*I.J.Modern Education and Computer Science,* **2013, 9, 1-10**
Published Online October 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2013.09.01

# Mathematical Framework for A Novel Database Replication Algorithm

Sanjay Kumar Yadav
Dept. of Computer Science & Information Technology, Sam Higginbottom Institute Of Agriculture, Technology & Sciences- Deemed University, Allahabad, India
Email: yadav_sk@rediffmail.com

Gurmit Singh
Dept. of Computer Science & Information Technology, Sam Higginbottom Institute Of Agriculture, Technology & Sciences- Deemed University, Allahabad, India
Email: gurmitsingh3@rediffmail.com

Divakar Singh Yadav
Department of Computer Science & Engineering, Institute of Engineering and Technology, Lucknow, India
Email: divakar_yadav@rediffmail.com

***Abstract —*** In this paper, the detailed overview of the database replication is presented. Thereafter, PDDRA (Pre-fetching based dynamic data replication algorithm) algorithm as recently published is detailed. In this algorithm, further, modifications are suggested to minimize the delay in data replication. Finally a mathematical framework is presented to evaluate mean waiting time before a data can be replicated on the requested site.

***Index Terms*** — database replication, throughput, average delay

## I. INTRODUCTION

A database system is one of the computer systems which offer efficient data storage facilities to the applications. A database system is used to control the collection of data items. Database systems play a vital role in contemporary applications, such as administration, social sites, search-engines, and banking systems. Database systems offer abstractions; data consistency and concurrent data access, due to these database systems have got huge success in real world applications. A database system [1]

1) provides an interface which can be used to solve the problems of data storage and retrieval;

2) allows concurrent data access while maintaining data integrity;

3) survives server crashes or power failures without corrupting data ;

Scalability and performance are the key problems as the database system gets bigger. When database system increases from a smaller system to a larger system performance is degraded and at one point performance can become a bottleneck in the database system. Because of this, much research has been done in these areas of database systems [1]. Replication is one of the good ways to increase the performance of the database system by separating out the database by maintaining different servers. Workload on a single server can be decreased by maintaining the different database servers [2].

Replication is an efficient method to achieve optimized access to data and high performance in distributed environments [3]. Replication has been used in distributed computing for a long time [4]. Replication creates several copies of the original file (called replicas) and distributes them to multiple sites. This provides remarkably higher access speeds than having just a single copy of each file. Besides [4,5] it can effectively enhance data availability, fault tolerance, reliability, system scalability and load balancing by creating replicas and dispersing them among multiple sites. The three [6] fundamental questions any replication strategy has to answer are: When should the replicas be created? Which files should be replicated? Where the replicas should be placed? Depending on the answers, different replication strategies are born.

Ming Tang et al. suggested two replication algorithms in [7]: Simple Bottom Up (SBU) and Aggregate Bottom Up (ABU) for multitier data sites. The basic idea of these algorithms is to create the replicas as close as possible to the clients that request the data files with high rates exceeding the pre-defined threshold.

In [8] a Popularity Based Replica Placement (PBRP) algorithm was proposed. This algorithm tries to decrease data access time by dynamically creating replicas for popular data files.

Ruay-Shiung Chang et al. proposed a dynamic data replication mechanism in [9], which is called Latest Access Largest Weight (LALW). The design of the architecture is based on a centralized data replication management. LALW selects a popular file for

 

replication and calculates a suitable number of copies and grid sites for replication.

In [10] a dynamic data replication strategy called FIRE was proposed. In this method each site maintains a file access table to keep track of its local file access history.

In another paper [11] a new replication algorithm named Modified BHR was proposed. The proposed algorithm was based on the network level locality. The algorithm tries to replicate files within a region and stores the replica in a site where the file has been accessed frequently based on the assumption that it may require in the future.

As detailed above, related work in the data replication data throughput and average fetching delay are the two important parameters. In this work an existing PDDRA algorithm [12] is discussed and modifications are suggested to improve its performance. Finally a mathematical model is presented to obtain throughput and average delay.

The remainder of this paper is organized as follows: Section 1 of this paper describes the introduction to distributed database and different replication strategies; section 2 presents the overview of the database replication. The concept and context of database replication and design issues in distributed real time replicated database system detailed in section 3. In section 4, the existing PDDRA algorithm is detailed, with its limitation. Section 5 explains our proposed scheme and the mathematical frame work for a novel database replication algorithm. Conclusion and future work are given in the final section.

## II. OVERVIEW OF DATABASE REPLICATION

Replication is the method of sharing information so as to make sure of consistency between redundant resources, such as hardware or software components, to enhance reliability, defect-tolerance, or accessibility. It could be data replication if the same data is stored on multiple storage devices. Replication is the mechanism that automatically copies directory data from one directory Server to another. Fig. 1 shows the basic replication model. In this model user or client does not know the multiple physical copies of data exits. Data replication is a combination of database and distributed system. Database replication can be defined as the process of creation and maintenance of the duplicate copy of database objects in a distributed database system [13].

