permits user to store and identify the logical file name. In addition, it also include the overall lifetime of a files and its alias, security data, provenance data.
- *Provenance meta-data:* It provide the time and location for the origin of data and by whom it was generated.
- *Security meta-data:* It records all information related to authorization and authentication such as user name, passwords etc.
- *Secretarial meta-data:* It keep track of logs, job monitoring and accounting.

3.3.6 Replica Location Service

Replication facilitates the global access to geographically distributed data which leads to better system scalability, robustness and reduced access latency. However, number of replicas of a file at various locations in a system introduces additional issues, such as they must be located at places which can be accessed easily, must be consistent and their lifetime can be managed. The Replica Location Service(RLS) is a system that provides and maintains access to information about the physical location of file and its replicas [143]. The RLS system contain two types of components:

(i) *Replica Location Index (RLI):* The RLI maintains information of replicas at single storage resource, thus containing up to date and reliable information about independent local state [5].

(ii) *Local Replica Catalogue (LRC):* The LRC maintains the soft state collective information obtained from any number of LRCs [5].

Each data file on the grid is provided with Globally Unique Identifier (GUID) based on Universally Unique Identifiers (UUID). In LRC each GUID is mapped to one or more physical file names, which represents the physical location of the each replica of the data. The RLI contains mapping of LRCs and the GUIDs. When a request for replica is made, at first client queries RLI to find out which LRC holds the mapping for given GUID. At second stage, the identified LRCs are queried to find the actual physical location of file or its replica. At the time of deployment the LRC is configured to construct RLIs. List of GUIDs which are maintained by the LRCs to maintain RLIs are published periodically. The periodic information send to RLIs in compressed using bloom filter object [5]. The operational RLS architecture is shown in Figure 3.5.

3.3.7 Agents in RMM Model

For managing and creating the agents, Java Agent Development Framework (JADE) has been used. Every agent is associated with a container that executes inside its JVM. RMM model contains main agent and child agents, which are located at master node and head node respectively. The main agent has additional properties for handling child agent. Basic structure of communication among agents is

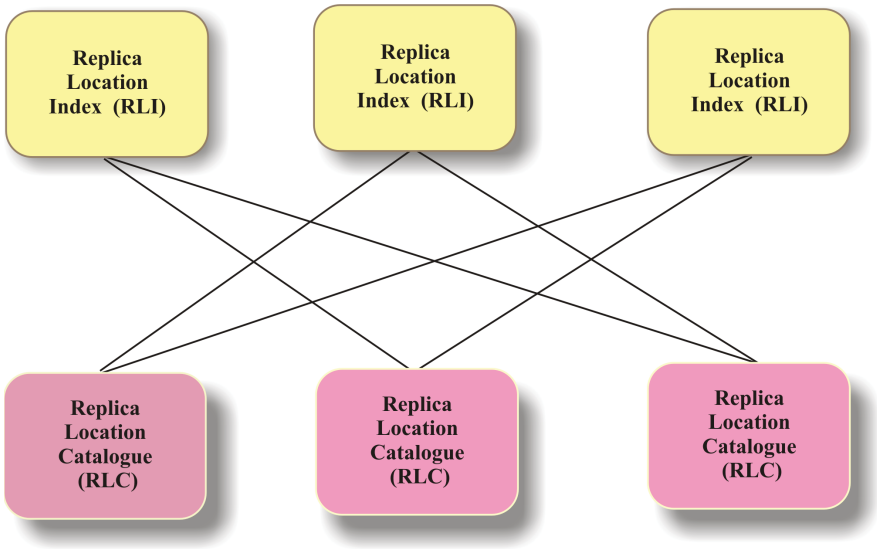

Figure 3.5: RLS Architecture and its Components [143]

shown in Figure 3.6. The main agent located at master node has additional agents: Master Node Analyser (MA), Request Scheduler (RS), Request Allocator (RA), Access Information Database (AID) and Global Information Collector (GIC). The head node contains child agent with additional agents: Information Probe (IP) and Local Node Controller (LNC). These agents are running to manage replicas and their files, handling request and provide interaction among different regions. All the message exchange between these agents is controlled by Agent Communication Channel which is JADE's message transport system.
For optimising the communication among the agents JADE-MTP is used. It uses existing connections instead of creating new one each time when request is transported to remote site. The communication between main and child agent follows push-pull mechanism. The master node can ask the status of any region or subregion by requesting the head node at any instance of time. Consequently, if any update is made at master node it pushes the information to head node and can pull information from the logs of head node after specific time interval.

3.4 System Software Stack for RMM Model

Agent based RMM model architecture identifies the basic components of the system. The architecture defines the purpose and functions of its components, while indicating how these components interact with each other. Each layer shares the

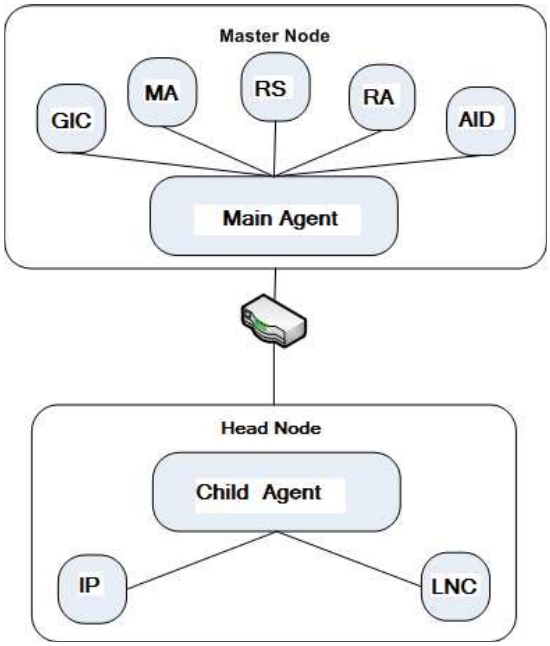

Figure 3.6: Interaction of Agents in RMM Model

behaviour of the underlying component layers. The software stack diagram describes functioning of each component in the layer from bottom to top layer as shown in 3.7. The contribution of each layer is as follows.

- *Bottom layer :* It links the nodes to the fabric layer in the grid so that middleware can have the knowledge of the available resources which can be used by grid jobs. Major components of bottom layer are Information Provider (IP) and Local Resource Manager (LRM). IP provides information at two levels, one at machine level other at network level. The machine level information include number of processors and their processing capacity, availability, storage space, load of the node etc. Network bandwidth between the node and latency between each link are provided by network level information. Job Manager provides the interface between grid and non-grid components through LRM. The resource manager provides the monitoring information during the lifetime of the node such as its configuration, current status, logs etc.
- *Middle layer :* The physical computing resources are clustered together to form a region. The application jobs running on grid having different constraints is handled by Workload Management System (WMS), whose re-

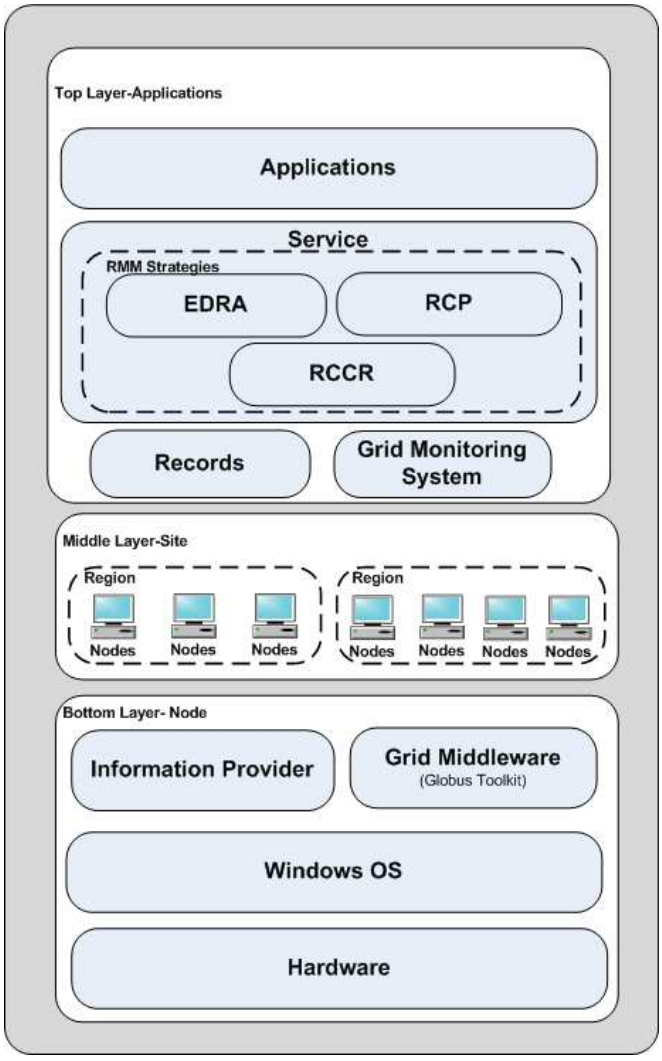

Figure 3.7: System Software Stack

sponsibility is to map resources to jobs in an effective manner. The Resource Broker (RB) is the integral part of the WMS, which optimize the mapping process of resource and job by obtaining information of all possible resources which can satisfy the requirements of the job. The resources are ranked according to certain criteria such as time to access a file, data required for execution of job or some scheduling strategy that can be easily interchangeable by the RB.

- *Top Layer :* The Components in this layer are strategies and applications. This is the single entry point for the user to interact with the grid system. User can easily manage the grid environment with the help of strate-

gies through applications. Strategies namely, Efficient Dynamic Replication (EDRA), Replica Creation and Placement (RCP) and Replica Consistency and Conflict Resolution (RCCR) are incorporated in RMM model. Services take information from the bottom layers to gather records and provide a interface for user to interact with the system.

3.5 Parameters and Access Patterns for RMM Model

A choice of parameters can be varied for simulating various grid scenario, which might affect the results obtained from the simulation. OptorSim provides three configuration files for varying input parameters. They are:

(i) Grid configuration file.

(ii) Job configuration file.

(iii) Parameter configuration file.

The grid configuration file helps in deciding the network topology in OptorSim. It offers range of network topologies from flat, centralized, peer to peer to hierarchical structure. The structure of the overall grid is greatly affected by the network topology used, while creating a grid. During the simulation of the proposed work, the CMS (Compact Muon Solenoid) testbed hierarchical layout has been adopted. In addition to that some more characteristics can be defined in this configuration file such as network connection between grid sites and their bandwidth, number of storage elements and their storage capacity, computing element and their processing capacity.
Job configuration file helps in defining the jobs which are being executed on the CEs and the file stored on the node during simulation. The *jobtable* parameter in job configuration file consist of job name and the file needed for the completion of the job. Moreover, the job is defined as the collection of data files it need to analyse. It also includes reference for each logical file and a scheduling table. The scheduling table contains information about the CE and the jobs it can run. Every job has certain probability of being selected to run on the grid which is declared in *jobselectionprbability* parameter table. In this work, the selection probability of

each job is equal and is defined in job configuration file.
In parameter configuration file various parameters for simulation can be specified. This contains number of jobs for execution, time required for processing a file, limit of waiting queue for each CE, delay between job submission and access pattern used. The jobs are submitted to the Resource Broker which act as a meta scheduler. It schedules the job to the appropriate node containing the requested file. The selection of the CE is done from the pool of nodes which are offering requested file. Access pattern is the key parameter to determine the order in which files can be requested. Various access patterns are used during the simulation. These are:

- *Sequential:* The order for accessing the file is same as stated in job configuration file.
- *Random:* The files are accessed in random fashion from the group of files with uniform probability distribution.
- *Unitary Random Walk:* The file is accessed in the neighbourhood of previously accessed file but the direction of selection is random.
- *Gaussian Random Walk:* The files are selected as in Unitary random walk but the selection uses Gaussian distribution around the previous file request.
- *Zipf :* A Zipf like distribution is an inverse power law distribution [5], which is represented as:

$$p_i = \frac{K}{i^s} \tag{3.1}$$

  where, P_i is the frequency of occurrence of the i^{th} ranked item and K is popularity index of file which is more likely to be accessed and s determines the shape of distribution. Zipf indicates that some files are likely to be accessed more while others have less probability to get accessed.

For sequential access pattern every file assigned to a job will be accessed in the order they are described during job description process. The occurrence of accessing the file for remaining access patterns can be zero or more. Though the number of occurrence of a file depends on the number of file request made in job configuration file. The Gaussian and Zipf distribution for accessing files shows similar behaviour as of real world applications which uses data grids.
In this work, the *optimizer* class of OptorSim is extended for the implementation

of RCP and EDRA strategies. The Resource Broker uses information of each resource on the grid and schedules the job to the best node. The resource broker decision is affected by various factors like the heterogeneity of the node, access cost, availability, network bandwidth etc. In addition, to the scheduling of the job sometimes there is a need to replicate the data file so as to get the better output. Again, *Optimizer* class has been extended through *ReplicatingOptimizer* class. The decision for replication is made based on popularity or availability of file. The file is either replicated on the local storage or it must be fetched from remote node. After creating replicas of a file they are placed at appropriate location. In this work, the placement of the replicas are based on placement cost, access frequency, storage etc., which is implemented by overriding the *assignFilesTosites()* method in *JobConfFileReader* class.

3.6 Functional Architecture of RMM Model

Various terminologies used in this thesis

3.6.1 Common Terminologies used in this Work

- *Master Node:* It facilitates data submission with an interface and helps in scheduling and monitoring of the job. Master node contains the global view of all the regions in the data grid. Master node is shown in Figure 3.11.
- *Head Node:* The head node has the local information of the region which helps in executing and scheduling the request at region level as shown in Figure 3.11. The master node and head node in data grid uses RM component of data grid node. Rest of the data grid nodes use only storage and computing element.
- *Sub Region:* Number of grid nodes collectively form one sub region as shown in Figure 3.11.
- *Region:* Collection of sub regions creates a region as shown in Figure 3.11.
- Load gauge: It is the length of the queue at master node and head node where jobs are waiting to get executed.
- *Availability:* This parameter is to check whether a node is available at the time of execution of request or not.
- *Access Frequency:* It is the measure of number of file requests made by users

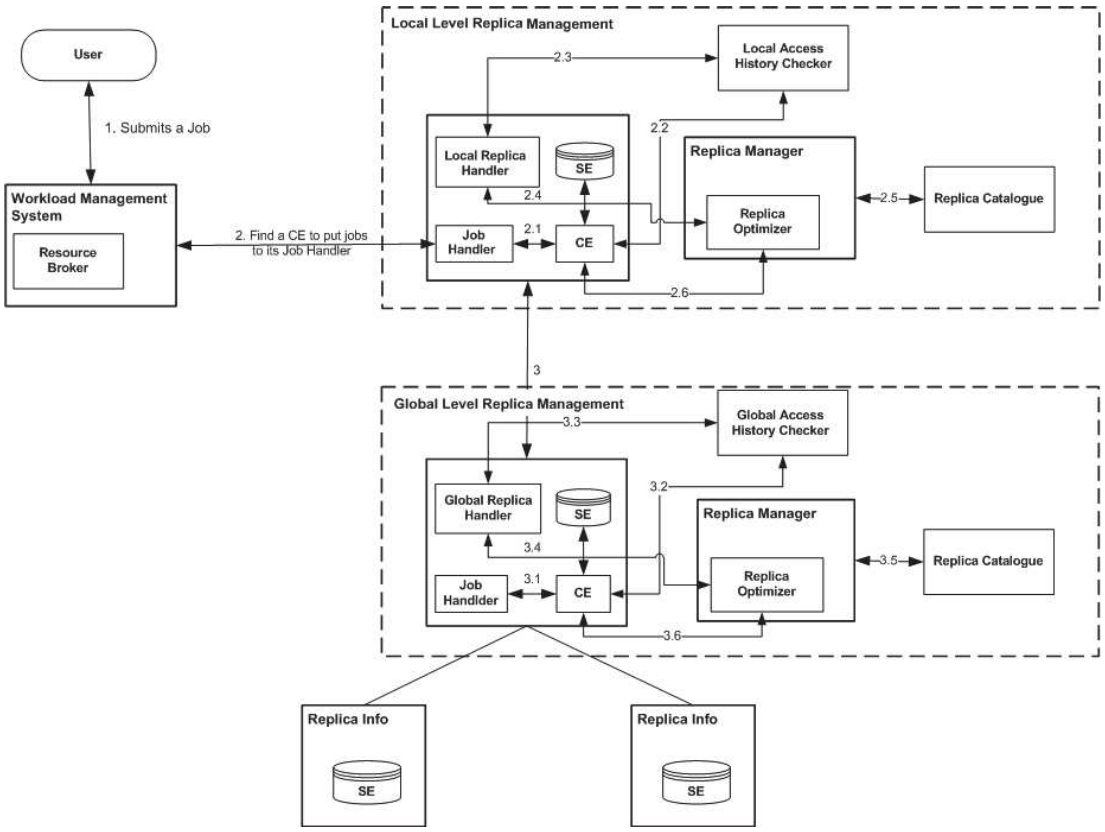

Figure 3.8: Functional View of RMM Model

running on the nodes in the region [144]. Access frequency helps in deciding the popularity of the file based on the access history. Higher access frequency of a file on a node means that the file is very popular.

- *Computing Capacity:* This helps in scheduling the jobs to the node having maximum computing power, which results in faster execution of job.

3.6.2 Functional View of RMM Model

While initializing the simulator, the master files are distributed randomly on the various storage elements. The Replica Catalogue (RC) registers the distributed files. The jobs are submitted by user and is picked by the Resource Broker as shown in Figure 3.8. The Resource Broker is responsible for scheduling the jobs to the node which contains desired file. The files can be accessed locally or remotely by various jobs using its logical filename (LFN). Logical filename is name given to a file for the reference of the client irrespective of its physical location. Physical filename

(PFN) represents specific file with respect to its physical location. The LFN are mapped to its respective PFN by Replica Location Service (RLS). There can exists multiple physical location of a logical file. To acquire the exact physical location of a file, Replica Manager (RM) is consulted by the CE. The RM, in turn, consults the Replica Catalogue (RC) for acquiring the information regarding the physical location of the file. In general, the selection of the best replica of a file relies on the minimum transfer time with some network constraints such as bandwidth, latency etc. The bandwidth in RMM model is fixed at each level during the setup. Bandwidth with in the sub-region is highest. Bandwidth among the sub-regions of a region is slightly less than the inter sub-region. Inter region bandwidth is the lowest. Additionally, the available bandwidth can also be determined by knowing the minimum available bandwidth between the interconnection of two nodes. The request is forwarded to the appropriate region from higher level to lower level, where the nodes are identified at local level, which contain file for execution.
Due to dynamic nature of the grid, sometime the files or nodes are busy or not available. To overcome this problem new replication strategy is introduced which will optimize the data grid system. The optimization of replication strategy is supported by replica manager through *optimizer* and is performed at two phases. In first phase, the optimizer chooses the CE on the node where the job should execute. To reach the desired CE a proper scheduling strategy is required. Various scheduling strategies that are offered by OptorSim and is used for comparative analysis in this work are:

- *Random Scheduling:* In Random scheduling selection of the CE for the execution of the job is done from the pool of available CEs which fulfils the request requirement of the job. Every CE has same probability to get selected for the execution of the job.
- *Shortest Queue Scheduling:* In this policy, the node containing minimum number of jobs in waiting queue is selected for running the job. In shortest queue scheduling every CE get an equal number of jobs.
- *Job Access Cost scheduling:* Job access cost is an estimated time and is calculated based on current network configuration and time to attain all the required file for complete execution of job. The node having the minimum job access cost is selected for the submission of the job.
- *Queue Access Cost Scheduling:* The queue access cost is calculated as sum

of the access costs for all the jobs in the queue of a specific node. Minimum access cost is the deriving factor for the selection of a node containing the CE.

In this work, a new scheduling policy the scheduling policy, namely, *Hybrid Scheduling (HS)* policy that take advantage of both shortest queue scheduling and job access cost scheduling. It is a two level scheduling policy. At first level, selection of region is done that has the minimum placement cost and appropriate storage capacity to store the replicas. In the second level, each node of the region is evaluated and the replica is placed on the node with minimum job access cost. The job access cost is based on the access frequency of the file accessed by the job. Additionally, the replica selection policy is optimized dynamically during the run time of the job. Moreover, the new replicas can be created which are prompted by replica placement algorithm. In this thesis, an optimization of replica selection and scheduling is done by proposing and implementing an agent based strategy namely, EDRA. The automatic creation and placement optimization is achieved by proposing and implementing an agent based novel strategy namely, Replica Creation and Placement (RCP). Additionally, the consistency mechanism is also extended on the top of OptorSim based on logical clock of the transactions and version of the file . The Replica Manager handles the consistency mechanism, it also keep track of conflicts among the data files in multi-master environment. If any update is required on the file, it must be propagated using proposed hybrid strategy namely, Replica Consistency and Conflict Resolution (RCCR). Once the file is updated, it must inform the replica manager at head node and master node, so that updates are propagated to the entire system and in case any conflicts are encountered that must be taken care by master node. The consistency of a file and its replicas are maintained so as to ensure the efficiency and effectiveness of the entire system.

