ISSN 2278-3091
Volume 8, No.5, September - October 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse117852019.pdf
https://doi.org/10.30534/ijatcse/2019/117852019

A Survey on Data Replication and Resource Management in Distributed Systems

Wan Nor Shuhadah Wan Nik[1], Bing Bing Zhou[2]
[1]Faculty of Informatics and Computing, University Sultan Zainal Abidin, Terengganu, Malaysia
wnshuhadah@unisza.edu.my
[2]School of Computer Science, University of Sydney, Sydney, Australia
bing.zhou@sydney.edu.au

ABSTRACT

Data replication provides an efficient mechanism in dealing with data (or storage that hold these data) in large-scale distributed systems. Generally, data replication is one of the optimization strategies which have been implemented in the distributed systems including database communities where several copies of data are kept at two or more resource sites to achieve performance at high level, availability and reliability of the distributed systems. Consequently, it is realized that the resource selection process (as part of resource management) is tightly coupled with data replication strategies and resource management in distributed systems. Accordingly, this paper discusses of the existing data replication techniques in distributed systems, in particular distributed DBMS, Data Grids and Cloud computing environments. Further, a review of resource management in distributed systems is also provided, with specific focus on both Grids and Cloud computing.

Key words: Replication of Data, Grid and Cloud Computing, Resource Management, Utility Computing.

1. DATA REPLICATION IN DISTRIBUTED SYSTEMS

To give a better understanding of data replication in distributed systems, a simple example of data replication environment is provided and is shown in Figure 11, which is adopted from [1]. In this example, there are *n* resource sites with $\{ S_1, S_2, \cdots, S_n \}$ geographically distributed and connected through middleware architecture. Further, assume that Object X represents data stored at S_1 and replicated to all other sites. For simplicity, let assume that the distance shown in the figure be directly proportional to the cost of access for Object X. Therefore, if User 1 would like to access an Object X, he or she can obtain a cheaper cost when this object is accessed at either S_2 or S_3, since S_2 and S_3 are closer to the user as compared to S_1, where the data was originally stored. In such a situation, the benefits of replication are apparent where it increases the performance (access cost) of distributed system and the data is still accessible despite of 3 out of 4 sites are fails (and therefore improving the reliability and availability).

Numerous replication schemes have been developed for different distributed system architectures such as Distributed Database Management System (DDBMS), Peer-to-Peer (P2P) systems, Data Grids/Grid Computing, World Wide Web (WWW), etc. Among them, both DDBMS and Data Grids can be considered to be the most active research domains in distributed systems in recent years. Furthermore, while Grid Computing is the predecessor for both Enterprise Grid and Cloud computing (as discussed in Chapter 1), it is realized that database replication (in particular, DDBMS) plays a pivotal role in supporting a transactional online application which may exist in both Enterprise Grids and Cloud computing systems [2]. With this regards, the discussions focus on the existing data replication techniques in DDBMS and Data Grid environments in section II and III, respectively, before a survey on resource management in Distributed Systems is provided in section IV.

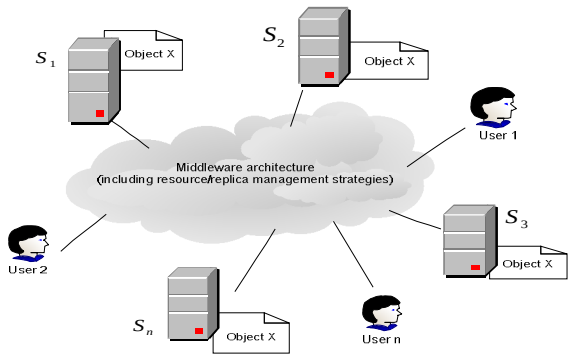

Figure 1: An example scenario of "Object X" being replicated at all resource sites [1].

2. DATA REPLICATION IN DISTRIBUTED DBMS

Replication of data in the area of distributed DBMS has been discussed for many years [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. Indeed, preserving the consistency and ensuring the correctness of replicated data is one of the major issues in this type of distributed system. Many

replication approaches have been developed to address this issue. Generally, these approaches can be generally categorized into two categories: *asynchronous* replications and *synchronous* replications.

Update transactions are propagated before commit to provide consistency guarantees among data which has been replicated in synchronous replications (or *eager* replication). The design of this type of replication must satisfy a property called *one-copy-serializability* (1SR) [16]. This property ensures the strict consistency among replicated data where the intermittent executions that are involved in replicated database need to establish a view of a *one-copy* database. In order to guarantee serializability, conflicts are usually resolved by using a mechanism called *concurrency control* protocol, which is defined as a process of managing simultaneous operations in the distributed database without having them interfere with one another [17]. Locking methods (e.g., two-phase locking (2PL)) and time-stamping methods are the two most well-known approaches which have been used for concurrency control. In locking methods, when one operation is accessing the database, a lock may deny access to other operations to avoid incorrect results. On the other hand, time-stamping-based methods ensure the correctness and the seriliazability of interleaved operations by using a unique identifier that indicates the start time of an operation. This identifier is used to order operations in such a way that operations with a smaller timestamp get priority to be executed in the event of conflict. In other words, the order of operations that are conflicting need to be preserved the scheduling of job operations is done [17].