Using replication, copying of any directory tree or sub-tree (stored in its own database) could be done between servers. The Directory Server, holding the master copy of the information, automatically copies every update to all replicas, whereas computation replication of the same computing job is executed several times.

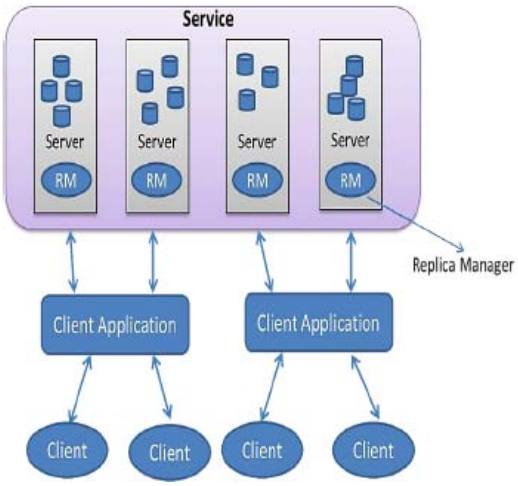


Fig. 1: basic data replication model

A computational job is typically replicated in space, i.e. executed on different devices, or it could be replicated in time, if it is executed again and again on a single device. The access to a replicated entity is generally uniform with access to a single entity which is not replicated. The replication itself should be transparent to an external user. The main features of database replication are as follows [14, 15]

**1) Database Locality**

This feature of database replication maintains the database locally so that geographically far distance users can access data with high speed. These users can access data from local servers instead of far distance servers because data access speed will be much higher than far distance area network. Providing database as closer to the user as possible contributes to higher performance of a system**.**

**2) Performance**

Database replication typically focuses on improving both read performance and write performance, while improving both read and write performance simultaneously is a more challenging task. When application is widely used across the large network but database is stored at a single server in that case database server can be a bottleneck of that system and the whole system slows down, i.e. slow response time and low request throughput capacity. Multiple replicas offer the system which serves the data in parallel.

**3) Availability and Fault Tolerance**

High availability of database requires low downtime of database system. In database systems there are two downtimes exit, first is planned and another one is unplanned. Planned downtime is incurred during the maintenance operation of all the software and hardware. Unplanned downtime can strike at any time and it is due to predictable or unpredictable failures such as hardware failures, software bugs, human error, etc. Downtime is usually the primary optimization area of database replication to increase the database availability. If a database item is stored at a single server and that

 

server does not respond or is down or it might have crashes. In that case database replication is the solution of this problem, to provide fault a tolerance database system.

### 2.1Types of Database Replication

The replica of database server can provide the data item to the users during server failure. This replica can also be used for restoring the data of failed servers. In this way database replication increases the data availability and forms a defect-tolerant system. [14] There are three different ways of Database replication:

**1) Snapshot Replication**

Data on one database server is plainly copied to another database server, or else to another database on the same server (Fig. 2). The snapshot replication method functions by periodically sending data in bulk format. Usually it is used when the subscribing servers can function in read-only environment and also when the subscribing server can function for some time without updated data. Functioning with un-updated data for a period of time is referred to as latency. Snapshot replication works by reading the published database and creating files in the working folder on the distributor. These files are named as snapshot files and contain the data from the published database as well as some additional information that will help create the initial copy on the subscription server [16].

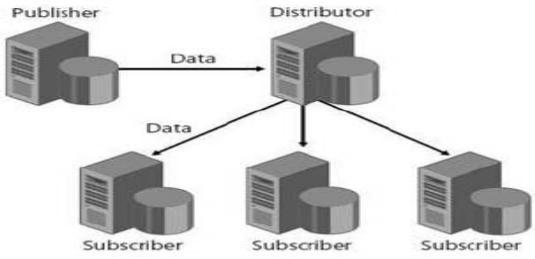


Fig. 2: schematic of snapshot replication

**2) Merger Replication**

Data from two or more databases is combined into a single database. Merge replication is the process of distributing data from Publisher to Subscribers, allowing the Publisher as well as Subscribers to make updates during connected or disconnected state, and then merging the updates between sites when they are connected. Merge replication allows distinct sites to work autonomously and at a later time merge updates into a single and uniform result. Merge replication includes default and custom choices for conflict resolution that you can define as you configure a merge publication. When a conflict happens, a resolve is invoked by the Merge Agent and determines which data will be accepted and propagated to other sites.