3.7 Simulation Testbed for Functional Architecture of RMM model

For the simulation of RMM Model, first step is to choose the simulation tool. There exist varitey of tool for the simulation of the grid such as OptorSim, Sim-

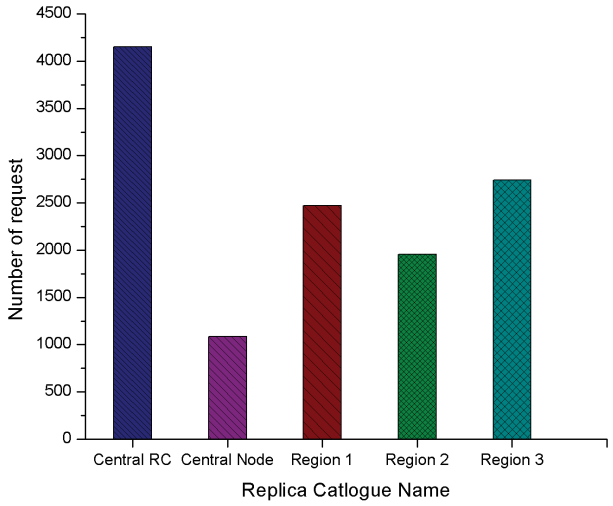

Figure 3.9: Comparison of Centralized and Hierarchical Model

Grid, DataGrid, GridSimetc. Optorsim has been chosen as it provide all the functionalities that can be used for replica management system. OptorSim offers two simulation testbeds namely, EU DataGrid and EDG testbed [58] with two different topologies. EU DataGrid follows hierarchical topologye whereas EDG testbed follows centralized topology. The evaluation for selecting the topology has been done on the basis of two parameters i.e. workload and availability. For initial simulation 250 users are considered. Initially, 10 random users start submitting the job after every 5 minutes, afterwards the arrival of users are controlled using Random Distributed. 40 to 150 jobs can be requested by each user at same amount of time. There are 100 types of data intensive jobs. For execution, each job require 10-15 files each. Total 2000 files are used during simulation. Average size for each file is 1 GB. Files are accessed using Zipf access pattern.

3.7.1 Workload

During evaluation, the comparison of centralized and hierarchical model has been done. Figure 3.9 depicts that in hierarchical topology number of request are distributed at each region where as in centralized model the requests are handled by central entity. The experiment results shows that load in the hierarchical model

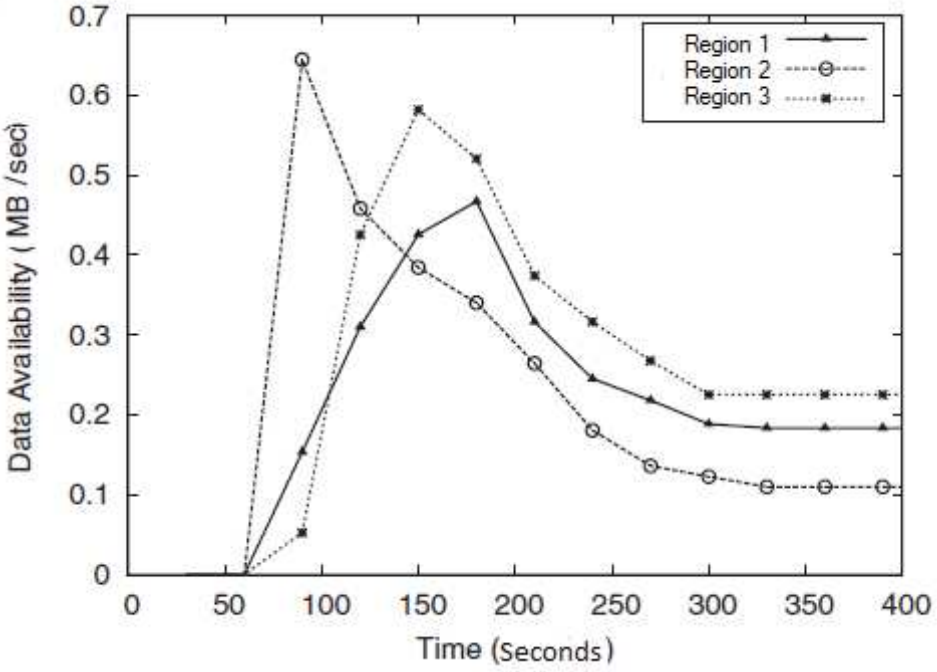

Figure 3.10: Average Availability in Time for Three Regions

decreases as compared to centralized model. It is evident that there is significant reduction in job processing in hierarchical model. The results clearly demonstrates that the selection of hierarchical model is more appropriate than centralized model in grid environment, where more jobs are likely to occur. To justify the selection of hierarchical model one more parameter has been considered and is discussed below.

3.7.2 Availability

The availability of the files can be obtained as amount of data required by the job upon the time needed by the job to get that data.

$$avail = \frac{Req_{file}}{Time} \tag{3.2}$$

The availability of the data depends on the network condition and replication frequency of the file as they changes over a period of time. Figure 3.10 shows the availability of different regions vary over a period of time. Availability depends

Use the relevant command to insert your figure file.

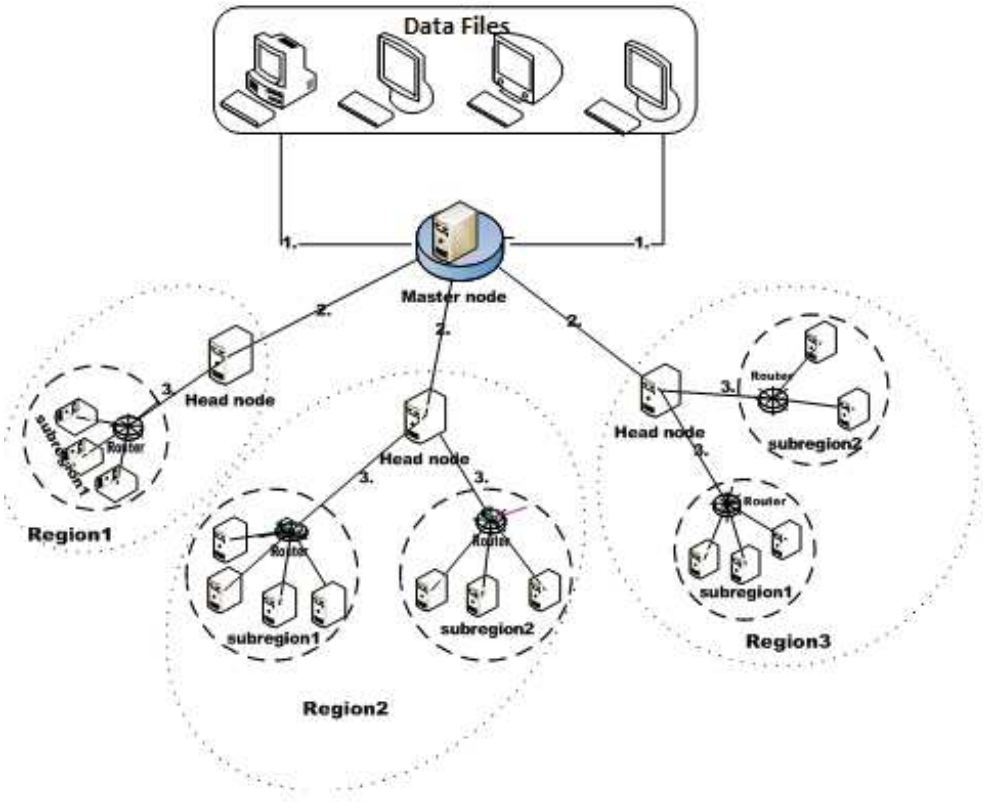

Figure 3.11: Network Topology of Proposed System

on network status so centralized model is not considered due to high workload at central node. In hierarchical model nodes are distributed in three regions each contain certain replicas of file and hence increases its availability. The results demonstrate the consideration of hierarchical model for replica management in distributed environment.
These two parameters helps in deciding that topology used in EU DataGrid will provide more efficient results.

EU DataGrid has been constructed for LHC experiment, where huge volume of data is produced and are read only. This model maintains a single source for data but the data is distributed across the world for collaborations. For example, the MONARC (Models of Networked Analysis at Regional Centres) group within CERN has proposed a tiered infrastructure model for data distribution. This model is presented in Figure 3.11 and specifies requirements for transfer of

data from CERN to various groups of physicists around the world. The first tier is the compute and storage farm at CERN which stores the data generated from the detector. This data is then distributed to Regional Centers (RCs) located around the world. From the RCs, the data is then passed downstream to the national and institutional centers and finally onto the physicists working on the data. The massive amounts of data generated in these experiments motivate the need for a robust data distribution mechanism. Also, researchers at participating institutions may be interested only in subsets of the entire dataset that may be identified by querying using metadata. One advantage of this model is that maintaining consistency is much simpler as there is only one source for the data.
The choice for selecting the best place in such cases is critical decision. Placement of data at different region should be such that it can be accessed in a minimum time. In this work, RCP strategy is responsible for the placing the replica at appropriate place. After placing the replica, the selection of best replica is taken care by proposed EDRA strategy which is very effective on EU DataGrid environment.

3.8 Use Cases for Different Strategies in RMM Model

The Use Cases for proposed strategies that are used in RMM model is described in this section. The working of the strategies are given in forthcoming chapters.
Use Case 1: Replication of a file to another grid location (RCP)

- t =t_0: The master node specifies a unique logical file name for the file (LFN) to be replicated at the destination SE,
- t =t_1: The RMS queries the metadata service to obtain a list of all replicas LFNs of the file,
- t =t_1: The RMS gets a list of all extant replicas for each LFN in the dataset from the RLS,
- t =t_2: The RMS calculates the total space needed for the file and reserves sufficient space on the destination SE,
- t =t_3: The RMS determines the best replica to use on the basis of access frequency. This is calculated using the replica optimisation component of

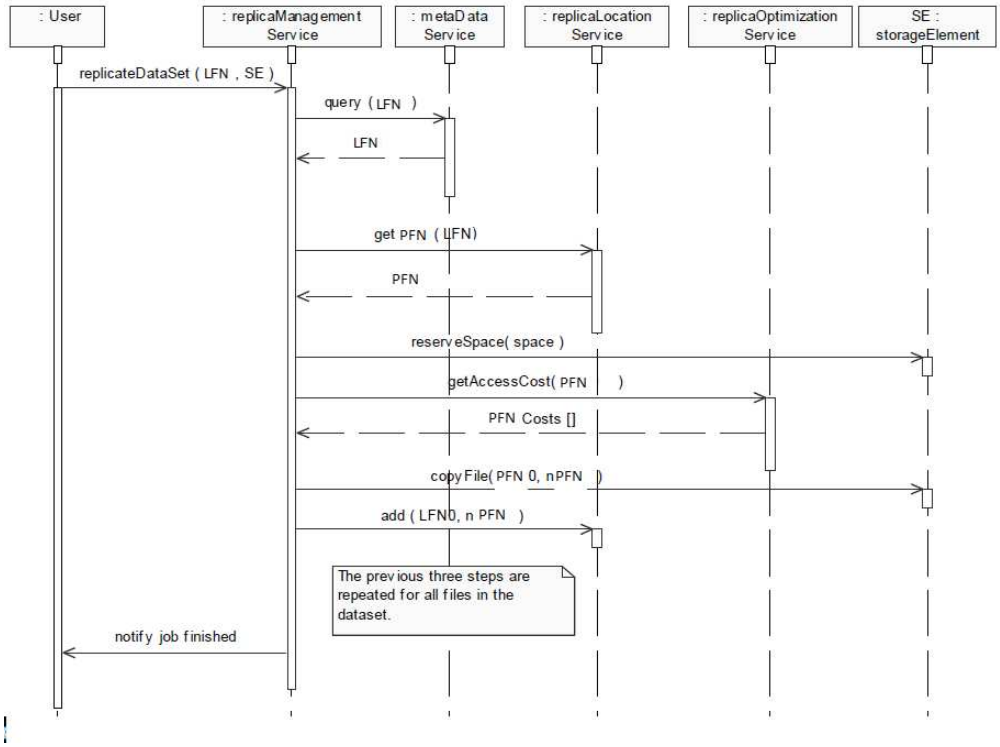

Figure 3.12: Replicate a Set of Data to Another Grid Location

the RMS which provides average access frequency of the file by given SE. The file will not necessarily be located on the same SE, or at the same region,

- t =t_4: The RMS copies all files in the dataset to the specified destination SE,
- t =t_5: The Replica Location Service (RLS) is updated with information about the new replicas,
- t =t_5: The user is notified of termination of job.

Figure 4.6 shows a sequence diagram for this use case. Errors may occur in the case where the destination SE does not have enough space for the dataset to be replicated. In such a case, the SE, in conjunction with the RMS, must decide how to proceed. Possible options include:

- t =t_6: RMS returns an error stating that the job cannot be completed due to lack of resources on the specified destination SE
- t =t_6: SE is identified that satisfies the job requirements.

Use case 2: Selection of Replica (EDRA)

- t = t_0: User submits a job to execute read or write request,
- t = t_1: The RB determines the best CE from the list of candidate CEs with respect to physical locations, network traffic and access frequency of the data,
- t = t_2: The RB reserves space on an SE for the jobs output data,
- t = t_3: The RB determines the best CE based upon the network costs returned by the RMS for each candidate CE and schedules the job on the best CE,
- t = t_4: The job queries the application metadata managed by the RMS to extract a list of LFNs that match the requirements,
- t = t_5: The job calls the RMS interface to determine the corresponding SFNs to access,
- t = t_6: The job reads and processes the data,
- t = t_7: The job writes the output dataset to the output SE,
- t = t_8: The job registers the output dataset with the RMS,
- t = t_9: The job updates the metadata service for the output dataset

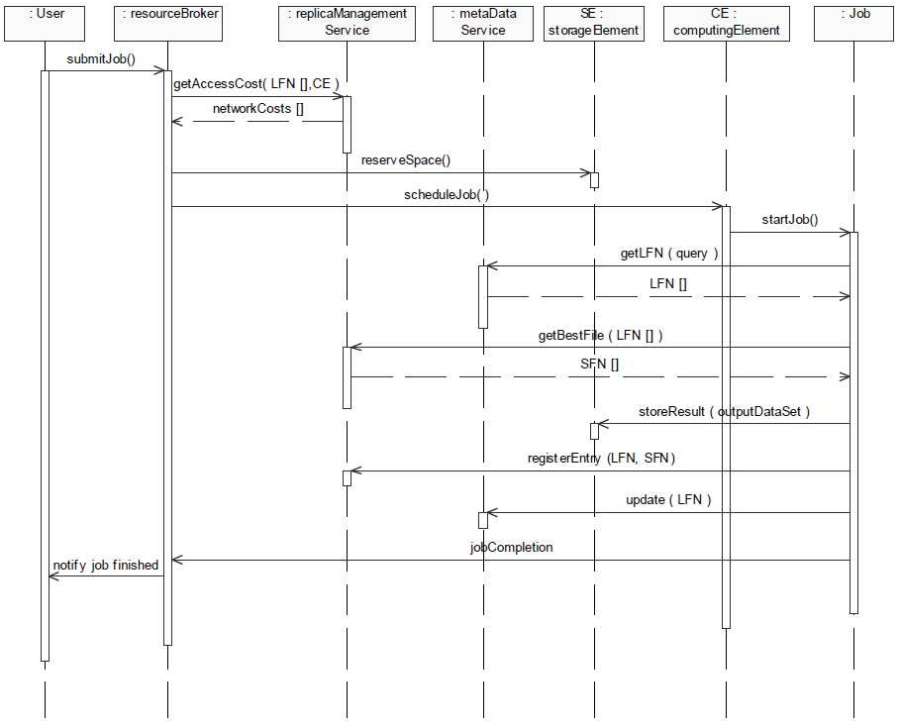

Figure 3.13: Sequence Diagram for Selecting "Best Replica"

Use case 3: Consistency Maintenance of the file (RCCR)

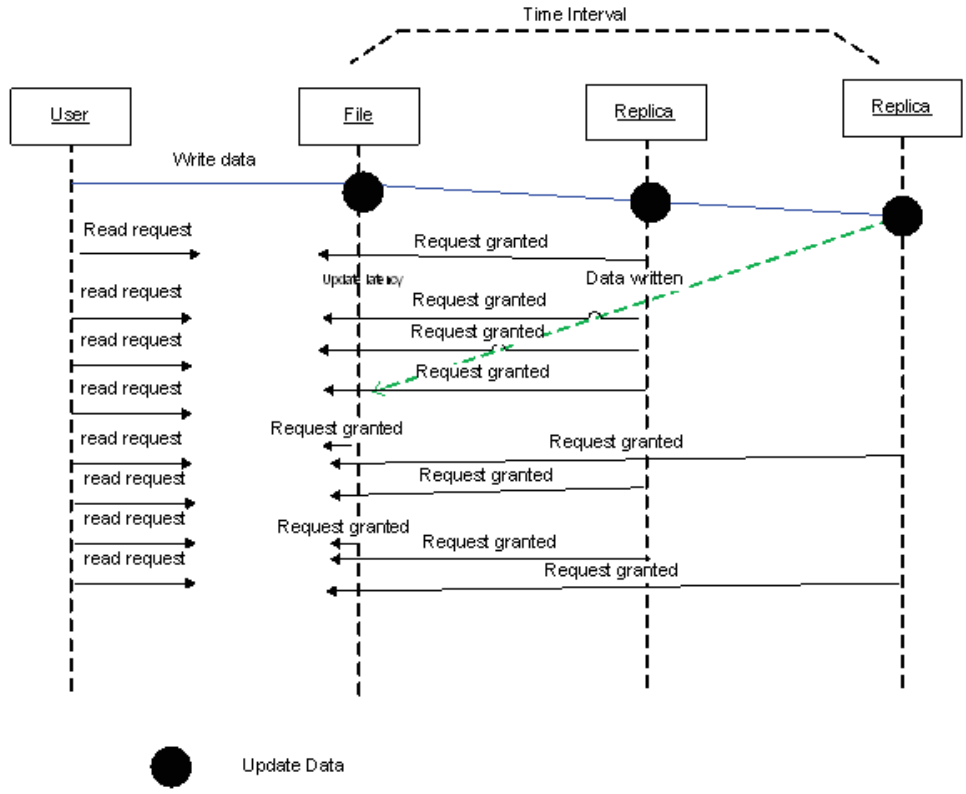

Figure 3.14: Sequence Diagram of Consistency Maintenance

- t = t_0: A write request is submitted to be executed,
- t = t_1: The RB determines the CE from the list of candidate CEs with respect to request made by the user,
- t = t_2: The RB schedules the job to CE,
- t = t_3: Job is executed,
- t = t_4: The RB determines the best CE to be elected as "leader node" based upon the version vector and vector clock in a region ,
- t = t_5: The job queries the application metadata managed by the RMS of head node to extract the list of LFNs that matches the requirements,
- t = t_6: The job calls the RMS interface to determine the corresponding SFNs to access within the region,
- t = t_7: The job propagates the update on each SE containing replica within the region,

- t = t_8: The job writes the updated dataset to the output SEs of the region,
- t = t_9: The job registers the updates with the RMS of region,
- t = t_{10}: The job updates the metadata service for the updated dataset in the region
- t = t_{11}: The step t_3 to t_{10}: are repeated for inter-region updates.

3.9 Conclusion

The aim of this chapter is to demonstrate architecture and functionality of RMM model which helps in optimizing the replica management issues in grid environment. An agent based replica management model namely, RMM has been proposed to address the problems of replica management in data grid. This model offers an automatic creation, placement, selection of replicas. RMM model also support consistency maintenance mechanism for replicas as per the requirement of data grid system.

The domain analysis for RMM model with its main component from user prospective is also provided. Working of the RMM model in the hierarchical structure is discussed. Moreover, use cases for strategies namely i.e. RCP, EDRA and RCCP are also presented. These strategies assist RMM model while replica management. A system software stack is presented in RMM model which defines the basic functionality of various components and their interaction at different grid layers.

Various scheduling strategies and input parameters are also elaborated. A simulation testbed is provided to check the feasibility of RMM model in terms of workload and availability. The results depicts that the hierarchical topology provide more flexibility and scalability by distributing the jobs at various regions. The availability helps in improving the efficiency of the RMM Model.

CHAPTER 4

Agent Based Replica Creation and Placement in Data Grids

Data grid is emerging as a key technology to support data intensive applications. Grids rely on data replication to enhance the performance of the system. Maintaining replicas is the primary concern of replica management in data grid environments. This chapter addresses replica management issues such as availability of replica, creation of replica and their placement at appropriate location, which in turn enhances the availability and accessibility of replicated data in a grid environment. An agent based strategy namely, Replica Creation and Placement (RCP) is proposed and implemented for optimized RMM model. A popularity driven policy is adapted by RCP, which dynamically creates an optimal number of replicas and places them on nodes with minimum placement cost. Access frequency of a file is the key parameter for computing number of replicas, placement cost and replication cost. The proposed strategy is implemented on Optorsim simulator and the results are compared with existing approaches to judge its effectiveness in data grids. The simulation results shows significant benefits of proposed RCP strategy in terms of a job execution time, storage utilization, effective network utilization and number of replicas created.