In asynchronous replications, also known as *lazy* replication, a replica can proceed with the execution of write operations without delay. In such, other replicas are allowed to get the updated value from this updated replica via update propagation when this updated replica has committed the operation. In other words, asynchronous replications do not have to satisfy strict consistency as in 1SR property.

3. DATA REPLICATION IN DATA GRIDS

Applications in Grid environments may involve many events and different scenarios of access patterns. Two main categories of the scenarios of replication can be generally categorized: (1) operations of read-only, and (2) operations of update (or write) [1]. Accordingly, for applications that involve read-only queries, the strategies of data replication can further be categorized into two main streams: *dynamic* and *static* replication. Meanwhile, with regard to update-intensive applications, data replication strategies are adopted from the distributed DBMS community, i.e., *synchronous* replication and *asynchronous* replication. These categorizations are illustrated in

Figure2. However, replication technology used in Grid environments is mostly targeted for read-only applications, where the main goal is to ensure the minimization on the latency of access and also the consumption of bandwidth are achieved. Most importantly, replication is done on coarse-grained data, i.e., flat data files rather than fine-grained data such as datasets from databases [18], [19], [20], [21], [22].

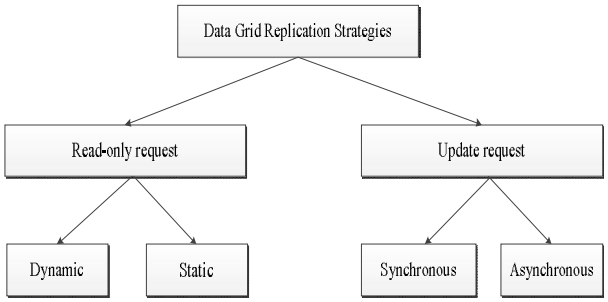

Figure 2: Categorization of data replication strategies which may involves in Data Grids environment

3.1 Data Grid Replication Strategies in Read-Only Queries

For a request that does not amend the value of data (i.e. read-only query), the data consistency is not an issue because the replicated data can be read by an operation without need to concern on data correctness. Typically, the data may be created at some resource node and can be accessed (and not modified) by other resource nodes. These data can be safely kept at other replicated resources. Dynamic replication approaches can tolerate changes in a pattern of user requests, the bandwidth and the capacity of storage, and able to create new replicas on new nodes as well as removing replicas that are no longer needed, which is depends on the global information available in the Data Grid system. A Grid Component such as Replica Manager (RM) is responsible to make decision on the removal and creation of replica. This functionality is usually a part of the data/replica management system [23], [24], [21], [25]. Further, [26] provides the most recent comprehensive survey on dynamic data replication in the area of Data Grids.

In contrast, the strategy of static replication means that a static number of replicas is determined at the beginning of the life cycle; where no other replicas are migrated or created later on [27], [19], [20], [28], [29]. With this static approach, data are replicated in advance to make as many resource sites as possible that is sufficient for job operations. For example, the implementation of well-known Data Grid services such as Globus [27], [19], [20], [28], [29] and the EUDataGrid [30], [31] are designed in such a way that the indices of replica location is preserved throughout the systems by a dedicated nodes. Indeed, this replication strategy offers a centralized and static approach in managing replicas on Data Grids.

It is apparent that dynamic, rather than static, replication strategy has been receiving more attention among existing researchers in the Grid environment as it able to make smart

decisions on data placement which is depends on the information that is available in the Grid environment. However, both dynamic and static replications come with the price of their own disadvantages. Despite of the advantages offers by the dynamic approach, this strategy also comes with the price of the creation of new replicas while continuous assessment on the availability of the resources. Notwithstanding this, the static replication strategy has several advantages such as faster job scheduling, improved fault-tolerance, with no latency which usually occurred in dynamic data replication [32].

3.2 Data Grid Replication Strategies in Update Request

Special consideration needs to be concerned in designing data replication for update request. As compared to read-only queries, an update request may modify data which further compromise data consistency. A replica is modified locally in the synchronous model. In this case, synchronization among all other replicas is achieved via a replica propagation protocol. However, the concept of data replication in distributed DBMS is also applied in this case, especially when local replicas at other nodes are being modified and/or updated. If conflict occurs in such cases, it is crucial to re-done the job must with the latest or updated replica.

With regard to asynchronous replication in the area of Data Grid, very few researchers have discussed the asynchronous model such as in [33]. In this research, different levels of consistency are considered, including possible inconsistent copy (consistency level -1), consistency file copy (consistency level 0) and consistent transactional copy (consistency level 1). The authors argue that because of the latency caused by locking process in strict consistency requirement, the strategy is not practical for a Grid computing environment. They propose that data consistency with various levels should be supported by Grid consistency services. The work is primarily focused on data sources with the flat file. The concept of database transaction theory which includes locking for establishing consistent data becomes the primary reference on their discussions.