**3) Transaction Replication**

Users obtain complete initial copies of the database and then obtain periodic updates as data changes. In transactional replication, each committed transaction is replicated to the subscriber as it takes place. You can control the replication process so that it will either accumulate transactions or send them at timed intervals; or transmit all changes as they occur. Transaction replication is used in environments having a lower degree of latency and higher bandwidth connections.

Transactional replication requires a reliable and continuous connection, because the Transaction Log will grow quickly and if the server is unable to connect for replication it might not be manageable. Transactional replication begins with a snapshot. This snap shot sets up the initial copy. Later then, the copied transactions update that copy. You can choose how often to update the snapshot, or select not to update the snapshot after the very first copy. Once the initial snapshot has been copied, transactional replication, using the Log Reader agent, reads the Transaction Log of the published database and stores new transactions in the distribution Database. The transactions are then transferred from the publisher to the subscriber by the Distribution agent.

## III. CONCEPT OF REPLICATED DATABASES

To better understand the method behind Database Replication we start with the term “Replication” which represents the process of sharing information to ensure consistency between redundant resources, like software or hardware components, to improve reliability, accessibility or fault-tolerance. It could be data replication if the same data is stored on multiple storage devices or computation replication if the same computing task is executed many times [17]. The availability of certain replication databases could be improved by using Database mirroring. Support for combining transactional replication with database mirroring depends on which replication database is being considered. Peer-to-Peer replication in combination with database mirroring is not supportive. The replication agents that connect to the publication database can automatically fail over to the mirrored publication database. In the occurrence of a failure the agents that connect to the publication database will automatically reconnect to the new principal database.

The source, in a replication building block, is generally a database that contains data to be replicated. One database can be the source for various replication building blocks. Further, the source database can also serve as the target for another replication building block. Following example will make it clearer. The same pair of data stores swap roles in the Master-Master Replication pattern, (source becomes target, and target becomes source) for a common movement set that is updateable in either data store.

A computational task is typically replicated in space, i.e. executed on different devices or it could be replicated in time, if it is executed again and again on a individual device. The access to replicated entity is

 

typically uniform with access to a non-replicated, single entity. The replication itself should be transparent to an external user. Additionally, in a failure scenario, a failover of replicas is concealed as much as possible

Replication is the key characteristic in improving the availability of data distributed real time systems. Replicated data is stored at multiple server sites so that it can be accessed by the user even when some of the copies are not available due to server/site failures [18]. A Major restriction to using replication is that replicated copies must behave like a single copy, i.e. internal consistency as well as mutual consistency must be preserved, Synchronization techniques for replicated data in distributed database systems have been studied in order to increase the degree of consistency and to reduce the possibility of transaction rollback [19].

In replicated database systems, copies of the data items can be stored at multiple servers and a number of places. The potential of data replication for high data availability and improved read performance is crucial to DRTDBS. In contrast, data replication brings up its own complications. Access to a data item is no longer controlled exclusively by a single server; rather, the access control is distributed across the servers each storing a copy of the data item. It is essential to ensure that mutual consistency of the replicated data is provided; it must fulfill the ACID properties of database.

It is common to talk about active and passive replication in systems that replicate data or services. In Active replication, the same request is processed at every replica, while in passive replication, each single request is processed on a single replica and then its state is transferred to the other replicas. If at any time, one master replica is entitled to process all the requests, then we are discussing about the primary-backup scheme (master-slave scheme) predominant in high-availability clusters. On the other hand, if any replica processes a request and then distributes a new state, then this is a multi-primary scheme (in the database field called multi-master). In the multi-primary scheme, it is necessary to use some form of distributed concurrency control, like distributed lock manager.

Load balancing is different from task replication, as it distributes a load of different (not the same) computations across machines, and it allows a single computation to be dropped in case of a failure. Load balancing, however, sometimes uses data replication especially for multi-user internally, to distribute its data among machines.

To cope with the complexity of replication, the notion of group (of servers) and group communication primitives have been introduced [20]. The notion of a group acting as a logical addressing mechanism, allows the client to ignore the degree of replication and the identity of the individual server processes of a replicated service. Group communication primitives provide one-to-many communication with various powerful semantics. These semantics hide much of the complexity of maintaining the consistency of replicated servers. The two main group communication primitives are Atomic Broadcast (ABCAST) and View Synchronous Broadcast (VSCAST). We give here an informal definition of these primitives. A more formal definition of ABCAST and of VSCAST can be found in [21] and [22] respectively (see also [23, 24]). Group communication properties can also feature FIFO order guarantees.

Even though the process of Data Replication is used to create instances of the same or parts of the same data, we must not confuse the process of Data Replication with the process of backup since replicas are frequently updated and quickly lose any historical state. While Backup on the other hand, saves a copy of data unchanged for a long period of time.