4.1 Introduction

Data grids aim to combine with high-performance networking, high-end computing technologies and wide-area storage management techniques. It allows grid users to discover distributed datasets in collaboration with these high end devices, technologies or intelligent replication techniques for faster access of desired files. Efficient access and availability of files are the crucial requirement of data intensive jobs. Replication of file at distributed locations is an efficient method for achieving high network performance in data grids. In addition to achieving efficient access

and availability, data replication can also be used to improve data locality and increase scalability and efficiency of the system. Additionally, replication helps in balancing the workload across multiple locations across the grid. The frequency of the replicating a data is limited by available storage capacity of the node, available bandwidth between the nodes and the access location of the file. To maximize the gains of replication, strategic placement of replica is essential. The replica placement helps in selecting the appropriate node for placing the replica. Locating a node for placement depends on several parameters such as high bandwidth, less transfer time, network latency, size of file etc.
In this chapter, an agent based RCP strategy is discussed which provides an efficient replica management for the data grid. RCP dynamically creates and places the replica on optimal nodes adaptively. The strategy systematically organizes the grid nodes into discrete regions. Grid nodes are selected for optimal placement of replicas. RCP allocate file to the request in accordance with their access frequency. The main contributions of the current study are as follows:

(i) RCP presents replica creation strategy on the basis of popularity of the file, which is further dependent on average access frequency of the file. To save storage space on a grid node, numbers of replica are restricted by considering replication cost and access frequency of file.

(ii) Replica placement issue is also addressed in RCP, which considers parameters such as; replication cost, placement cost, latency, processing capacity of the node etc., which helps in enhancing the overall system performance.

(iii) Investigating various trade-off in terms of mean job execution time, percentage of storage usage, effective network utilization and number of replicas created.

4.2 Replica Creation and Placement (RCP) Strategy

To enhance the utility of the RMM model, RCP strategy has been implemented in the *Optimizer* class of OptorSim. The RCP helps in optimizing the replica management process in simulated data grid environment. Various components

that are used by the RCP strategy are discussed in next section.

4.2.1 Components Used in RCP Strategy

- Master Node: Global view of the entire system is kept by the master node with the help of an agent and stores it in the global database (GDB).
- Master Node Analyzer (MA): It analyses the request based on the access frequency of the file and maps them to different regions.
- Request Scheduler (RS): It generates a schedule for request to be executed.
- Request Allocator (RA): Helps in allocating region to the request based on the access frequency of the file.
- Global Information Collector (GIC): It receives data from information probe (IP) of every region and stores it in the global database (GDB).
- Global Database (GDB): It contains the physical and logical mapping of the files in different regions for future request allocation.
- Access Information Database (AID): It stores information associated with each request
- Head Node: An agent on head node facilitates collection and management of information of nodes on Information Probe (IP).
- Information Probe (IP): It monitors resource utilization like number and names of files, storage space, the processing capacity of node in a region and observed data is stored at the local database (LDB).
- Local Database (LDB): It keeps updated status of all nodes in a region.
- Local Node Controller (LNC): An agent on the head node which analysis the request scheduled by a master node and map it to the resource.
- Agent: Software entity which manages and manipulates information intelligently.

RCP is a centralized approach, the decision to place replicas are handled by master node. The front end server stores the request requirements on master node for analysing, scheduling, and allocation of file resources. Master node analyser (MA), an agent on master node analyses the request depending upon access frequency of files and then map it to an appropriate region. A counter, namely, *enu* is incremented by one whenever a request is made for replica, as it keeps track of the access frequency of the file. An agent called request allocator (RA) coordi-

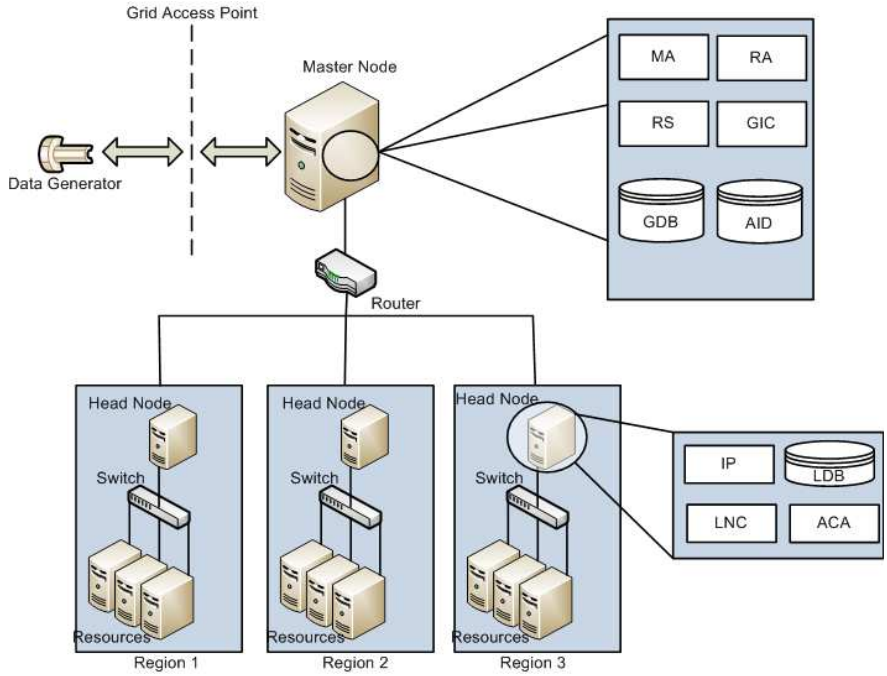

Figure 4.1: Various Components of RCP Strategy

nates with global database (GDB) to map the request with an appropriate region. Requests are then scheduled to head node of the allocated region based on the information collected by request allocator (RA). Once the request is deployed on a particular grid node by the head node, its resource utilization is monitored by IP after every fixed interval and observed information is stored in LDB.

In case, a resource is not available for the request on scheduled resource due to dynamic nature of data grids. The request is migrated to one of the neighbouring node containing replica. After rescheduling the request available number of replica are checked in the region. If number of replicas on the region is less than the required replicas, new replicas are created. Once the decision to create replica has been made, determining place for data replica is the prime concern. Optimal placement of data is achieved by acquiring the current status of the system. The replica placement algorithm selects the node with lower replication cost. On occasion, it is not feasible to replicate a file on node due to insufficient available free resources, then old file are replaced using Least Recently Used (LRU) algorithm [27]. In LRU, files which are not being accessed for quite some time will be replaced to accommodate the newly generated replica.

4.3 System Model for RCP Strategy

While developing a replica management strategy, significant decisions are to be made such as determination of the number of replicas required for effective usage of system and identification of a node where the replica will be placed in order to accomplish the aims of data replication. To address the above issues a strategy namely, Replica Creation and Placement (RCP), has been proposed, which comprises of following steps:

(i) Data grid regions are created where replicas should be placed.

(ii) Creation of replicas: This step is achieved by calculating:

- Number of replicas needed for the entire data grid.
- Establish number of replicas required for the region in data grid.

(iii) Determine a node in data grid for placing the replica. This is attained by calculating the placement cost of replica and processing speed of node.

4.3.1 System Model

The data grid D considered here, is a set comprised of discrete nodes $D = \{n_1, n_2, \ldots, n_l\}$ and data user $U = \{u_1, u_2, \ldots, u_m\}$ connected by some communication link. Initially, data files $f = \{f_1, f_2, \ldots, f_k\}$ can be produced at any node and replica of the file $R = \{r_1, r_2, \ldots, r_n\}$ can also be stored at any node $n_i \in D$. $f_{p_i}, where\{i = 1, 2, \ldots, n\}$, is a primary copy of each file. Each node has the following characteristics: $n_i \in D$ $\langle P_{cap}, RC_i, S_i \rangle$ where, P_{cap} is the processing capacity of the node, RC_i is name of replica catalogue to which node n_i is connected and S_i is total storage capacity of the node. Number of grid nodes in a group form one region as shown in Figure 3.11. The data nodes are clustered together based on the physical proximity of nodes to form a region. In all there are N regions whose nodes are partitioned into several non-overlapping regions, $Reg = \{Reg_1, Reg_2, \ldots, Reg_N\}$ such that, $Reg = \{Reg_1 \cap Reg_2 \cap \ldots Reg_N = \emptyset\}$. The nodes are connected to each other directly or indirectly through some communication link. Nodes within a region have low latency and maximum bandwidth. One node in a region is called head node comprising of replica catalogue $RC = \{RC_1, RC_2, \ldots, RC_N\}$, which provides the mapping of file and its physical location locally. The local replica catalogue is connected to global level replica

catalogue such that $GRC = \{RC_1 \cup RC_2 \cup \ldots \cup RC_N\}$, which offers global view of data grid at master node.

4.3.2 Replica Creation

Replica improves the reliability and availability of the grid system. The proposed RCP strategy, divides the replica creation process into two sub parts:

(i) Determining the number of replicas required for the executing a job in entire data grid system.

(ii) Calculating number of replicas in a region.

However, finding the optimal number of replicas (Num_{f_i}) of a file (f_i) for a set of regions $\{Reg_1, Reg_2, \ldots, Reg_N \in D\}$ is a distinguished problem for discovering Num_{f_i} combinations which is a complex combinatorial optimization problem. In this chapter, number of replicas of a file depends on the access frequency which can be obtained from the log history (H) stored at their respective replica catalogue ($RC_i, i = 1, 2, \ldots, N$). The average access frequency $Freq(f_i)$ of a file (f_i) in data grid D is calculated as [**?**]:

$$Freq(f_i) = \frac{\sum_{n=0}^{L_\tau}(\alpha f_{(\tau)} \times 2^{\Delta(\tau)})}{L_\tau} \tag{4.1}$$

where, L_τ is number of the passed time interval, α_τ is the number of calls for replica r_i in period τ and $2^{\Delta}(\tau)$ is weight based on access frequency of a replica r_i during the period τ. In fact, $\alpha f_{(\tau)}$ has tight relationship to the value in log history (H) stored at their respective replica catalogue (RC_i). The average access frequency $Freq(f)$ of all existing files that are requested by the user is calculated as [**?**]:

$$Freq(f) = \frac{\sum_{i=1}^{n}(Freq(f_i))}{F_{num}} \tag{4.2}$$

where, F_{num} is sum of all files that have been requested. The number of replicas (Num_{f_i}) required for a file (f_i) in data grid D is calculated as [**?**]

$$Num_{f_i} = \left\lceil \frac{Freq(f_i)}{Freq(f)} \right\rceil \tag{4.3}$$

After calculating the number of replicas (Num_{f_i}) required for the data grid (D) in order to make the system more sustainable, the number of replicas (Num_{Reg_i}) needed within a region (Reg_i) is determined. This can be calculated as:

$$Num_{Reg_i} = \left\lceil \frac{Num(f_i)}{N} \times \frac{Freq_{Reg_i}(f_i)}{Freq_{Reg_i}(f)} \right\rceil \tag{4.4}$$

where, $Freq_{Reg_i}(f_i)$ is average access frequency of a file (f_i) in region (Reg_i), $Freq_{Reg_i}(f)$ is average access frequency of all files in region (Reg_i), N is total number of regions. Replica creation increases system performance in terms of reliability and availability. In contrary, creating and maintaining large amount of replicas in the system reduces the effectiveness and efficiency of the system many folds. Restricting the number of in a system helps in reducing the overhead of maintaining large amount of replicas in a system. For optimizing the replication process further, an appropriate location for replica should also be determined for enhancing the overall system performance. In the next section, the replica placement algorithm has been discussed.

4.3.3 Replica Placement

Replicas should be placed at appropriate location so as to reduce the access cost of replica effectively. The precondition for placing the replica is to calculate placement cost (P_{cost_i}) of replica (r_i), and is calculated as:

$$P_{cost_i} = \frac{\alpha_{f_i} \times S_f}{P_{cap_i}} \times Rep_{cost_i} \tag{4.5}$$

where, α_{fi} is number of times the request for replica ($r_i \in Reg_i$) is made by user, S_f is the size of file, $P_{cap_i}$is the processing capacity of node (n_i) containing replica ($r_i \in f_i$) and Rep_{cost_i} is the replication cost. Further, replication cost is governed mainly by two factors i.e. communication cost and the accessing cost of storage media. The communication cost (C_{cost}) is determined by:

$$C_{cost} = \sum_{i=1}^{\delta} \sum_{j=1}^{k} C_{ij} + P_{network} \tag{4.6}$$

where, C_{ij} is communication cost between two node n_i to n_j such that $n_i \neq n_j$ but are connected to each other via some communication link in the network, , δ is number of nodes holding the replica of file and k is number of nodes connecting link between n_i and n_j and $P_{network}$ is the propagation delay of network. The access cost of storage media $C_{storage}$ is calculated as:

$$C_{storage} = \frac{S_f}{D_{Rate}} + L_{storage} \tag{4.7}$$

where, D_{Rate} is the transfer rate of storage media and $L_{storage}$ is the storage media latency. Storage media latency occurs due to reading and writing to and from different blocks of memory. Consequently, the comprehensive replication cost (Rep_{cost_i}) of file (f_i) can be formulated as storage access cost ($C_{storage}$) of a file and communication cost (C_{cost}) during the transfer of file from one node to another. Hence, the replication cost (Rep_{cost_i}) is determined as:

$$Rep_{cost_i} = C_{cost} + C_{storage} \tag{4.8}$$

Placement of replica on a node is done if it has minimum placement cost and sufficient storage capacity is available. After calculating placement cost, the storage capacity of the node is also investigated. The available storage capacity $Aval_{str}$ on a node is calculated as:

$$Aval_{str} = S_{total} - S_{usage} \tag{4.9}$$

where S_{total} is the total storage capacity of a node and S_{usage} is storage space consumed by the node. If available storage space ($Aval_{str}$) of the node is less than the size of file (S_f), then replacement algorithm namely, Least Recently Used (LRU) algorithm is used [**?**]. LRU strategy compares size of file to be replicated with the available space on the selected node, if sufficient space is not available then oldest file is deleted. However, if the deleted file is less in size than the required space, the second oldest file is deleted and so on.

4.3.4 Algorithms for RCP Strategy

The pseudo code for RCP strategy is elaborated in Algorithm 4.1, is composed of two phases namely; replica creation and replica placement. As discussed earlier RCP strategy at first level is applied to schedule the request for a region and at second level the desired file is allocated to the request made by the user. An effi-

Algorithm 4.1 Pseudocode of RCP strategy

```
1: procedure RCP
2:     initialization
3:     while not Termination Condition  do
4:         Schedule Request
5:         Replica Creation and Placement
6:      end while
7:     return(New Replicas)
8: end procedure
```

cient scheduling of a request in RCP strategy is determined by selecting the most appropriate region and node respectively. An appropriate region is the one which is close to the user and holds the most of the requested files which considerably reduces the transfer time. To resolute which region contains most of requested file a record is being kept by a counter *(enu)* for read request as shown in Algorithm 4.2. The given pseudo code can be easily modified for the write request to keep

Algorithm 4.2 Pseudocode for Scheduling Request to the node

```
1: procedure Initialization (Request for file)
2: [enu] ← 0
3: if read request received then
4:     [enu] ← [enu] + 1
5: end if
6: if r_i exists on scheduled n_i of Reg_i  then
7:     process request;
8: else
9:     reschedule request to nearest replica
10:     replica creation and placement
11: end if
```

track of access frequency of a file. In the replica creation and placement phase as shown in Algorithm 4.3, the new replicas are produced and every region provides

some nodes for the replica placement. As discussed in previous section a replica is created based on the access frequency of a file and the appropriate location is chosen on the basis of placement cost. N_{reg} represents number of regions in data grid, Num_{Reg_i} is total number of replicas in a region and n_{ij} represent i^{th} node and j^{th} region. Placement cost refres to the transfer cost of a file from one location to another. Low transfer cost of a file means that the node on which replica is to be created is in the vicinity of the primary file. Low cost uses less resources and hence improve the system performance. The *GridFTP()* function, copies the contents of primary file to the destined node. Afterwards, the location of newly placed replicas is updated by using *updateLocation()* function. The complexity of the replica creation and placement algorithm is $n_{ij}Nlog(n_{ij}N)$ where n_{ij} is number of nodes processed for placing replica and N is total number of regions.

4.4 Evaluation Parameters and Experimental Results

The RCP strategy has been evaluated using OptorSim simulator [58]. For RCP, European Data Grid EU testbed architecture has been considered [57]. There are total 52 nodes in experimental setup out of which two are considered as master nodes, where the initial requests are being made. These two nodes support each other by sharing workload. Additionally in case one master node crashes another will take the responsibility. Remaining nodes are divided into 7 regions where nodes are distributed randomly. Requests are processed on nodes having storage and computing elements. The storage capacity of master nodes are 10PB each and storage capacity of all other nodes are 100 GB. The size of the file varies from 10 MB-1GB. Maximum number of requests generated and executed by each node is 100, which require file ranging from 1-10 in number. The router helps in routing the request to the appropriate location in the grid environment. Network latency varies between 10ms-20ms whereas, storage media latency of the nodes are 2ms-5ms. To replicate real world grid environment, background traffic is introduced which generate random traffic patterns in simulated environment. Various simulation parameters are used for execution of requests are shown in Table 4.1. The grid topology used for proposed algorithm is specified in grid configuration file of Optorsim. Various parameters, such as access pattern of jobs, number of job

Algorithm 4.3 Pseudocode for Replica Creation and Placement

1: **procedure** *replica creation and placement*
2: **input:** node id (n_{ij}), primary file(f_{pi}) and region (N)
3: Evaluate :

$$Num_{Reg_i} = \left\lceil \frac{Num(f_i)}{N} \times \frac{Freq_{Reg_i}(f_i)}{Freq_{Reg_i}(f)} \right\rceil \forall Reg_i \in Reg$$

4: **if** Num_{Reg_i} = Sum of required replicas **then**
5: do nothing
6: **else**
7: **for** $N_{Reg} = 1$ to N **do**
8: **for** $n_{ij} = 1$ to Num_{Reg_i} **do**
9: Evaluate:

$$P_{cost_i} = \frac{\alpha_{f_i} \times S_f}{P_{cap_i}} \times Rep_{cost_i}$$

10: Arrange list in ascending order of placement cost
11: **if** $Avail_{str} \geq S_f$ **then**
12: **for** each selected node **do**
13: $n_{ij} \leftarrow GridFTP(f_{pi})$
14: $GRC_{ij} \leftarrow updateLocation(f_{pi}, n_{ij})$
15: $RC_{ij} \leftarrow updateLocation(f_{pi}, n_{ij})$
16: **end for**
17: **else**
18: **if** $Avail_{str} < S_f$ **then**
19: delete another replica using LRU
20: **end if**
21: **end if**
22: **end for**
23: **end for**
24: **end if**
25: **end procedure**

Table 4.1: Parameters Used in Simulation

Parameter	Value
File size	10 MB - 1GB
Number of files	1000
Number of jobs	100
Network latency	10ms-20ms
Storage media latency	2ms-5ms
Transfer rate	50 Mb/s
Storage capacity	10PB

Table 4.2: Performance Metrics Used in Simulation

Performance Metrics	Description
Mean Execution Time	$\frac{Totalexecutiontime+TotalWaitingtime}{Numberofcompletedrequests}$
Effective Network Usage	Estimates the efficiency of network resources
Number of Replicas	Specifies number of replicas produced using RCP

etc. are set in parameter configuration file. The background traffic is introduced generated using bandwidth configuration file. Various performance metrics for the evaluation are presented in Table 4.2.

The efficiency of RCP strategy is evaluated by considering various scheduling and replication strategies. Five scheduling strategies,namely, Random, Shortest Queue, Access Cost, Queue Access Cost and HS have been considered for evaluation. Detailed description of these scheduling policies were discussed in Chapter 3. Moreover six replication strategies have also been considered for evaluation. These replication strategies are:

- *Least Recently Used (LRU):* In this strategy, replications always take place at the node where the execution of job is done. If the available space for the new replica is not sufficient, the oldest file is removed from the SE.
- *Least Frequently Used (LFU):* In this strategy, replications always take place at the node where the execution of job is done. If the available space for the new replica is not sufficient, the file with minimum access frequency is removed from the SE.
- *Bandwidth Hierarchy Replication (BHR):* In this strategy, the files having the high probability to be requested in near future. The replicas are stored

within the region having high bandwidth.

- *Modified BHR:* In Modified BHR, replication of file is dependent on the access frequency of the file.
- *3-Level Hierarchical (3LHA):* This strategy considers hierarchical structure with three levels. Bandwidth of the file plays a vital role while selection and deletion.
- *Replica Creation and Placement (RCP):* In RCP strategy the replicas are placed at appropriate nodes with highest access frequency of the file. This strategy reduces the average access latency of the file by selecting the best node for placing the replica considering the access frequency, storage and transfer time.