Meanwhile, in [34], two Grid replication protocols have been proposed, namely aggressive copy and lazy copy. In later concept (i.e. lazy copy), the replica content is synchronized with the primary replica only when Grid sites requests an operation. Otherwise, replicas are inconsistent for some time until the request for operation is made. Meanwhile, an aggressive copy strategy provides full consistency at all time. This strategy is similar to synchronous replication method.

3.3 Data Grid Replication in Cloud Computing Environments

Similar to Grids, Cloud applications also benefit data replication to get high performance, availability and reliability of the Cloud computing systems. However, while Data Grids merely deal with files where very rare update operations are involved, data replication schemes developed for Cloud applications usually deals with replicated data which are vulnerable for frequent updates. In such cases, issues of data consistency become a focus. In other words, Cloud systems share a similar characteristic with distributed DBMS in terms of data replication, where the consistency issue is paramount and needs to be addressed when dealing with data replication.

However, this issue becomes more complex and it is undoubtedly difficult to managed, i.e. ACID (atomicity, consistency, isolation, and durability) properties in Cloud computing environment as compared to traditional distributed DBMS. This is due to the nature of Cloud characteristics in which data may be replicated across large geographic distance [35]. The CAP theorem found in [36] proves that a replicated system can only choose at most two out of three properties: consistency, availability and fault-tolerance. That is, when data are replicated over a wide area, the trade-offs between “Consistency” (which is part of ACID) and availability for a system are typically compromised [35].

Further, research in [37] proposed a new transaction scheme paradigm on replicated data for Cloud applications. The concept called *Consistency Rationing* is introduced in order to achieve an optimal cost of runtime for a system database in the Cloud when inconsistencies introduce a penalty in terms of monetary cost. The basic concept of the idea in this research is to allow applications to achieve a sufficient level of consistency at the very minimum cost as possible. That is, consistency requirements are categorized into three types, A, B and C. Strong consistency is guaranteed at the very high monetary cost per transaction in an A category. The C category represents a scheme similar to the one in SimpleDB and Yahoo PNUTS, that is, eventual/timeline/session consistency; this result an inconsistencies despites its minimum transaction cost. Meanwhile in a B category, depending on the specified requirement, data is managed with either session or strong consistency. The authors show the practicality of the proposed scheme through extensive experiments implemented on Amazon S3 Cloud storage running the TPC-W benchmark.

In [38], the authors proposed a method that exploits lazy (*asynchronous*) update propagation strategy for data updates of data replicas in Cloud computing. The fundamental idea of this research is to differentiate the updating data replica process and data access in Cloud in order to preserve the consistency of data replicas while preserving the availability and accessibility and of data services in Cloud systems. In other words, the proposed method allows the update propagation to be done before the local copy of the replica at the server on the master site commits. The rationale behind this approach is to avoid inconsistencies in case updating at

the master or primary site fails while it may succeed at the secondary or slave site.

Other research on dealing with data consistency in Cloud application are found in [39], [40], [41] and [42]. In these studies, another variant of asynchronous update protocol on replicated data is proposed in the Cloud computing environment. The authors proposed a scheme called Re:GRIDiT which is developed to deal with distributed update transaction on replicated data. This approach addresses the requirements of novel data-intensive e-Science applications in Grid by eliminating the need of a global coordinator to synchronize updates on multiple replicas. Later, in [42], the same authors enhance the Re:GRIDiT approach with strategies on deciding an optimal replica placement locations to address the load balancing issue among replicas before a refined version, called. Re:FRESHiT [43], is developed. Similar to the one proposed in [38], these schemes seem to be excellent to be implemented in the Cloud environment on their own advantages. However, none of these researches addresses the issue of monetary cost incurred on executing jobs by resources (which host the replica data required by jobs), which is a trademark characteristic of Cloud systems.

4. RESOURCE MANAGEMENT IN DISTRIBUTED SYSTEMS.

When data replication is considered, two main entities of the resource management system are *replica placement* and *replica selection*. In other words, the efficient strategies on the placement and the selection of replica (or resource that holds this replica) is of prime importance in order to ensure that jobs are served with the most appropriate resources to meet the goals of the distributed system. From here onward throughout this thesis, the term "*resource*" will be used to refer to the replica or physical resource (storage) that holds this replica data in distributed systems. Therefore, with regard to the scope of this research, the following will review the existing resource management in the area of Data Grid and Cloud storage systems in order to give a better insight on the significant challenge of resource management, which includes resource selection, in both Grid and Cloud system environments.

4.1 Resource Management in Data Grids

Generally, the Resource Management Systems (RMS) as a central unit in Grid systems is responsible to address many issues in order to ensure the efficiency of Grid in supporting various applications. These issues include (a) scalability, extensibility and adaptability, (b) preserving site autonomy while allowing systems with various policies on administrative to inter-operate, (c) resource co-allocations, (d) QoS support, and (e) comply with constraint of computational cost.

