Scientific Research and Essays Vol. 6(28), pp. 5966-5975, 23 November, 2011
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE11.210
ISSN 1992-2248 ©2011 Academic Journals

Full Length Research Paper

A novel design and development of persistence layer for heterogeneous synchronous replication in data grid

A. H. Beg[1], A. Noraziah[1]* A. N. Abdalla[2]

[1]Faculty of Computer Systems and Software Engineering, University Malaysia Pahang, 26300 Gambang, Kuantan, Pahang, Malaysia
[2]Faculty of Electrical and Electronic Engineering, University Malaysia Pahang, 26600 Pekan, Pahang, Malaysia.

Accepted 31 October, 2011

Data grid provided a way to solve the problems of large-scale data management. Data-intensive applications such as high-energy physics and Bioinformatics required both computational and data grid features. Replication is a process to achieved efficient and fault-tolerant access in the data grid. Usually, data replication schemes maintain an entire replica in every site where a file is replicated. It provided an important role in this involving world of the distributed database system. Replication technology should be efficient, thereby facilitates the maintenance costing. Most traditional replication technique is cost effective and also not considers the heterogeneous system. This paper presented a new model of persistence layer for synchronous replication and implemented based on multi-threading in heterogeneous environment. The main objective of this model is to make the persistence layer adaptive and make the replication process reliable and faster than other existing replication processes concerning cost minimization. In the proposed replication technique, the replication servers are operating system independent and the entire replication process is not inter dependent nevertheless on the main server. Adding a new replication server is easier than other processes. The technique also introduces the modification of replication servers without making impairment to the entire process. The performance of the proposed technique has been compared with SQL Server in terms of transactional inserts and synchronization time. The result shows that PLSR performs outstandingly than SQL server for transactional insert and synchronization in compare to time (seconds).

Key words: Data replication, data grid environment, synchronous, heterogeneous replication, multi-threading technique, persistence layer.

INTRODUCTION

The data grid mainly deals with large computational problems. It can provide geographically distributed resources for large-scale data-intensive applications that generate large scientific data sets. Therefore, it requires the modern scientific community to involve in managing the massive amounts of very large collection of geographically distributed data (Abdullah et al., 2008). Nowadays, in many fields, such as scientific experiments and technological applications, generate a huge amount of data. The appropriate use of data sharing and collaboration, these generated data should be shared and distributed in wide area networks. Therefore, the effective management of these wonderful sources of information shared and distributed is becoming a very important topic of scientific research and commercial applications. Therefore, data replication is a very useful technique to manage the large scale data across the widely distributed networks (Sriram and Cliff, 2010; Li and Shen, 2009; Lei et al., 2008). Several researches have been carried out regarding the replication process from the last decade. Among them were those by Ibison (2010), Pucciani et al. (2010), Tanga et al.

*Corresponding author. E-mail: noraziah@ump.edu.my. Tel: 09-5492121.

(2010), Lou et al.(2009), Sato et al. (2009), Tong and Shu (2009), Elghirani et al. (2007), Boyera and Hura (2005) and Ma et al. (2004). Those articles revealed that data replication in heterogeneous system are one of the current issues that still are unsolved in distributed system. Therefore; the study on this basis is initiated.

Data replication is a process that maintains multiple copies of data known as replicas. The replication can improve availability by providing access to data, even when some of the replicas are unavailable. Replication can also improve the performance as follows: i) reduce the latency, because users can use the nearby replicas therefore avoiding remote network access; ii) to accelerate the performance, since multiple computer is potential in serving the data simultaneously (Wahid et al., 2007; Sashi and Thanamani, 2011). Replication can improve the availability and performance in distributed systems. In data-centric distributed systems, replication, consistency and data integrity (constraint consistency) are used as correctness criteria (Osrael et al., 2007).

Data replications used in the various field such as bank, insurance, group of industries to protect their secure data to prevent unwanted crashes. Data replication in terms of duplication of data creates a backup copy of the data on the different servers. In the current enterprise software system, typically there used a persistence layer which persists in different current objects, which in terms help the application to avoid fault tolerance. So basically, on an enterprise system, replication helps to avoid fault of the data server system. Currently, the data replication system consist of the following impediments:

1. Usually replication process depends on the main server
2. Introducing the up-gradation of the replication process usually mute or pause the system for a routine of time
3. Fail or cashes of the main server, usually make the entire system stop working (For a database driven system)

This research has been focused to design and development of a Persistence Layer for Synchronous Replication (PLSR) in the data grid environment that support the heterogeneous system. The proposed layer is used as a multi-threaded application and which also provides an interface between the database and the main system. The persistence layer have a single thread which is responsible for making communication with the main server and have another thread running to manage the replicated databases. So it helps the entire system to reduce dependency of the replicated server on the main server. Replication server also used as a main server in the case of the crashes or fall of main server. Therefore, adding more replication servers are like plug and play features.

BACKGROUND

Data replication is a process to improve the performance of data access in a distributed system. Through this technique, an object request (read and write) will be accessed from multiple locations such as Local Area Network (LAN) or in the worldwide distributed network. For example, the results of a student in college will be read and updated by lecturers from various departments. The price of financial instruments will be read and updated from around the world (Noraziah et al., 2009; Chidambaram et al., 2008; Gu et al., 2006).