Next section will discusses the experimental results for RCP strategy with various parameters such as Average Execution Time, Effective Network Usage and Number of Replications.

4.4.1 Average Execution Time

The average execution time is calculated as ratio of time taken for complete execution of a request, time spent by the request in waiting queue and amount of requests. For evaluating the mean job execution time three scenarios are considered:

4.4.1.1 Mean Job Execution Time using Various Replication and Scheduling Strategies

Figure 4.2 represents that No replication has performed worst. The results predicts that LRU and LFU strategies have almost the same execution time. Both strategies performed better than No Replication but there execution time is very high as compared to other strategies. BHR algorithm performed better than LRU and LFU. By avoiding the network congestion in data grid BHR is able to reduce the data access time. The performance of 3LHA is further improved as it considers the variation in inter and intra region communication. RCP performed best in the scenario as it considers the access frequency, waiting time, replication cost and placement cost for creating and placing the replicas on a node.
In Random scheduling the mean job execution time increases because it does not

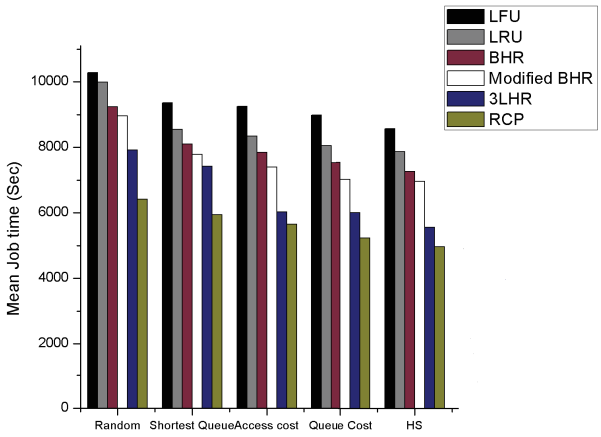

Figure 4.2: Mean Job Execution Time using Various Replication and Scheduling Strategies

consider any factors. In Shortest Job Queue Scheduling each CE receives approximately the same number of jobs. If CEs have low network bandwidth, then file transfer time will be high and overall job execution time will increase. Access Cost Scheduling selects a CE based on its access cost. CEs with lower access cost may receive large number of jobs to execute. So, overall performance is decreased. The Queue Access Cost considers not only shortest job queue but also access cost. Therefore, the Queue Access Cost decreases total job execution time. Finally, HS is the best job scheduling algorithm for the large number of jobs because it schedules jobs close to the data whilst ensuring sites with high network connectivity are not overloaded and sites with poor connectivity are not left idle. It also takes into account hierarchical grid structure and considers computational capability.

4.4.1.2 Mean Job Execution Time using Various Access Patterns

Figure 4.3 shows that RCP has the least value of mean job execution time in all the experiment and all of file access patterns. Obviously, the No Replication strategy has the worst performance as all the files requested by jobs have to be transferred from master node. The Random access patterns comprises Random, Unitary random walk and Gaussian random walk, where a certain set of files is more likely to be requested by grid nodes, so a large percentage of requested files have been replicated before. Therefore, RCP strategy and also all the other strategies have more improvement for random file access patterns.

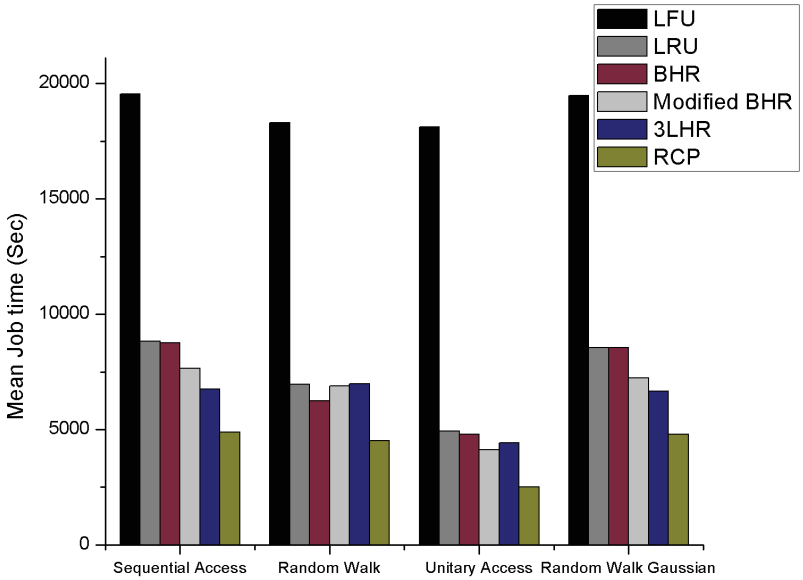

Figure 4.3: Mean Job Execution Time using Various Access Patterns

4.4.1.3 Mean Job Execution Time with Various Number of Jobs

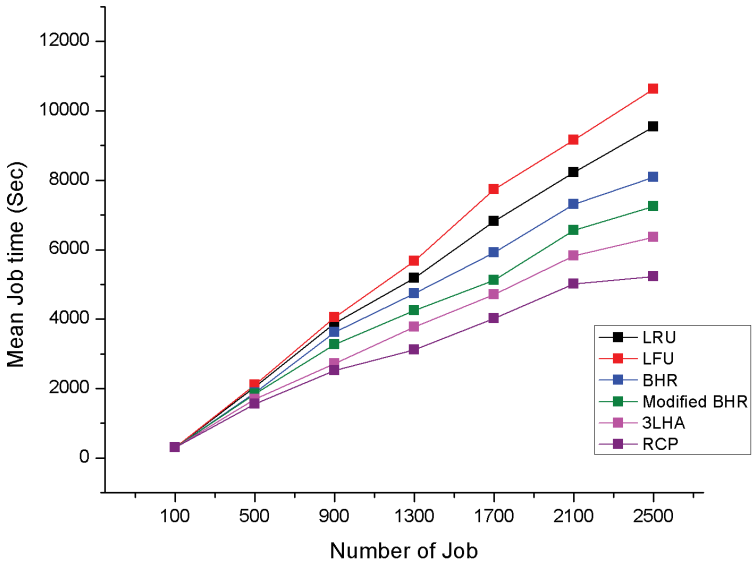

Figure 4.4: Mean Job Execution Time using Various Number of Jobs

Figure 4.4 displays the mean job time based on changing number of jobs for seven algorithms. It is clear that as the job number increases, RCP is able to process the jobs in the lowest mean time in comparison with other methods. It is similar to a real grid environment where a lot of jobs should be executed.

4.4.2 Effective Network Usage (ENU)

The ENU is used for estimating the efficiency of network resources. The ENU is dependent on the access frequency of the file locally and remotely and the number of times replicas of the file is created. This gives the ratio of files transferred across the network to total files requested. For a given network topology, a lower effective network usage indicates the optimisation strategy is better at replicating files to the suitable location. The effective network usage (ENU) is defined as [5]:

$$ENU = \frac{N_{rem} + N_{rep}}{N_{toc}} \tag{4.10}$$

where N_{rem} is number of access times file is read from remote site, N_{rep} is number of times the file is replicated and N_{toc} is total number of files requested by all CEs.

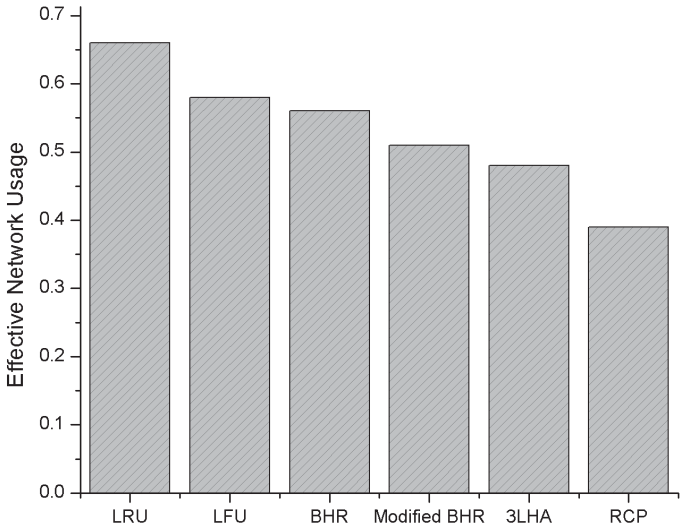

Figure 4.5: Effective Network Usage for Various Strategies

Figure 4.5 depicts the comparison of effective network usage with various replication strategy. The RCP utilizes the network resources most efficiently. The No Replication uses maximum number of network resources, results in poor performance. As compared to BHR and modified BHR the performance of 3LHA has improved. LRU has the least job execution time and highest effective network usage, showing that it is poor at making replication decisions.

4.4.3 Number of Replication

Large number of replications shows that large numbers of files were not stored locally at the time of execution, so replication was needed in order to access the required file. As it is obvious in Figure 4.6, RCP performs better in comparison with other algorithms and the total number of replications decreases in this method. The reason is that in this strategy, access frequency of the file helps in determining the future needs of grid sites and pre-fetches the replica at local site. Therefore more numbers of files are stored locally at the time of execution, decreasing total number of replications remarkably. As it can be seen in Figure 4.6, LRU and LFU create large number of replica, this indicates that sufficient number of the replicas were not stored locally at the time of execution of a job.

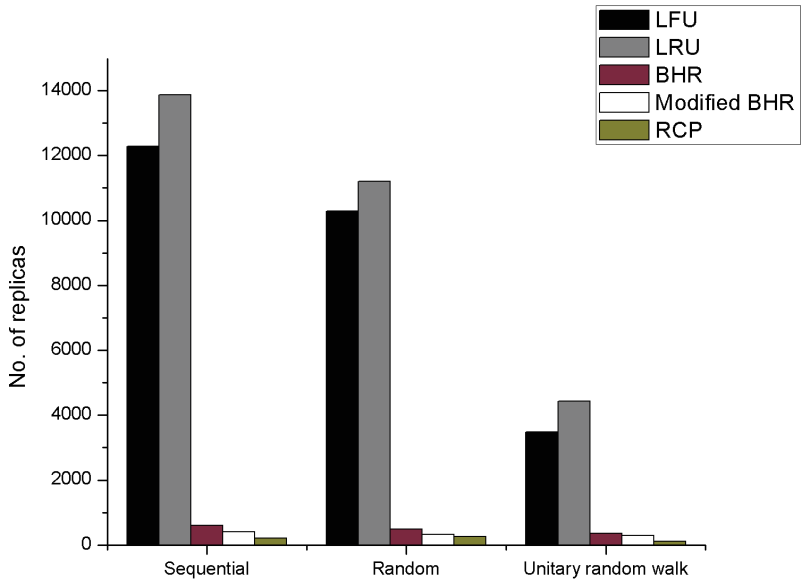

Figure 4.6: Number of Replications Vs Access Pattern

4.5 Conclusions

In this chapter, a popularity-driven replica creation and placement (RCP) strategy is discussed. The proposed RCP strategy have significant benefits. First, RCP is based on distributed and decentralized model; and second it can dynamically adapt to change to both network and user behaviour while improving the performance of the overall system. RCP strategy is divided in two phases. First replicas are created for the frequently accessed file by calculating the access frequency of the file. Secondly, the newly placed replicas are placed at appropriate location based on minimum placement cost.

The proposed strategy increases the data availability by creating the replicas dynamically. It also reduces the unnecessary replication by restricting number of replicas in a region. It places the replica at an appropriate location so as to reduce the placement cost. A comparative study of various existing strategies has been done to evaluate the performance of the RCP by considering various access patterns and strategies. RCP outperforms other existing strategies in terms of effective network usage, mean job execution time and number of replicas produced.

CHAPTER 5

Agent Based Replica Selection in Data Grids [2]

A replica selection strategy discovers the optimized replicas from the pool of replicas. An optimized selection strategy always select the best replica that fulfils the requirement of the user. In this chapter, an agent based optimized replica selection strategy, namely, Efficient Dynamic Replication using Agent (EDRA) has been proposed and implemented. To evaluate the impact of proposed optimisation EDRA strategy, simulation of the grid environment has been done to achieve important performance results before any strategy is deployed on the data grid. In this chapter, the effects of several scheduling and replica selection strategies are compared in different grid scenarios using various performance metrics. The results depicts that scheduling algorithms which take into account parameters namely, the workload of computing resources, network bandwidth, availability, computing capacity and file access cost of jobs are the most effective parameters at optimising grid resources as well as improving the job throughput. The results depicts that, in most scenarios, the EDRA strategy helps in enhancing the performance of system under different network loads.

5.1 Introduction

Replica selection problem has been explored by many researchers who considered response time as the only criterion for the selection process. Earlier studies focuses on binding a client to its nearest replica on the basis of some static parameters such as the topological distance in number of hops and geographical distance in miles. However, several experimental work shows, that the static metrics are not good predictors for the expected response time of client requests [**?**]. The main drawback of topological and geographical network parameter is that they ignore the dynamic conditions of network parameter. The objective in data grids is to

[2]Priyanka Vashisht, Rajesh Kumar, and Anju Sharma, *"Strategies for Replica Consistency in Data GridA Comprehensive Survey"*, Concurrency and Computation: Practice and Experience, 26(4), doi: 10.1002/cpe.3907

reduce access time and transfer latencies of a data file, as well as to avoid single site congestion by the numerous requesters. To facilitate access and transfer of the data, the files of the data grid are distributed across the multiple sites. The effectiveness of a replica selection strategy in data grids depends on its ability to serve the requirement posed by the user's request. Most requests are required to be executed at a specific execution time. Total execution time needs to factor latencies due to search of location of file and network transfer rates. Network resources affect the speed of moving the required data files and searching methods can reduce scope for replica selection. The purpose of replica selection is to select a replica among the sites that constitute a data grid [**?**]. The criteria of selection depends on characteristics of the application. Replica selection is important for the data-intensive applications as it can provide location transparency. When a user requests a data set, the system determines an appropriate way to deliver the replica to the user. Another issue concerning replica selection is the prediction of the transfer time, which involves the inspection of many characteristics and is a complex piece of work. In situations, where replicas are to be selected based on access time, grid information services can provide information about network performance and perhaps the ability to reserve network bandwidth, whereas the metadata repository can provide information about the size of the file. Based on this, the optimizer can rank all of the existing replicas to determine which one will yield the fastest data access time. Alternatively, the optimizer can consult the same information sources to determine whether there is a storage system that would result in better performance if a replica was created on it. In data grids, data replication provides a solution for managing data files efficiently. The data replication helps in enhancing the data availability which reduces the overall access time of the file. In this chapter, an optimized strategy, namely, EDRA (Efficient Dynamic Replication using Agents) for data grid has been proposed and implemented, which is an extension of BHR (Bandwidth Hierarchy Replication) strategy. EDRA consists of dynamic replication of hierarchical structure that take into account dynamic parameters such as network load, node availability, bandwidth etc. for the selection of best replica. Decision for selecting best replica is based on certain scheduling parameters. The scheduling parameters are: network bandwidth, load gauge, access frequency, availability, storage capacity and computing capacity of the node. The scheduling in data grid helps in reducing

the data access time. The distribution of the load on the nodes of data grid is done evenly by considering scheduling parameters [**?**]. The EDRA strategy is compared with Binary Hierarchy Replication (BHR) strategy. There are certain issues with BHR that are addressed in EDRA

(i) BHR strategy rely on one replica within a region.

(ii) Dynamic nature of the grid is ignored and assumption is that all the replicas are available throughout.

(iii) There is no scheduling strategy considered for load balancing.

5.2 Efficient Dynamic Replication (EDRA) Strategy

In this chapter, two stage optimization is provided which aims to increase network bandwidth, availability of data, and reduce total access time. Whenever, a request is submitted to optimizer, the following process happen:

- *Replication Decision*: In EDRA, if file is not available at the time of execution of job then replica of file is created as discussed in Chapter 4 so that grid can adapt according to its dynamic nature.
- *Replica Selection*: In this process best replica is selected locally among the pool of available replicas. The criterion for selecting the best replica in EDRA is based on the parameters such as, workload of the node, availability status, available bandwidth, access frequency of file and computing capacity of the node.
- *Replica Replacement*: When the replication decision is made the amount of storage space should be taken into account. If less storage space is available then there should be some replacement strategy involved such as Least Frequently Used (LFU), Least Recently Used (LRU), Exponential based Replica Replacement Strategy (ERRS) [**?**]. In EDRA, LRU is used for file replacement where the available storage space is checked before replacing the file. File having less access frequency is replaced when the replica of the file which is being requested.

EDRA strategy is divided into two parts i.e. region optimizer and sub region opti-

mizer. Region optimizer is executed at the master node of the data grid where, the agent helps in scheduling the job to the region for the execution. The node containing the required replica is selected on the basis of following parameters:

- *Network Bandwidth* This is one of the most significant data grid parameter as the files in data grid environments are usually large in size and distributed in nature. In order to relocate file from one location to another, transfer times plays a vital role and are tightly dependent on network bandwidth situations.
- *Load Gauge:* It is the length of the queue at master node and head node where jobs are waiting to get executed.

Region optimizer helps in getting the global view of all the regions in the data grid. At the second stage, the sub-region optimizer is invoked by master node on the basis of information gathered by agents at master node to get the best replica. The sub-region optimizer at head node of the region has the local view of all the sub regions in that region. The local information includes:

- *Availability:* This parameter is to check whether a node is available at the time of execution of request or not.
- *Access Frequency:* This parameter is the measure of number of file requests made by users running on the nodes in the region [144]. Access frequency helps in deciding the popularity of the file based on the access history. Higher the access frequency of a file on a node, more popular the file is.
- *Computing Capacity:* This helps in scheduling the jobs to the node having more computing power. The value of available computing capacity is calculated using equation 5.1.

Above mentioned parameters helps in scheduling the job to the destined node containing the file/replica in the region by efficient usage of the resources. The optimisation of replicas are performed in a distributed manner by number of replica optimisation agents, one for each grid region. An agent at sub-region performs local replica optimisation whose aim is to achieve global optimisation as the emergent result of local optimisation and every optimiser therefore has two main goals:

- *To minimise a job execution cost:* Users want their jobs to get executed at minimum cost; an optimiser therefore aims to minimise the execution cost

of every job that is run on grid node. In this chapter, the cost of executing a job is derived as the total running time of the job.

- *To maximise the usefulness of locally stored files:* Efficient utilisation of available data grid resources is another goal of optimisation. An optimiser should also, therefore, aim to keep locally those files that are most useful for jobs and are executed either locally or at neighbouring sub-region or region with maximum network connectivity. This also reduces the execution time and hence the performance of the grid system.

Algorithm 5.1 Pseudocode for Region Optimizer

```
1: procedureInput(Grid Topology, Bandwidth and Storage Space, Load gauge)

2: job is submitted to master node of grid by the agents/user
3: master node has the global view of all regions in the grid
4: do
5: send request to head nodes asking for (available bandwidth, load gauge)
6: end;
7: find maximum available bandwidth and minimum load gauge
8: rank the job according to maximum bandwidth and minimum load gauge in
   descending order
9: schedule job to region with highest rank
10: call optimizer for sub region to get best replica
11: execute all the jobs
12: optimizer(sub region)
```

5.3 Algorithms for EDRA Strategy

In EDRA, the request is submitted by the agent to the master node, where another agent is placed having the global view of the whole data grid. The jobs are submitted randomly. The agent located at master node schedules the job submitted by agent to the regions, where the replica/file has been stored with the goal of improving the overall throughput of grid. The available bandwidth between the master and head node of region and the load gauge of the head node is checked. The load gauge is a pointer to the waiting queue of head node which tells the number of jobs waiting for their turn to get executed. Available bandwidth and network gauge decides on which region the job is to be scheduled. This also helps master node to balance the load on the regions. The head node located in regions

Algorithm 5.2 Pseudocode for Sub-Region Optimizer

```
1: Agent at head node of each region keeps track of stored files, availability status,
   and computing capacity of node and access frequency with in region.
2: if file exist then
3:    process job;
4:    terminate optimizer
5: else
6:    if request file is not available in local sub region then
7:       fetch from nearby sub region
8:       proceed to execute the job with replica
9:       create new replica on node where it was initiated
10:   end if
11: end if
12: if free space available in SE of the node where request was initially scheduled,
   store new replica then
13:   store data
14: else
15:   check free space in SE of nearby node within in sub region and replicate
16: end if
17: if (!enough free space with the sub region) then
18:   sort file in least frequently accessed order
19:   for each file in sorted list do
20:      if access frequency of new replica > access frequency of the old file then
21:         delete old file
22:         if enough free space then
23:            store new replica
24:         end if
25:      end if
26:   end for
27: end if
```

is responsible for scheduling the jobs to the nodes where the replica is placed. An agent is placed on each head node of the region, which keeps information of the nodes located in the sub-regions. The information stored on the head nodes is local to the region like availability of the node, access frequency, computing capacity of node etc. within region. The replica is placed on the nodes based on two factors:(i) high data access frequency and (ii) sufficient storage capacity. The high access frequency shows the popularity of the file based on the access history stored at head node. The popularity of the file can be decided by number of request made for the file during the job runs on the nodes within the region. The available storage capacity in EDRA is based on the storage usage of the node. The available storage capacity $Aval_{str}$ on a node is calculated in equation 4.9. If the available storage space has more capacity than the size of the file, the replica is placed on the node. At the time of selection of the replica, the agent also checks the current available status of the node, computing capacity of the node for execution of request and maximum available bandwidth and load gauge. The computing capacity of the node is the factor which decides how fast a job can be processed by that node. The available computing capacity $Aval_{cap}$ of the node is calculated as:

$$Aval_{cap} = Comp_{node} - Comp_{usage} \tag{5.1}$$

where $Comp_{node}$ is the CPU total computing capacity of the node and is measured in (MHz) and $Comp_{usage}$ is the CPU capacity utilized by the node. If file is available in the sub-region, the job is processed otherwise agent sends the job to the neighbouring sub-region. After scheduling the request to the neighbouring sub-region, the job is processed on node containing replica. The information of replica in replica catalogue at head node is updated by the agent of that head node. After processing the job on neighbouring sub-region, a new replica of the file is created on the node where initially the job was scheduled. The replica catalogue is updated and the file is made available for executing the jobs. Though the number of replica of the file has been increased within a region by the EDRA algorithm but this consumes more storage space as compared to BHR. By increasing the number of replicas of the file, the availability of the file has been increased which helps in reducing the data access time of the file. There is always a trade off between the storage and the access time.