Active replication, also called the state machine approach [25], is a non-centralized replication technique. Its key concept is that all replicas receive and process the same sequence of client requests. Consistency is made certain by assuming that, when supplied with the same input in the same order, the same output will be produced by the replicas. This assumption implies that servers process requests in a deterministic way. Clients do not contact one specific server, but address servers as a group. In order for servers to receive the same input in the same order, an Atomic Broadcast can be used to propagate the client requests to servers. Weaker communication primitives can also be used if semantic information about the operation is known (e.g., two requests that commute do not have to be delivered at all servers in the same order).

The main advantage of active replication is its simplicity (e.g., same code everywhere) and failure transparency. Failures are fully concealed from the clients, because if a replica fails; the requests are still processed by the other replicas. The major drawback of this approach is the determinism constraint.

The basic principle of passive replication, also named as Primary Backup replication, is that clients send their requests to a primary, which after executing the requests; sends update messages to the backups. The invocations are not executed by the backups, but apply the alterations produced by the invocation execution at the primary that is; updates. By doing this, no determinism constraint is necessary on the execution of invocations, the main disadvantage of active replication. Communication between the backups and the primary has to guarantee that updates are processed in the same sequence, which is the case if primary backup communication is based on FIFO channels. However, an only FIFO channel is not enough to ensure correct execution in case of failure of the primary. For example, consider that the primary fails before all backups receive the updates for a definite request, and another replica takes over as a new primary. Some mechanism has to ensure that updates sent by the new primary will be “properly” ordered with regard to the updates sent by the primary, which is faulty. VSCAST is a mechanism that guarantees these constraints can usually be used to implement the primary backup replication technique [26]. Passive replication can tolerate non-deterministic

 

servers (e.g., multi-threaded servers) and uses little processing power when compared to other replication techniques. However, when the primary fails, passive replication suffers from a high reconfiguration cost.

### 3.1 Context of Database Replication

Replication Techniques in Distributed Systems organizes and surveys the spectrum of replication protocols and systems that achieve high availability by replicating entities in failure-prone distributed computing environments.

The transaction level data can be duplicated to the replica database. The output is greater data integrity and availability. However, the increased availability is dependent on how independent the database replica is from the primary database. Replica independence must be taken in consideration in terms of disk spindles, disk controller, system, power supplies, room, city and building.

While data copying can provide users with local and much quicker data accessing, the problem is to provide these copies to users so that the overall systems operate with the same integrity and management capacity that is available within a centralized model. It is significantly more complicated to manage a replicated data than running against a single location database. It deals with all of the implementation and design issues of a single location and additionally with complexity of distribution, remote administration and network latency.

### 3.2 Issues in Distributed Real Time replicated Database Systems: Design Issues

#### 1) Replication Set Size

Decide whether to replicate a subset of a table, an entire table, or data from more than one table. This is a trade-off among the amount of data that changes the complexity of the link and the overall table size.

#### 2) Transmission Volume

To transmit, the right amount of the data should be chosen. The decision between sending all changes for any single row, or just the net effect of entire changes, is a key one.

#### 3) At the target, Replication Set Data Changes

If these have to occur and if the source wants to view the changes, then try to make the changes naturally non-conflicting to avoid the need for conflict detection and resolution.

#### 4) Replication Frequency

Decide the appropriate timing of the replication for the requirements and optimize the use of computing resources.

#### 5) Replication Unit

As explained earlier, a replication set consists of a group of replication units. Recognize the unit of data that will be transmitted to the target from the source. In the extreme requirements, this will be a transaction as it has been executed on the source. Easier to achieve (easily achievable) but a less precise requirement is to move a changed row. For environments with a big (huge, immense) risk of conflicts, it can also be a distinctive change in a cell within a record.

#### 6) Initiator

Decide whether the target pulls the data or the source pushes it, and make sure that throughout your replication topology these decisions do not cause later replication links to have problems meeting their operational requirements.

#### 7) Locking Issues

Verify that you can accept the locking impact of the replication on the source. If not, verify that a minor (small) decrease in consistency at a point in time is acceptable for the targets so you can avoid lock conflict.

#### 8) Replication Topology

The players, their roles and the overall integrity must be identified.

#### 9) Security

Ensure that the replicated data is treated with the right level of security at the target given the source security conditions. Along with (it), verify that your replication link is secure enough in the overall topology requirements.

#### 10) Key Updates

Verify whether the source allows updates to the key of records belonging to the replication set. If so, take special care for a consistent replication of such operations. Key updates are SQL updates to the columns of the physical key within a replication set. Such key updates must be handled particularly by the replication (Specially, the replications must handle such key updates).