Data grid solution

In the data grid, data is typically replicated to improve the file access time and data availability. The recognition of data-intensive scientific applications, wherein millions of files are generated from scientific experiment and thousands of user's world-wide access this data, has resulted in the appearance of Grid computing. The resources of many computers, spanning geographic locations and organizations, are utilized to solve large-scale problems in the system (Lei et al., 2008).

The data transmission is inevitable though, the channel indicating the geographical distribution of data sources which also contains time, bandwidth consumption, and latency and consequently, cause the performance to decrease. So the serious problem is that, data grid is a massive resource of data objects. Data replication is a solution for this matter. The main objective of replication is to create multiple copies of files in different location to increase their data availability ((M.Mat Deris et al., 2004; Atakan, 2009; Khanli et al., 2011). Data replication is a practical and effective method to attain efficient fault tolerant data access in data grids. It can be static or dynamic. In static replication, location is predetermined and also well defined. Dynamic replication works is to create and delete replicas according to changing access patterns and therefore, ensures the benefits of replication even in behavior changes (Jose et al., 2010; Sashi et al., 2011; Bsoul et al., 2010).

Shena and Zhu (2009) proposed a proactive low-overhead file replication scheme, which is responsible for file replication among physically close nodes, low cost and consistency maintenance. Atakan (2009), studied performance of dynamic file replication algorithms for real-time file access in data grids. In their study, the performances of eight dynamic

replication algorithms are evaluated under various data grid settings. They were motivated to elaborate on the real-time performance of dynamic file replication algorithms. Data replication is usually use in data grid to enhance data availability and fault tolerance (Bsoul et al., 2010). However, replication should be used wisely because of the limitation of the storage size of each node that resides on the data grid.

Data replication can improve the job response time and data availability (Lei et al., 2008) has proposed a line optimizer algorithm that can minimize the data missing rate in order to maximize the data availability, In (Chang et al., 2007), job scheduling policy, named HCS (Hierarchical Cluster Scheduling), and a dynamic data replication strategy, called HRS (Hierarchical Replication Strategy) was implemented to improve the data access efficiencies in a cluster grid and evaluate this in various combinations of data access patterns.

Heterogeneous system

A Distributed Heterogeneous Computing (DHC) is a collection of autonomous dissimilar computing machines that are linked by a network and synchronized by software that functions as a single powerful computing facility. As computer tasks can be broken into different parts so is it possible to distribute the task for parallel execution. A DHC system has some advantages over homogeneous computing because some parts of an application perform better in some system, and some parts may perform better in another system (Boyera and Hura, 2005). The homogeneous computing system is easier to control as the processing times are not dependent and identical with an arbitrary distribution (Tong and Shu, 2009). Heterogeneous computing system can achieve both capability and capacity based jobs where capability based (aimed at minimization of the completion time of one big job) and capacity based (aimed at maximization the number of completions of small jobs within a given time) (Guest editorial, 2005).

Recently, heterogeneous systems have emerged as a major high-performance computing platform in parallel and distributed system. Chen et al. (2008) proposed an isospeed-efficiency model based on mark speed for heterogeneous computing. Chi et al. (2006) proposed App.Net.P2P architecture to implement effective content delivery on peer-to-peer networks for heterogeneous system. The main objective of their App.Net.P2P is to allow delivering intermediate objects to other peers as well as, the final presentations. Therefore, the recipient peers can share the intermediate objects and adapt their presentations for other peers using the associated service logic.

Persistence layer

Persistence layer is used in data dictionary and load balancing systems. It is used in simulation checkpoint and restart, and it is very important in parallel and distributed system. Qiao et al. (2006) described two frameworks: SPEEDES Persistence Framework (SPF) and Boost Serialization Library (BSL) and applied persistence framework in parallel and distributed systems. The main contribution of their work tested a C^{++} template based persistence framework BSL in a Parallel and Distributed Simulation (PADS) application.

In a collaborative computing environment, the collaborative applications required a simple and transparent persistence middleware to deal with complex data accesses. Wang et al. (2005) proposed a data persistence mechanism and implemented a persistence server, called Tree-Structured Persistence Server, which is known as TSPS. Their TSPS allowed states of collaborative applications to stored in a tree fashion beside tables. Their main goal is to develop the TPS to serve as a persistence layer in supporting the project of computer collaborative work.

Transactions

A transaction is a group that contains a set of tasks inherent of any part application that collects or manipulates data. SQL Server to ensure data integrity. This means that two users should not modify the same data item simultaneously. In the SQL server, a transaction is a single statement or multiple data modification language (DML) statements executed simultaneously. In a transaction all statements are treated as a group of work. If any statement fails, then the transaction is treated as a failed transactions and the whole transaction is rolled back. Accordingly, changes are not saved. On the other hand, if all of the statement succeed the transaction is treated as a succeed transaction and committed (Dewald, 2011; Poddar, 2003).