5.4 Evaluation Parameters and Experimental Results

The efficiency of EDRA is calculated on the basis of six parameters i.e. Mean Job Execution Time, Effective Network Usage, Storage Usage, Computing Usage, Hit Ratio and Transfer Time of the nodes. EDRA is compared with three other dynamic replication strategies i.e. BHR, No Replication and LRU on the Optorsim simulator using the above mentioned parameters. At the time of simulation, the complete data is available at the root of the hierarchy in No Replication strategy. In case of LRU, file is replicated when it is needed. BHR algorithm is the enhancement of Least Recently Used (LRU) algorithm. BHR algorithm follows hierarchical topology where nodes are positioned closely within a region having maximum bandwidth by taking advantage of "Network level locality" [144]. First the file is searched within the region, if the requested file is situated in the same region, less amount of time will be used for fetching the file. BHR expands existing site-level replica optimization to more scalable way by taking advantage of network-level locality. In BHR, there exists only one replica in a region on the basis of the popularity of the file. In EDRA, the number of replicas is decided dynamically based on the access frequency and degree of availability of file. Comparative Analysis of LRU, BHR, No Replication and EDRA is discussed in next section.

5.4.1 Mean Job Execution Time (MJET)

The mean job execution time is calculated as time to execute a file, the time spend by a job in waiting queue divided by number of jobs completed. It can be represented as:

$$MJET = \frac{\sum_{n=1}^{i} T_i + W_i}{n} \tag{5.2}$$

where n is the number of jobs processed by the system, T_i is time to execute the i^{th} job and W_i is waiting time of i^{th} job that has been spent in the queue. The EDRA along with LRU, No replication and BHR were tested using different job numbers of 100, 200, 300, 400 and 500 jobs. The job execution time for Random Zipf Access Pattern Generator is shown in Figure 5.1. It is evident that as the

number of jobs increases, EDRA is able to process the job in the lowest mean execution time as shown in Figure 5.1. EDRA is able to access the file in less time, which reduces the waiting time and increases the availability. Moreover, the scheduling strategy used is able to process the work load in less amount of time. The No replication strategy performs worse in all the cases. BHR performs better than LRU and No Replication.

Figure 5.1: Mean Job Execution Time for Varying Number of Jobs

5.4.2 Effect of Background Traffic

In the previous evaluation, grid background traffic was not included in the grid model. In this experiment, the effect of network traffic was included to get its effect on execution time and network usage. A comparison of the results with and without background traffic is shown in Figure 5.2 and Figure 5.3. As would be expected, there is a large increase in mean job time when we simulate the background network traffic. For all the optimisation strategies, this increase is around a factor of 3.

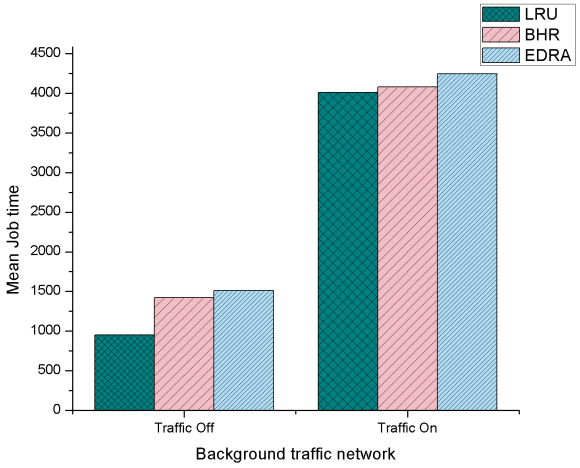

Figure 5.2: Mean Job Execution Time with Network Traffic

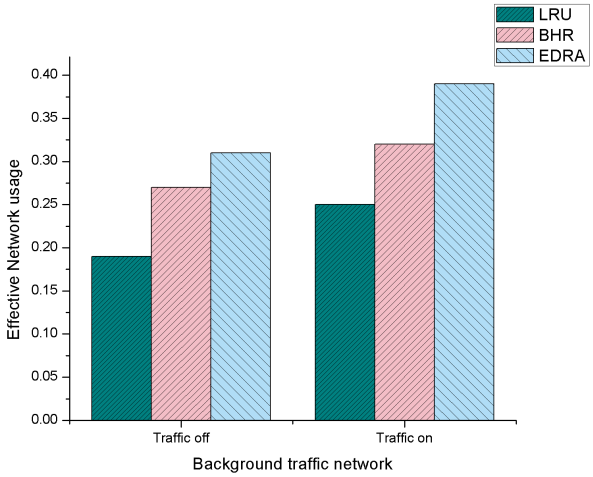

Figure 5.3: Effective Network Usage with Network Traffic

outperforms all the available scheduling strategies.

5.4.3 Storage Used (SU)

The percentage of storage used (SU) by files in MB is specified by storage used within region under this strategy. This can be calculated as:

$$SU = \frac{S_{total} - \sum_{i=1}^{n} Aval_{str}}{S_{total}} \times 100 \tag{5.3}$$

where S_{total} is total storage capacity of a region and $Aval_{str}$ is calculated from equation 4.9.

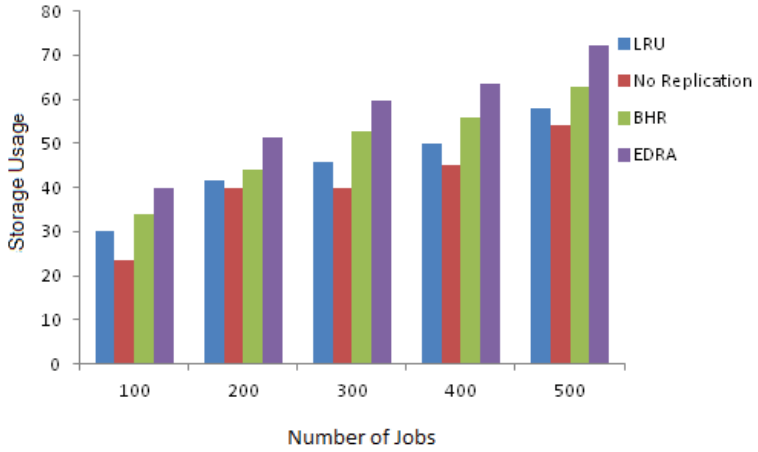

Figure 5.4: Comparison of Storage Usage

The storage used in No Replication strategy is best as no replica is created. EDRA performance is worst as the number of replicas has increased. The LRU and BHR strategy uses moderate storage space. The results are depicted in Figure 5.4.

5.4.4 Computing Usage

Monitoring the usage of computing power of data grid resources is a valuable source of information. This can be defined as the percentage of time the computing element is active or executing a job. The total computation power is derived as:

$$CU = \frac{Comp_{node} - \sum_{i=1}^{n} Aval_{cap}}{Comp_{node}} \times 100 \tag{5.4}$$

More computational usage indicates that the uniform distribution of jobs among the regions, reduces the execution time of the job. The computing usage is compared with three optimizing strategies; BHR, LRU, and EDRA.

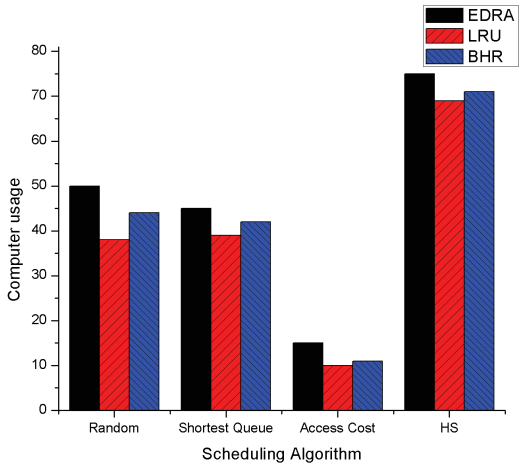

Figure 5.5: Comparison of Effective Network Usage

Figure 5.5 shows that the scheduling strategy mainly Access Cost Scheduling has lowest computing usage as in this strategy jobs are scheduled to regions with high network bandwidth. In case of HS scheduling strategy, the jobs are scheduled to the regions with high network bandwidth and lower load gauge, which helps in ensuring the uniform distribution of the job in the existing system. The Random and Shortest Queue strategies show similar behaviour and perform better than Access Cost Scheduling but HS strategy performed best.

5.4.5 Hit Ratio

In order to evaluate hit ratio, the numbers of local and remote file accesses of each strategy for different access patterns are considered. The hit ratio is dependent on the total number of replications. More number of replication indicates that the file is not available locally for execution. If number of replications are large, hit ratio of the file is less which indicates for executing the job, file are accessed remotely. No Replication strategy has very few files located locally, resulting in lower hit rate. Figure 5.6 shows the hit ratio evaluation for the five strategies. It is observed that our strategy has the highest value of hit ratio for almost all access patterns.

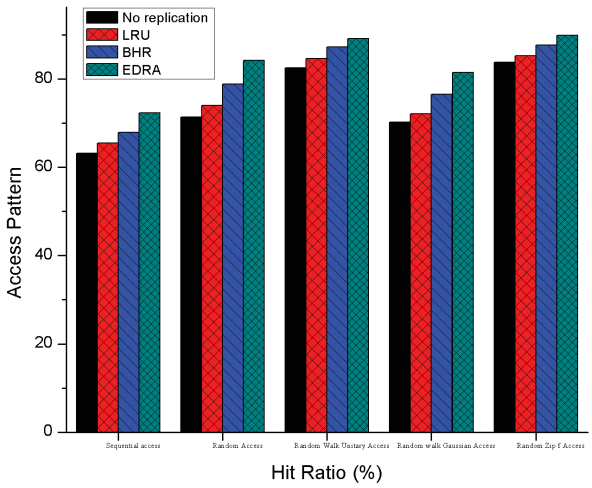

Figure 5.6: Access Pattern vs. Hit Ratio

5.4.6 Transfer Rate

Transfer rate is measured from the time when the file starts transferring from one node to another till the time when all transmission of files are finished. In this section, the data transfer time of the EDRA strategy from the data site to the requesting site is compared with LRU, BHR, and No Replication strategies. The data transfer rate is calculated by finding the time required for transferring the number of bytes sets from one site to another site. Figure 5.7 shows the comparison of mean data transfer rate. The mean data transfer rate was reduced in case of EDRA, as the files are located very close to the client. No Replication Strategy perform worst, whereas, LRU and BHR take more time to transfer files than EDRA.

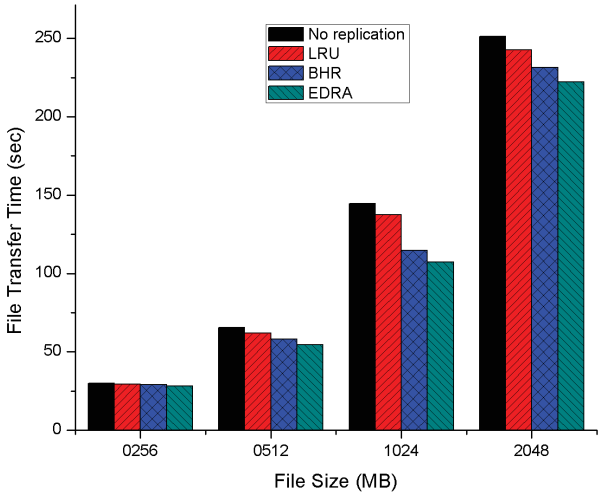

Figure 5.7: Transfer Time

5.5 Conclusion

In this chapter, an optimised EDRA strategy has been proposed and implemented. This work shows that Hybrid Scheduling (HS), a scheduling strategy take into account both the file access cost and workload of computing resources, which effectively optimizes the resources usage and reduces the executing time of the job. Moreover, the suitable choices for selecting replica can improve the performance of grid system. The EDRA strategy has shown significant results particularly when large number of jobs are to be executed. Moreover, EDRA perform better with or without the presence of network traffic.

The EDRA (Efficient Dynamic Replication using Agent) is used for optimization of replication in grid environment. In this work, the issues of BHR algorithm have been addressed. EDRA uses RCP strategy to create number of replicas which increases number of replicas within a region. The number of replicas is increased at the cost of storage capacity. In the previous works based on BHR, the assumption was that all the nodes are available all the time. Due to dynamic nature of grid this assumption is not appropriate. The proposed algorithm checks the availability status of particular node within region and takes the decision accordingly. To increase the performance of EDRA, two fold scheduling policy

has been adapted first at master node and second at head node. The scheduling has incorporated load balancing factor which results in better execution of jobs.

CHAPTER 6

Agent Based Replica Consistency and Conflict Resolution

Data replication is widely used mechanism aiming at ensuring efficient access, improving performance and providing quality data in data grids. The properties of file and their replicas can vary with time resulting to inconsistency. In this chapter, an agent based replica consistency approach, namely, Replica Consistency and Conflict Resolution (RCCR) has been proposed to improve the efficiency of the system. RCCR is hybrid approach which is used to address the problem of inconsistency in multi master environment. Additionally, the conflict resolution approach is introduced based on vector clock and version of the replica. Finally, simulated results are presented and compared with existing consistency approaches such as Pessimistic, Optimistic and One-way Replica Consistency (ORCS). The simulated results depicts that proposed RCCR approach performs better when writeable replicas are more likely to occur.

6.1 Introduction

In recent years, data-intensive applications and their management in distributed environment has become crucial task in order to enhance the performance, throughput and fast access to data. Distributed environment like data grid rely on sharing data and resources between various scientists and researchers to execute common tasks [**?**]. Scientific applications such as high energy physics, data mining, satellite images processing, and climate simulation, generates large amounts of data which is stored and managed by data grids. The management and access to huge volume of data located at various distributed locations across the data grid, degrades due to certain network restrictions. Replication helps in maintaining and optimizing access time and the availability of data in distributed environment. Replication provides a means for allocating similar data at various distributed locations in

data grid. Despite advantages of data replication technique, the main challenging issues are consistency maintenance and conflict resolution of geographically distributed data. Replica consistency can be achieved using pessimistic or optimistic approaches in distributed environment.[34]. Pessimistic technique provides single copy serialization for achieving consistency in datasets. One copy serialization uses locking technique for propagating updates across distributed files, as a result large number of replication and frequent updates decreases the performance of pessimistic approach. This is the reason why pessimistic technique is not suitable in distributed environments such as grids and clouds. In optimistic replication technique, relaxed approach is used for propagating the updates to replicas. Delayed updates propagation helps in improving the write access even in the presence of unreliable network link. In optimistic replication the performance of the system decreases when frequency of the write request is very high and problem will worsen as the magnitude of data grid grow. Moreover, if the updates are made concurrently in multi-master environment, issue of conflict arises among the replicas. There exists two methods for resolving conflicts i.e. *backward* and *forward* conflict resolution. If the divergence among replicas is small backward conflict resolution method is used, in which request is simply rolled back and the client that submitted the request is notified of the result. The client may then resubmit the request for execution. In forward conflict resolution method, the difference among stale data and latest data find a way of merging conflicting replicas to make all replica in consistent state. The forward resolution method is more suitable for real time systems such as grids since such systems must interact with a continuously evolving environment. Roll backing to an older state is counter intuitive in dynamic systems like data grids and may cause transactions to miss their deadlines. Therefore, the key issues in data grids is how the file and its replicas can be used efficiently and effectively to reduce inconsistencies and resolving conflicts.
Main contribution of this work consists of proposal of consistency management approach in large scale distributed system, which uses hybrid approach to take advantage of both optimistic and pessimistic approach. The optimistic approach enhance the performance of the system whereas the quality of service is assured by pessimistic approach. The contribution of this research has two essential aspects:

(i) an agent based hybrid approach is followed for maintaining the consistency

among replicas, which rely on access frequencies of the file.

(ii) the conflicts are handled using agents at various regions in multi master environment.

6.2 Replica Consistency and Conflict Resolution (RCCR) Strategy

The main goal of proposed approach is to provide efficient results by handling inconsistencies and resolving conflicts among replicas. The proposed RCCR approach follows the steps mentioned below during the execution of the request made by the client.

(i) **Request:** A client submits a request for a file f_{ij}. Request can be either read only or write anywhere.

(ii) **Execution:** Request is initially handled at local level. Once the request is deployed on a particular node, its resource utilization is monitored by IP after every fixed interval and observed information is stored in LDB of the head node.

(iii) **Response:** After execution of the request, the node provides response to the client.

(iv) **Coordination and Agreement:** In this step, an agent on the head node will update the corresponding information and propagates the results to the other nodes containing replicas of the same file. During this phase, the operation is scheduled for execution at all relevant nodes within the region, and same is updated with the master node. The head node will ensure if there is any inconsistency with in a region, it will propagate the updates to corresponding replicas by electing leader node of the region. A node with highest number of updates and latest timestamp is elected as a leader. After maintaining the consistency at regional level. The information regarding the replicas is propagated to the master node of the network. Master node is responsible to maintaining consistency across the a single networks and also among the masters of another network. It will check whether the all the

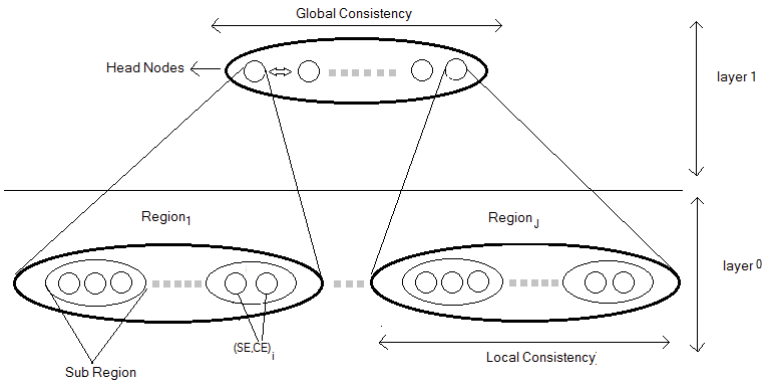

Figure 6.1: High Level View of RCCR Strategy

replicas within the network are convergent, divergent or there exists some conflicts. Convergence indicates that two replicas are in consistent state, divergence signifies that two replicas are different by only one component. Two replicas are in conflicting state, if at least two or more components of a file are different.

(v) **Conflict handling**: If no conflicts are detected during the coordination and agreement step, updates are integrated permanently into the GIC and changes are made in GDB as well. If conflicts are detected, they must be handled mutually by exchanging information among various master nodes.

6.3 Working of RCCR Strategy

A hierarchical network structure has been implemented which offers the system with flexibility, scalability and tolerance of certain faults. The work is divided into two parts, first the local level consistency is maintained at region and sub-region using head node. Secondly, the conflicts and consistencies are checked among multi-masters of various regions to ensure global consistency. A two level architectural view of RCCR strategy is presented in Figure 6.1 and the functional view of RCCR strategy is presented in Figure 6.2.

At *Level 0:* group of nodes form a region or sub-region. Each node contains number of replicas of several files. Each replica at level 0 contains some additional information such as ($VV_{ij_{kp}}$, $timestamp_{ij_{kp}}$, rep_{id}) where, $VV_{ij_{kp}}$ is a vector for storing version of replica p of file k located at node i of region j, $timestamp_{ij_{kp}}$ is timestamp for recent update to keep track of local update and r_{id} is Id of replica.