#### 11) Referential Integrity

Verify whether the target has implemented referential integrity. If so, you need rules to stop (prevent) changes from the replication link being applied twice if the change triggers a target change in another replicated table.

## IV. RELATED WORKS

**PDDRA: Pre-fetching Based Dynamic Data Replication Algorithm [12]**

Replication is an efficient method to achieve optimized access to data and high performance in distributed environments Replication has been used in distributed computing for a long time. This technique appears clearly applicable to data distribution problems such as High Energy Physics community where several thousand physicists want to access the terabytes and

even petabytes of data that is produced every year. It is a reasonable way to make copies or replicas of the dataset and store these replicas among multiple sites because of the geographic distribution of the corporation in a data grid. Replication creates several copies of the original file (called replicas) among the data grid and distributes them to multiple grid sites. This provides remarkably higher access speeds than having just a single copy of each file. Besides it can effectively enhance data availability, fault tolerance, reliability, system scalability and load balancing by creating replicas and dispersing them among multiple sites.

Data replication not only reduces data access costs but also increases data availability in many applications. If the required files are replicated in some sites where the job is executed, then the job is capable of processing data without communication delay. However if the required files are not stored locally, they will be fetched from remote sites. This fetching takes a long time due to the large size of files and the limitation of network bandwidth between sites. Therefore it is better to pre-fetch and pre-replicate the files that are probable to be requested in near future. This will increase data availability.

In this section algorithm for data replication will be presented; this algorithm is based on pre-fetching [12]. For increasing system performance and reducing response time and bandwidth consumption it is better to pre-fetch some replicas for requester grid site, these replicas will be requested in the near future with a high probability and is better to replicate these files to requester node so the next time that the grid site needs them, it will access them locally, decreasing access latency and response time. The architecture of the algorithm is illustrated in Fig. 3. As shown in the Fig. 3, the grid sites are located in lowest level of the architecture. These grid sites consist of Storage and/or

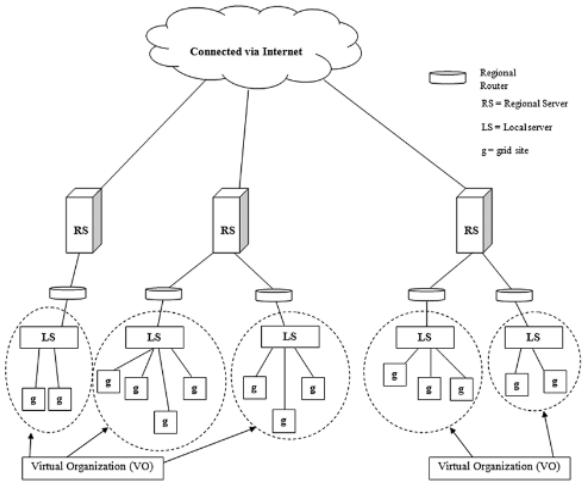


Fig. 3: existing pre-fetching based dynamic data replication algorithm (PDDRA)

Computing Element. Multiple grid sites constitute a Virtual Organization (VO), there is a Local Server (LS) for every Virtual Organization (VO) and the Replica Catalog (RC) is located at Local Server. It is worth mentioning that as available bandwidth between the sites within a VO is higher than bandwidth between Virtual Organizations. Hence accessing a file that is located in the current VO is faster than accessing the one that is located in the other VO. In the upper layer there is a Regional Server (RS) and each RS consists of one or more VOs. Regional Servers are connected through the internet, so transferring files between them takes a long time. There is also a Replica Catalog located at each RS that is a directory of all the files stored at that region. Whenever a file that is not stored in the current VO is required, the RC of RS is asked for determining which VOs have the requested file. Suppose that grid site 'A' requests a file that is not stored locally. Therefore it asks the RC to determine which sites have the requested file. For reducing access latency, bandwidth consumption and response time, it is better to pre-fetch replicas that are probable to be requested by the requester grid site in the near future. When a required file is not in the current VO and is stored in the other VOs, a request is sent to RS. Then RS searches on its Replica Catalog table and determines the locations of the requested file in other VOs. In such situations only the required file will be replicated and because of low bandwidth between VOs, high propagation delay time and consequently high replication cost, pre-fetching will not be advantageous and will not be done. In addition in this paper [17] the authors have assumed that members in a VO have similar interests of files so file access patterns of different VOs differ and consequently a file from different VO should not be pre-fetched for the requester grid site in other VO, because their requirements and access patterns are different. So only the required file will be replicated and pre-fetching will not be performed. The algorithm is constructed on the bases of an assumption: members in a VO have similar interest in files. For predicting the future accesses, past sequence of accesses should be stored. Files that will be accessed in the near future can be predicted by mining the past file access patterns. PDDRA consists of three phases:

**1) Phase 1: Storing file access patterns**

In this phase, file access sequences and data access patterns are stored in a database.