These issues must be addressed in addition to matters such as stability and fault-tolerance [44]. Figure3 illustrates the general RMS system context in a Grid environment [44]. Based on this diagram, the implementation of user applications is done via a usage of Grid toolkits services. This toolkit presents an abstraction of suitable application via services offered by the RMS. One of the most well-known RMS architectures in the Data Grid area is the one developed for the EUDataGrid project [45], [46]. The main components of RMS considered in this project are shown in Figure4 [45]. From this figure, it is obvious that replica selection is an optimization process that must be considered in the replica management service in the area of Data Grid.

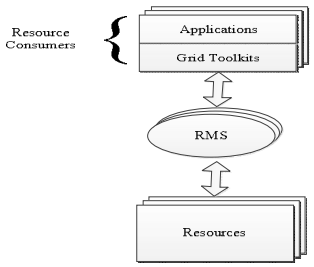

Figure 3: General RMS system context in Grid environment [44].

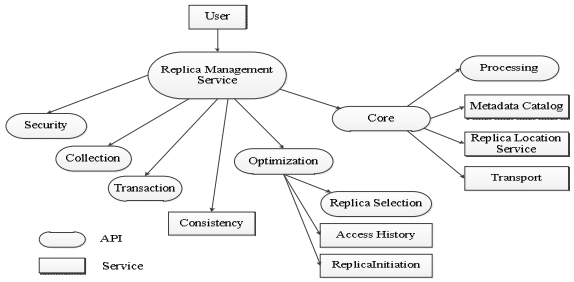

Figure 4: The main entities in Replica Management System in EUDataGrid project [45].

Also based on the EUDataGrid project [45], [47], the simulation tool called *OptorSim* can be considered as one of the most well-known simulation packages which have been designed to investigate the efficiency of algorithms for replica optimization in Data Grid environment. This is especially true when many existing researches in replication optimization in the area of Data Grid have been working with this simulation tool such as in [48], [49], [50]. The architecture of this simulator is illustrated in Figure5 [51]. The construction of the simulator was guided by the assumption that Grids are composed of two or more sites. These sites may provide storage resources and computational resources when jobs are submitted. *User Interface* in a Grid systems is used by the user to submit jobs. Each of these sites may equipped with zero or more *Storage Elements* (SE) and zero or more *Computing Elements* (CE). CE is responsible to execute jobs which may used the data which are kept in SE.

The job scheduling to CE is controlled by the *Resource Broker* (RB). Network nodes or routers are represented by the sites with no SE or CE. *Replica Manager* (RM) is a component that is responsible to decide the movement of data together with their jobs between sites. Another component that is within RM is called *Optor*, which control the decision on the creation and deletion of replicas. Replica optimization algorithm is the main function of *Optor*. OptorSim performs two different types of optimizations: (1) the RB uses the *scheduling algorithms* for job allocation, and (2) the RM at each site uses replication algorithms to make decision on which data (file) to replicate, which data to delete and when to replicate a data [52], [53]. The ultimate goal is to minimize the execution time for jobs, together with the efficient use of resources in Grid system.

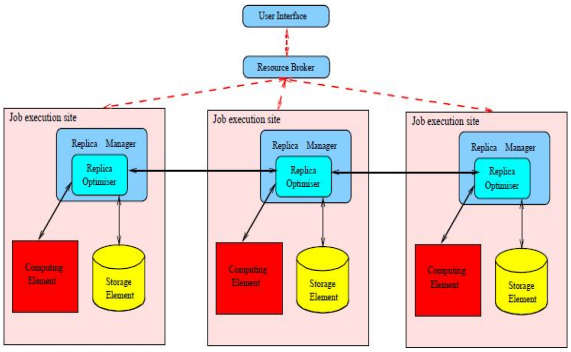

Figure 5: Architecture of simulated Data Grid architecture (OptorSim) [51].

Indeed, the design of resource management in the Data Grid area is tightly coupled with the implementation of replication strategies employed in the respective Grid systems. That is, replica (or the resource that hosts the replica) management is a Grid service which has a replica manager to delete or create the replicas in the storage systems. Therefore, the replacement, creation and selection of replicas together with the maintenance are all considered in replica management. However, only the first three functionalities (i.e. replica creation, placement and deletion) are considered as a main function of replica management. Based on different strategies of replacements, creations and selection of replicas, the replica management can be classified into several types as shown in Figure **6** [54], [55]. Based on this classification and due to the dynamic nature of Grid environment, the simple replica management strategy is not suitable for such environment as it frequently replicate data which may introduce high overhead. A model in hierarchical form is designed for EUDataGrid [30], [31], [23] and is not suitable for the structure of hybrid and P2P network. The main problem for economic model lies on its complexity on the evaluation of computing. Further, many other aspects are also considered by other replica policies.

Li et al. [56] propose a scalable method on replica location where the determination of effective location for various replicas of the same data is used by home nodes. Also in this method, the local replica is used to support local query for replica.

