Open Archive TOULOUSE Archive Ouverte (OATAO)
OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : http://oatao.univ-toulouse.fr/
Eprints ID : 14981

To link to this article : DOI :10.1007/s11227-015-1508-7
URL : http://dx.doi.org/10.1007/s11227-015-1508-7

To cite this version : Tos, Uras and Mokadem, Riad and Hameurlain, Abdelkader and Tolga, Ayav and Bora, Sebnem *Dynamic replication strategies in data grid systems: A survey.* (2015) Journal of Supercomputing, vol. 71 (n° 11). pp. 4116-4140. ISSN 0920-8542

Any correspondance concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr

Dynamic replication strategies in data grid systems: a survey

Uras Tos[1,2,3] · **Riad Mokadem**[1] · **Abdelkader Hameurlain**[1] · **Tolga Ayav**[2] · **Sebnem Bora**[3]

Abstract In data grid systems, data replication aims to increase availability, fault tolerance, load balancing and scalability while reducing bandwidth consumption, and job execution time. Several classification schemes for data replication were proposed in the literature, (i) static vs. dynamic, (ii) centralized vs. decentralized, (iii) push vs. pull, and (iv) objective function based. Dynamic data replication is a form of data replication that is performed with respect to the changing conditions of the grid environment. In this paper, we present a survey of recent dynamic data replication strategies. We study and classify these strategies by taking the target data grid architecture as the sole classifier. We discuss the key points of the studied strategies and provide feature comparison of them according to important metrics. Furthermore, the impact of data grid architecture on dynamic replication performance is investigated in a simulation study. Finally, some important issues and open research problems in the area are pointed out.

✉ Uras Tos
urastos@gmail.com; tos@irit.fr

Riad Mokadem
mokadem@irit.fr

Abdelkader Hameurlain
hameurlain@irit.fr

Tolga Ayav
tolgaayav@iyte.edu.tr

Sebnem Bora
sebnem.bora@ege.edu.tr

1 Institut de Recherche en Informatique de Toulouse (IRIT), Paul Sabatier University, 118 Route de Narbonne, 31062 Toulouse, France

2 Department of Computer Engineering, Izmir Institute of Technology, 35430 Urla, Izmir, Turkey

3 Department of Computer Engineering, Ege University, 35100 Bornova, Izmir, Turkey

Keywords Data grid · Dynamic replication · Data grid architecture · Performance

1 Introduction

Data grid provides a scalable foundation for computationally intensive applications, in which large amounts of data are processed [15]. Many research areas, including high-energy physics [25,50,53], astronomy [19], biology [34], and climate science [14] employ data grid based infrastructures for their computational needs. Experiments in these scientific research areas create very large amounts of data that are often measured in petabytes [25,56]. Frequent access to these data sets would strain network links, overload remote data stores, and affect computational performance. On the other hand, placing local copies at each node is costly and not realistic. As a result, the placement of data plays an important role in data grid systems.

Dealing with data placement problem, data replication is a common solution in data grid systems. It consists of strategically placing copies of data to increase availability, access performance, reliability, and fault-tolerance, as well as to reduce bandwidth usage, and job completion times. Many replication strategies were proposed [11,42, 45,54] to satisfy these constraints.

Any replication strategy should address at least three challenges [45]: (i) which files should be replicated? It is generally not feasible to replicate every file; therefore, establishing criteria on choosing files to be replicated is important. (ii) When should the replication of a file occur? Finding a good balance on when to replicate is also crucial as replicating too early might be wasteful on storage space, while replicating too late may not yield the full benefits of replication. (iii) Where should the replica of a file be placed? Placing replicas closer to the clients with the most access requests plays an important role in overall performance of replication strategy. Optimal replica placement is an NP-complete problem [55]. Therefore, each replication strategy approaches the issue as an optimization problem and answers these three questions differently.

There are many ways to look at data replication, e.g. fault-tolerance, security, load balancing points of view. However, from the most general perspective, data replication strategies can be divided into two categories, namely static [13,18,22,32] and dynamic [11,41,42,54] replication. In this paper, we are focusing on dynamic replication strategies, where all replication decisions are made adaptively while the system is in operation. In addition, allowing updates on replicas creates serious consistency problems and introduces a significant amount of management overhead. Consequently, most of the work done on dynamic replication focus only on read-only access performance [13], which is also the scope of this paper.

Different classification criteria are often intertwined in recent surveys on data replication. Some papers deal with the already cited static vs. dynamic classification [13,18], while some other works deal with centralized vs. decentralized replication management [4,20,33], push-based vs. pull-based replication [16,20,41,52], and objective function-based classification [40].

In general sense, most dynamic replication strategies are developed for a target data grid architecture. In this context, important issues as replica placement and

selection depend significantly on data grid architecture. Few papers consider target data grid architecture as a classification criteria. Kingsy Grace and Manimegalai [28] discuss various replica placement and selection strategies with a focus on decentralized replication management. In their survey, architecture is not the only classifier, but instead it is used alongside objective function, and load balancing criteria in a mixed classification. In another paper, Amjad et al. [4] show interest in another replication classification. In their classification, both the centralized/decentralized aspect of replication management and storage assumption are considered as the most important criteria, while the main focus is on improved data availability. Furthermore, performance evaluation is not present in these recent surveys. We believe it is important to point out the impact of the classification criteria in an experimental analysis.

In this paper, we provide a new classification of dynamic replication strategies with respect to the data grid architecture only. We regard data grid architecture as a system that consists of components and their relationships [26]. Our classification, following a brief description of each architecture, discusses the current state of the art on dynamic data replication strategies. Furthermore, we evaluate and discuss the impact of data grid architecture on data replication by means of a simulation study. In addition, we show how the features of an architecture may contribute to which types of benefits in terms of performance.

We also study some factors that impact the performance. Considering the dynamic nature of the data grid, nodes can join or leave the data grid at any time. As a result, the number of active nodes at any given time varies. Dynamic replication strategies must consider these dynamic aspects of the data grid, as well as taking advantage of file access patterns, making realistic storage assumptions, managing available storage space in the nodes, and calculating costs of replication. Indeed, for any dynamic replication strategy, benefits of replication should always outweigh the overhead. The simulation study presented in this paper enables us to investigate the effect of data grid architecture on replication performance.

The organization of this paper is as follows: Sect. 2 discusses existing classifications and how dynamic replication can be classified with respect to different criteria. Section 3 describes our classification and analyzes recent work on dynamic replication. Section 4 evaluates the performance of selected strategies on different architectures to investigate the impact of architecture on data replication performance. Section 5 discusses the important issues and some open problems for dynamic replication strategies. Section 6 presents a conclusion to the paper.

2 Existing classifications

Replication strategies vastly vary, as all have different implementations. While every replication strategy is different, they may have common features with respect to certain aspects. Therefore, it is a sensible approach to classify replication strategies, as it helps building a coherent and organized foundation for studying them. In this section, existing classification schemes for dynamic replication are discussed.

2.1 Static versus dynamic replication

In most general sense, replication strategies can be classified into two groups, namely static and dynamic replication. In static replication, all decisions regarding the replication strategy are made before the system is operational and not changed during operation [13,18,22,32]. On the other hand, in dynamic replication, what, when, and where to replicate are decided as a response to the changing trends of the data grid [11,41,42,54].

In a non-changing grid environment, where nodes do not join or leave the grid and file access patterns are not varied, static replication might be a good choice. Compared to dynamic replication, static replication does not have the overhead caused by replication decision and management. On the other hand, when a replication scenario needs to be periodically reconfigured according to dynamic grid properties, it causes significant administrative overhead and affects scalability and optimal resource use of the system. In a dynamic environment where nodes are free to join or leave, and file access patterns change over time, dynamic replication excels static replication due to its nature of adaptability.

2.2 Centralized versus decentralized replication

Data replication involves many tasks, including but not limited to, choosing files to replicate, and deciding where to place replicas. Each task requires having a priori information about that particular state of the data grid environment. Which party or parties will collect this information, process it, and take actions regarding replication is in the scope of this classification scheme [4,20,33].