**2) Phase2: Requesting a file and performing replication and pre fetching**

In this phase a grid site asks for a file and replication is accomplished for it, if it is beneficial. Adjacent files of the requested file are also pre-fetched for the requester grid site in this phase.

**3) Phase 3: Replacement**

If there was enough space in storage element for storing a new replica, it will be stored; otherwise an existing file should be selected for replacement.

 

**Limitations of Existing PDDRA**

1) The PDDRA algorithm tries to minimize the access time using pre-fetching mechanism. However, due to the limited bandwidth of the access network sometimes it may not be possible to the fetch data as per our will, and request will be in queue, that leads to the further waiting and in turn will increase the replication time.

2) Members of VO may have different interests.

## V. PROPOSED SCHEME

1. In the proposed scheme the internet cloud will be considered as master node as it can be assumed that the data is available in the internet for the replication.
2. If any VO searches for any data first it will search in RS and then it will search in internet, if data is locally available at any RS then it will be replicated and there will not be any need to connect through the master node.
3. There is a possibility that the data may not be available at RS, hence, a simultaneous request is send to both RS and master node, if access of master node is in queue for let's say time $t_q$ then local search at RS will be done for time $t_s < t_q$.
4. The three phases of the above PDDRA will be implemented as explained above.

### 5.1 Simplified Mathematical Framework

As the replicated data is either available locally or it is available globally. Therefore, some of the generated request will be full-filled locally and leftover request will be fetched from internet (master node). In this section a mathematical framework is developed to estimate the average response time of all the transactions.

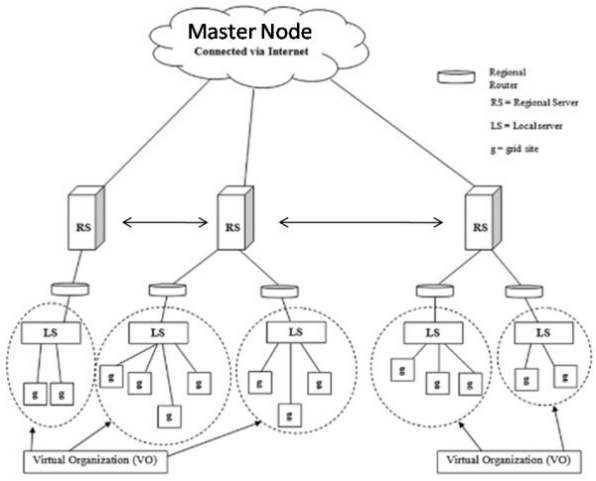


Fig. 4: modified pre-fetching based dynamic data replication algorithm (PDDRA)

Table 1: List of Parameters

| Parameters | Meaning |
|---|---|
| $n$ | Number of cites |
| $\eta$ | Transaction type |
| $\zeta$ | Percentage of transaction of type $i$ |
| $\lambda$ | Transaction arrival rate |
| $\mu_i$ | Mean service time for $i^{th}$ transaction |
| $p_i$ | Probability of local transaction execution |
| $t_{send}$ | Mean time to send a transaction type |
| $t_{return}$ | Mean time to return query result |

**Transaction Processing and Arrival Rates**

The arrival of the update and query transaction are random in nature. However, in most of the cases for a particular node it's a rare event. Hence, the arrival of the update and query transactions from every node is assumed to be a Poisson process, and then their sum is also Poisson. However, where the arrival of updates and query transactions are frequent, then, Bernoulli model can be used. Update transactions are assumed to be propagated asynchronously to the secondary copies. Furthermore, transactions are also assumed to be executed at a single site, either the local or a remote site.

The performance of replicated databases can be improved if the requirement of mutual consistency among the replicas of a logical data item is relaxed. Various concepts of relaxed coherency can be denoted by coherency conditions which allow calculating a coherency index $k \in [0,1]$ as a measure of the degree of allowed divergence. Small values of k express high relaxation, $k$ = 0 models suspend update propagation, and for $k$ = 1 updates are propagated immediately.