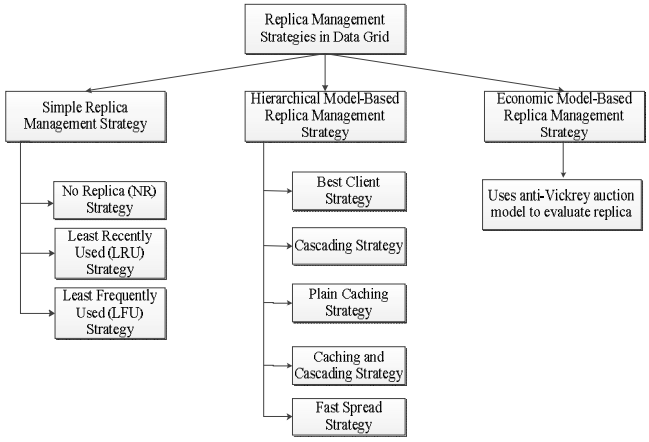

Figure 6: Replica Management Strategies in Data Grid [54], [55].

4.2 Resource Management in Cloud Storage

Resource management in Cloud storage deals with the issue to provide durability, high data availability, and cost-efficiency both for Cloud storage providers and Cloud users. Similar to Data Grids, the goals of achieving this high quality of system service are usually gained via an indirect replication (i.e. customer intervention or request is not required as an automatic data replication) in Cloud storage systems. In such situations, it is realized that replica placement and/or resource selection are also paramount in resource management in Cloud storage systems.

Replica placement in the management of replication has become an active research focus in Cloud storage. A scheme of replication management that is cost-effective has been proposed in [57], named CDRM, for a Cloud storage cluster. Two main issues have been addressed in this research: system availability and load balancing. In terms of availability, the fundamental concept of the proposed approach lies in the relationship between the number of replica and availability [58]. That is, more replicas creation will increase the system availability as it can mask any resource failure which allows the operation of distributed systems to continue despite this interruption. However, this comes with the price of higher resource management cost when the number of replicas is increasing. Therefore, the proposed CDRM scheme is leveraged in order to calculate and maintain the minimum number of replicas for a given availability requirement, which further minimizes the resource management cost while preserving good system availability for Cloud storage systems. Meanwhile, with regards to load balancing issues, this research proposed a new scheme of replica placement in order to distribute load efficiently across data nodes clusters. Similarly to [57], research done in [59] adapts replica placement strategy described in [58] by expressing availability as a function of replication degree in order to improve the availability of data, its reliability and the utilization of network bandwidth, while minimizing the

trade-offs between data availability and resource management cost. The main difference brought by this research is that it developed a PC cluster-based Cloud storage system (cheaper than the high performance server), which is implemented with *Hadoop Distributed File System* (HDFS) [60] by enhancing replication management schemes in order to provide inexpensive storage.

Further, research in [61] proposed a Cloud storage management service called *ecStore*, an elastic Cloud storage system to support automatic data replication and partitioning, efficient range query, load balancing, and transactional access. The proposed system is part of a project called *epiC* (elastic power-aware data-intensive Cloud) [62] , which is developed to support OLTP and analytical workloads. Specifically, the design of *ecStore* provides dealing with consistency issues of load-balancing and data replication problems in ranged-partitioned systems. The system handles two categories of issue in data replication, namely, *which* data should be replicated, and *where* to replicate data. A two-tier replication mechanism is used in order to improve load balancing and data availability for *ecStore*. Two kinds of replicas are considered for each data object – slave and secondary e replicas – in addition to its primary copy. The first tier of replication represents *K* level of replication for all data objects. The objective of the replication scheme at tier 1 is similar to the one in [59], which is to maintain the minimum replica creation and named secondary replicas, together with the primary copy for the data reliability requirement. Meanwhile, at the second tier, the additional replicas, called slave replicas, are associated with popular data objects in order to support frequently accessed objects for load balancing. When a flash crowd (a sudden increase in query requests) is faced by primary copy or secondary replica, the scheme will create slave replicas to help support the dynamic nature of workload pattern. By doing this, the minimization of all possible replication cost can be preserved. These costs may include replica consistency maintenance cost and replica storage cost. In addition, Figure7 illustrates the stratum architecture of the transactional Cloud storage considered in [61]. In *ecStore*, the storage system consists of three main stratums: a transaction management layer, a a replication layer and a distributed storage layer. The bottom stratum (storage nodes) is organized as a balanced tree-structured overlay and assigns a data range for each storage node based on BATON [63]. Meanwhile, in the middle layer, the structure of BATON is extended with a two-tier partial replication strategy, as previously discussed. Finally, the optimistic and multi-version protocol for concurrency is implemented in the transaction management module on the top stratum of this architecture.

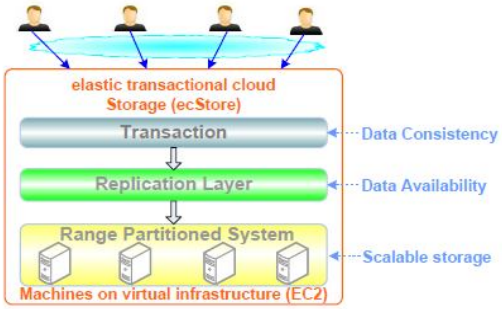

Figure 7: Stratum architecture of transactional cloud storage [61].

