Benchmarking Data Replication Performance for The Defense Integrated Military Human Resources System

Venkata Mahadevan, Mahdi Abdelguerfi, Shengru Tu, Golden Richard
Department of Computer Science
University of New Orleans
New Orleans, LA 70148

Abstract

The Defense Integrated Military Human Resources System (DIMHRS) is an initiative by the Department of Defense (DoD) to develop a web-based personnel and pay management system. The participant subsystems of DIMHRS and its intended users are highly geographically distributed throughout all regions of the world. This varied distribution of subsystems and users over vast distances requires some mechanism to ensure that services are delivered to users with reasonable performance and reliability. The deployment of data replication in DIMHRS is intended to provide users with fast, reliable access to data regardless of their location. Several different data replication schemes can be applied to achieving this goal. The purpose of our research is to define a set of benchmarks that measure various aspects of data replication performance and evaluate the effectiveness of particular data replication schemes under simulated test environments designed to match expected real-world DIMHRS usage. We conclude with a recommendation on the data replication scheme(s) that we feel best serve the aforementioned goals.

1. Introduction

Data replication technology makes it possible to provide users with rapid, dependable access to data regardless of their location on the globe. This makes it especially attractive for deployment in large distributed information systems where high availability and reliable access to data is critical. The Defense Integrated Military Human Resources System (DIMHRS) is such a system. It is an initiative by the Department of Defense (DoD) to develop a web-based personnel and pay management system that will be used by all branches of the military. The participant subsystems of DIMHRS and its intended users are highly geographically distributed throughout all regions of the world. This varied distribution of subsystems and users over vast distances requires some mechanism, namely data replication, to ensure that services are delivered to users with reasonable performance and reliability. Several different data replication schemes can be applied towards achieving this goal and various Commercial Off The Shelf (COTS) replication products are available that provide the necessary functionality. The goal of our research is to define a set of benchmarks that measure various aspects of data replication performance and evaluate the effectiveness of particular data replication schemes under simulated test environments designed to match expected real-world DIMHRS usage[1].

There are two major kinds of data replication: primary-site replication and multiple-site replication [1]. Primary-site replication involves the copying of data from a primary ("master") site to many target sites ("data stores"). Data is modified only at the primary site. For applications that require greater flexibility in updating data, multiple-site replication should be considered. In multiple-site replication, there can be numerous master sites that can contain complete sets or subsets of data which can be accessed and modified at any of the sites. Modifications to the data are propagated to all the other sites using synchronous or asynchronous propagation. In the context of DIMHRS, it is inevitable that multiple-site replication should be used as there are many scenarios in the military where it will be necessary to have read/write access to data subsets. For example, disconnected operation in mobile units such as submarines should be possible.

Several mechanisms exist for replicating data across multiple sites. These are briefly reviewed below:

[1] This work was funded in part by Science & Engineering Associates, Inc (SEA). The views expressed herein are those of the authors and not of SEA.

Snapshots: These are locally stored copies of table data from a master site. Snapshots may be read-only (immutable) or updateable. The primary advantage of snapshots is that they do not need to contain a complete set of data from a master site. Snapshots can be comprised of only certain rows or columns of a table. For example, consider a table that contains two columns: the first column is plain text while the second column is a Binary Large Object (BLOB). To save storage space at the remote site (which may be a mobile unit with limited storage capacity), the snapshot at the remote site can be comprised of only the first column of the table as the BLOB may have considerable storage requirements. A snapshot can be refreshed by pulling changes in table data from the master table that it is associated with. An updateable snapshot can push changes to its associated master table.

Synchronous multiple master replication: Multiple master replication in general entails executing transactions that modify data across both local and replicated copies of data. In a synchronous multiple master replication scheme, local transactions that modify data marked for replication are not committed until corresponding replicated copies of that data are updated first i.e. all replicated copies are updated within the same transaction. If the number of replicated copies is large, such a scheme may result in extremely poor performance. Furthermore, if for some reason a replicated copy cannot be updated, then none of the copies, including the local (master) copy, are updated.