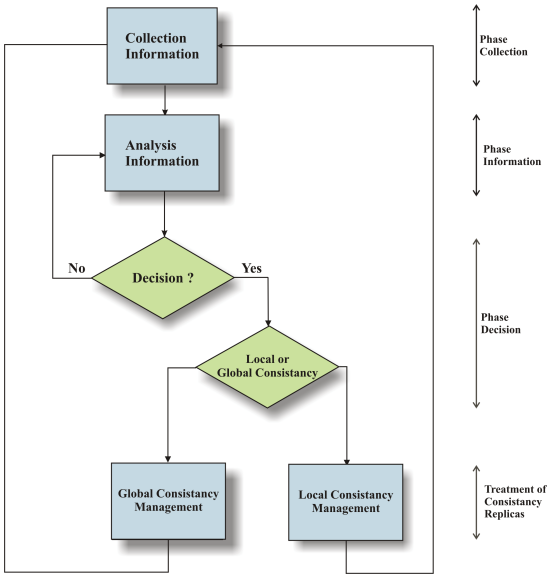

Figure 6.2: Functional View of RCCR Strategy

At level 0 the consistency with in the region or local consistency is achieved. To achieve the local consistency following steps are needed:

(i) Collection of Information: The agents on the head node start collecting the information of all the node that belongs to a region. The collected information are:

- Number of replicas of a file f_i in the region.
- Number of requests for a file f_i in the region.
- Latest version (Ref_{VV}) and timestamp (Ref_{time}) associated with each replica r_{ik} of a file f_i that exist in the region.

(ii) Selection of Leader Node::

- Leader node is selected by analysing the information collected in the previous step.
- Replica having the latest version (Ref_{VV}) and timestamp (Ref_{time}) is elected as the leader node.

(iii) Propagating the Updates in the Region:

- The elected leader node is responsible for propagating the updates to other replicas of same file in the region.

- The updated replicas are added to the list of leader and start propagating the updates to other replicas of the sub-region of same region.
- When all replicas in the sub-region is updated, then the leader nodes propagate updates to remaining sub-regions of the region. The multiple leader nodes propagate the updates in parallel, hence reducing the update time.

(iv) After updating all the replicas in a region the information is send to the master node of the network, so as to propagate the updates in entire network.

At *Level 1*, *master node* acts as a representative for that network. The master node ensure consistency at two levels. Firstly, it is responsible for converging all the replicas to a consistent state in the entire network. Secondly, the consistency is achieved among several networks. The global consistency among the regions uses optimistic approach where as the local consistency at head node follows pessimistic approach. To achieve the consistency globally across the grid system, following steps are needed:

Case I Consistency maintenance by a single master in its network

- If Divergence is there in version and timestamp between two regions.
- Identify the missing operation.
- Propagate the updates for missing operations to the head node region.
- Master node will maintain the consistency among various regions as discussed above.

Case II Consistency maintenance among various master nodes

- If Conflict occur among replicas of a file at are located in different networks.

(i) Selection of Representative: A master node is selected among the pool of available master nodes on the basis of maximum number of updates.

(ii) Collaboration: A representative master node will collaborate with other master nodes for propagating the updates of the replica.

(iii) Detection and Resolution of Conflicts: Here, divergent and conflict are two cases for representing the inconsistencies. If the amount of inconsistency among two replicas is small then it is divergent else there exist certain conflicts among two replicas of different regions. Mutual exchange of information is done by selecting representative at global level.

6.4 Algorithms for RCCR Strategy

The user input requests to grid system. Once the request is deployed on a particular node, its resource utilization is monitored by IP after every fixed interval and observed information is stored in LDB. Moreover the reference vector version Ref_{VV} and reference timestamp Ref_{time} is also updated as shown in Algorithm 6.1. The request is executed and results are send back to client.

Algorithm 6.1 Pseudo code for Consistency Maintenance

```
1: procedure Consistency maintenance
2: input: Replica Location
3: Ref_VV ← φ
4: Ref_time ← φ
5: if read/write request received then
6:    [enu] ← [enu] + 1
7:    execute operation on n_ij
8:    if read operation then
9:       return (result)
10:   else
11:      Ref_VV ← VV_ijkp
12:      Ref_time ← timestamp_ijkp
13:   end if
14: end if
```

For dealing with inconsistency at local level the head node will elect a leader node as depicted in Algorithm 6.2. The leader node is selected on the basis of latest timestamping and vector version of the file and is depicted in Algorithm 6.3. The leader node propagate its update to other replicas within a region. The updates are propagated among regions with leader nodes. Local level consistency follows optimistic strategy so that no divergence or inconsistency exists among replicas.

After that information is sent to the master node to achieve global consistency. Master node analyser (MA), an agent on master node analyses the request and then map it to an appropriate region. An agent called request allocator (RA) coordinates with global database (GDB) to map the request with an appropriate region. Requests are then scheduled to head node of the allocated region based on the information collected by request allocator (RA). Due to multi-master approach

Algorithm 6.2 Pseudocode for electing leader node

1: $node_{led} \leftarrow 0$
2: **for** all n_{ij} in the region **do**
3: **if** $(VV_{ij_{kp}} = Ref_{VV}) \vee (timestamp_{ij} = Ref_{time})$ **then**
4: $node_{led} \leftarrow n_{ij}$
5: **end if**
6: $VV_{ij_{kp}} \leftarrow VV_{ij_{kp}} \cup Ref_{VV}$
7: $timestamp_{ij_{kp}} \leftarrow timestamp_{ij_{kp}} \cup Reftimestamp$
8: $node_{led} + +$
9: **end for**
10: **return**(consistent replicas)
11: **end procedure**

the client can make request to any master which may lead to have conflicts at global level. If conflicts are detected, then the conflict resolution mechanism is invoked. In this work, the conflicts are categorized as Replica Conflict and Update Conflict.

Two updates u and $u^{'}$ to a file f_i are in conflict, if they are performed concurrently on a replica r_{ij} located on some nodes. There is an update conflict state if neither of the update has observed the effects of the other before executing. The update conflict has a consequence which results into replica conflict. If the updates are applied to different replicas r_{ij}, r_{ik} of a single file f_i, they will represent different versions of f_i. This make replicas r_{ij} and r_{ik} in conflict state after the updates have been applied.
However, when a single update u is applied to a replica r_{ij} of file f_i , it does not lead replicas in conflicting state. Though the state of r_{ij} differs from the state of another replica r_{ik} until update u has been applied to r_{ik}. As in optimistic approach the state of two replicas can vary upto a certain time but eventually achieve consistency with due course of time. To resolve the problem of update conflicts a logical time-stamping is used, while the replica conflicts are taken care by using version vectors and is depicted in Algorithm 6.3.

The complexity of the replica creation and placement algorithm is $n_{ij}Nlog(n_{ij}N)$ where n_{ij} is number of nodes processed for placing replica and N is total number of regions.

Algorithm 6.3 Pseudocode for removing conflicts

1: Multicast(Ref_{VV}, Ref_{time})
2: **if** Convergence(($VV_{ij_{kp}}, Ref_{VV}$) $\vee$ ($timestamp_{ij}, Ref_{time}$) **then**
3: convergence, no propagation
4: **else**
5: **if** Divergence(($VV_{ij_{kp}}, Ref_{VV}$) $\vee$ ($timestamp_{ij}, Ref_{time}$) **then**
6: Divergence $\leftarrow$ missing operations
7: $VV_{ij_{kp}} \leftarrow VV_{ij_{kp}} \cup Ref_{VV}$
8: $timestamp_{ij_{kp}} \leftarrow timestamp_{ij_{kp}} \cup Reftimestamp$
9: **else**
10: **if** Conflict(($VV_{ij_{kp}}, Ref_{VV}$) $\vee$ ($timestamp_{ij}, Ref_{time}$) **then**
11: Mutual exchange of information by selecting leaders at global level
12: **end if**
13: **end if**
14: **end if**
15: **return**(consistent replicas)
16: **end procedure**

6.5 Evaluation Parameters and Experimental Results

The efficiency of RCCR is considered based on six performance metrics i.e. access cost of read and write operation, percentage of accessing consistent data, amount of update transaction, retired request rate , conflict rate and mean job execution time. RCCR is compared with optimistic, pessimistic and ORCS approaches on OptorSim using performance metrics mentioned in Table 6.2 with varying parameters as shown in Table 6.1. During simulation pessimistic approach keeps entire replicas consistent all the times. In optimistic approach, consistency takes place after a small interval. ORCS follows hierarchical structure where data is stored primarily at root node and is automatically replicated to various leave nodes. Updates can be made only on original data as the files at other hierarchical level are read only. ORCS considered two issues (i) check the storage capacity of the replicas and (ii) find the access frequency of the replicas. OCRS follows asynchronous replication whereas in RCCR hybrid approach for maintaining the consistency among the replicas is used. RCCR take advantage of both pessimistic and optimistic approach. Pessimistic approach helps in ensuring consistency of the files all

Table 6.1: Parameters Used in Simulation

Parameter	Value
File size	10 MB - 1GB
Number of files	1000
Number of jobs	100
Network latency	10ms-20ms
Storage media latency	2ms-5ms
Transfer rate	50 Mb/s
Storage capacity	10PB

Table 6.2: Performance Metrics used for Simulation

Performance Metrics	Description
Access Cost	Specifies cost for read and write operation based on access frequency, size of file and communication cost
Percentage of Accessing Consistent Data	Specifies percentage of consistent replicas of files at specific time interval
Amount of Update Transaction	Number of write request performed on a file at some time interval
Mean Execution Time	$\frac{Total execution time + Total Waiting time}{Number of completed requests}$
Retired Request Rate	Specifies the number of reissued request due to failure or conflict
Conflict Rate	Specifies the number of times stale data is encountered

the time whereas optimistic approach improves the system performance.

6.5.1 Percentage of Accessing Consistent Data

As compared to the Pessimistic consistency approach, during propagation phase, the number of update message decreases considerably in proposed approach. When RCCR strategy is implemented, the performance of the system increases significantly. The update actions on replicas are not frequent due to hybrid approach followed in RCCR. The replica consistency approach is applied from time to time so that the probability of inconsistent consequences can be dealt. Consequently, system attains a balance between consistency, performance, and availability. The

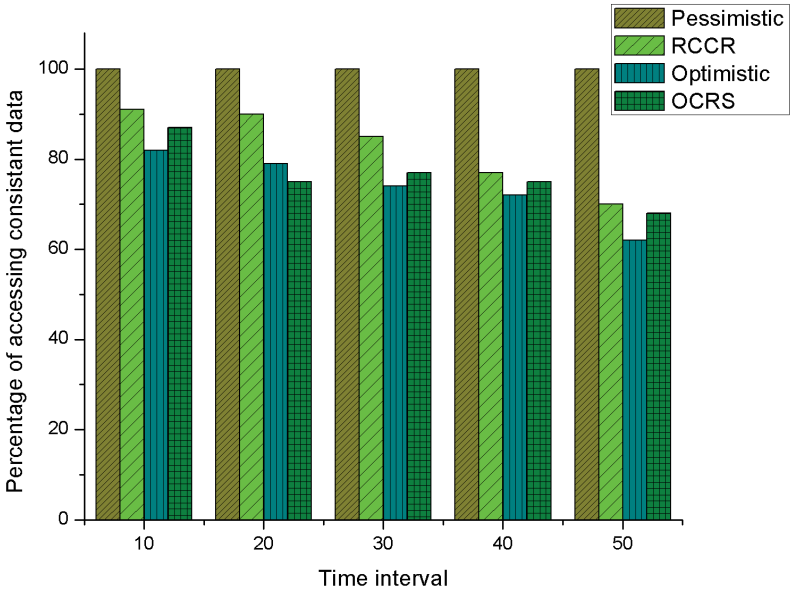

Figure 6.3: Percentage of Accessing Consistent Data

percentage of accessing consistent data is calculated as:

$$P_{cons} = \frac{Num_{con}}{Num_{f_i}} \times 100 \tag{6.1}$$

where Num_{con} represents number of consistent replicas of file f_i at time τ and Num_{f_i} is total number of replicas of file f_i at time τ. Figure 6.3 shows the evaluation of the percentage of reading up-to-date data at each predefined checkpoint time interval τ between Pessimistic, Optimistic, ORCS and RCCR consistency approaches. Pessimistic approach almost guarantees that every single request for read is satisfied by accessing the updated data. Optimistic and ORCS approaches guarantee weaker consistency, so the percentage of accessing updated data is lower than Pessimistic approach. In RCCR strategy the guarantee for consistency is better than Optimistic strategy whereas, ORCS is marginal with Synchronous consistency strategy.

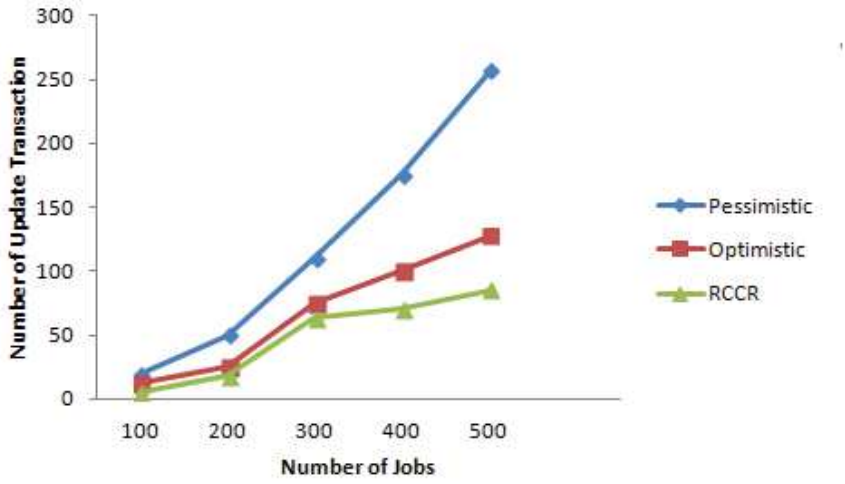

Figure 6.4: Number of Update Transactions

6.5.2 Amount of Update Transactions

In RCCR algorithm, number of update transaction decreases significantly as compared to Pessimistic consistency. The number of operations for data update has reduced, consequently enhances the performance of the system considerably. The amount of update transaction is calculated by:

$$UP_{trans} = \text{number of write request executed for replica} r_i \in f_i \text{ at time } \tau \quad (6.2)$$

In Figure 6.4, comparison of total amount of updates for Pessimistic, Optimistic and proposed RCCR approach has been done. As time goes by, a number of transactions for Pessimistic consistency approach increases rapidly, whereas, in Optimistic consistency approach, the volume of transactions increases gradually. A number of transactions for proposed RCCR algorithm exceed the total transactions for Optimistic approach marginally. As compare to Pessimistic consistency, the number of transactions for RCCR algorithm reduces significantly. Also, the RCCR consistency approach decreases cost of update transaction significantly, while the requirements of application for consistency are fulfilled. Moreover, RCCR consistency approach assures higher level of consistency than Optimistic consistency while the transaction cost increases marginally. Consequently, a better balance between consistency, availability, and performance is achieved.

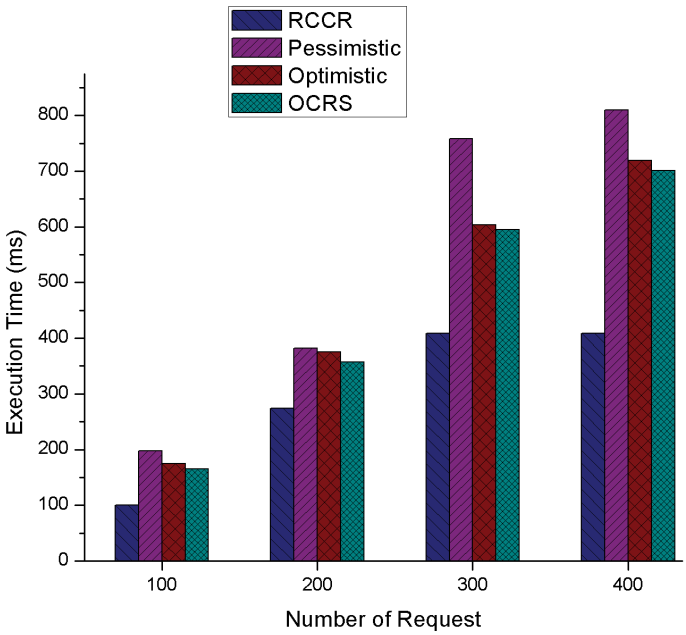

Figure 6.5: Average Execution Time

6.5.3 Average Execution time

Average Execution Time is calculated as ratio of time taken for complete execution of a request, time spent by the request in waiting queue and the total amount of requests. It can be represented as:

$$\text{Average Execution Time} = \frac{\sum_{n=1}^{i}(T_i + W_i)}{n} \tag{6.3}$$

where, n is number of requests processed by the system, T_i is time to execute the i^{th} request and W_i is time spent by i^{th} request in waiting queue. The RCCR along with Pessimistic and Optimistic approach was tested using 100, 200, 300, 400 and 500 numbers of requests. The comparative analysis of RCCR, Pessimistic, Optimistic and ORCS is presented in Figure 6.5. It is apparent that execution time of RCCR is lowest as the number of requests increases. The access time of the file has reduced in RCCR, which moderates the waiting time. The Pessimistic approach performs worst in all the cases whereas, RCCR performs better than Optimistic and ORCS approaches.

6.5.4 Access Cost

The access cost for read and write operation is dependent on number of times request for read are made for the file f_i , number of bytes processed by the request i.e the size of the file and their communication cost. The access cost of read and write request is defined as :

$$A_{read} = r_k \times S_f \times C_{cost} \tag{6.4}$$

$$A_{write} = w_k \times S_f \times C_{cost} + \sum_{i=1}^{\delta} \sum_{j=1}^{k} C_{ij} \tag{6.5}$$

where r_k is a number of read requests performed on the data file f_i, w_k represents number of write requests performed on the data file f_i, δ is number of nodes holding the replica of file and k is number of nodes connecting link between n_i and n_j. The communication cost (C_{cost}) is determined by:

$$C_{cost} = \sum_{i=1}^{\delta} \sum_{j=1}^{k} C_{ij} + P_{network} \tag{6.6}$$

where, C_{ij} is communication cost between two node n_i to n_j such that $n_i \neq n_j$ but are connected to each other via some communication link in the network, δ is number of nodes holding the replica of file and k is number of nodes connecting link between n_i and n_j, and $P_{network}$ is the propagation delay of network. Figure 6.6 and 6.7 shows the comparisons of access cost for read and write operations with various numbers of requests. When the number of read requests is more than the write requests than Pessimistic consistency approach outperforms RCCR approach. With time when the number of write requests increases then, pessimistic consistency approach performs worst. In proposed approach, the number of replicas is limited and the updates are propagated only after a certain period of time. This helps in limiting the use of bandwidth required to propagate updates of replicas. Henceforth, the overall performance of the system increases.

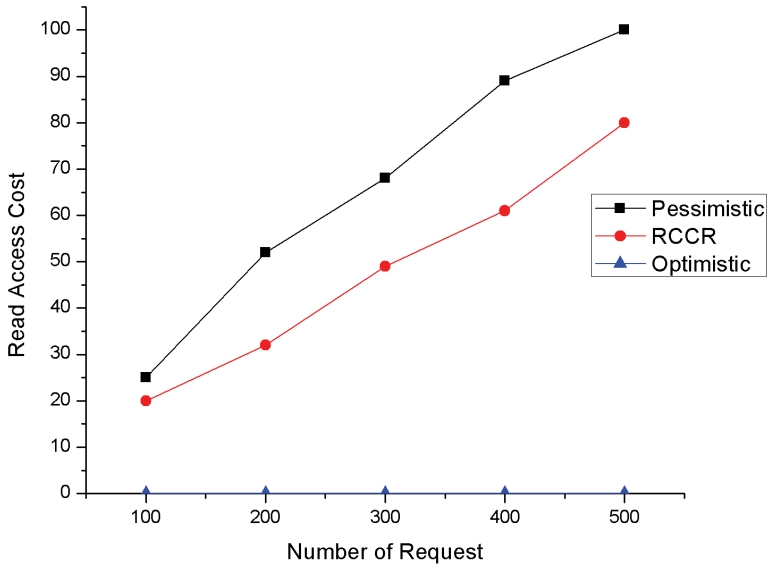

Figure 6.6: Access Cost of Read request

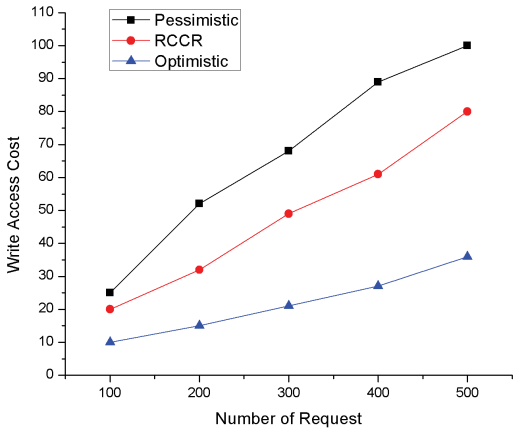

Figure 6.7: Access Cost of Write Request

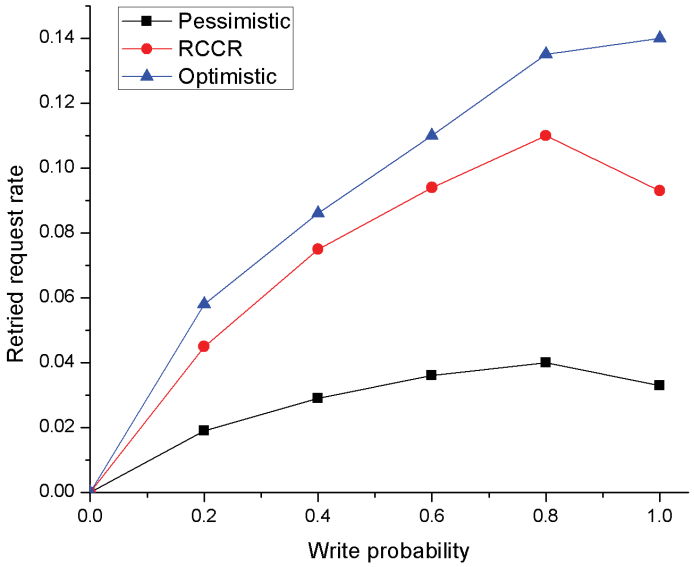

Figure 6.8: Comparison of Retried Request Rate

6.5.5 Retired Request Rate

Retired Request rate is defined as number of requests than have been reissued due to conflict upon the total number of requests made for the file. As compared to Optimistic approach the proposed RCCR approach has shown an improvement, whereas, the results are marginal to Pessimistic approach. The Retired Request rate Ret_{RR} is calculated as:

$$Ret_{RR} = \frac{R_{ret}}{Total_{req}} \tag{6.7}$$

where, R_{ret} is defined as amount of reissued requests after failure and $Total_{req}$ is the total number of requests made by the client for a file. However, $Total_{req}$ is calculated as

$$Total_{req} = r_k + w_k \tag{6.8}$$

Figure 6.8 shows the evaluation of the retired request rate with respect to various

write probability of the file. For the simplicity, an assumption has been made that all files have same write probability. The simulation is done in batches, each batch having different write probability. Pessimistic consistency approach guarantees that the Ret_{RR} of each request is minimum. The number of retired requests rate increases as the number of write request for a file increases. The proposed RCCR approach shows a significant improvement by lowering the number of reissued requests as compared to Optimistic approach. Retirement rate of the request is not increasing much with respect to amount of write requests. It provides better handling of retired requests as compared to Optimistic approach.