Centralized replication strategies contain a central authority to control all aspects of data replication. All metrics are either collected by or propagated to this central authority. Replication decisions are given by this point of control and all the other nodes report to it. In contrast, decentralized approach encourages no central control mechanism to exist in the system. Nodes themselves decide on how replication will occur. With no central control, no single node can hold complete information about the entirety of the data grid. In a decentralized replication management strategy, coordination of a replication event is usually performed with the collaboration of a number of nodes. As the system scales up, the inter-node communication overhead should not increase to a point that surpasses the benefits of the replication. Similarly, if a centralized replication management is chosen, the capabilities of the central replica manager should not cause bottlenecks if the system scales up in the future.

Each approach has its advantages and drawbacks. Centralized replication is easier to implement and generally more efficient, as a single entity is responsible for all the decisions and has knowledge about every aspect of the data grid. On the other hand, central authority is also a point of failure and thus is not ideal for reliability and fault-tolerance. Decentralized replication is good for reliability as there is no single point of failure in the system and the system can still behave predictably even a number of nodes are lost. However, having no central control and nodes acting on incomplete information about the state of the system may yield non-optimal results, e.g. excessive replication [46].

2.3 Push versus pull based replication

In any replication strategy, if we take a close look at the replication event of any particular file, there are two main actors involved. The former is the server that holds the file in its storage elements, and the latter is the client that needs that file in its local storage. Push- vs. pull-based classification is focusing on which of these two actors triggers the replication event [16,20,41,52].

In push based replication, the replication event is triggered by the originator of the file, as the server pushes the file to the requestor node. Servers receive requests from a number of clients, thus they require enough information about the state of the system to be able to trigger replication events. Therefore, push based replication is often proactive.

In pull-based replication, the replication event is triggered by the requestor node, as the client pulls the file from the server. Compared to push-based strategies, pull-based replication can be regarded as a reactive approach as the replication event is realized on-demand. Client-side caching is also regarded as pull replication due to the fact that in this form of caching, clients decide to temporarily store a file in their local storage [52].

2.4 Objective function based classification

When approaching data replication as an optimization problem, observing an objective function can be expected. Considering that each data replication strategy aims to minimize or maximize some objective, it is possible to make a classification with regard to the definition of this objective function [40].

A popular approach to define an objective function is to set data locality goals [45, 54]. In this approach, the primary aim is to place replicas as close to the requestors as possible, preferably in the local storage. There is also the possibility to extend this locality goal with some other objective. For example, instead of increasing the locality of all requested files, some strategies use heuristics to selectively increase the locality of just the popular files [51].

Cost model based objective functions enable the replication decision to take a number of parameters into account [5,36,44]. In these works, replication decision is generally given according to the output of a mathematical model. These models can include many parameters including collective file access statistics, bandwidth availability, replica sizes, etc.

Market-like mechanisms and economic behaviors are also evaluated [8,23]. In the economic models, files are treated as tradable goods on the market. During a replication event, clients tend to buy files from remote sites that offer the lowest price and remote sites try to sell their files for the greatest profit.

3 Classification of dynamic replication strategies

In this section, we classify the existing dynamic replication strategies with respect to target data grid architecture. Each different data grid architecture has different properties, and these properties necessitate different strategies concerning data replication.

Table 1 Dynamic replication strategies for multi-tier architecture

Replication strategy	Ranganathan et al. [45]	Tang et al. [54]	Shorfuzzaman et al. [51]	Abdurrab et al. [2]	Khanli et al. [27]
Features					
Architecture	Multi-tier	Multi-tier	Multi-tier	Multi-tier	Multi-tier
Replication decision	Decentr.	Centr.	Centr.	Decentr.	Decentr.
Storage assumption	Limited	Limited	Limited	Limited	Limited
Objective function	Locality	Locality	Locality	Locality	Locality
Measured metrics					
Availability	No	No	No	No	No
Number of replicas	No	No	No	No	No
Response time	Yes	Yes	No	No	Yes
Request success rate	No	No	No	No	No
Total execution time	No	No	Yes	Yes	No
Storage usage	No	No	Yes	No	No
Network usage	Yes	Yes	Yes	Yes	No
Replication frequency	No	Yes	No	No	No

Tables 1, 2, 3, 4, 5 and 6 provide a summary of studied replication strategies in each respective subsection.

3.1 Dynamic replication strategies for hierarchical architecture

Hierarchical architectures assume a structured network generally in the form of a tree or a star. Whether the replication strategy is developed for a multi-tier hierarchy, or it takes advantage of some hierarchy with network-level locality, all hierarchical approaches are studied in this subsection.

3.1.1 Dynamic replication strategies for multi-tier architecture

Multi-tier architectures follow the data grid model of GriPhyN project [45]. It is hierarchical in nature, and it has a well-defined, strict structure. On the other hand, due to this strict organizational structure, multi-tier architectures are not very flexible to allow arbitrary addition of removal of nodes. Multi-tier data grid is organized in four tiers as Fig. 1 shows. Tier 0 denotes the source, e.g. CERN, where the data are generated and master copies are stored. Tier 1 represents national centers, Tier 2 the regional centers, Tier 3 consists of work groups, and Tier 4 contains desktop computers. In this model, generally, the storage capacity increases from bottom to upper levels of the data grid.

Ranganathan and Foster [45] propose six dynamic replication strategies for multi-tier data grid. These strategies are *No Replication or Caching*, *Best Client*, *Cascading Replication*, *Plain Caching*, *Caching plus Cascading Replication*, and *Fast Spread*. No Replication or Caching is implemented as a base case for comparing other strategies

Table 2 Dynamic replication strategies for bandwidth hierarchy architecture

Replication strategy	Park et al. [42]	Horri et al. [24]	Sashi et al. [49]	Mansouri et al. [35]	Mansouri et al. [36]	Mansouri et al. [37]
Features						
Architecture	BW hierarchy	BW hierarchy	BW hierarchy	BW hierarchy	BW hierarchy	BW hierarchy
Repli. decision	Decentr.	Centr.	Centr.	Decentr.	Decentr.	Decentr.
Stor. assump.	Limited	Limited	Limited	Limited	Limited	Limited
Obj. function	Locality	Locality	Locality	Locality	Cost model	Locality
Measured metrics						
Availability	No	No	No	No	No	No
Num. of replicas	No	No	No	No	No	No
Response time	No	No	No	No	No	No
Req. success rate	No	No	No	No	No	No
Tot. execution time	Yes	Yes	Yes	Yes	Yes	Yes
Storage usage	No	No	Yes	No	No	Yes
Network usage	No	No	Yes	No	No	Yes
Replication frequency	No	No	No	No	No	No

Table 3 Dynamic replication strategies for other hierarchical architectures

Replication strategy	Chang et al. [11]	Perez et al. [43]	Zhao et al. [58]	Lee et al. [30]	Saadat et al. [48]
Features					
Architecture	Hier.	Hier.	Hier.	Hier.	Hier.
Replication decision	Centralized	Centralized	Centralized	Centralized	Centralized
Storage assumption	Limited	Limited	Limited	Limited	Limited
Objective function	Locality	Locality	Locality	Locality	Locality
Measured metrics					
Availability	No	No	No	Yes	No
Number of replicas	No	No	Yes	No	Yes
Response time	No	No	No	No	No
Request success rate	No	No	No	No	Yes
Total execution time	Yes	No	Yes	No	Yes
Storage usage	Yes	No	No	No	Yes
Network usage	Yes	Yes	Yes	Yes	Yes
Replication frequency	No	No	No	No	No

to a no-replication scenario. In Best Client strategy, access history records are kept for each file on the grid. When a certain threshold is reached, the file is replicated only on the client that generates most requests. Cascading Replication introduces a tiered replication strategy, in which when a threshold for a file is exceeded at the root node,