SQL server replication

SQL Server is defined as a Relational Database Management System (RDBMS) from Microsoft that is designed for the enterprise environments. It's run on Transact-SQL (T-SQL), a set programming extension from Sybase and Microsoft. The original SQL server code has been developed by Sybase in the 1980's, Microsoft. Sybase and Ashton-Tate have collaborated to produce the first product version, SQL Server 4.2 for OS / 2. Later, Sybase and Microsoft SQL Server provide the products. In November 2005, SQL Server

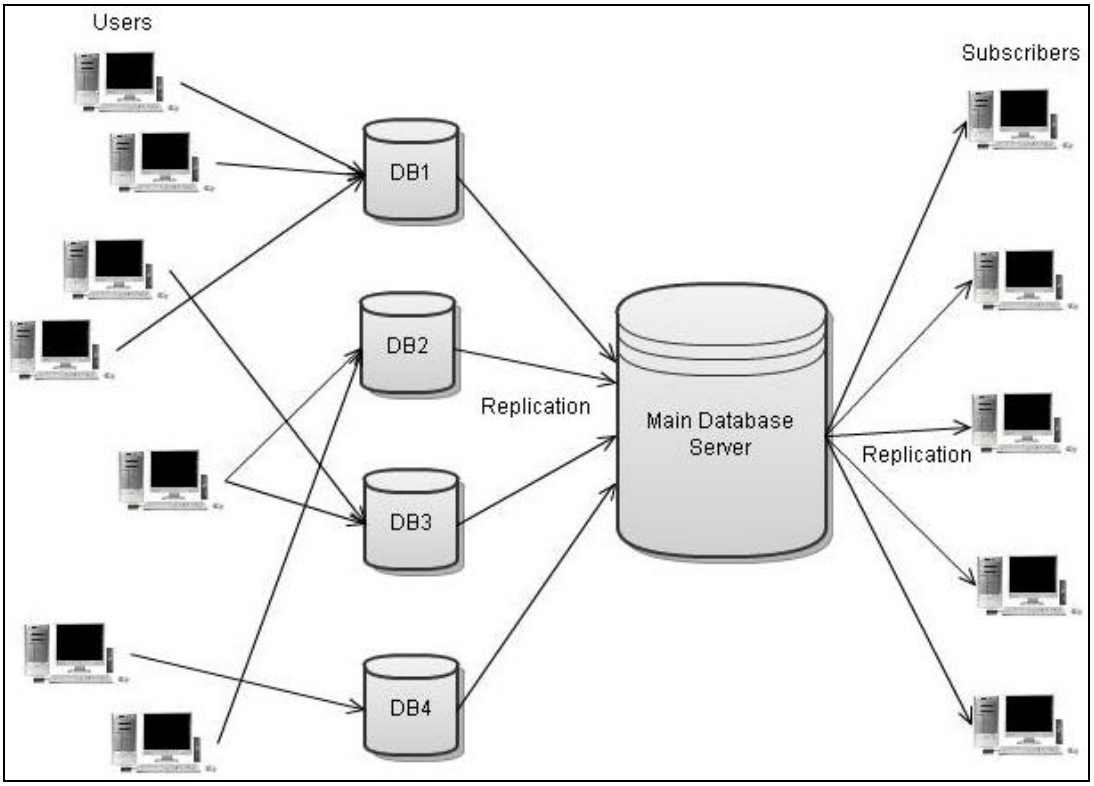

Figure 1. SQL server replication architecture.

Table 1. Basic comparison of SQL server replication time between transaction and merge insert.

# rows	Snapshot inserts	Sync time	Transactional inserts	Sync time	Merge inserts	Sync time
100	0	4	0	0	1	2
500	1	4	1	1	3	6
1000	2	5	2	1	5	12
5000	6	7	7	2	21	42
10000	13	12	26	5	42	96

2005 was released. The features of the products were to offer flexibility, scalability, reliability and security of database applications and were easier to build and deploy, and also reduced the complexity and tediousness involved in database management. This version also includes a more administrative support (SearchSQlserver.com, 2006).

Gutzait (2007) has described SQL server database design, replication design and architecture. The architectures shows that the databases are replicated with a common structure and the changes are replicated to the subscribers with fewer publications. During the replication consolidated information, users can publish a piece of information about the specific site. This allows users to create one publication and add the “where” clause according to the site or database code. The database servers are connected with the main server. The main database also connected with replication servers. Figure 1 show the SQL server replication.

Ibison (2010) described a basic comparison of SQL server replication times between Merge and Transactional insert and synchronization as shown in Table 1. The Table used a sample table having fixed width columns: The fixed-length data types were chosen because this table was also used to calibrate the problems of network bandwidth. The table data shows that for the inserts, merge takes significantly longer than transactional and snapshot. For a small number of insertions, transactional and snapshot are quite the same. When the number of row increases, transaction begins to take longer than the snapshot.

Conversely, there are a few rows with not much difference, but as there is increase in the number of rows, transactional and snapshot stay very similar, while the merge becomes much larger.

Multi-threading technique

Multi-threading is the ability to run multiple processor threads. It seems at the same time that CPUs are very fast in executing instructions. Modern PCs can run almost one billion instructions per second. Instead of running the same program for a second, the processor executes a program, perhaps a few hundred microseconds, and then switch to another and work for a short period and so on. It is also possible for an application to have multiple parts that run simultaneously. For example, a background task could be to respond to the mouse, while a file is loaded into RAM, and second task updates a progress bar on the screen (Bolton, 2011). Wang et al. (2007) presented a Software-based Redundant Multi-Threading (SRMT) approach for transient fault detection. Their technique used compiler to automatically generate redundant threads, and they can run on general-purpose chip multi-processors (CMPs). The result shows that their technique can provide a flexible program execution environment where flexible scheduling legacy binary codes and reliability-enhanced codes can co-exist in a fashion mix-and-match according to the desired level of reliability and software compatibility.