6.5.6 Conflicts Rate

The parameter is defined as number of time stale data of a file come across when the update requests is made over a period of time. The conflict rate is calculated as:

$$Conf_{Rate} = \frac{Conf_{num}}{W_{rep}} \tag{6.9}$$

where, $Conf_{num}$ is number of conflicts and W_{rep} is number of write request of a replica over a period of time. W_{rep} is calculated as:

$$W_{rep} = \frac{Total_{req} - r_k}{\tau} \tag{6.10}$$

Figure 6.9 shows the comparison of conflict rate of replicas. The results depict the conflict rate RCCR performed better than Optimistic approach, as the number of write/update request increases over a period of time. As RCCR uses hybrid approach which ensure that the performance of the system is maintained by taking advantage of both Pessimistic and Optimistic approaches. Pessimistic approach is not considered as there does not exist any conflict.

6.6 Conclusion

In this strategy a hybrid approach namely, RCCR is proposed and implemented for maintaining consistency and resolving conflicts in multi-master grid environment, which take advantage of both optimistic and pessimistic approaches. The pessimistic approach assures the quality of service and the optimistic approach

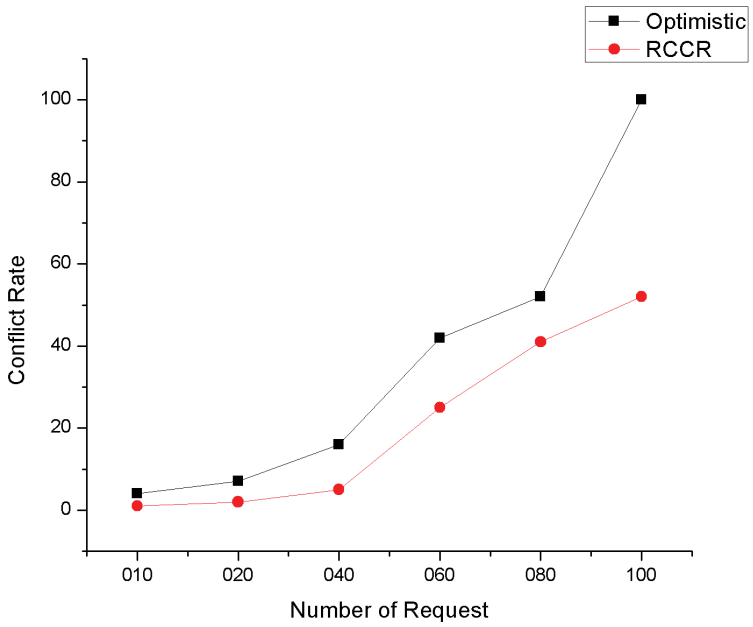

Figure 6.9: Comparison of Conflict Rate for Various Strategies

is responsible for enhancing performance of the system. The contribution of this work, aims to resolve conflicts through forward conflict resolution, which means that conflicting updates are merged rather than rolled back and re-executed. The forward resolution generates a new consistent state which is more suitable for dynamic environments like data grids.

The RCCR is compared with well known approaches such as Optimistic, Pessimistic and ORCS. Various Parameters has been considered such as percentage of accessing consistent data, amount of update transactions, average execution time, access cost, retired request rate and conflict rate. The results shows that proposed RCCR approach assures consistency of replica while maintaining efficiency of the system.

CHAPTER 7

Conclusion and Future Scope

This chapter presents concluding remarks on the thesis by highlighting the principal contributions of this research study. Referable to the potential involvement of extremely large numbers of files and their replicas located at distributed nodes, replica management remains to be a major challenge for grid computing. A number of issues related to it have been identified and addressed. This thesis focuses upon replica creation, placement, selection and consistency maintenance in distributed grid environment. To address the problem, an agent based Replica Management Model (RMM) for efficient management of replicas has been presented. Further, various agent based replica creation, replica placement, replica selection and replica consistency maintenance strategies have been designed and implemented to support RMM model. This chapter concludes the thesis by highlighting the main contributions of this research work. This chapter presents the outcome of each chapter and then discusses the contributions of the RMM for grids. Subsequently, it focuses upon the future scope of the study.

7.1 Conclusion

With the proliferation of scientific data, the management and access to huge data sets at geographically distributed location become slow due to several network restrictions. To increase the availability of data, various data grid service providers has increased the storage and computing resources in their data centres in order to accommodate the user demands. The data is replicated on distributed storage resources to enhance the availability data and minimize the access rate. A crucial challenge of data grids is how to optimize the replica management issues using available resources. Replica management in large scale distributed system like grids is one of the most challenging and popular research topic. In this thesis, several replica management issues such as creation, placement, selection and

consistency maintenances are tackled. The major contributions of this thesis are achieved in phases. The brief summary of each phase is as follows:

(i) In the initial phase, in depth review and analysis of existing research in the area of replica management, and consistency maintenance in data grids has been carried out as detailed in Chapter 2. The classification of related work on various factors is also presented. More specifically, various types of asynchronous replication strategies have been presented. This chapter helped to identify the open challenges in the area of replica management and consistency control in large distributed system and helps in defining research directions. The existing literature analysis helped to identify research gaps and challenges in the area of large and distributed datasets, and clearly defines the research path followed in this thesis.

(ii) An optimised Replica Management (RMM) model for efficient management of replication and consistency management is designed and developed as discussed in Chapter 3. The RMM model is layered in nature. The layered approach provides the flexibility and increase the scalability of the system. The proposed RMM model has been devised and implemented in simulated grid environment based on Optorsim. The RMM model support various replica management strategies, namely, Replica Creation and Placement (RCP), Efficient Dynamic Replication (EDRA) and Replica Consistency and Conflict Resolution (RCCR). Different use cases for each strategy is also elaborated in Chapter 3.

(iii) An agent based efficient Replica Creation and Placement (RCP) strategy is presented in Chapter 4. A popularity driven dynamic replica creation and placement is developed to increase the access rate of file. The idea behind RCP strategy is to create replicas dynamically and placing them close to the users that frequently make requests for a particular file. The number of replicas are restricted in each region to optimize the usage of storage resources. The optimization of network resources such as network bandwidth, computational and storage capacity are also provided. The RCP strategy helps in determining the appropriate number of replicas and place them at suitable locations. The goal of the strategy is to minimize the replication cost, placement cost, execution time of the job and effective usage

of network resources.

(iv) Beside creating and placing replicas efficiently, the selection of appropriate replica is other critical issue resolved in this work. As replica selection in data grid is an optimization problem itself. Two level replica selection strategy using agents is presented in Chapter 5. The strategy is to choose the best replica from multiple replicas with minimum cost. A replica selection strategy based on the replica response time was presented to select the best replica. The selection is done on the basis of certain scheduling parameters such as network bandwidth, queue length, computing capacity, access frequency etc. These parameters helps in maximising the usefulness of grid resources. The two fold scheduling policy is also presented done first at master node and secondly at head node. This helps in distributing the workload among various regions of the system in a very effective manner.

(v) The replica management strategies presented in Chapter 4 and Chapter 5 considers read only files, where no modification is done. The issue is resolved by extending the RMM model with RCCR strategy, which allows the write operations on the files as detailed in Chapter 6. The RCCR strategy has been introduced to provide the consistency among the replicas of same file distributed across the grid system. The consistency maintenance is carried out at two levels: The strategy determine three data flow steps to manage the consistency among replicas: (i) At first, the optimistic approach is used for attaining the consistencies among subregions. The head node in the region need to collect information about the existing replicas locally. Then, leader node is elected which exchange updated information among replicas of a region to allow all the replicas converge to a coherent state. (ii) In the second flow, every head node at upper layer will interact with their corresponding head node of the region to exchange the updated information. The information provides the global view about the replicas in the region. (iii) The last flow consist in exchanging updated information among several master nodes. The global consistency of the grid will be achieved after mutual exchange of information. A hybrid approach is used to take advantage of both optimistic and pessimistic approach. The RCCR strategy reduces the execution time while providing the updated replicas with marginal update

operations. Moreover the conflicts are handled very efficiently.

7.2 Scope and Future Work

The work presented in this thesis relates to replica management and consistency control in data grids. The aim is to optimally handle the distributed replicas and their consistencies in order to improve system efficiency. The centralized popularity-driven replica placement (RCP) and selection (EDRA) strategies and its adaptive in nature which aims to balance the storage utilization and access latency trade-off by determining the frequency and degree of replication in changing grid conditions. The simplicity of the approach makes it easy to deploy on a large number of nodes. The data in this scenario is read-only and so there are no consistency issues. This assumption holds at least reasonably for grid applications where data updates are expected to be infrequent.

Another important aspect of replication-based systems is the protocol used to maintain consistency among replicas. The updates are delivered from the original copy to all replicas, in which each head node receives the updates from its parent and is responsible for further distributing the updates to its children. The main issue in such systems is maintaining scalability with large numbers of replicas distributed over the grid while maintaining the same view of all replicas (i.e. consistency). It is assumed that in grid applications, updates to the data are infrequent and that the consistency can be more relaxed than in, for example, high-performance commercial databases.

Moreover, in this work hierarchical structure is used. Not all potential applications fit this sort of data model. In some cases, data can be collected at multiple sites, replicated to other locations, and then shared among collaborators. This type of replication scenario can be found, for example, still within the E-science domain in the gravitational wave community, where there are multiple detectors (two in the US and two in Europe) [39]. This proposed replication scheme can be fit into a peer-to-peer organization model as other non-E-science applications (e.g. media content distribution). Further this can be extended to resolve the replica management problem and consistency maintenance in distributed environment such as smart grid, Internet of Things (IoT) etc.

With growing alternative energy resources and poorly evolved power delivery in-

frastructure, smart grid are emerging as a feasible alternative. A smart grid is an electrical grid which includes a variety of operational and energy measures including smart meters, smart appliances, renewable energy resources, and energy efficient resources [145]. The dynamic resources and the distributed architecture offers processing of huge volume of data at several locations. The management and processing of large data is a crucial problem in distributed environment. Replication provide solution for managing large data produced by placing at different locations and processing them parallel at various locations. The Internet of Things (IoT) is a networking paradigm where interconnected, smart objects continuously generate huge volume of data and transmit it over the Internet. The Internet of Things presents a new set of challenges for database management systems, such as absorbing large volumes of data in real time or near real time, processing events as they flow rather than on a schedule of fixed intervals and dealing with significantly larger volumes of data than those often encountered in enterprise applications [146]. An IoT database should be fault tolerant and highly available. If an object in the database cluster is down, the database service should still be able to accept read and write requests. Distributed databases make copies, or replicas, of data and write them to multiple servers. In the case of the failure of one of the servers that stores a particular dataset, then one of the other servers storing a replica of this dataset can respond to a read query. Regarding write requests, if the server that would normally accept a request is down, another node in the server cluster can accept the request and forward it to the target server when it is back to an operational status.

Bibliography

[1] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint For a New Computing Infrastructure*. Elsevier, 2003.

[2] Sushant Goel and Rajkumar Buyya. Data Replication Strategies In Wide-Area Distributed Systems. In *Enterprise Service Computing: From Concept to Deployment*, pages 211–241. IGI Global, 2007.

[3] Houda Lamehamedi, Boleslaw Szymanski, Zujun Shentu, and Ewa Deelman. Data Replication Strategies in Grid Environments. In *Algorithms and Architectures for Parallel Processing, 2002. Proceedings. Fifth International Conference on*, pages 378–383. IEEE, 2002.

[4] Minakshi Memoria and Mukesh Yadav. On Fault Tolerance of Resources in Grid Environment. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2(1):pp–056, 2013.

[5] David G Cameron. *Replica Management and Optimisation for Data Grids, January 2005*. PhD thesis, Ph. D. Thesis, University of Glasgow.

[6] Types of grid.

[7] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The Data Grid: Towards an Architecture For the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*, 23(3):187–200, 2000.

[8] Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[9] Vassiliki Andronikou, Konstantinos Mamouras, Konstantinos Tserpes, Dimosthenis Kyriazis, and Theodora Varvarigou. Dynamic QoS-aware Data

Replication in Grid Environments Based on Data importance. *Future Generation Computer Systems*, 28(3):544–553, 2012.

[10] Tehmina Amjad, Muhammad Sher, and Ali Daud. A Survey of Dynamic Replication Strategies for Improving Data Availability in Data Grids. *Future Generation Computer Systems*, 28(2):337–349, 2012.

[11] Ann Chervenak, Robert Schuler, Carl Kesselman, Scott Koranda, and Brian Moe. Wide Area Data Replication for Scientific Collaborations. *International Journal of High Performance Computing and Networking*, 5(3):124–134, 2008.

[12] Nagamani H Shahapure and P Jayarekha. Replication: A Technique for Scalability in Cloud Computing. *International Journal of Computer Applications*, 122(5), 2015.

[13] Chang-Ming XING, Fang-Ai LIU, Lin YANG, and Wen LI. Education Resource Grid Model and Replica Creation Strategies. *Journal of Software*, 10:022, 2009.

[14] Priyanka Vashisht, Rajesh Kumar, and Anju Sharma. Efficient Dynamic Replication Algorithm Using Agent for Data Grid. *The Scientific World Journal*, 2014, 2014.

[15] Leyli Mohammad Khanli, Ayaz Isazadeh, and Tahmuras N Shishavan. PHFS: A Dynamic Replication Method, to Decrease Access Latency in the Multi-Tier Data Grid. *Future Generation Computer Systems*, 27(3):233–244, 2011.

[16] Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Şebnem Bora. Dynamic Replication Strategies in Data Grid Systems: A Survey. 2015.

[17] Nazanin Saadat and Amir Masoud Rahmani. PDDRA: A New Pre-Fetching Based Dynamic Data Replication Algorithm in data grids. *Future Generation Computer Systems*, 28(4):666–681, 2012.

[18] Chao-Tung Yang, Chun-Pin Fu, and Ching-Hsien Hsu. File Replication, Maintenance, and Consistency Management Services in Data Grids. *The Journal of Supercomputing*, 53(3):411–439, 2010.

[19] Replication and Synchronization Modes. `http://docs.oracle.com/cd/E19359-01/819-6148-10/chap2.html`. Accessed: 2010-09-30.

[20] Jim Gray, Pat Helland, Patrick O'Neil, and Dennis Shasha. The dangers of Replication and a Solution. *ACM SIGMOD Record*, 25(2):173–182, 1996.

[21] Gianni Pucciani. The Replica Consistency Problem in Data Grids. Technical report, 2008.

[22] Philip A Bernstein and Nathan Goodman. The Failure and Recovery Problem for Replicated Databases. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, pages 114–122. ACM, 1983.

[23] Andrea Domenici, Flavia Donno, Gianni Pucciani, and Heinz Stockinger. Relaxed Data Consistency With CONStanza. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, volume 1, pages 5–pp. IEEE, 2006.

[24] Leanne Guy, Peter Kunszt, Erwin Laure, Heinz Stockinger, and Kurt Stockinger. Replica Management in Data Grids. In *Global Grid Forum*, volume 5, pages 278–280, 2002.

[25] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing. *ACM Computing Surveys (CSUR)*, 38(1):3, 2006.

[26] M Ciglan. *Replica Delivery Optimization and Content Synchronization in Data Grids*. PhD thesis, Dissertation Thesis, Institute of Informatics, Slovak Academy of Sciences, 2008.

[27] Kavitha Ranganathan, Adriana Iamnitchi, and Ian Foster. Improving Data Availability Through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities. In *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, pages 376–376. IEEE, 2002.

[28] Mohammad Shorfuzzaman. *Placement of Replicas in Large-Scale Data Grid Environments*. PhD thesis, University of Manitoba (Canada), 2012.

[29] Donald Kossmann. The State of the Art in Distributed Query Processing. *ACM Computing Surveys (CSUR)*, 32(4):422–469, 2000.

[30] Bezalel Gavish and Olivia R Liu Sheng. Dynamic File Migration in Distributed Computer Systems. *Communications of the ACM*, 33(2):177–189, 1990.

[31] R Kingsy Grace and R Manimegalai. Dynamic Replica Placement and Selection Strategies in Data GridsA Comprehensive Survey. *Journal of Parallel and Distributed Computing*, 74(2):2099–2108, 2014.

[32] Karim Y Kabalan, Waleed W Smari, and Jacques Y Hakimian. Adaptive load sharing in heterogeneous systems: Policies, modifications, and simulation. *International Journal of Simulation, Systems, Science and Technology*, 3(1-2):89–100, 2002.

[33] Mohammad Shorfuzzaman, Rasit Eskicioglu, and Peter Graham. The State of the Art and Open Problems in Data Replication in Grid Environments. In *Handbook of Research on Scalable Computing Technologies*, pages 486–516. IGI Global, 2010.

[34] Priyanka Vashisht, Anju Sharma, and Rajesh Kumar. Strategies For Replica Consistency in Data Grid-A Comprehensive Survey. *Concurrency and Computation: Practice and Experience*, 29(4), 2017.

[35] Hongzhang Shan, Leonid Oliker, Warren Smith, and Rupak Biswas. Scheduling in heterogeneous grid environments: The effects of data migration. In *International Conference on Advanced Computing and Communication, Gujarat, India*, 2004.

[36] Charlie Catlett. Standards for Grid Computing: Global Grid Forum. *Journal of Grid Computing*, 1(1):3–7, 2003.

[37] Steve Tuecke, I Foster, and C Kesselman. The Open Grid Services Architecture. *The Grid: Blueprint for a new Computing Infrastructure*, pages 215–242, 2004.

[38] PJW Faulkner, LS Lowe, CLA Tan, PM Watkins, DS Bailey, TA Barrass, NH Brook, RJH Croft, MP Kelly, CK Mackay, et al. GridPP: Development of the UK Computing Grid for Particle Physics. *Journal of Physics G: Nuclear and Particle Physics*, 32(1):N1, 2005.

[39] Ewa Deelman, Carl Kesselman, Gaurang Mehta, Leila Meshkat, Laura Pearlman, Kent Blackburn, Phil Ehrens, Albert Lazzarini, Roy Williams, and Scott Koranda. GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists. In *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pages 225–234. IEEE, 2002.

[40] Ewa Deelman, Carl Kesselman, Gaurang Mehta, Leila Meshkat, Laura Pearlman, Kent Blackburn, Phil Ehrens, Albert Lazzarini, Roy Williams, and Scott Koranda. GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists. In *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pages 225–234. IEEE, 2002.

[41] Lauri Loebel-Carpenter, Lee Lueking, Carmenita Moore, Ruth Pordes, Julie Trumbo, Sinisa Veseli, Igor Terekhov, Matthew Vranicar, Stephen White, and Victoria White. SAM and the Particle Physics Data Grid. In *Proceedings of Computing in High-Energy and Nuclear Physics*. Beijing, China, 2001.

[42] Ian Bird, Kors Bos, N Brook, D Duellmann, C Eck, I Fisk, D Foster, B Gibbard, M Girone, C Grandi, et al. LHC Computing Grid. *Technical design report*, page 8, 2005.