Table 4 Dynamic replication strategies for P2P architecture

Replication strategy	Ranganathan et al. [46]	Bell et al. [8]	Abdullah et al. [1]	Challal et al. [10]	Chettaoui et al. [17]
Features					
Architecture	P2P	P2P	P2P	P2P	P2P
Replication decision	Decentr.	Decentr.	Decentr.	Decentr.	Decentr.
Storage assumption	Unlimited	Limited	Unlimited	Limited	Limited
Objective function	Locality	Economic behavior	Locality	Locality	Locality
Measured metrics					
Availability	Yes	No	No	No	No
Number of replicas	Yes	No	Yes	Yes	No
Response time	No	No	Yes	No	Yes
Request success rate	No	No	Yes	No	No
Total execution time	No	Yes	No	Yes	No
Storage usage	No	No	No	Yes	No
Network usage	No	No	No	Yes	Yes
Replication frequency	No	No	No	No	No

Table 5 Dynamic replication strategies for hybrid architecture

Replication strategy	Lamehamedi et al. [29]	Rasool et al. [47]
Features		
Architecture	Hybrid	Hybrid
Replication decision	Decentralized	Centralized
Storage assumption	Limited	Limited
Objective function	Cost model	Locality
Measured metrics		
Availability	No	No
Number of replicas	No	Yes
Response time	Yes	Yes
Request success rate	No	No
Total execution time	No	No
Storage usage	No	No
Network usage	No	Yes
Replication frequency	No	No

a replica is placed at the level on the path towards the best client, progressively. In Plain Caching, the client requests a file and stores it locally. Caching plus Cascading Replication combines Cascading Replication and Plain Caching. Fast Spread is the final strategy in which, upon client file requests, a replica of the file is placed on each tier on the path to the client. Popularity and file age are used as parameters to select

Table 6 Dynamic replication strategies for general graph architecture

Replication strategy	Rahman et al. [44]	Lei et al. [31]	Chen et al. [12]	Bsoul et al. [9]	Andronikou et al. [5]
Features					
Architecture	General graph	General graph	General graph	General graph	Not mentioned
Replication decision	Decentr.	Decentr.	Decentr.	Centr.	Centr.
Storage assumption	Limited	Limited	Limited	Limited	Limited
Objective function	Cost model	Locality	Locality	Locality	Cost model
Measured metrics					
Availability	No	Yes	No	No	No
Number of replicas	No	No	No	No	No
Response time	Yes	No	No	Yes	No
Request success rate	No	No	No	No	No
Total execution time	No	Yes	Yes	No	Yes
Storage usage	No	No	Yes	No	No
Network usage	No	No	No	Yes	No
Replication frequency	No	No	No	No	No

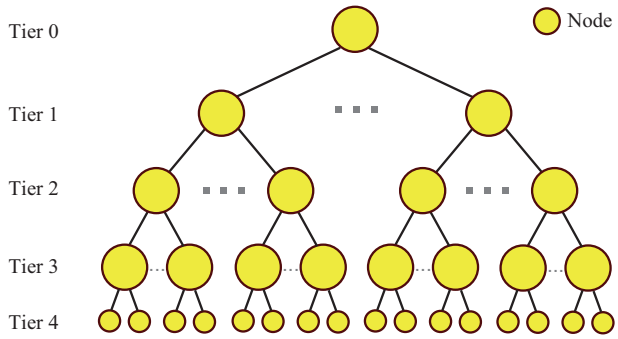


Fig. 1 Multi-tier architecture

files for the replica replacement approach. In simulations with three different access patterns, they show that Best Client strategy performs worst. Fast Spread works better with random data access patterns and Cascading Replication performs better when locality exists in data access patterns.

Two dynamic replication strategies, *Simple Bottom-Up* (SBU) and *Aggregate Bottom-Up* (ABU) were proposed by Tang et al. [54] to reduce the average response time. Popular files are identified by analyzing the file access history. When an access threshold is exceeded, SBU places replicas close to the nodes that request files with higher frequencies. ABU, on the other hand, calculates the aggregate access records for each sibling of a node and passes this information to higher tiers until

the root node is reached. At each level, replication decision is given when aggregate access values pass a predefined threshold. Both strategies employ *Least Recently Used* (LRU) [6] replica replacement approach. In the performance evaluation, ABU gives the best average response time and average bandwidth cost among studied strategies.

Shorfuzzaman et al. [51] propose two dynamic replication strategies for multi-tier data grid, *Popularity Based Replica Placement* (PBRP), and its adaptive counterpart, *Adaptive-PBRP* (APBRP). PBRP aims to balance storage utilization and access latency trade-off by replicating files based on file popularity. The replication strategy is run periodically in a way that access records are aggregated bottom-up and replica placement is done in a top-down manner. APBRP improves PBRP by introducing an adaptive access rate threshold. In simulations, APBRP shows improvement over PBRP while both strategies perform better than Best Client, Cascading, Fast Spread, and ABU in terms of job execution time, average bandwidth use, and storage use.

Abdurrab and Xin [2] present a replication strategy, called *File Reunion* (FIRE), which takes advantage of the correlation between file requests. FIRE assumes that there is a strong correlation between a group of jobs and a set of files. Based on this assumption, FIRE aims to reunite the file set onto the sites by means of replication. Replication is performed when a file is not locally available, and there is enough storage space to store it. If there is not enough storage space, a file with a lower group correlation degree is removed before replicating the new file. In a simulation scenario, FIRE performed better than *Least Frequently Used* (LFU) [6] and LRU replication strategies.

Khanli et al. [27] mention that most of the recent work on data replication focus on temporal and geographical locality, but not on spatial (file) locality. They propose *Predictive Hierarchical Fast Spread* (PHFS) as an improvement over Fast Spread. PHFS works in three stages. In monitoring stage, file access records from all clients are collected in a log file. In analyzing stage, data mining techniques are used to discover the relationships between files. For a file A, any file B with a relationship greater than a threshold is considered in the *predictive working set* (PWS) of A. In the final stage replication configuration is applied according to the calculated PWSs. They left the simulation for future work but showed on an example that PHFS improved access latency over Fast Spread.

3.1.2 Dynamic replication strategies for bandwidth hierarchy architecture

Park et al. [42] propose a dynamic replication strategy that takes advantage of the bandwidth hierarchy in the data grid. In their approach, they present that bandwidth between regions, e.g. countries, are narrower compared to bandwidth available inside a region. Their strategy, *Bandwidth Hierarchy Replication* (BHR) takes advantage of this relationship between regions to introduce a new type data locality, namely network-level locality as depicted in Fig. 2. BHR replicates popular files as many times as possible within a region, where intra-region bandwidth is abundant. In simulations, BHR performs better than delete LRU and delete oldest replication when narrow inter-region bandwidth or low node storage space exists. However, as the inter-region

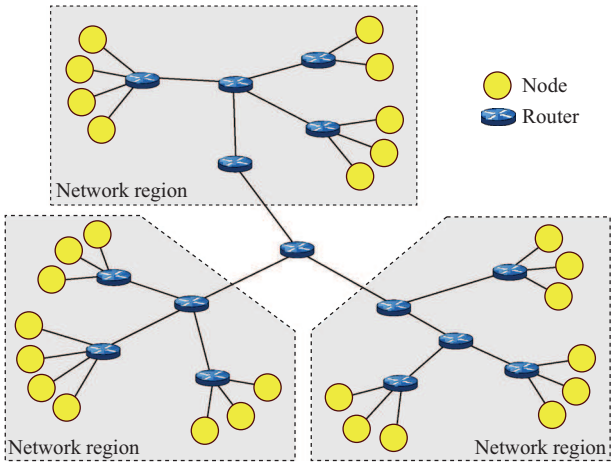


Fig. 2 A hierarchical architecture based on network-level locality

bandwidth or available storage space of the nodes increase, BHR performs similarly to the traditional strategies.

Horri et al. [24] presented *3-Level Hierarchical Algorithm* (3LHA) for network level hierarchy. The proposed strategy targets a hierarchical architecture that consists of three levels. The first level consists of regions, i.e. having low bandwidth availability. Levels two and three represent local area networks (LAN) and clients in the LANs, respectively. When a client accesses a file, if it has enough storage, the file is replicated. However, if files needed to be deleted before the replication, first, the local files that also already exist on the LAN are chosen for deletion. Then, the local files that already exist in the region are selected, and if there is still not enough space available, other local files are deleted. They compared their strategy with BHR and LRU and showed that the proposed strategy performs better in terms of mean job time.