Asynchronous multiple master replication: This scheme attempts to compensate for the deficiencies inherent in the previous scheme by separating transactions from the replication process. If a transaction updates a local replica of a table, the other replicas of that table are not updated within the same transaction. Instead, the other replicas of the table are updated when the replicator (data replication software) is scheduled to perform a refresh at some preset interval or time. This type of replication scheme can often provide more effective utilization of network resources and cause less database contention than a synchronous replication scheme [2].

As mentioned earlier, the use of data replication technology in DIMHRS in intended to provide services to users quickly and reliably. The primary advantages of replication that are most useful in the framework of DIMHRS include:

Increased availability of data: This is especially useful for users who may be occasionally disconnected (e.g., mobile users). For example, an application on the mobile unit can access a local database when a network connection to a remote database server is not available. Or, since data is replicated at more than one site, if a site experiences system failure, an application can still obtain the data it needs from another site. In the case of DIMHRS, which is a highly geographically distributed system, replication can also provide improved performance to users in different regions of the world by providing them with access to databases that are closest to them.

Improved security: An organization can choose to keep their complete database on a highly secure system while non-classified data may be replicated to various other sites. All current replication packages support partitioning of data -- the ability to select only certain rows (horizontal) or columns (vertical) from a table to replicate.

Information transport: Data can be periodically transported from an OLTP database to an OLAP data warehouse.

Information off-loading: If the speed of transaction processing is important, a copy of the database can be created to handle decision support applications separately.

Unfortunately, despite the many advantages of replication aforementioned, there are several issues to consider before deployment:

Latency: This is the lag introduced between when an application modifies data on a local database and when the changes are replicated to the remote replicas. To ensure that the users of DIMHRS are provided with up-to-date data on a consistent basis, it is important to determine how much latency is acceptable for the applications that support DIMHRS to function adequately.

Transaction integrity: A single transaction can update many rows from many tables on a database. The order of the updates must be maintained by the replicator and the updates must be performed in that order on all replicas. This will ensure that transaction integrity is enforced and updates at the replicas are not rejected because of referential integrity constraints.

Replication conflicts: In an advanced replication environment like DIMHRS, several kinds of replication conflicts can transpire under certain scenarios. For instance, if two or more different sites attempt to replicate an update to the same row at the same time, data inconsistencies could result [3].

The remainder of this paper will focus on the performance aspects of data replication as they pertain to the DIMHRS replicated environment and is organized as follows. In Section 2 we describe the fundamental design of the DIMHRS replicated environment. Section 3 discusses the tools selected to provide data replication facilities in our simulated test environments and the miscellaneous tools chosen for use in the benchmarking

process. In Section 4 we explain the methodology used to experimentally gauge data replication performance. An analysis of the results obtained from benchmarking is provided in Section 5. In Section 6, we conclude with a recommendation on the data replication schemes that are most appropriate for use in DIMHRS and suggestions for future work.

2. Overview of the DIMHRS Replicated Environment

Figure 1 shows the fundamental design of the DIMHRS replicated environment. There are 3 clearly distinguishable layers comprising this replicated environment. These are:

Data warehouse (OLAP): The central store of all DIMHRS data. Data from the lower layers is eventually propagated to this site.

Regional OLTP master sites: There are 4 such sites, one for each region of the world where DIMHRS usage is expected to be high. Each site contains data pertaining to its regional location. No replication of data occurs between these sites although they are connected together by a high-speed bus. They are linked to the data warehouse via high bandwidth network connections through which replication of data occurs.

Snapshot sites: There are many snapshot sites containing subsets of data from the regional master sites. Snapshots are linked to regional master sites via low to moderate bandwidth network connections. End users and applications will most likely interact with the system at this level.