Wei et al. (2009) presented the Multi-Staged Engine (MSE) for high performance and flexibility in the application of concurrent continuous query processing, using the pipeline strategy and departs from the continuous query processing on three parallel phases: preprocessing, execution and dispatching modules to improve the parallelism with multi-threaded technology. They developed an algorithm multi-threaded (MTCNN) for k nearest massive continuous query processing. MTCNN algorithm uses parallel threaded workload and cache-conscious implementation of the reorganization to improve spatial and temporal locality.

PLSR MODEL DESCRIPTION

In the proposed system PLSR modifies the persistence layer to adopt multi processing use of multi-threading. The persistence layer is connected with different database servers, from those one will be the main server that will take care of Create Read, Update and Delete (CRUD) with the entire system and the several others are known as the replicated server. A thread with higher priority, that is; main thread keeps the consistent connection with the main server.

Thread is defined as the single stream of execution within a process. Programs that execute within its own address space are known as Process. The persistence layer creates one thread for each of the replication servers. One thread is the higher priority used for main server, and rest of all has the lower priority. As the main server is maintained by a high priority thread, thus the data should be saved or deleted or modified immediately with high priority. On the other hand, replication servers are maintained by a low priority thread. Consequently, the data transaction will be kept in a queue, and then it makes its own copy and does the transaction with that thread.

Programming implementation

The programming implementation has been developed using Java programming language. NetBean 6.9.1 has been used to write the source code of persistence layer. This is because NetBean is a comfortable Integrated Development Environments (IDE) in implementing Java source code. NetBean is developed and maintained by Sun Microsystems. After installing NetBean, it can be accessible to the start menu. NetBean has its own file format to maintain the source code. MySQL, SQLServer and MSAccess have been used as the database. To implement the SQL Query testing in MySQL, SQLyog (which is a free tool) was used. SQL Server has its own SQL IDE. To connect to the database from Java, JavaMySQL connector API was used, and to connect to MSSQL, JavaSQLServer connector API was used. Finally, Java can easily connect to MS Access from the user control panel.

Hardware and software components

The implementation of PLSR requires some minimum hardware and software specifications. To demonstrate PLSR system, prototypes across three replication servers as in Figures 2 and 3 are deployed. Each server or node has been connected to one another through a fast Ethernet switch hub. Theoretically, each of the replication servers and the main server has to be connected to each other logically. The hardware specifications as shown in Table 2 were used in each replication server for implementation.

The implementation of the PLSR was carried out by using Java programming language and has been deployed in different OS environment. Table 3 shows the system development tools specification for this implementation.

PLSR Environment

From the user's perspective, the functionality is offered by PLSR framework for heterogeneous system.

Table 2. Server main components specifications.

Hardware	Specifications
Processor	Intel (R) Core ((TM) 2 Quad CPU Q9650 @3.00 GHz 2.99 GHz
Memory	4.00 Gigabyte
Cache	3624 Megabyte
Hard disk	300 Gigabyte
Chip set	ATI Radeon HD 3450- Dell Optiplex
Network card	Intel (R) 82567 LM-3 Gigabit Network Connection

Table 3. System development tools specifications.

System development software	Specifications
Java	SE (Jdk 6.0)
SQLyog	Community edition 8.53
MySQL server	Version 5.0.89
Ubuntu	Version 10.0.4
NetBeans IDE	Version 6.9.1
Wine	Version 1.14
Windows vista	TM business
SQL server	Version 2008
MSAccess	Version 2007

system. Nevertheless, PLSR is tested in different server under the local area network (LAN) for this implementation. Two experiments have been done. In the first experiment, SQL server was the main server, while in the second experiment, MySQL server in Linux OS was the main server.

Experiment 1

In this experiment, the heterogeneous replication has been done with one main server and three replication servers. The entire servers are connected with the persistence layers. SQL server is the main server and connected with the persistence layer with high priority thread. Ms Access is the first replication server. MySQL in Windows OS is the second replication server and MySQL in Linux OS is the third replication server that is connected with the persistence layer with low priority thread as shown in Figure 2.

The host name and IP address for each server depicted in Table 4 Server A with IP address 172.21.202.232 is the main server. Server B with IP address 172.21.202.235 is the first replication server. Server C with IP address 172.21.202.231 is the second replication server, and Server D with IP address 172.21.202.230 is the third replication server.

Experiment 2

In this experiment, heterogeneous replication has been done with one main server and three replication servers. The entire server is connected with the persistence layers. MySQL in Linux OS is the main server and connected with the persistence layer with high priority thread. MS Access is the first replication server. MySQL in Windows OS is the second replication server and SQL Server is the third replication server that is connected with the persistence layer with low priority thread as shown in Figure 3.

On the other hand, in this experiment, the host name and IP address for each server is depicted in Table 5. Server A with IP address 172.21.202.230 is the main server. Server B with IP address 172.21.202.235 is the first replication server. Server C with IP address 172.21.202.231 is the second replication server and Server D with IP address 172.21.202.232 is the third replication server.