An improved implementation of BHR was proposed by Sashi and Thanamani [49] as *Modified BHR* algorithm. In their strategy, the data are generated at the master site and replicated to region headers before any jobs are scheduled on the grid. They assume that the files accessed by a node will also be accessed by nearby nodes and popular files will be accessed more frequently. Replicas are only placed in the region header and the node that makes the most requests. The access records are kept in the region header and least frequently used replicas are chosen as the deletion strategy. Modified BHR algorithm is compared with no replication, LFU, LRU, and BHR in a simulation study and the results show improved mean job time than to other strategies.

Mansouri and Dastghaibyfard [35] extended 3LHA further and proposed *Dynamic Hierarchical Replication* (DHR) algorithm. They emphasize that 3LHA places replicas in all of the requestor sites. On the other hand, DHR creates a per-region ordered list of sites with respect to the number of accesses to a file. The site that is at the top of the order is chosen to place the new replica. By placing replicas at best sites, DHR aims to lower storage cost and mean job execution time. They compare the effectiveness of DHR against no replication, LFU, LRU, BHR, and 3LHA. The results show that

DHR shows better job execution times compared to other studied strategies, especially when grid sites have smaller storage space.

In another paper, Mansouri and Dastghaibyfard [36] added economic cost model calculation to DHR and presented *Enhanced Dynamic Hierarchical Replication* (EDHR). By predicting future economic value of files, they made better assessment of which replicas will not be beneficial and get deleted, and which files will be beneficial and get replicated. Simulations indicate that EDHR yields even better mean job times than DHR.

Mansouri et al. [37] also present *Modified Dynamic Hierarchical Replication Algorithm* (MDHRA), which is another extension of DHR strategy. In MDHRA, replica replacement decision mechanism is altered to take last request time, number of accesses, and size of the replica into account. They note that the new approach improves the availability of valuable replicas. Simulations show that, compared to DHR and other studied strategies, MDHRA performs better in terms of mean job completion time and effective network usage. However, performance comparisons do not include EDHR.

3.1.3 Dynamic replication strategies for other hierarchical architectures

An access-weight based dynamic replication strategy is proposed by Chang and Chang [11]. Their work, *Latest Access Largest Weight* (LALW), defines a strategy for measuring popularity of files on the grid, calculating the required number of replicas, and determining sites for replica placement. In the presented strategy, clients are connected to cluster headers that manage all replication decisions. File access history records are aggregated to the cluster headers and system-wide popularity of files are calculated. Recently accessed files have larger weights, and the replica placement is based on weighted access frequencies. In simulations, LALW shows similar total job execution times compared to LFU while consuming less storage space and having more effective bandwidth usage.

Perez et al. [43] present *Branch Replication Scheme* (BRS) that has three key features: (i) sub-replica creation optimizes storage usage, (ii) data access performance is increased via parallel I/O, and (iii) consistency on updates are maintained to allow replica modification. In BRS, files are divided into several disjoint sub-replicas that are placed on different nodes. With this approach, BRS aims to create high levels of fault-tolerance without increasing the storage use. A simulator based on Omnet++ was developed for performance evaluation. In the simulations, BRS is compared with hierarchical replication strategy and shows improved data access performance on both read and write operations.

Zhao et al. [58] propose a replication strategy, called *Dynamic Optimal Replication Strategy* (DORS). A file is replicated when the number of replicas of that particular file is less than a threshold. This threshold is the ratio of total grid capacity to the total size of the files on the grid. For the replica replacement policy, they designed a model that calculates value of the replicas. When a shortage of storage space is detected, replicas with the lowest value are replaced. Replica value is dependent on access frequency and access cost of the replicas. They compared DORS with LFU

and LRU in a simulation environment. The results show that DORS performs better in terms of mean job execution time and effective network use metrics.

Saadat and Rahmani [48] propose *Pre-fetching Based Dynamic Data Replication Algorithm* (PDDRA) with the assumption that members of a virtual organization (VO) have similar interests in files. PDDRA predicts the future accesses of files and pre-replicates them before the requests are placed. In the algorithm, file access records of VOs are collected and logged. When a file request is put forward for a file A, PDDRA scans the logs and determines which files follow file A, and which of the follower files has the greatest number of accesses. Using five different access patterns, they compared PDDRA with six existing strategies. PDDRA shows better performance than other strategies in terms of mean job execution time and effective network usage under all simulated access patterns.

Lee et al. [30] developed an adaptive replication strategy, called *Popular File Replicate First* (PFRF). Their algorithm runs periodically in four stages: (i) file access records are aggregated by header nodes to the replication manager. (ii) Popularity of each file is calculated. (iii) Top 20 % popular files are chosen to be replicated for every grid site, and (iv) files are replicated to destination sites from the closest site that holds the required files. In case of a storage shortage, less popular files are deleted prior to replication. In a simulation scenario using five access patterns, PFRF shows better performance on average job turnaround time, average data availability, and bandwidth cost ratio metrics, compared to other strategies.

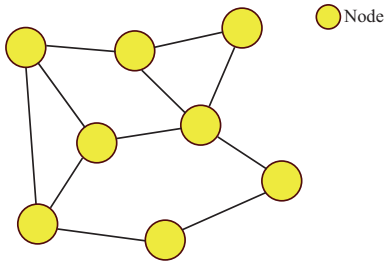
Meroufel and Belalem [39] propose a replication strategy, called *Placement Dynamic* (PD). The aim of PD is to minimize the number of replicas to ensure certain degree of availability without degrading performance. In the strategy, placement of the replicas and the failures in the system are taken into account. If a failure suspicion is observed, the data are moved to other nodes in the system to maintain the availability level. Authors compared PD with random replication approach via simulations performed with FTSim. The results show that PD demonstrates better recovery times compared to random replication, and unlike random replication, PD can keep the desired availability.

3.2 Dynamic replication strategies for peer-to-peer architecture

In P2P architectures, nodes act in an autonomous way, without intervention of a central authority (Fig. 3). Nodes normally possess enough functionality to be both servers and clients at the same time. This decentralized structure allows for even higher volatility than other architectures, as nodes can connect to any part of the grid and leave without notice. Replication strategies for P2P architecture are developed by keeping this highly dynamic nature of P2P grids in mind [57].

Ranganathan et al. [46] propose a replication strategy which is dynamic and model driven. Their model takes single system stability, transfer time between nodes, storage cost of files, and accuracy of replica location mechanism to calculate the required number of replicas to achieve a desired availability level. They show with a simulation scenario that their replication strategy performs better than static replication. They also report that their strategy accurately predicts the number of replicas in the system.

Fig. 3 Peer-to-peer architecture



However, they note that nodes act on incomplete information and this sometimes lead to unnecessary replication.

Bell et al. [8] presented an economy-based data replication strategy for P2P data grids. In the proposed strategy, data grid is treated as a marketplace, where files represent goods that are traded by the optimization agents in the system. Computing elements purchase files and aim to minimize the purchasing cost. Similarly, storage elements try to maximize their profits and make investments based on file access predictions to increase revenue. In simulations, the proposed model is compared with LRU strategy. The results indicate that the proposed model reduces total job execution times in sequential file access; however, LRU performed better in Gaussian random walk distribution.

Abdullah et al. [1] propose two dynamic replication strategies, *Path and Requestor Node Placement Strategy*, and *N-hop Distance Node Placement Strategy*. Path and Requestor Node Placement Strategy replicates files on all nodes on the path to the requestor node, including the requestor node. In N-hop Distance Node Placement Strategy, the replicas are placed on all neighbors of the provider node with a distance of N. In simulations, proposed strategies increase availability and decrease response time with the expense of using more bandwidth.

Challal and Bouabana-Tebibel [10] presented a priori data replication strategy for P2P data grid systems. Their strategy supplements dynamic replication by finding optimal nodes to place initial replicas before the jobs are started. Maximizing the distance between identical replicas and minimizing distances between different replicas, they increase availability and ensure that each node has replicas of different file in its vicinity. In their simulations, a priori replica placement strategy is compared with random initial replica placement and no initial replica placement. The proposed strategy improves job completion times and reduces file transfer times without increasing bandwidth and storage costs.