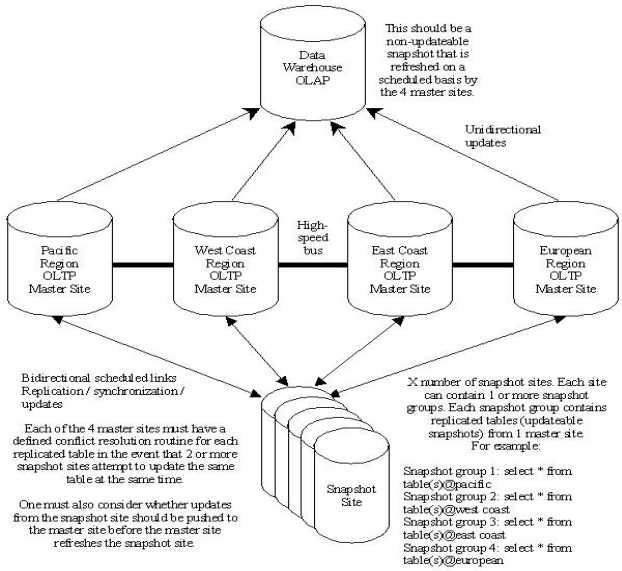

Figure 1. DIMHRS Replicated Environment

Replication software running at every site (snapshot or regional master) is responsible for the propagation of changes to replicated data within the environment.

3. Tools Selected for Use in the Benchmarking Process

In order to construct simulated test environments for use in benchmarking data replication performance in DIMHRS, a tool that provided a full suite of data replication features for a RDBMS was needed. Oracle Corporation's Oracle8i Enterprise Edition v. 8.1.6 was selected as the RDMBS for use in our simulations. This narrowed the options for selecting the replication means to products that were compatible with it. Oracle8i itself contains its own trigger-based, bi-directional replication facilities collectively called Oracle Advanced Replication. The data replication schemes supported by Oracle Advanced Replication include: synchronous and asynchronous immutable and updateable snapshot replication, synchronous and asynchronous multiple-master replication, or any combination of the two. Flexible horizontal and vertical partitioning of data, guaranteed transaction integrity, automated conflict detection and resolution routines, and fault tolerance provided via the Oracle8i RDBMS round out the list of desirable features. In addition, complete application independence is possible because all replication is done at the database level. Other tools that were required are as follows:

Database access and programming language: SQL is the standard language for retrieving and modifying information from a RDBMS. The selection of the API for interfacing an application with a RDBMS, however, is directly related to the programming language in which the application is written. Java was chosen because of its inclusion of the Java Database Connectivity (JDBC) API, which makes it easy to rapidly develop and deploy portable applications.

Network statistics monitor: In any application that entails transmission of data across a LAN, it is often useful to collect information about the state of the LAN such as bandwidth utilization and traffic level (in packets or bytes). Since it is expected that data replication will generate measurable network traffic, this information could be useful in evaluating the kind of network conditions that data replication operates best under. For this purpose, LANScan Software's Network Monitor was used. The most attractive feature of this software is its

capability to measure network bandwidth usage and traffic across heterogeneous environments.

Using COTS software products as building blocks has been a strong trend in businesses and organizations, as highlighted by the COTS-based systems initiative at the Carnegie Mellon Software Engineering Institute [5]. The advantages of using COTS are numerous, including market-tested reliability, market-approved features, and an opportunity for expanding software capabilities and improving system performance by the commercial marketplace [4].

4. Benchmarking Methodology

The first step in the benchmarking process was to generate test data. A subset of the DIMHRS Implementation Data Model that will be used in the final production system was used as the basis for data generation. The subset consists of 14 tables that contain information about military personnel. The PER (Person) table is the central table in this data model. It contains all the information pertaining to a person such as name, birth date, and a description field holding some miscellaneous data about the person. The additional tables hold data pertaining to a person's military affiliation such as the organization they belong to, skills they possess, and their occupation and position. The PER table is shown in Figure 2.

Due to the fact that the actual DIMHRS database contains highly confidential information about military personnel, no actual data could be used for data generation purposes. Therefore, the entire database had to be comprised of dummy data. Data for each field in every table of the data model had to be composed of random letters and numbers (depending on the field type). For this purpose, a Java application was developed to interface with the database via JDBC and automatically generate 500,000 random personnel records. The test database size was approximately 1 GB.