Similar concerns on both system availability and cost-efficiency are the focus in research work done in [64]. Specifically, this research proposed a new scheme on managing data replication in Cloud storage systems which can dynamically allocate the resources of a data Cloud to several applications in a cost-efficient and fair way. Three objective functions are simultaneously defined to address the problems in hand: (1) maximizing data availability, (2) minimizing communication cost, and (3) maximizing net benefit. With regard to the first objective, the proposed scheme provides high data availability by placing all replicas of any particular data to a set of different storage resources (servers) which may be geographically diverse. Further, to address the issue of high communication cost induced by this diversity, the second objective is achieved by maximizing data proximity among distributed replicas. Meanwhile, the last objective function is defined based on the fact that the operational cost of a server is influenced mainly by the query processing and communication overhead, its physical hosting, quality of the hardware, its storage, and the access bandwidth allocated to the server. That is, the net benefit is minimized by replacing expensive servers with cheaper ones, while maintaining a certain minimum data availability promised by SLAs to clients. The experimental results demonstrate the feasibility, the effectiveness and the low communication overhead of the proposed model.

5. SUMMARY AND REMARKS.

In this paper, a survey of data replication and resource management in distributed system is presented. In particular, we provide insight on the importance and how both data replication and resource management are facing their challenges in distributed systems, especially in the area of Grid and Cloud computing systems.

From the perspective of replication technique, we show that asynchronous replication is more desirable than synchronous replication in a highly dynamic large-scale distributed environment as it allows weaker consistency and is not required to satisfy *1-copy-serializability* (1SR) property. Meanwhile, from the resource management point of view, we show that resource selection is insufficiently addressed in the

utility-based computing environment, both in the area of Enterprise Grid and Cloud systems.

ACKNOWLEDGEMENT

This work is partially supported by Fundamental Research Grant Scheme (FRGS, Grant No: RR268, FRGS/1/2018/ICT03/UNISZA/02/1) under the Ministry of Education (MOE) and Universiti Sultan Zainal Abidin (UniSZA), Malaysia. The authors are also grateful and wish to acknowledge the support of all members of Centers for Distributed and High Performance Computing at University of Sydney and all staff of the Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin who have provided a vibrant and intellectually stimulating environment for this research.

REFERENCES

1. Goel, S. and R. Buyya, *Data Replication Strategies in Wide Area Distributed Systems," Enterprise Service Computing: From Concept to Deployment*. 2006: Idea Group Inc.
2. Jiménez-Peris, R., M. Patiño-Martínez, and B. Kemme, *Enterprise Grids: Challenges Ahead.* Journal of Grid Computing, 2007. **5**(3): p. 283-294. https://doi.org/10.1007/s10723-007-9071-y
3. Amza, C., A.L. Cox, and W. Zwaenepoel, *Distributed versioning: consistent replication for scaling back-end databases of dynamic content web sites*, in *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*. 2003, Springer-Verlag New York, Inc.: Rio de Janeiro, Brazil. p. 282-304. https://doi.org/10.1007/3-540-44892-6_15
4. Breitbart, Y. and H.F. Korth, *Replication and consistency: being lazy helps sometimes*, in *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 1997, ACM: Tucson, Arizona, United States. p. 173-184.
5. Elnikety, S., F. Pedone, and W. Zwaenepoel. *Database replication using generalized snapshot isolation*. in *24th IEEE Symposium on Reliable Distributed Systems, 2005 (SRDS 2005).* . 2005.
6. Irún-briz, L., et al. *Madis: A slim middleware for database replication*. in *Proc. 11th Euro-Par, LNCS 3648*. 2005: Springer. https://doi.org/10.1007/11549468_41
7. Kemme, B. and G. Alonso. *Don't be lazy, be consistent: Postgres-R, A new way to implement Database Replication*. in *International Conference on Very Large Data Bases (VLDB2000)*. 2000.
8. Lin, Y., et al., *Middleware based data replication providing snapshot isolation*, in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. 2005, ACM: Baltimore, Maryland. p. 419-430. https://doi.org/10.1145/1066157.1066205
9. Pacitti, E. and E. Simon, *Update propagation strategies to improve freshness in lazy master replicated databases.* The VLDB Journal, 2000. **8**(3-4): p. 305-318.
10. Pape, C.L., S. Gancarski, and P. Valduriez. *Refresco: Improving Query Performance through Freshness Control in a Database Cluster*. in *In Int. Conf. On Cooperative Information Systems (CoopIS)*. 2004. https://doi.org/10.1007/978-3-540-30468-5_13
11. Patiño-martínez, M., et al., *Middle-R: Consistent Database Replication at the Middleware Level.* ACM Transaction and Computational System, 2005. **23**.
12. Pedone, F., et al., *The Database State Machine Approach.* Distrib. Parallel Databases, 2003. **14**(1): p. 71-98.
13. Plattner, C. and G. Alonso, *Ganymed: scalable replication for transactional web applications*, in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*. 2004, Springer-Verlag New York, Inc.: Toronto, Canada. p. 155-174. https://doi.org/10.1007/978-3-540-30229-2_9
14. R¨ohm, U., et al., *FAS: A Freshness-Sensitive Coordination Middleware for a Cluster of OLAP Components*, in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*. 2002, VLDB Endowment: Hong Kong, China. p. 754-765.
15. Wu, S. and B. Kemme, *Postgres-R(SI): Combining Replica Control with Concurrency Control Based on Snapshot Isolation*, in *Proceedings of the 21st International Conference on Data Engineering*. 2005, IEEE Computer Society. p. 422-433.
16. Bernstein, P.A., V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*. 1987, Massachusetts: Addision-Wesley Publishers.
17. Connolly, T. and C. Begg, *Database Systems - A Practical Approach To Design, Implementation, and Management*. 2005: Addison Wesley.
18. Stockinger, H., et al., *File and Object Replication in Data Grids.* Cluster Computing, 2002. **5**(3): p. 305-314. https://doi.org/10.1023/A:1015681406220
19. Tatebe, O., et al. *Grid Datafarm Architecture for Petascale Data Intensive Computing*. in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.* . 2002.
20. Chervenak, A., et al. *Giggle: A Framework for Constructing Scalable Replica Location Services*. in *Conference on High Performance Networking And Computing*. 2002: IEEE Computer Society Press. https://doi.org/10.1109/SC.2002.10024
21. Lamehamedi, H., et al. *Simulation of dynamic data replication strategies in Data Grids*. in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2003)* 2003.
22. Lamehamedi, H., et al. *Data replication strategies in grid environments*. in *Fifth International Conference on Algorithms and Architectures for Parallel Processing, (ICA3PP 2002)* 2002.