Taking locality, update propagation, and relaxed coherency into account, the total arrival rate of transactions of type $i, (1 \le i \le \eta)$, at a single site amounts to

$$\lambda_i^T = p_i\lambda_i + (n-1)(1-p_i)\lambda_i.\frac{1}{(n-1)} \qquad (1)$$

The first term $p_i$ describe a share of the incoming $\lambda_i$ transactions which can be executed locally, whereas the remaining transactions $(1-p_i)\lambda$ are forwarded to nodes where appropriate data is available. The other $n$-1 nodes also forward $(1-p_i)$ of their $\lambda_i$ transactions, which are received by each of the remaining databases

with equal probability $\frac{1}{(n-1)}$. The above formula simplifies to $\lambda_i^T = \lambda_i$

$$\lambda^{Tot} = \sum_{i=1}^{\eta} \lambda_i^{Tot} = \sum_{i=1}^{\eta} \lambda_i \quad (2)$$

The mean waiting time $\overline{W}$ at a local database is found to be:

$$\overline{W} = \frac{\sum_{i=1}^{\eta} \lambda_i^{Tot}.\mu_i^2}{1-\sum_{i=1}^{\eta} \lambda_i^{Tot}.\mu_i} \quad (3)$$

The mean waiting time at local database site is the time that user or transaction spends in a queue waiting to be serviced. Meanwhile, the response time is the total time that a job spends in the queuing system. In other words, the response time is equal to the summation of the waiting time and the service time in the queuing system. On average, a transaction needs to wait for $\overline{W}$ seconds at a database node to receive a service of $\mu_i$ seconds. Additionally, with probability $(1-p_i)$ a transaction needs to be forwarded to a remote node that takes $\overline{W}_C$ seconds to wait for plus the time to be sent and returned. Thus, the response time is given by

$$\overline{R}_i = \overline{W} + \mu_i + (1-p_i).(\overline{W}_C + t_{send}^i + t_{return}^i) \quad (4)$$

And the average response time over all transaction type results in

$$\overline{R} = \sum_{i=1}^{\eta} \zeta_i \overline{R}_i \quad (5)$$

## VI. CONCLUSION

In this paper, PDDRA (Pre-fetching based dynamic data replication algorithm) algorithm [12] as recently published is modified. Finally a mathematical framework is presented to evaluate mean waiting time before a data can be replicated on the requested site. In the future work, the simulation results will be presented to obtain the mean waiting time and throughput.

## REFERENCES

[1] R.Elmasri and S. B. Navathe. Fundamentals of Database Systems [B]. The Benjamin/Cummings Publishing Company, Inc., 1994.

[2] Fredrik Nilsson, Patrik Olsson. A survey on reliable communication and replication techniques for distributed databases [B].

[3] A. Dogan, A study on performance of dynamic file replication algorithms for real-time file access in data grids, [J] Future Generation Computer Systems 2009, 25 (8): 829–839 .

[4] R.-S. Chang, P.-H. Chen, Complete and fragmented selection and retrieval in data grids, [J] Future Generation Computer Systems, 2007, 23: 536–546.

[5] Y. air Amir, Alec Peterson, and David Shaw. Seamlessly Selecting the Best Copy from Internet-Wide Replicated Web Servers [C]. Proceedings of the International Symposium on Distributed Computing (Disc98), LNCS 1499, pages 22-33 Andros, Greece, September 1998.

[6] I. Foster, K. Ranganathan, Design and evaluation of dynamic replication strategies a high performance Data Grid, [C] in: Proceedings of International Conference on Computing in High Energy and Nuclear Physics, China, September 2001.

[7] M. Tang, B.S. Lee, C.K. Yao, X.Y Tang, Dynamic replication algorithm for the multi-tier data grid, [J] Future Generation Computer Systems 2005, 21 (5) : 775–790.

[8] M. Shorfuzzaman, P. Graham, R. Eskicioglu, Popularity-driven dynamic replica placement in hierarchical data grids, [C] in: Proceedings of Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008, 524–531.

[9] R.-S. Chang, H.-P. Chang, Y.-T. Wang, A dynamic weighted data replication strategy in data grids, [J] The Journal of Supercomputing , 2008, 45 (3) : 277–295

[10] A.R. Abdurrab, T. Xie, FIRE: a file reunion data replication strategy for data grids, [C] in: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, 215–223.

[11] K. Sashi, A.S. Thanamani, Dynamic replication in a data grid using a modified BHR region based algorithm, [J] Future Generation Computer Systems 2010, 27 :202–210.

[12] N.Saadat and A.M. Rahmani. PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids. [J] Springer: Future Generation Computer Systems, 2012, 28:666-681.

[13] Salman Abdul Moiz, Sailaja P., Venkataswamy G., Supriya N. Pal. Database Replication: A Survey of Open Source and Commercial Tools. [J] International Journal of Computer Applications (0975 – 8887) 2011, 13(6), 1-8.

[14] Heinz Stockinger. Data Replication in Distributed Database Systems, 1999 [B].

[15] Marius Cristian MAZILU, "Database Replication", [J] Database Systems Journal 2010, 1(2), 33-38.

[16] Mark A.Linsenbardt, Shane Stigler. McGraw-Hill/Osborne Media Book SQL Server2000Administration-Chap.10,'.Replication' [B].