Chettaoui and Charrada [17] propose DPRSKP, *Decentralized Periodic Replication Strategy based on Knapsack Problem*. Two main features of the proposed strategy are the limited storage assumption for grid sites and dynamicity of the data grid, i.e. the number of grid sites that exist at any given time. DPRSKP selects what to replicate by creating a prioritized list according to the popularity and availability of each file. Replicas of popular files are then placed on nodes that are stable and having good bandwidth to the requestor nodes. This objective is accomplished by formulizing and solving it as the Knapsack problem.

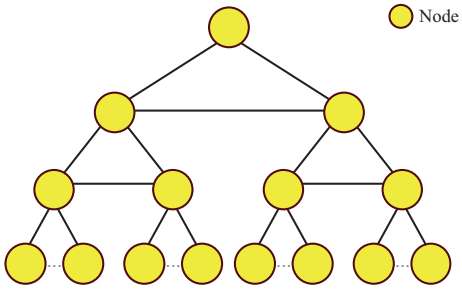


Fig. 4 An example hybrid architecture (sibling tree)

3.3 Dynamic replication strategies for hybrid architecture

In this subsection, we studied dynamic data replication strategies for hybrid architectures. Hybrid data grid architectures generally combine at least two other architectures with different properties. For example, as depicted in Fig. 4, if a replication strategy is created for a sibling tree hybrid architecture that combines P2P-like inter-sibling communication with hierarchical parenthood relationships, that particular strategy is studied in this subsection.

Lamehamedi et al. [29] present a hybrid replication strategy that combines the hierarchical architecture with P2P features. They implemented a cost model and based the replication decisions on how the gains of the replication measure against the costs. A runtime component constantly monitors the grid to collect important parameters, i.e. replica size, and network status. These information used in the calculation of the replication costs. They evaluated three different simulation scenarios on a single architecture. The results indicate that, average response time is improved as replicas are placed closer to the clients.

Rasool et al. [47] propose *Two-Way Replication* (TWR) that combines a multi-tier architecture with P2P-like features. In the target architecture, in addition to being connected to the parent node, each node (except at the leaf level) is connected to its siblings as well. Replication decision is handled by a central authority, called *Grid Replication Scheduler* (GRS). GRS targets the files that have higher-than-average access frequency are replicates them at the parent of the client that generate the most requests. Files with lower access frequencies are replicated at the grandparent level. A simulation study shows that, in terms of response time TWR performed similarly to Fast Spread, while consuming less resources.

3.4 Dynamic replication strategies for general graph architecture

In this subsection, we discuss dynamic data replication strategies that are proposed for general graph architectures. In general graphs, nodes are freely connected. From a scalability point of view, these architectures are at an advantage because there is no strict limitation on the organization of the nodes. Scale-free, social network based data grid architectures (Fig. 5), and other general strategies that do not focus on one particular architecture are classified in this subsection.

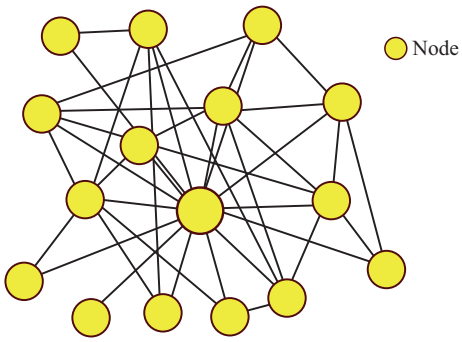


Fig. 5 An example data grid with scale-free topology

Rahman et al. [44] present a multi-objective approach to the replica placement problem. They use *p*-median and *p*-center models to select nodes for placing replicas. The *p*-median model finds *p* replica placement nodes to optimize the request-weighted average response time. The *p*-center model selects *p* replication nodes to minimize maximum response time. Their strategy aims to minimize *p*-median model by restricting the increase in the *p*-center objective. By doing this, they minimize average response time without having a requestor too far from a replication node. Their simulation study show that the multi-objective strategy has better response time than single-objective strategies that employ *p*-median and *p*-center models.

Lei et al. [31] propose a dynamic replication strategy, called *Minimize Data Missing Rate* (MinDmr). MinDmr measures and manages availability of the entire system. They introduce two data availability metrics, *System File Missing Rate* (SFMR) and *System Bytes Missing Rate* (SBMR). The former represents ratio of the missing number of files to total files requested by jobs and the latter represents the ratio of unavailable bytes to total bytes requested by all jobs. With the objective of improving SFMR or SBMR, all files are assigned weights by calculating the availability of the file, number of predicted future accesses, number of copies, and size of the file. The files with lower weights are called cold data and files with higher weights are called hot data. During replica replacement, cold data are deleted first, and hot data have the greater probability of replication. In performance evaluation, MinDmr performed better in terms of job execution times, SFMR, and SBMR compared to other strategies.

Chen et al. [12] developed the *Dynamic Multi-replicas Creation Algorithm* (DMRC) for data grids with scale-free complex network topology. Their strategy measures the degree of distribution of nodes, i.e. the number of links connected to a node. They assume that nodes with higher degrees are more important and better suitable for replica placement. Candidate nodes for replica placement are selected from two pools: frequency-based candidate pool and degree-based candidate pool. They established a cost model to calculate costs of placing replicas on candidate nodes. DMRC is compared to economic model and always replicate strategies of OptorSim. In both total job completion time and storage usage metrics, DMRC showed better performance.

Bsoul et al. [9] propose an improved Fast Spread replication strategy, called *Enhanced Fast Spread* (EFS). Different from Fast Spread, in EFS strategy, a replica is created only under two conditions: (i) when enough storage is available, or (ii) replica

to be created is more important than the replicas it is replacing. The replica replacement decision is based on a dynamic threshold that takes the number of requests, frequency of requests, size of the replica, and last request time into account. EFS is compared to Fast Spread with LFU, and Fast Spread with LRU in three different scenarios. In both total response time and total bandwidth consumption, EFS performed better compared to other studied strategies.

Andronikou et al. [5] present a quality of service (QoS) aware centralized dynamic replication strategy. In their approach, replication decisions are given by measuring data importance. The importance of data is defined as maximizing profits by satisfying QoS requirements of the system. The mechanisms for replica placement, relocation, and retirement are reduced to a search problem. In order to solve this search problem, they proposed a greedy algorithm and an adaptable heuristic algorithm. They compared both strategies against each other by measuring the execution time of the optimization algorithm. The heuristic approach outperformed the greedy algorithm in terms of execution speed.

4 Performance evaluation

Using data grid architecture the classification criteria raises the necessity to investigate the effect of data grid architecture on replication performance. Rather than comparing replication strategies individually, we are focusing on contrasting the benefits and disadvantages presented by the key properties of each architecture.

4.1 Simulation environment

There are a number of simulation tools available for the data grid. For our simulations, we have chosen OptorSim [7], as it is extremely popular and already used in many of the studied strategies in this paper. Originally developed as a part of the European Data Grid Project, OptorSim is easily extensible as a result of being an open source project.

In OptorSim, simulation scenarios are defined in three configuration files. First, general parameters file defines which access pattern, replica optimizer, and scheduling algorithm will be used, among other system-wide settings. Second, grid configuration file contains the topology definition of the grid, the storage and computational capabilities of nodes, and bandwidth capability of the network links. Finally, a job configuration file defines the available jobs on the grid for processing, as well as including a list of files that are required for each job.

A typical topology of a simulation scenario consists of computing elements (CE), storage elements (SE), and network links. When users submit jobs to the system, the resource broker (RB) assigns jobs to CEs as defined by the parameter files. During the computation of the jobs, the replica optimizer handles data replication.