23. Ranganathan, K. and I. Foster. *Design and Evaluation of Dynamic Replication Strategies for a High-Performance Data Grid*. in *International Conference on Computing in High Energy and Nuclear Physics*. 2001.
24. Yuan, Y., et al., *Dynamic Data Replication based on Local Optimization Principle in Data Grid*, in *Proceedings of the Sixth International Conference on Grid and Cooperative Computing*. 2007, IEEE Computer Society. p. 815-822.
25. Byoung-Dai, L. and J.B. Weissman. *Dynamic replica management in the service grid*. in *Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing, 2001*. . 2001.
26. Amjad, T., M. Sher, and A. Daud, *A survey of dynamic replication strategies for improving data availability in data grids.* Future Generation Computer Systems, 2012. **28**(2): p. 337-349. https://doi.org/10.1016/j.future.2011.06.009
27. Chervenak, A.L., et al., *Performance and Scalability of a Replica Location Service*, in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC '04)*. 2004, IEEE Computer Society. p. 182-191.
28. Vazhkudai, S. and J.M. Schopf. *Using Disk Throughput Data in Predictions of End-to-End Grid Transfers*. in *Proceedings of the third International Workshop on Grid Computing*. 2002.
29. Vazhkudai, S. and J.M. Schopf, *Using Regression Techniques to Predict Large Data Transfers.* International Journal of High Performance Computing Applications, 2003. **17**.
30. Bosio, D., et al., *Next-Generation EU DataGrid Data Management Services Computing*, ed. H.E.P.C. 2003). 2003.
31. Hoschek, W., et al., *Data Management in an International Data Grid Project*, in *Proceedings of the First IEEE/ACM International Workshop on Grid Computing*. 2000, Springer-Verlag. p. 77-90.
32. Uroˇs ˇ Cibej, Boˇstjan Slivnik, and B. Robiˇ, *The complexity of static data replication in data grids.* Parallel Comput., 2005. **31**(8+9): p. 900-912. https://doi.org/10.1016/j.parco.2005.04.010
33. Dullmann, D., et al. *Models for replica synchronisation and consistency in a data grid*. in *Proceedings. 10th IEEE International Symposium on High Performance Distributed Computing (HPDC '01)* 2001.
34. Yuzhong, S. and X. Zhiwei. *Grid replication coherence protocol*. in *Proceedings. 18th International Parallel and Distributed Processing Symposium (IPDPS '04)* 2004.
35. Abadi, D.J. (2009) *Data Management in the Cloud: Limitations and Opportunities*.
36. Gilbert, S. and N. Lynch, *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.* SIGACT News, 2002. **33**(2): p. 51-59.
37. Tim, K., et al., *Consistency rationing in the cloud: pay only when it matters.* Proc. VLDB Endow., 2009. **2**(1): p. 253-264.
38. Aiqiang, G. and D. Luhong. *Lazy update propagation for data replication in cloud computing*. in *5th International Conference on Pervasive Computing and Applications (ICPCA), 2010* 2010.
39. Voicu, L.C., et al. *Re:GRIDiT - Coordinating distributed update transactions on replicated data in the Grid*. in *10th IEEE/ACM International Conference on Grid Computing, 2009* 2009.
40. Voicu, L.C. and H. Schuldt, *How replicated data management in the cloud can benefit from a data grid protocol: the Re:GRIDiT Approach*, in *Proceeding of the first international workshop on Cloud data management*. 2009, ACM: Hong Kong, China. https://doi.org/10.1145/1651263.1651272
41. Voicu, L.C., et al., *Replicated data management in the grid: the Re:GRIDiT approach*, in *Proceedings of the 1st ACM workshop on Data grids for eScience*. 2009, ACM: Ischia, Italy. p. 7-16.
42. Voicu, L.C. and S. Heiko, *Load-Aware Dynamic Replication Management in a Data Grid* R. Meersman, T. Dillon, and P. Herrero, Editors. 2009, Springer Berlin / Heidelberg. p. 201-218.
43. Voicu, L.C. *Flexible Data Access in a Cloud Based on Freshness Requirements*. in *IEEE 3rd International Conference on Cloud Computing*. 2010. Miami, Florida https://doi.org/10.1109/CLOUD.2010.75
44. Krauter, K., R. Buyya, and M. Maheswaran, *A taxonomy and survey of grid resource management systems for distributed computing.* Software: Practice and Experience, 2002. **32**(2): p. 135-164.
45. Guy, L., et al., *Replica Management in Data Grids*. 2002, Global Grid Forum Informational Document, GGF5.
46. Cameron, D., et al., *Replica Management in the European DataGrid Project.* Journal of Grid Computing, 2004. **2**(4): p. 341-351.
47. *The DataGrid Project - The Data Grid Architecture*. 2001; Available from: http://eu-datagrid.web.cern.ch/eu-datagrid/.
48. AL-Mistarihi, H.H.E. and C.H. Yong, *Response Time Optimization for Replica Selection Service in Data Grids.* Journal of Computer Science, 2008. **4**(6): p. 487-493. https://doi.org/10.3844/jcssp.2008.487.493
49. Rahman, R.M., R. Alhajj, and K. Barker, *Replica selection strategies in data grid.* Journal of Parallel and Distributed Computing, 2008. **68**(12): p. 1561-1574.
50. Husni Hamad, E.A.L.M. and Y. Chan Huah, *On Fairness, Optimizing Replica Selection in Data Grids.* IEEE Trans. Parallel Distrib. Syst., 2009. **20**(8): p. 1102-1111.
51. Bell, W.H., et al., *Simulation of Dynamic Grid Replication Strategies in OptorSim*, in *Proceedings of the Third International Workshop on Grid Computing*. 2002, Springer-Verlag. p. 46-57.
52. Cameron, D.G., et al., *Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using OptorSim.* Journal of Grid Computing, 2004. **2**(1): p. 57-69.
53. Cameron, D.G., et al., *Evaluating Scheduling and Replica Optimisation Strategies in OptorSim*, in *Proceedings of*