PER_Id:	NUMBER
PER_Last_Nm: VARCHAR2(50)	
PER_Frst_Nm: VARCHAR2(50)	
PER_Mid_Nm: VARCHAR2(20)	
PER_Birth_Clndr_Dt:	DATE
SEX_CAT_Cd:	VARCHAR2(1)
PER_Ethnic_Affnty_Cd:	VARCHAR2(1)
PER_Adult_Dpndnt_Qy:	INTEGER
PER_Mnr_Dpndnt_Qy:	INTEGER
PER_Total_Dpndnt_Qy:	INTEGER
PER_Dscrptn_Text:	

Figure 2. The PER (Person) Table

3 different test environments were constructed to simulate real world DIMHRS usage. These are as follows:

Environment 1: This environment is intended to simulate a single link between a regional master site and data warehouse. Refer to Figure 1 to notice where this environment fits into the larger DIMHRS Replicated Environment framework. The regional master site (local test database) was set up on a Pentium III, 450Mhz PC with 128MB RAM and an adjustable 2Mbps wireless Ethernet card. The Data Warehouse (replicated remote database) was set up on a Pentium III, 800Mhz PC with 256MB RAM and a 10Mbps Ethernet connection. The local test database accepts updates from an application. The replication software running at both sites then propagates the changes to the replicated remote database. The way that changes are propagated depends on the replication scheme configured in the replication software.

Environment 2: This environment is similar to the first one, but the speed of the network connectivity between the two sites is vastly reduced. Since the primary goal of this environment is to evaluate various replication schemes under low to moderate speed network connections, infrared devices (IRDA) that allow for various low to moderate bandwidth connections to be simulated were used as the network medium instead of the high speed wireless and wired Ethernet connections used in environment 1. An updateable snapshot site (local test database) was set up on a Pentium III, 450Mhz PC with 128MB RAM and an adjustable IRDA link. A regional master site (replicated remote database) was set up on a Pentium II, 450Mhz PC with 256MB RAM and an adjustable IRDA link. TCP/IP runs on top IRDA to provide reliable networking. As in the first environment, the local test database accepts updates from an application and the replication software running at both sites then propagates the changes to the replicated remote database. In the context of the greater DIMHRS Replicated Environment framework, this environment sits between the bottom and middle layers.

Environment 3: This environment is intended to simulate DIMHRS usage under a production quality scenario. 3 SUN SparcServer20 machines (256MB RAM, 10Mbps Ethernet) each represent a regional master site and a Sun Enterprise 420R (4GB RAM, 10Mbps Ethernet) machine represents the data warehouse. In the context of the greater DIMHRS Replicated Environment framework, this environment is intended to simulate the top 2 layers. There are, however, 2 important differences. There is a

bidirectional link between the data warehouse and each of the regional master sites and the data warehouse is not restricted to being a non-updateable snapshot. The reason for this is that in our benchmarking process, it would be beneficial to be able to actually update data at the data warehouse and have it replicated to the other sites. Therefore, instead of just benchmarking data replication performance in one direction (from the regional master sites to the data warehouse), it would be possible to also measure data replication performance in the opposite direction i.e. from the data warehouse to the regional master sites.

The 2 most important factors in measuring overall data replication performance in DIMHRS were determined to be replication latency and the number of transactions that have been propagated from a local database to its remote replica(s) per unit time. It is fairly straightforward to understand why knowing the replication latency of a particular data replication scheme under different test environments is crucial in determining the overall performance of a replicated environment. If the replication latency is too large, updates to replicated data will not be propagated to remote replicas in a timely fashion and any users retrieving data from these remote replicas may be presented with stale data. For certain applications and users this may not be such a dilemma, but for mission critical applications that require the latest updates in near real-time, keeping the replication latency within certain parameters is necessary. Counting the number of transactions that have been propagated per unit time in a particular replication scheme under different test environments is useful in ascertaining the efficiency of the scheme, environment, and the replication software itself. For instance, if there is heavy transaction activity at a local database and the remote replicas need to be updated in near real-time, a replication scheme that propagates the transactions as fast as possible would satisfy this scenario better than a scheme that can propagate fewer transactions per unit time. Of course, other conditions such as system hardware and network bandwidth availability also play a role in contributing to the overall number of transactions propagated.