OptorSim comes with LFU, LRU, and Economic replication strategies built-in. Our aim is to show the effect of data grid architecture on data replication. Therefore, we use these built-in replication strategies as tools and do not implement every single replication strategy studied in this paper. Moreover, these strategies are very suitable for

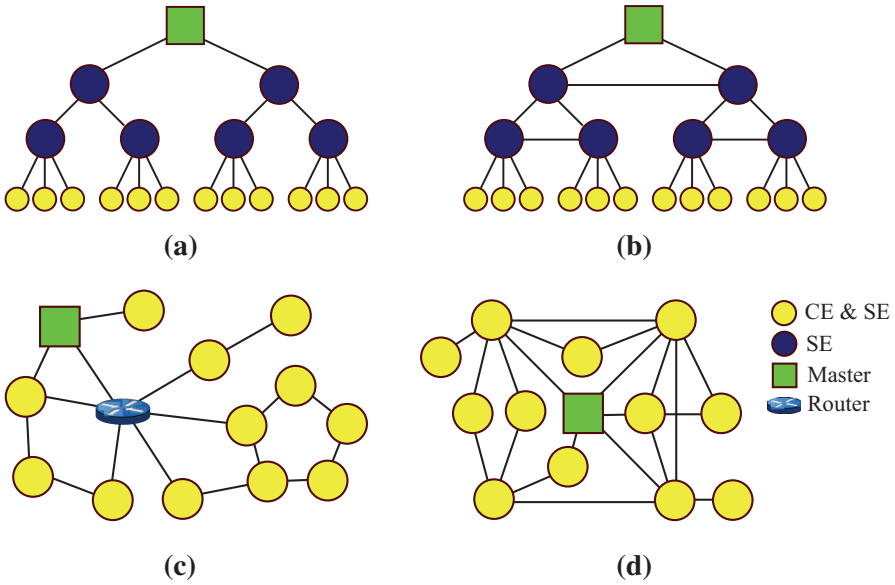


Fig. 6 Architectures used in the simulations. **a** Multi-tier **b** Hybrid **c** P2P **d** General graph

our purposes as they are architecture-independent, i.e. they do not favor any particular architecture.

Every data grid architecture has different topologies and various key properties that differentiate it from other architectures. In order to be able to make a clear and meaningful comparison of data grid architectures, it is necessary to set some constraints. These constraints help the results of the simulation to be the measurement of the initial intention. For example, without the constraint of network links, it would not be possible to comment on whether the results are due to the difference in architecture or due to the variation of the available bandwidth. Therefore, in the simulations, we keep the number of master nodes, number of CEs, capacity of SEs, and inter-node bandwidths the same for each architecture.

As the test cases, we created four architectures: (i) multi-tier, (ii) hybrid, (iii) P2P, and (iv) general graph. The topologies of these architectures are depicted in Fig. 6. In all scenarios, there is only one master node with enough storage capacity to hold all files initially. Storage capacity for all nodes is the same and of 5 GB. All network links in all architectures are of 100 Mbps. Some important simulation parameters and their respective values are included in Table 7.

We have used the same job configuration file for all architectures, in order to make sure that there would be no differences in the total amount of jobs processed by each architecture. In each run, a total of 100 jobs are distributed to the nodes by the RB. Each job requires 16 files on average, and each file has a size of 1 GB. File access patterns are also an important aspect in the simulations. We wanted our simulations to follow a realistic file access pattern. As a result, we have used a Zipf distribution, as Zipf distribution is the most accurate representation of the file requests on the Internet [3].

Table 7 Simulation parameters

Parameter	Value
Number of master nodes	1
Number of files	97
Size of a single file	1 GB
Number of jobs	100
Average number of files per job	16
Total number of CEs	12
Storage capacity per SE	5 GB
Inter-node bandwidth	100 Mbps

All three replication strategies are run on each of the four data grids. We have performed five simulation runs per scenario, and a total of 60 simulation runs were executed.

4.2 Simulation results

We are primarily focusing on the impact of data grid architecture on data replication performance. For this purpose, we consider response time as the most important metric. We measured response time as the elapsed time between the start of the first job and the completion of all jobs. It includes file access, computation, and communication time. Therefore, we can further comment on whether file access, computation or communication time contribute more to faster response time by evaluating other additional metrics.

Effective network usage (ENU) is another measured metric, and provided by OptorSim. By looking at the source code of OptorSim we can see that it is calculated as shown in Eq. 1:

$$\text{ENU} = \frac{N_{\text{remotereads}} + N_{\text{replications}}}{N_{\text{total}}} \tag{1}$$

The number of remote reads ($N_{\text{remotereads}}$) and the number of replications ($N_{\text{replications}}$) contribute to an increase in bandwidth consumption. ENU shows the ratio of the number of file operations that consume bandwidth to the number of total file requests (N_{total}). Therefore, a lower ENU value is an indication of efficiency in replicating the files. In order to investigate ENU deeper, we also recorded the metrics contribute to ENU, namely replications, number of local reads, and number of remote reads. And as the final metric, we measured CE usage to observe its effect on response time.

Analyzing response times depicted in Fig. 7a, it is obvious that all replication strategies performed best on general graph architecture. Also, P2P architecture performed better than hybrid, and hybrid architecture performed better than multi-tier in the same manner. In the results, comparison between different strategies among architectures are not the focus of this study. What is important is the performance comparison of any particular strategy with changing grid architecture. Despite the limited set of test

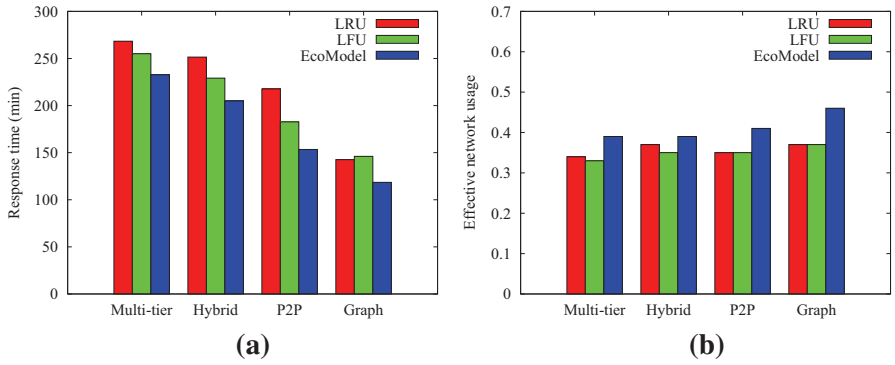


Fig. 7 Response time and ENU. **a** Response time (min). **b** ENU

cases with limiting constraints, there is a noticeable relation between the increasing connectivity of the nodes and the response time.

In order to explain how a more relaxed architecture results in an increase in performance, we need to look at other measured metrics. We see relatively similar levels of ENU as Fig. 7b shows. However, ENU is not a measure of bandwidth usage in terms of the amount of data transferred per unit time. It is a ratio, and we need to take a closer look at the values that contribute the calculation to correctly assess the result.

Economic model may decide to perform a remote read if it finds that a required file is not valuable enough to store in the local storage as a replica. While LFU and LRU always replicate instead of performing a remote read, Economic model performed 251, 256, 307, and 363 remote reads in multi-tier, hybrid, P2P, and general graph data grid scenarios, respectively. Multiple access paths to remote data files lessens the bottleneck during file transfers. As the connectivity of the nodes increased in the tested architectures, remote access cost is reduced, and economic model could afford a greater number of remote reads. As a result, both the number of local reads (Fig. 8a) and number of replications (Fig. 8b) decreased for the economic model, and this leads to a slight increase in ENU.

Another component that contributes to the calculation of the response time is CE usage. For some strategies, we observed decreased CE usage (Fig. 8c) as the architecture is relaxed. Even though this has a negative impact on the response time, the benefits of the easier file access surpasses this negative effect, as the response time is still decreased for these strategies.

4.3 Analysis

The analysis of the results show that more relaxed architectures offer easier data access, with multiple network routes to remote sites. In applications where frequent remote requests are a necessity, these architectures should be more suitable than other architectures with stricter topologies. It is safe to say that, in the architectures where nodes are connected in a less restrictive manner, the response time is decreased, regardless of the replication strategy we have used.