the 4th International Workshop on Grid Computing. 2003, IEEE Computer Society. p. 52.
54. Chervenak, A., *The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets.* Journal of Network and Computer Applications, 2000. **23**(3): p. 187-200.
55. Zhong, H., Z. Zhang, and X. Zhang, *A Dynamic Replica Management Strategy Based on Data Grid*, in *Proceedings of the 2010 Ninth International Conference on Grid and Cloud Computing*. 2010, IEEE Computer Society. p. 18-23.
56. Dongsheng, L., et al. *Dynamic self-adaptive replica location method in data grids*. in *Proceedings. 2003 IEEE International Conference on Cluster Computing (CLUSTER '03)*. 2003. https://doi.org/10.1109/CLUSTR.2003.1253345
57. Qingsong, W., et al. *CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster*. in *IEEE International Conference on Cluster Computing (CLUSTER), 2010*. 2010.
58. Jalote, P., *Fault tolerance in distributed systems*. 1994: Prentice-Hall, Inc. .
59. Myint, J. and T.T. Naing, *Management of Data Replication for PC Cluster Based Cloud Storage System.* International Journal on Cloud Computing: Services and Architecture(IJCCSA), 2011. **1**(3): p. 31-41.
60. Shvachko, K., et al. *The Hadoop Distributed File System*. in *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010* 2010.
61. Vo, H.T., C. Chen, and B.C. Ooi, *Towards elastic transactional cloud storage with range query support.* Proc. VLDB Endow., 2010. **3**(1-2): p. 506-514.
62. *epiC project*. Available from: http://www.comp.nus.edu.sg/~epic/overview.html.
63. Jagadish, H.V., B.C. Ooi, and Q.H. Vu, *BATON: a balanced tree structure for peer-to-peer networks*, in *Proceedings of the 31st international conference on Very large data bases*. 2005, VLDB Endowment: Trondheim, Norway. p. 661-672.
64. Bonvin, N., T.G. Papaioannou, and K. Aberer, *A self-organized, fault-tolerant and scalable replication scheme for cloud storage*, in *Proceedings of the 1st ACM symposium on Cloud computing*. 2010, ACM: Indianapolis, Indiana, USA. p. 205-216. https://doi.org/10.1145/1807128.1807162