 

[17]Microsoft MSDN Library - http://msdn.microsoft.com [W]
[18] B. Kemme, F. Pedone, G. Alonso, and A. Schiper. Processing transactions over optimistic atomic broadcast protocols. [C] In Proceedings of the International Conference on Distributed Computing Systems, Austin, Texas, June 1999.
[19] M. Raynal, G. Thia-Kime, and M. Ahamad. From serializable to causal transactions for collaborative applications. [R] Technical Report 983, Institut de Recherche en Informatique et Systèmes Aléatoires, Feb. 1996.
[20] K. P. Birman. The process group approach to reliable distributed computing. [J] Communications of the ACM, 1993, 36(12):37–53.
[21] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. [B] In S. Mullender, editor, Distributed Systems, chapter 5. adwe, second edition, 1993.
[22] Sanjay Kumar Tiwari et al. Distributed Real Time Replicated Database: [J] Concept and Design International Journal of Engineering Science and Technology (IJEST) ISSN: 0975-5462 230, 2011 3(6) 4839-4849.
[23] K. P. Birman and T. A. Joseph. Exploiting virtual synchrony in distributed systems. [C] In Proceedings of the 11th ACMSymposium on OS Principles, pages 123–138, Austin, TX, USA, Nov. 1987. ACM SIGOPS, ACM.
[24] K. P. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atomic group multicast.[J]ACM Transactions on Computer Systems, 1991, 9(3):272–314.
[25] F. B. Schneider. Implementing fault-tolerant services using the state machine approach:[J] A tutorial. ACM Computing Surveys, 1990, 22(4):299–319.
[26] R. Guerraoui and A. Schiper. Software-based replication for fault tolerance.[J] IEEE Computer, 1997, 30(4):68–74.

**Authors' Profiles**

**Sanjay Kumar Yadav:** is Assistant Professor of Computer Science in Dept. of Computer Science & Information Technology at Sam Higginbottom Institute Of Agriculture, Technology & Sciences" (Formerly Allahabad Agricultural Institute), (Deemed-to-be-University) Allahabad. He obtained batchelor degree in B.Sc.(Maths) from University of Allahabad, MCA degree from Institute of Engineering and Technology, Lucknow. M.Tech. in Software Engineering from Motilal Nehru National Institute of Technology Allahabad and pursuing his Ph.D. in Computer Science & IT at Sam Higginbottom Institute Of Agriculture, Technology & Sciences" (Formerly Allahabad Agricultural Institute), (Deemed-to-be-University) Allahabad. His research interest includes distributed system and mobile ad-hoc network.

**Prof. Gurmit Singh:** is Emeritus Professor of Computer Science in Dept. of Computer Science & Information Technology at Sam Higginbottom Institute Of Agriculture, Technology & Sciences" (Formerly Allahabad Agricultural Institute), (Deemed-to-be-University) Allahabad. He served the department as professor and Head for several years and retired in year 2012. He was also served the University as Dean, Shepherd School of Engineering & Technology and is on the program committees of the University. He is the author/co-author of several publications in technical journals and conferences. His research interest includes distributed system and mobile ad-hoc network, wireless sensor network and evolutionary computing.

**Prof. Divakar Singh Yadav:** is Professor of Computer Science at Institute of Engineering and Technology, Lucknow. He obtained B.Tech in Computer Science& Engineering, M.Tech in Computer Science from IIT, Kharagpur and Ph.D from University of Southampton, U.K. Before joining Gautam Buddh Technical University, Lucknow as Pro-Vice Chancellor, he was at South Asian University, New Delhi, an international university established by South Asian Association for Regional Cooperation (SAARC) nations, where he was Chairperson of Department of Computer Science at Faculty of Mathematics and Computer Science.
Dr. Yadav possesses more than 20 years of experience in academics/research in India and Abroad. Besides serving as member of several expert committees of U. P. Technical University, Lucknow, AICTE, New Delhi and Govt. of Uttar Pradesh, he also served as member of Advisory Boards, Technical Program Committees and reviewer for several international conferences /workshops /journals. He has long standing academic interests in database systems and distributed computing. His primary research interests are in formal methods, refinement of distributed systems using Event-B, verification of critical properties of business critical systems and reasoning about distributed database systems. He has also participated in prestigious Daghtuhl seminar at Schloss Dagstuhl-Leibniz Center for Informatics, Germany in 2006, in addition to

invitation at Commonwealth Scholarship Commission, U.K. seminar held at the University of the West England, Bristol in 2007. Dr. Yadav is author of four (04) books in the area of computers and information technology including best seller 'Foundations of Information Technology' published in 2001. His research contributions in the area of computer science and information technologies appeared in the international journals and refereed conference proceedings published by Springer-Verlag, Elsevier and IEEE.

 