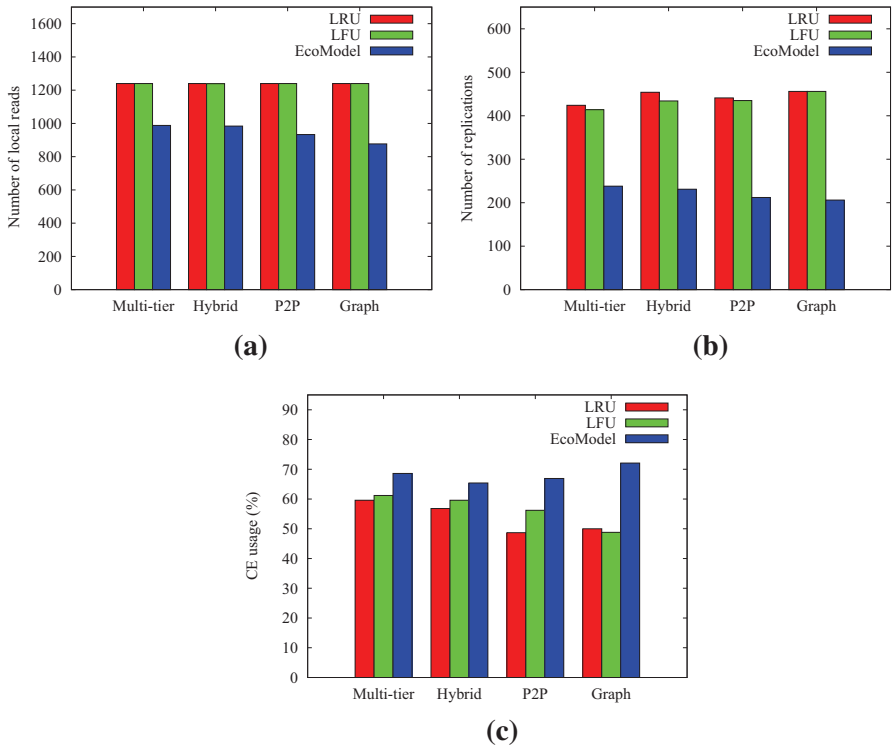


Fig. 8 Number of local reads, number of replications, and CE usage. **a** Number of local reads. **b** Number of replications. **c** CE usage (%)

As in many studied methods, we used the built-in replication strategies of OptorSim. Although not evaluating the most recent strategies for simulations is frowned upon [4], it is irrelevant for our simulations as we are comparing data grid architectures. For this reason, we deliberately did not use the replication strategies from our classification to avoid favoring any particular architecture. We would like to note that any future work on comparing data grid architectures should be aware of this issue.

File access patterns are shaped by the user requests [20,49]. Therefore, a popular file of today may not be popular in the future or vice versa. There are a number of different access patterns used in the literature, including sequential, and random access patterns that are generated from statistical distributions, e.g. Gaussian and Zipf. While some papers use three [45] or five [30,48,51] different access patterns in the simulations, we used just one and the most realistic access pattern [3], namely Zipf-based, to keep the focus on architecture comparison.

5 Open problems in dynamic replication strategies

In recent years, data grid systems have seen a constant evolution as the available technologies changed. Foster et al. [21] describe the change of focus in data

grids from an infrastructure-based grid that delivers storage and computational resources, to economy-based ones that the serve resources in a more abstract way. In the infrastructure-based systems, we see the institutions sharing a federated set of resources. On the other hand, in the economy-based systems, these set of resources are typically offered by a third party for a profit. In this direction towards the economy-based systems, elastically scalable computing solutions bring new challenges into the scene. Monetary aspect of the resources makes it as must to consider the point of view of the resource providers as well.

Many replication strategies aim to increase performance through increasing availability. As a result, there are many studies pointing out replication strategies that always replicate [20,45] or create as many replicas as possible [42,49]. We have seen, especially in the replication strategies that target bandwidth hierarchy, the economic aspect of the assumptions is taken into account, e.g. bandwidth cost difference between different network regions. However, there still is an open research issue that considers other aspects, e.g. cost of storage and cost of utilizing more nodes, especially for the economy-based systems. An example case is when a traditional strategy aims to increase availability by filling all of the available storage. Considering the cost of increased storage capacity, it is apparent that replication strategies that create as many replicas as possible will create economic burden for both the consumer and the provider.

We believe that another open problem in dynamic replication strategies is accomplishing the performance goals while maintaining a certain degree of replication, i.e. a certain number of replicas. Dynamically adjusting the optimal number of replicas with economic consideration presents new research challenges [38]. Finding optimal number of replicas puts emphasis on replica placement as well. Studying the combined effectiveness of the optimal number of replicas and strategically placing them is still an open issue and plays a key role in achieving optimality for both consumers and providers of the resources.

6 Conclusion

In this paper we presented a survey of dynamic replication strategies for data grid systems. Only few works classified dynamic data replication by taking data grid architecture as one of the classification criteria. Also, these works did not provide any performance evaluation. We have studied and classified recent dynamic replication strategies only according to data grid architecture and discussed their contributions. Furthermore, we performed a simulation study to investigate the impact of data grid architecture on data replication performance. Our simulations evaluate a number of replication strategies on different architectures. We highlighted how key aspects of each data grid architecture impact the performance. The simulation study indicates that more relaxed architectures yield better response times while keeping relatively similar levels of effective network use. As a result, we regard these relaxed architectures, e.g. general graphs, as the most interesting and realistic representation of the data grid systems. The simulation study can be further expanded in the future to include more recent replication strategies. Following the simulation study, we discussed some

open problems in dynamic replication strategies, including finding the optimal number of replicas.

Acknowledgments The work presented in this paper is supported in part by TUBITAK.

References

1. Abdullah A, Othman M, Ibrahim H, Sulaiman MN, Othman AT (2008) Decentralized replication strategies for P2P based scientific data grid. In: International symposium on information technology (ITSim 2008), vol 3, pp 1–8. IEEE
2. Abdurrab AR, Xie T (2010) FIRE: a file reunion based data replication strategy for data grids. In: 10th IEEE/ACM international conference on cluster, cloud and grid computing, pp 215–223. IEEE
3. Adamic L, Huberman B (2002) Zipf's law and the Internet. Glottometrics 3(1):143–150
4. Amjad T, Sher M, Daud A (2012) A survey of dynamic replication strategies for improving data availability in data grids. Future Gener Comput Syst 28(2):337–349
5. Andronikou V, Mamouras K, Tserpes K, Kyriazis D, Varvarigou T (2012) Dynamic QoS-aware data replication in grid environments based on data importance. Future Gener Comput Syst 28(3):544–553
6. Arlitt M, Cherkasova L, Dilley J, Friedrich R, Jin T (2000) Evaluating content management techniques for web proxy caches. ACM SIGMETRICS Perform Eval Rev 27(4):3–11
7. Bell WH, Cameron DG, Capozza L, Millar AP, Stockinger K, Zini F (2012) Simulation of dynamic grid replication strategies in optorsim. In: IEEE workshop on grid computing (Grid'2002), pp 46–57
8. Bell WH, Cameron DG, Carvajal-Schiaffino R, Millar AP, Stockinger K, Zini F (2003) Evaluation of an economy-based file replication strategy for a data grid. In: Proceedings of the 3rd IEEE/ACM international symposium on cluster computing and the grid (CCGRID'03), pp 661–668. IEEE
9. Bsoul M, Al-Khasawneh A, Abdallah EE, Kilani Y (2011) Enhanced fast spread replication strategy for data grid. J Netw Comput Appl 34(2):575–580
10. Challal Z, Bouabana-Tebibel T (2010) A priori replica placement strategy in data grid. In: International conference on machine and web intelligence, pp 402–406. IEEE
11. Chang RS, Chang HP (2008) A dynamic data replication strategy using access-weights in data grids. J Supercomput 45(3):277–295
12. Chen D, Zhou S, Ren X, Kong Q (2010) Method for replica creation in data grids based on complex networks. J China Univ Posts Telecommun 17(4):110–115
13. Chervenak A, Deelman E, Foster I, Guy L, Hoschek W, Iamnitchi A, Kesselman C, Kunszt P, Ripeanu M, Schwartzkopf B, Stockinger H, Stockinger K, Tierney B (2002) Giggle: a framework for constructing scalable replica location services. Proc ACM/IEEE Conf Supercomput 3:1–17
14. Chervenak A, Deelman E, Kesselman C, Allcock B, Foster I, Nefedova V, Lee J, Sim A, Shoshani A, Drach B, Williams D, Middleton D (2003) High-performance remote access to climate simulation data: a challenge problem for data grid technologies. Parallel Comput 29(10):1335–1356
15. Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S (2000) The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. J Netw Comput Appl 23(3):187–200
16. Chervenak A, Schuler R, Kesselman C, Koranda S (2008) Wide area data replication for scientific collaborations. Int J High Perform Comput Netw 5(3):124–134
17. Chettaoui H, Charrada FB (2014) A new decentralized periodic replication strategy for dynamic data grids. Scalable Comput: Pract Exp 15(1):101–119
18. Cibej U, Slivnik B, Robic B (2005) The complexity of static data replication in data grids. Parallel Comput 31(8–9):900–912
19. Deelman E, Kesselman C, Mehta G, Meshkat L, Pearlman L, Blackburn K, Ehrens P, Lazzarini A, Williams R, Koranda S (2002) GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists. In: Proceedings of the 11 th IEEE international symposium on high performance distributed computing (HPDC02), pp 225–234
20. Dogan A (2009) A study on performance of dynamic file replication algorithms for real-time file access in data grids. Future Gener Comput Syst 25(8):829–839
21. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. In: 2008 grid computing environments workshop, pp 1–10. IEEE