RESULTS

The proposed persistence layer synchronous replication (PLSR) has been compared with other replication processes in terms of replication time for transactional insert and synchronization. Table 6 shows the comparative time between SQL Server and PLSR replication server 1 and server 2 for insertion. At the first experiment, server 1 was the main server and at the second experiment, server 2 was the main server. The result demonstrates that, 1000 rows of

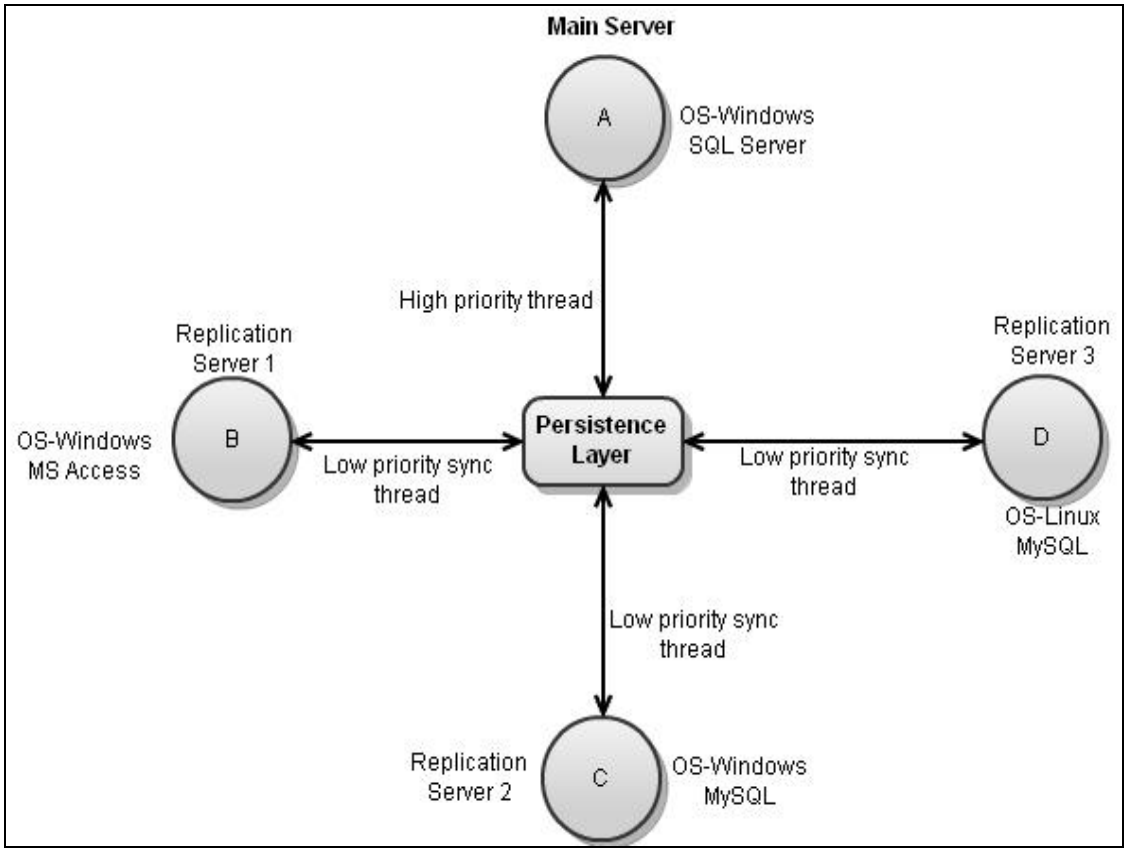

Figure 2. Data replication on heterogeneous system with 3 replication server and SQL server.

Table 4. The local IP address for each server based on SQL Server.

Server	Host Name	IP address	Operating System	Software
A	Main Server	172.21.202.232	Windows Vista	SQL Server
B	Replication Server 1	172.21.202.235	Windows Vista	MS Access
C	Replication Server 2	172.21.202.231	Windows Vista	MySQL
D	Replication Server 3	172.21.202.230	Linux (Ubuntu)	MySQL

data insertion, SQL server taken 2 s, where PLSR replication server 1 and server 2 taken 0.689 and 5.479 s respectively.

On the other hand, 5000 rows of data insertion, SQL server taken 7 s where PLSR main replication server 1 and server 2 taken 2.421 and 6.485 s respectively. Conversely it shows that in 10000 rows of data insertion, SQL server took 26 s where PLSR replication server 1 and server 2 took 4.754 and 7.508 s respectively. Table 7 shows the comparative time between SQL Server and PLSR replication Server 1 and Server 2 for synchronization. The result demonstrates that in 1000 rows of data synchronization, SQL server took 1 s, where PLSR replication server 1 and server 2 took 0.689 and 5.479 s respectively. On the other hand, in 5000 rows of data synchronization, SQL server took 2 s where PLSR replication server 1 and server 2 t00k 2.421 and 6.485 s respectively. Conversely, it shows that in 10000 rows of data synchronization, SQL server have taken 5 s where PLSR replication server 1 and server 2 took 4.754 and 7.508 s respectively. The total replication time (RT) has been calculated by using Equation (1) (Beg et al., 2011):

$$RT = \sum (TT+ST) \quad (1)$$

Here, RT represents the replication time, TT represents Transactional time and ST represents