[43] William E Johnston, Dennis Gannon, and Bill Nitzberg. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. In *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on*, pages 197–204. IEEE, 1999.

[44] Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, volume 1. Addison-Wesley Reading, 1999.

[45] Hosein Ebrahimi Ahangaran and Amir Masoud Rahmani. An innovative Approach of Data Grid Consistency Using Tree-Based Clustering. In *Computer*

and Network Technology (ICCNT), 2010 Second International Conference on, pages 258–261. IEEE, 2010.

[46] Andrea Domenici, Flavia Donno, Gianni Pucciani, Heinz Stockinger, and Kurt Stockinger. Replica Consistency in a Data Grid. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 534(1):24–28, 2004.

[47] Mohammed K Madi, Yuhanis Yusof, Hatim Mohamed Tahir, Khuzairi Mohd Zaini, and Suhaidi Hassan. Replica Maintenance Strategy for Data Grid. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(1-2):47–51, 2017.

[48] Dirk Dullmann, Wolfgang Hoschek, Javier Jaen-Martinez, Ben Segal, Asad Samar, Heinz Stockinger, and Kurt Stockinger. Models for Replica Synchronisation and Consistency in a Data Grid. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, pages 67–75. IEEE, 2001.

[49] Ghalem Belalem, Naima Belayachi, Radjaa Behidji, and Belabbes Yagoubi. Load Balancing to Increase the Consistency of Replicas in Data Grids. In *Technology Integration Advancements in Distributed Systems and Computing*, pages 42–57. IGI Global, 2012.

[50] Alan D Fekete and Krithi Ramamritham. Consistency Models for Replicated Data, Replication: Theory and Practice, 2010.

[51] Hongfei Guo, Per-Åke Larson, Raghu Ramakrishnan, and Jonathan Goldstein. Relaxed Currency and Consistency: How to Say Good Enough in SQL. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 815–826. ACM, 2004.

[52] Yasushi Saito and Marc Shapiro. Optimistic Replication. *ACM Computing Surveys (CSUR)*, 37(1):42–81, 2005.

[53] Synchronous and Asnchronous Replication. `http://ww2.paranet.com/blog/bid/133845/`

Difference-between-Synchronous-Asynchronous-Replication-Table. Accessed: 2015-04-24.

[54] Synchronous Replication: Procs, Cons and Myths. http://shadowbasesoftware.com/wp-content/uploads/2016/08/Synchronous-Replication-Pros-Cons-And-Myths.pdf. Accessed: 2015-04-24.

[55] Ghalem Belalem and Belabbes Yagoubi. Collaborative Negotiation to Resolve Conflicts among Replicas in Data Grids. In *Multimedia*. InTech, 2010.

[56] Jan-Jan Wu, Yi-Fang Lin, and Pangfeng Liu. Optimal Replica Placement in Hierarchical Data Grids with Locality Assurance. *Journal of Parallel and Distributed Computing*, 68(12):1517–1538, 2008.

[57] Xun-yi REN, Ru-chuan WANG, and KONG Qiang. Using Optorsim to Efficiently Simulate Replica Placement strategies. *The Journal of China Universities of Posts and Telecommunications*, 17(1):111–119, 2010.

[58] William H Bell, David G Cameron, A Paul Millar, Luigi Capozza, Kurt Stockinger, and Floriano Zini. Optorsim: A Grid Simulator For Studying Dynamic Data Replication Strategies. *International Journal of High Performance Computing Applications*, 17(4):403–416, 2003.

[59] Manghui Tu, Peng Li, I-Ling Yen, Bhavani M Thuraisingham, and Latifur Khan. Secure Data Objects Replication in Data Grid. *IEEE Transactions on Dependable and Secure Computing*, 7(1):50, 2010.

[60] P Krishna Reddy and Subhash Bhalla. Asynchronous Operations in Distributed Concurrency Control. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):721–733, 2003.

[61] Liangzhong Yin and Guohong Cao. DUP: Dynamic-Tree Based Update Propagation in Peer-to-Peer networks. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 258–259. IEEE, 2005.

[62] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent Hashing and Random Trees: Distributed Caching Protocols for relieving hot spots on the World Wide Web. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 654–663. ACM, 1997.

[63] Anwitaman Datta, Manfred Hauswirth, and Karl Aberer. Updates in Highly Unreliable, Replicated Peer-to-Peer Systems. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 76–85. IEEE, 2003.

[64] Mema Roussopoulos and Mary Baker. CUP: Controlled Update Propagation in Peer-to-Peer Networks. In *USENIX Annual Technical Conference, General Track*, pages 167–180, 2003.

[65] Xin Chen, Shansi Ren, Haining Wang, et al. SCOPE: Scalable Consistency Maintenance in Structured P2P Systems. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1502–1513. IEEE, 2005.

[66] Sung Chune Choi and Hee Yong Youn. Dynamic Hybrid Replication Effectively Combining Tree and Grid Topology. *The Journal of Supercomputing*, 59(3):1289–1311, 2012.

[67] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. *A Scalable Content-Addressable Network*, volume 31. ACM, 2001.

[68] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol For Internet Applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003.

[69] WK Lin, C Ye, and DM Chiu. Decentralized Replication Algorithms For Improving File Availability in P2P Networks. In *Quality of Service, 2007 Fifteenth IEEE International Workshop on*, pages 29–37. IEEE, 2007.

[70] Xianfu Meng, Yanli Wang, and Yalin Ding. An Optimized Strategy For

Update Path Selection in Unstructured P2P Networks. *Computer Networks*, 56(17):3744–3755, 2012.

[71] Dan-Wei Chen, Shu-Tao Zhou, Xun-Yi Ren, and KONG Qiang. Method for Replica Creation in Data Grids Based on Complex Networks. *The Journal of China Universities of Posts and Telecommunications*, 17(4):110–115, 2010.

[72] Ying Ding and Ying Lu. Automatic Data Placement and Replication in Grids. In *High Performance Computing (HiPC), 2009 International Conference on*, pages 30–39. IEEE, 2009.

[73] Stefano Ceri and Giuseppe Pelagatti. *Distributed Databases Principles and Systems*. McGraw-Hill, Inc., 1984.

[74] Chao-Tung Yang, Wen-Chi Tsai, Tsui-Ting Chen, and Ching-Hsien Hsu. A One-Way File Replica Consistency Model In Data Grids. In *Asia-Pacific Service Computing Conference, The 2nd IEEE*, pages 364–373. IEEE, 2007.

[75] Ghalem Belalem. A Hybrid Approach for Consistency Management in Large Scale Systems. In *Networking and Services, 2006. ICNS'06. International conference on*, pages 71–71. IEEE, 2006.

[76] José M Pérez, Félix García-Carballeira, Jesús Carretero, Alejandro Calderón, and Javier Fernández. Branch Replication Scheme: A New Model for Data Replication in Large Scale Data Grids. *Future Generation Computer Systems*, 26(1):12–20, 2010.

[77] Stefan Schmid and Roger Wattenhofer. Structuring Unstructured Peer-to-Peer Networks. In *International Conference on High-Performance Computing*, pages 432–442. Springer, 2007.

[78] Daniel Hughes, Geoff Coulson, and James Walkerdine. Free Riding on Gnutella Revisited: The Bell Tolls? *IEEE Distributed Systems Online*, 6(6), 2005.

[79] Nathaniel S Good and Aaron Krekelberg. Usability and Privacy: A Study of Kazaa P2P File-Sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 137–144. ACM, 2003.

[80] Suby Maria Jacob. Efficient Load Balancing in Peer-to-Peer Systems with Partial Knowledge of The System. *International Journal of Science and Research*, 2(5):1904–1907, 2013.

[81] Arijit Ganguly, Abhishek Agrawal, P Oscar Boykin, and RJ Figueiredo. Wow: Self-Organizing Wide Area Overlay Networks of Virtual Workstations. *Journal of Grid Computing*, 5(2):151–172, 2007.

[82] Xianfu Meng and Changyuan Zhang. An Ant Colony Model Based Replica Consistency Maintenance Strategy in Unstructured P2P Networks. *Computer Networks*, 62:1–11, 2014.

[83] Mohammed Radi. Improved Aggressive Update Propagation Technique in Cloud Data Storage. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 3(2):102–106, 2014.

[84] Po-Cheng Chen, Jyh-Biau Chang, Jia-Hao Yang, Yi-Chang Zhuang, and Ce-Kuen Shieh. Spigot: Fragment-Level File Sharing and Consistency on Grids. In *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*, pages 884–891. IEEE, 2009.

[85] AS Navaz, C Prabhadevi, and V Sangeetha. Data Grid concepts For Data Security in Distributed Computing. *arXiv preprint arXiv:1308.6058*, 2013.

[86] Xu PengZhi, Wu YongWei, Huang XiaoMeng, YANG GuangWen, and ZHENG WeiMin. Optimizing Write Operation on Replica in Data Grid. *SCIENCE CHINA Information Sciences*, 54(1):1–11, 2011.

[87] M Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Springer Science & Business Media, 2011.

[88] JoAnne Holliday, Robert Steinke, Divyakant Agrawal, and Amr El Abbadi. Epidemic Algorithms For Replicated Databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1218–1238, 2003.

[89] Henry Spencer and David Lawrence. *Managing Usenet*. O'Reilly & Associates, Inc., 1998.

[90] Nadège Pontisso, Philippe Quéinnec, and Gérard Padiou. Analysis of Distributed Multiperiodic Systems to Achieve Consistent Data Matching. *Concurrency and Computation: Practice and Experience*, 25(2):234–249, 2013.

[91] Yi Hu, Min Feng, and Laxmi N Bhuyan. A Balanced Consistency Maintenance Protocol for Structured P2P Systems. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[92] Xie Kun, ZHANG Da-Fang, Xie Gao-Gang, and Wen Ji-gang. A Trace Label Based Consistency Maintenance Algorithm in Unstructured P2P Systems. 2007.

[93] Jinjun Chen and Yun Yang. Multiple States Based Temporal Consistency for Dynamic Verification of Fixed-Time Constraints in Grid Workflow Systems. *Concurrency and Computation: Practice and Experience*, 19(7):965–982, 2007.

[94] OptorSim simulator. `https://sourceforge.net/projects/optorsim/`. Accessed: 2010-10-24.

[95] Kavitha Ranganathan and Ian Foster. Identifying Dynamic Replication Strategies for A High-Performance Data Grid. *Grid ComputingGrid 2001*, pages 75–86, 2001.

[96] Haiying Shen and Guoxin Liu. A Geographically Aware Poll-Based Distributed File Consistency Maintenance Method For P2P Systems. *IEEE Transactions on Parallel and Distributed Systems*, 24(11):2148–2159, 2013.

[97] Ping Zhang, Kunqing Xie, Xiujun Ma, Xiong Li, and Yixian Sun. A Replication Strategy Based on Swarm Intelligence in Spatial Data Grid. In *Geoinformatics, 2010 18th International Conference on*, pages 1–5. IEEE, 2010.

[98] Haifeng Yu and Amin Vahdat. Consistent and Automatic Replica Regeneration. *ACM Transactions on Storage (TOS)*, 1(1):3–37, 2005.

[99] Ruay-Shiung Chang and Jih-Sheng Chang. Adaptable Replica Consistency Service for Data Grids. In *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*, pages 646–651. IEEE, 2006.

[100] Jih-Sheng Chang and Ruay-Shiung Chang. An Innovative Replica Consistency Decision Model With Naive Bayesian Classifier in Data Grids. *Journal of the Chinese Institute of Engineers*, 31(7):1101–1111, 2008.

[101] Houssem-Eddine Chihoub. Self-Adaptive Cost-Efficient Consistency Management in the Cloud. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 2290–2293. IEEE, 2013.

[102] Tim Kraska, Martin Hentschel, Gustavo Alonso, and Donald Kossmann. Consistency Rationing in the Cloud: Pay Only When it Matters. *Proceedings of the VLDB Endowment*, 2(1):253–264, 2009.

[103] Ximei Wang, Shoubao Yang, Shuling Wang, Xianlong Niu, and Jing Xu. An Application-Based Adaptive Replica Consistency for Cloud Storage. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 13–17. IEEE, 2010.

[104] Jianwei Liao, Li Li, Huaidong Chen, and Xiaoyan Liu. Adaptive Replica Synchronization for Distributed File Systems. *IEEE Systems Journal*, 9(3):865–877, 2015.

[105] Yuri Breitbart and Henry F Korth. Replication and Consistency: Being Lazy Helps Sometimes. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems*, pages 173–184. ACM, 1997.

[106] Susan B Davidson, Hector Garcia-Molina, and Dale Skeen. Consistency in a Partitioned Network: A Survey. *ACM Computing Surveys (CSUR)*, 17(3):341–370, 1985.

[107] David K Gifford. Weighted Voting for Replicated Data. In *Proceedings of the Seventh ACM Symposium on Operating Systems Principles*, pages 150–162. ACM, 1979.

[108] Jemal H Abawajy and Mustafa Mat Deris. Data Replication Approach With Consistency Guarantee for Data Grid. *IEEE Transactions on Computers*, 63(12):2975–2987, 2014.

[109] Sushant Goel, Hema Sharda, and David Taniar. Replica Synchronisation in Grid Databases. *International Journal of Web and Grid Services*, 1(1):87–112, 2005.

[110] Mustafa Mat Deris, Jemal H Abawajy, and Ali Mamat. An Efficient Replicated Data Access Approach for Large-Scale Distributed Systems. *Future Generation Computer Systems*, 24(1):1–9, 2008.

[111] Mustafa Mat Deris, Jemal H Abawajy, David Taniar, and Ali Mamat. Managing Data Using Neighbour Replication on a Triangular-Grid Structure. *International Journal of High Performance Computing and Networking*, 6(1):56–65, 2009.

[112] Ching-Min Lin, Ge-Ming Chiu, and Cheng-Hong Cho. A New Quorum-Based Scheme for Managing Replicated Data in Distributed Systems. *IEEE Transactions on Computers*, 51(12):1442–1447, 2002.

[113] Keijirou Arai, Katsuya Tanaka, and Makoto Takizawa. Group Protocol for Quorum-Based Replication. In *Parallel and Distributed Systems, 2000. Proceedings. Seventh International Conference on*, pages 57–64. IEEE, 2000.

[114] Lei Gao, Michael Dahlin, Jiandan Zheng, Lorenzo Alvisi, and Arun Iyengar. Dual-Quorum: A Highly Available and Consistent Replication System for Edge Services. *IEEE Transactions on Dependable and Secure Computing*, 7(2):159–174, 2010.

[115] James Cowling, Daniel Myers, Barbara Liskov, Rodrigo Rodrigues, and Liuba Shrira. HQ Replication: A Hybrid Quorum Protocol For Byzantine Fault Tolerance. In *Proceedings of the 7th symposium on Operating systems Design and Implementation*, pages 177–190. USENIX Association, 2006.

[116] Rohaya Latip, Mohamed Othman, Azizol Abdullah, Hamidah Ibrahim, and Md Nasir Sulaiman. Quorum-Based Data Replication in Grid Environment. *International Journal of Computational Intelligence Systems*, 2(4):386–397, 2009.

[117] Ivan Frain, Abdelaziz M'zoughi, and Jean-Paul Bahsoun. How to Achieve High Throughput With Dynamic Tree-Structured Coterie. In *Parallel and*

Distributed Computing, 2006. ISPDC'06. The Fifth International Symposium on, pages 82–89. IEEE, 2006.

[118] Senhadji Sarra, Kateb Amar, and Belbachir Hafida. A Load Balancing Strategy for Replica Consistency Maintenance in Data Grid Systems. *Informatica*, 37(3):345, 2013.

[119] T Senthilnathan and T Purusothaman. A Multi-Agent Based Data Replication Mechanism for Mobile Grid. *American Journal of Applied Sciences*, 9(4):542, 2012.

[120] Huey-Ming Lee and Cheng-Hsiung Yang. A Distributed Backup Agent Based on Grid Computing Architecture. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 167–167. Springer, 2005.

[121] Ghalem Belalem. Economic Model for Consistency Management of Replicas in Data Grids with OptorSim Simulator. In *International Conference on Networks for Grid Applications*, pages 121–129. Springer, 2008.

[122] Ilir Fetai and Heiko Schuldt. Cost-Based Data Consistency in a Data-As-A-Service Cloud environment. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 526–533. IEEE, 2012.

[123] Elmootazbellah Nabil Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B Johnson. A Survey of Rollback-Recovery Protocols in Message-Passing Systems. *ACM Computing Surveys (CSUR)*, 34(3):375–408, 2002.

[124] Christopher Dabrowski. Reliability in Grid Computing Systems. *Concurrency and Computation: Practice and Experience*, 21(8):927–959, 2009.

[125] Aurelien Bouteiller, Thomas Herault, Géraud Krawezik, Pierre Lemarinier, and Franck Cappello. MPICH-V Project: A Multiprotocol Automatic Fault-Tolerant MPI. *International Journal of High Performance Computing Applications*, 20(3):319–333, 2006.

[126] Anand Natrajan, Marty Humphrey, and Andrew Grimshaw. Capacity and Capability Computing Using Legion. *Computational ScienceICCS 2001*, pages 273–283, 2001.

[127] Tom Goodale, Gabrielle Allen, Gerd Lanfermann, Joan Massó, Thomas Radke, Edward Seidel, and John Shalf. The Cactus Framework and Toolkit: Design and Applications. *High Performance Computing for Computational ScienceVECPAR 2002*, pages 15–36, 2003.

[128] Gerd Lanfermann, Gabrielle Allen, Thomas Radke, and Edward Seidel. Nomadic Migration: Fault Tolerance in a Disruptive Grid Environment. In *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, pages 280–280. IEEE, 2002.

[129] Adding High Availability. Replication to the Condor Central Manager.

[130] Edmond Lau and Samuel Madden. An Integrated Approach to Recovery and High Availability in An Updatable, Distributed Data Warehouse. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 703–714. VLDB Endowment, 2006.

[131] Javier García-García, Carlos Ordonez, and Predrag T Tosic. Efficiently Repairing and Measuring Replica Consistency in Distributed Databases. *Distributed and Parallel Databases*, 31(3):377–411, 2013.

[132] Haifeng Yu and Amin Vahdat. Design and Evaluation of a Continuous Consistency Model For Replicated Services. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation-Volume 4*, page 21. USENIX Association, 2000.

[133] Haifeng Yu and Amin Vahdat. The Costs and Limits of Availability For Replicated Services. *ACM Transactions on Computer Systems (TOCS)*, 24(1):70–113, 2006.

[134] Robin Qiu. *Enterprise Service Computing: From Concept to Deployment: From Concept to Deployment*. IGI Global, 2006.

[135] Yijun Lu, Ying Lu, and Hong Jiang. Adaptive Consistency Guarantees for Large-Scale Replicated Services. In *Networking, Architecture, and Storage, 2008. NAS'08. International Conference on*, pages 89–96. IEEE, 2008.

[136] NS2 Network Simulator. `http://www.isi.edu/nsnam/ns`. Accessed: 2012-02-3.

[137] Rajive Bagrodia, Richard Meyer, Mineo Takai, Yu-an Chen, Xiang Zeng, Jay Martin, and Ha Yoon Song. Parsec: A Parallel Simulation Environment For Complex Systems. *Computer*, 31(10):77–85, 1998.

[138] Fred Howell and Ross McNab. SimJava: A Discrete Event Simulation Library for Java. *Simulation Series*, 30:51–56, 1998.

[139] Catalin L Dumitrescu and Ian Foster. GangSim: A Simulator For Grid Scheduling Studies. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 2, pages 1151–1158. IEEE, 2005.

[140] Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: A Generic Framework For Large-Scale Distributed Experiments. In *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*, pages 126–131. IEEE, 2008.

[141] Rajkumar Buyya and Manzur Murshed. Gridsim: A Toolkit For the Modeling and Simulation of Distributed Resource Management and Scheduling For Grid Computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1175–1220, 2002.

[142] Krzysztof Kurowski, Jarek Nabrzyski, Ariel Oleksiak, and Jan Weglarz. Multicriteria Aspects of Grid Resource Management. In *Grid Resource Management*, pages 271–293. Kluwer Academic Publishers, 2004.

[143] Ann Chervenak, Ewa Deelman, Ian Foster, Leanne Guy, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Bob Schwartzkopf, et al. Giggle: A Framework For Constructing Scalable Replica Location Services. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 1–17. IEEE Computer Society Press, 2002.

[144] Sang-Min Park, Jai-Hoon Kim, Young-Bae Ko, and Won-Sik Yoon. Dynamic Data Grid Replication Strategy Based on Internet Hierarchy. *Grid and Cooperative Computing*, pages 838–846, 2004.

[145] Pierluigi Siano. Demand response and smart gridsa survey. *Renewable and Sustainable Energy Reviews*, 30:461–478, 2014.

[146] Mervat Abu-Elkheir, Mohammad Hayajneh, and Najah Abu Ali. Data Management for the Internet of Things: Design Primitives and Solution. *Sensors*, 13(11):15582–15612, 2013.