22. Fu X, xin Zhu X, yuhan J, chuan Wang R (2013) QoS-aware replica placement for data intensive applications. J China Univ Posts Telecommun 20(3):43–47
23. Goel S, Buyya R (2006) Data replication strategies in wide area distributed systems. In: Enterprise service computing: from concept to deployment, p 17
24. Horri A, Sepahvand R, Dastghaibyfard GH (2008) A hierarchical scheduling and replication strategy. Int J Comput Sci Netw Secur 8(8):30–35
25. Hoschek W, Jaen-martinez J, Samar A, Stockinger H, Stockinger K (2000) Data management in an international data grid project. In: IEEE, ACM international workshop on grid computing, pp 77–90
26. International Organization Of Standardization (2011) ISO/IEC/IEEE 42010:2011—systems and software engineering–architecture description. Technical report
27. Khanli LM, Isazadeh A, Shishavan TN (2011) PHFS: a dynamic replication method, to decrease access latency in the multi-tier data grid. Future Gener Comput Syst 27(3):233–244
28. Kingsy Grace R, Manimegalai R (2014) Dynamic replica placement and selection strategies in data grids. A comprehensive survey. J Parallel Distrib Comput 74(2):2099–2108
29. Lamehamedi H, Szymanski B, Shentu Z, Deelman E (2002) Data replication strategies in grid environments. In: Proceedings of 5th international conference on algorithms and architectures for parallel processing, pp 378–383. IEEE Comput Soc
30. Lee MC, Leu FY, Chen YP (2012) PFRF: an adaptive data replication algorithm based on star-topology data grids. Future Gener Comput Syst 28(7):1045–1057
31. Lei M, Vrbsky SV, Hong X (2008) An on-line replication strategy to increase availability in data grids. Future Gener Comput Syst 24(2):85–98
32. Loukopoulos T, Ahmad I (2004) Static and adaptive distributed data replication using genetic algorithms. J Parallel Distrib Comput 64(11):1270–1285
33. Ma J, Liu W, Glatard T (2013) A classification of file placement and replication methods on grids. Future Gener Comput Syst 29(6):1395–1406
34. Maltsev N, Glass E, Sulakhe D, Rodriguez A, Syed MH, Bompada T, Zhang Y, D'Souza M (2006) PUMA2—grid-based high-throughput analysis of genomes and metabolic pathways. Nucleic acids Res 34(Database issue):D369–D372
35. Mansouri N, Dastghaibyfard GH (2012) A dynamic replica management strategy in data grid. J Netw Comput Appl 35(4):1297–1303
36. Mansouri N, Dastghaibyfard GH (2013) Enhanced dynamic hierarchical replication and weighted scheduling strategy in data grid. J Parallel Distrib Comput 73(4):534–543
37. Mansouri N, Dastghaibyfard GH, Mansouri E (2013) Combination of data replication and scheduling algorithm for improving data availability in data grids. J Netw Comput Appl 36(2):711–722
38. Mansouri Y, Azad ST, Chamkori A (2014) Minimizing cost of K-replica in hierarchical data grid environment. In: IEEE 28th international conference on advanced information networking and applications (AINA), pp 1073–1080. IEEE
39. Meroufel B, Belalem G (2013) Managing data replication and placement based on availability. AASRI Procedia 5:147–155
40. Mokadem R, Hameurlain A (2015) Data replication strategies with performance objective in data grid systems: a survey. Int J Grid Util Comput 6(1):30–46
41. Nicholson C, Cameron DG, Doyle AT, Millar AP, Stockinger K (2008) Dynamic data replication in lcg 2008. Concurr Comput: Pract Exp 20(11):1259–1271
42. Park SM, Kim JH, Ko YB, Yoon WS (2004) Dynamic data grid replication strategy based on Internet hierarchy. Grid and cooperative computing. Springer, Berlin Heidelberg, pp 838–846
43. Pérez JM, García-Carballeira F, Carretero J, Calderón A, Fernández J (2010) Branch replication scheme: a new model for data replication in large scale data grids. Future Gener Comput Syst 26(1):12–20
44. Rahman R, Barker K, Alhajj R (2005) Replica placement in data grid: a multi-objective approach. In: Zhuge H, Geoffrey CF (eds) Grid and cooperative computing-GCC 2005. Springer, Berlin Heidelberg, pp 645–656
45. Ranganathan K, Foster I (2001) Identifying dynamic replication strategies for a high-performance data grid. In: Proceedings of the international grid computing workshop, vol 2242, Springer, pp 75–86
46. Ranganathan K, Iamnitchi A, Foster I (2002) Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In: Proceedings of the 2nd IEEE/ACM international symposium on cluster computing and the grid (CCGRID'02), pp 376–381

47. Rasool Q, Li J, Zhang S (2009) Replica placement in multi-tier data grid. In: 8th IEEE international conference on dependable, autonomic and secure computing, pp 103–108. IEEE
48. Saadat N, Rahmani AM (2012) PDDRA: a new pre-fetching based dynamic data replication algorithm in data grids. Future Gener Comput Syst 28(4):666–681
49. Sashi K, Thanamani AS (2011) Dynamic replication in a data grid using a modified BHR region based algorithm. Future Gener Comput Syst 27(2):202–210
50. Segal B (2000) Grid computing: the European data grid project. IEEE Nucl Sci Symp Med Imaging Conf 1:15–20
51. Shorfuzzaman M, Graham P, Eskicioglu R (2009) Adaptive popularity-driven replica placement in hierarchical data grids. J Supercomput 51(3):374–392
52. Steen MV, Pierre G (2010) Replicating for performance: case studies. In: Charron-Bost B, Pedone F, Schiper A (eds) Replication. Lecture Notes in Computer Science, vol 5959. Springer, Berlin Heidelberg, pp 73–89
53. Takefusa A, Tatebe O, Matsuoka S, Morita Y (2003) Performance analysis of scheduling and replication algorithms on grid datafarm architecture for high-energy physics applications. In: Proceedings of the 12th IEEE international symposium on high performance distributed computing (HPDC03), pp 34–43
54. Tang M, Lee BS, Yeo CK, Tang X (2005) Dynamic replication algorithms for the multi-tier data grid. Future Gener Comput Syst 21(5):775–790
55. Tang X, Xu J (2005) QoS-aware replica placement for content distribution. IEEE Trans Parallel Distrib Syst 16:921–932
56. Tatebe O, Morita Y, Matsuoka S (2002) Grid datafarm architecture for petascale data intensive computing. In: International symposium on cluster computing and the grid (CCGRID02), pp 102–110
57. Xhafa F, Kolici V, Potlog AD, Spaho E, Barolli L, Takizawa M (2012) Data replication in P2P collaborative systems. In: 2012 7th international conference on P2P, parallel, grid, cloud and internet computing, pp 49–57. IEEE
58. Zhao W, Xu X, Wang Z, Zhang Y, He S (2010) A dynamic optimal replication strategy in data grid environment. In: International conference on internet technology and applications, pp 2–5