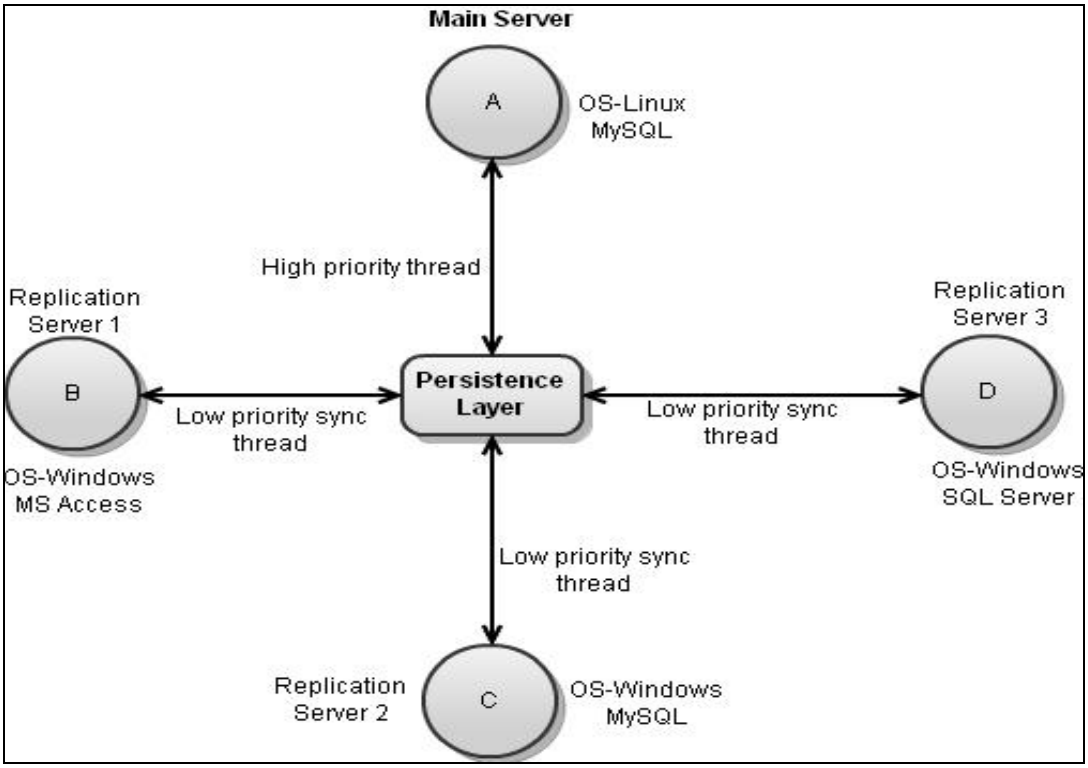

Figure 3. Data replication on heterogeneous system with three replication servers and MySQL server.

Table 5. The local IP address for each server based on MySQL server.

Server	Host name	IP address	Operating system	Software
A	Main server	172.21.202.230	Linux (Ubuntu)	MySQL
B	Replication server 1	172.21.202.235	Windows vista	MS access
C	Replication server 2	172.21.202.231	Windows vista	MySQL
D	Replication server 3	172.21.202.232	Windows vista	SQL server

Table 6. Transactional insertion time between SQL server and PLSR replication server 1 and server 2.

No. of rows	SQL server replication	PLSR replication server 1	PLSR replication server 2
100	0	0.375	5.129
500	1	0.459	5.385
1000	2	0.689	5.479
5000	7	2.421	6.485
10000	26	4.754	7.508

Synchronous time.

For 10000 rows of data insertion in SQL server, (Table 1)
RT = (26 + 5) s
= 31 s

For 10000 rows of data insertion in PLSR (Server 1), (Table 6).
RT = (4.754 + 0) s [ST=0, Because PLSR TT and ST done at the same time]
= 4.754 s

Table 7. Transactional synchronization time between SQL server and PLSR server 1 and server 2.

No. of rows	SQL server replication	PLSR replication server 1	PLSR replication server 2
100	0	0.375	5.129
500	1	0.459	5.385
1000	1	0.689	5.479
5000	2	2.421	6.485
10000	5	4.754	7.508

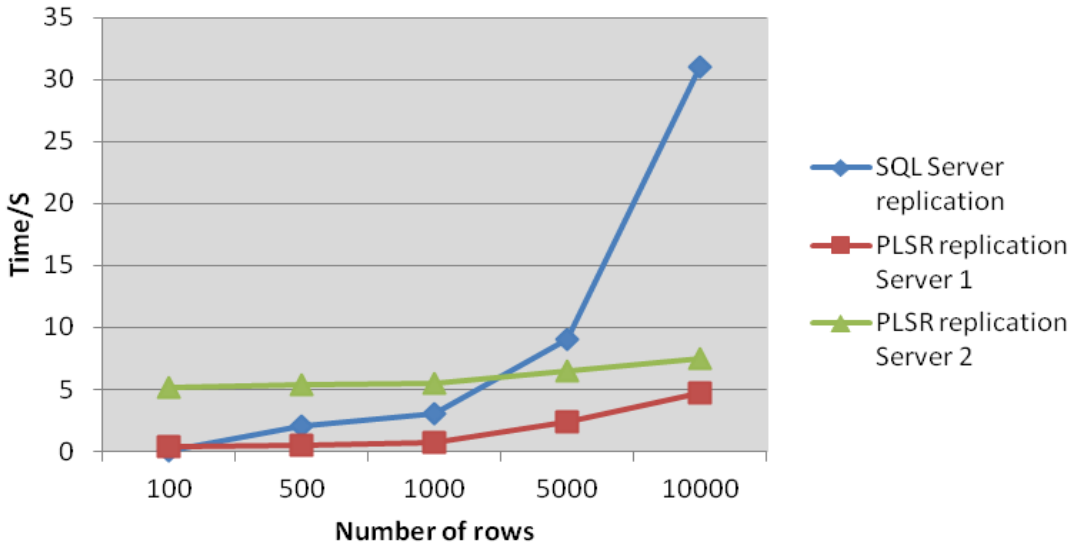

Figure 4. Total replication time between SQL server and PLSR replication server 1 and server 2.

The total replication time between SQL Server and PLSR replication server 1 and server 2 has been compared in Figure 4.

DISCUSSION

From the replication time, it shows that PLSR replication time is lower than SQL Server replication. As the number of data goes higher, SQL Server replication time gets much higher in comparison to the PLSR replication.

The motivation to compare the result of PLSR replication with SQL Server replication is the transactional replications which can alter the use of several trigger, similar to the proposed strategy, as the algorithm can perform rollback command from the persistence layer which can alter the result. The result shows that PLSR outstanding performs more than SQL server for transactional insertion and synchronization in comparison to SQL server replication in time per seconds. From the aforementioned result and the execution point of view, it can be found that PLSR is highly acceptable.

CONCLUSION

The data grid is a grid computing system to process and manage this very large amount of distributed data. At present, in the data grid community and clustering system, a lot of work has been focused on providing efficient and safe replication management services through designing of algorithms and systems. A lot of organizations used replication for many purposes. In this research, a new model PLSR has been proposed for managing data replication in the heterogeneous system. It can provide several advantages like, enterprise application, a more secured and reliable data transmission. In addition, one of the main goals is to make the database replication easier to handle. Thus, making it vastly configurable and the entire architecture, service oriented. It used latest technology trends and the replication done from the persistence layer. On the other hand, persistence layer is a part of a software engine and uses the latest customizable fourth generation language like Java. Therefore, a new era can move forward related to networking as well as database programming. Currently, PLSR is a support to only 3 types of

databases. Supporting all other popular databases can be another important improvement of PLSR. The developed prototype tool has few fields to insert data into different master and replication table. To make this more user friendly, various forms of input can be introduced. Therefore, many input type fields can be added in future.

ACKNOWLEDGEMENT

Appreciation conveyed to Ministry of Higher Education Malaysia for project financing under Fundamental Research Grant Scheme RDU100109; and University Malaysia Pahang under UMP Short Term Grant RDU080328.

REFERENCES

Abdullah A, Othman M, Sulaiman MN, Ibrahim H, Othman AT (2008). Towards a scalable Scientific Data Grid model and services. Int. Conf. Comput. Commun. Eng., pp. 20-25.

Atakan D (2009). A study on performance of dynamic file replication algorithms for real-time file access in Data Grids. Future Generation Computer Systems. 25: 829-839.

Beg AH, A. Noraziah , Abdalla AN, Mohd ZN, Sultan EI (2011). Synchronous Replication: Novel Strategy of Software Persistence Layer Supporting Heterogeneous System. 2nd International Conference on Software Engineering and Computer Systems, Pahang, Malaysia, ICSECS, Part 2 180, pp. 232-243, 2011.Springer-Verlag Berlin Heidelberg 2011.

Bolton D (2011). About.com Guide. Retrieved from http://cplus.about.com/od/glossar1/g/multithreading.htm(February 12, 2011).

Bsoul M, Khasawneh AA, Abdallah EE, Kilani Y (2010). Enhanced Fast Spread Replication strategy for Data Grid. Journal of Network and Computer Applications j.jnca.2010:12.006.

Boyera WF, Hurab GS (2005). Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments. J. Parallel Distr. Comput., 65: 1035-1046.

Chen Y, Sun X, Wu, M (2008). Algorithm-system scalability of heterogeneous computing. J. Parallel Distr. Comput., 68: 1403-1412.

Chi CH, Su M, Liu L, Wang HG (2006). An Active Peer-to-Peer System for Heterogeneous Service Provisioning. IEEE Int. Conf. Inf. Reuse Integrat., pp. 17- 22.

Chidambaram J, Rao PAN, Aneesh CS, Prabhu CSR, Wankar R, Agarwal A (2008). A Methodology for High Availability of Data for Business Continuity Planning / Disaster Recovery in a Grid using Replication in a Distributed Database. TENCON, IEEE Region 10 Conference, pp.1-6.

Dewald B, Kline K (2011). SQL Server: Transaction and Locking Architecture. Retrieved from http://www.informit.com/articles/article.aspx?p=26657.

Elghirani A, Zomaya AY, Subrata R (2007). An Intelligent Replication Framework for Data Grids. EEE/ACS Int. Conf. Comput. Syst. Appl., pp. 351- 358.

Gutzait M (2007). Simplify SQL Server replication. Retrieved from http://searchsqlserver.techtarget.com/tip/Simplify-SQL-Server-replication (June 23, 2010).

Guest editorial (2005). Heterogeneous computing. Parallel Computing, 31: 649-652.

Gu X, Lin W, Veeravalli B (2006). Practically Realizable Efficient Data Allocation and Replication Strategies for Distributed Databases with Buffer Constraints. IEEE trans. parallel distr. syst., 17(9): 1001-1013.

Ibison P (2010). Basic Comparison of Replication Times between Merge and Transactional Replication. Retrieved from http://www.replicationanswers.com/ReplicationTimesArticle.asp.

Jose M, Perez, Felix GC, Carretero J, Alejandro C, Fernandez J (2010). Branch replication scheme: A new model for data replication in large scale data grids. Future Gene. Comput. Syst., 26: 12-20.

Lei M, Vrbsky SV, Hong X (2008). An on-line replication strategy to increase availability in Data Grids. Future Gene. Comput. Syst., 24: 85- 98.

Li Z, Shen H (2009). A mobility and congestion resilient data management system for distributed mobile networks. 6th IEEE Int. Conf. Mobile Adhoc Sensor Syst. Macau., pp. 60-69.

Lou YS, Wang ZJ, Huang L, Yue L (2009).The Study of a Reflected Persistence Data Layer Framework.WRI World Congress Software Eng., pp. 291-294.

Ma D, Zhang W, Li Q (2004). Dynamic Scheduling Algorithm for Parallel Real-time Jobs in Heterogeneous System. The Fourth Int. Conf. Comput. Inf. Technol., pp. 462- 466.

Noraziah A, Deris MM, Saman MYM, Norhayati R, Rabiei M, Shuhadah WNW (2009). Managing Transaction on Grid-Neighbour Replication in Distributed System. Int. J. Comput. Math. Taylor Francis., 86(9): 1624-1633.

Osrael J, Froihofer L, Chlaupek N, Goeschka KM (2007). Availability and Performance of the Adaptive Voting Replication Protocol. The Second Int. Conf. Avail. Reliab. Secur. Vienna. pp. 53-60.

Poddar S (2003). SQL server transactions and error handling. Retrieved from http://www.codeproject.com/KB/database/sqlservertransactions.aspx (July 24, 2010)

Pucciani G, Domenici A, Donno F, Stockinger H (2010). A performance study on the synchronisation of heterogeneous Grid databases using CONStanza. Future Gene. Comput. Syst., 26: 820- 834.

Qiao H, Ju R, Li G, Huang K (2006). A New Persistence Framework for Parallel and Distributed Simulation. International on Multi-Symp. Comput. Computat. Sci., pp. 344- 348.

Sato H, Matsuoka S, Endo T (2009). File Clustering Based Replication Algorithm in a Grid Environment. 9th IEEE/ACM Int. Symp. Clust. Comput. Grid., pp. 204-211.

Sashi K, Thanamani AS (2011). Dynamic replication in a data grid using a Modified BHR Region Based Algorithm. Future Gene. Comput. Syst., 27: 202-210.

SearchSQlserver.com (2006) Definition SQL Server. Retrieved from http://searchsqlserver.techtarget.com/definition/SQL-Server (February 25, 2011).

Shena H and Zhu Y (2009). A proactive low-overhead file replication scheme for structured P2P content delivery networks, "J. Parallel Distrib. Comput., 69: 429-440.

Sriram I, Cliff D (2010). Effects of Component-Subscription Network Topology on Large-Scale Data Centre Performance Scaling. 15th IEEE Int. Conf. Eng. Complex Comput. Syst., pp. 72- 81.

Tanga X, Kenli L, Renfa L, Veeravalli B (2010). Reliability-aware scheduling strategy for heterogeneous distributed computing systems. J. Parallel Distrib. Comput., 70:941-952.

Tong X, Shu W (2009). An Efficient Dynamic Load Balancing Scheme for Heterogenous Processing System. IEEE Conf. Comput. Intell. Nat. Comput., pp. 319- 322.

Wahid SAW, Andonie R, Lemley J, Schwing J, Widger J (2007). Adaptive Distributed Database Replication Through Colonies of Pogo Ants. Parallel Distrib. Process. Symp. IPDPS. IEEE Int., pp. 1-8.

Wang CH, Kim H, Wu Y, Ying V (2007). Compiler-Managed Software-based Redundant Multi-Threading for Transient Fault Detection. Int. Symp. Code Gene. Optimiz. pp. 244- 258.

Wang CM, Chen HM, Lee GC, Wang ST, Hong SF (2005). A Tree-Structured Persistence Server for Data Management ofCollaborative Applications. IEEE Int. Conf. Adv. Inf. Network. Appl., pp. 503 -506

Wei L, Ping WX, Qi Z, Nong Z (2009). Improving Throughout of Continuous k-Nearest Neighbor Queries with Multi-threaded Techniques. IEEE Int. Conf. Intell. Comput. Intell. Syst., pp. 438- 442.