In addition to collecting the above statistics, the network bandwidth utilization percentage and total number of bytes transferred across the network during replication activity were recorded. LANScan Network Monitor was used to analyze the data flowing over the network and to isolate the traffic flowing between machines involved in replicating data. Network bandwidth utilization and the amount of data transferred were therefore accurately logged to reflect only the traffic generated by the data replication process.

A collection of benchmarking tools was developed to measure the statistics aforementioned. These tools are the transaction generator and the latency calculator.

Transaction generator: The purpose of the transaction generator is to apply SQL updates to data stored in tables on a local database. In this case, the local database contains the schema and data created by the data generation process described earlier. The motivation behind the transaction generator is to simulate a user or users updating their local database in DIMHRS. As the transactions are applied to the local test database, the Oracle Advanced Replication software propagates the updates to the remote replica sites.

Latency calculator: The purpose of the latency calculator is to measure the replication latency (in milliseconds) at a remote replica site. It does this by detecting changes in the rows of replicated tables at the site. As transactions are propagated to the remote replica site from another database by the replication software, the latency calculator detects if each row of a table has been updated (due to the propagation of changes) from its original value and calculates the replication latency of the update to the row.

The transaction generator and latency calculator must interact with each other to ensure the accuracy of the results. Figure 3 presents this process.

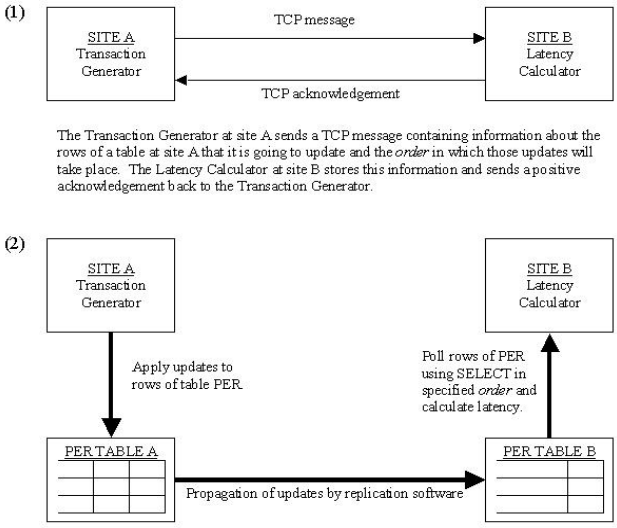

Figure 3. Interaction between the Transaction Generator and Latency Calculator

Under each of the 3 test environments, several replication schemes were employed and their performance benchmarked. For all of the experiments conducted, the SQL that comprises a transaction generated by the transaction generator was:

1. UPDATE PER
2. SET PER_Dscrptn_Text = ?
3. WHERE PER_ID = ?

The SQL statement shown above will cause a transaction to update the description field of the person table for a specified personnel identification number. The reasons that this particular definition for a transaction was used are because it is expected that the PER table will be one of the most updated and frequently used tables in real-world DIMHRS usage and the PER_Dscrptn_Text field is the largest field in the PER table. Updates to this field will generally consist of several hundred ASCII characters. The substantial amount of data contained in this field is beneficial when measuring the amount of traffic flowing over the network.

5. Results

Table 1 summarizes the results obtained from benchmarking under test environment 1. The Windows 2000 Professional OS was used in all of the benchmarks conducted in this environment.

It can be seen that the snapshot replication schemes are vastly more efficient than the multiple master schemes. Both asynchronous and synchronous snapshot schemes performed, on average, several orders of magnitude faster than either of the multiple master schemes. More interesting, however, is the fact that the asynchronous multiple master scheme was nearly twice as efficient as the same synchronous scheme.

Table 1. Summary of Benchmarking Results for Test Environment 1

Replication Scheme	Avg. No. of Transactions per minute	Avg. Replication Latency	Total Bytes Transferred	Avg. Bandwidth Utilization
Asynchronous Snapshot	8874	< 20ms	38.8MB	3.71%
Synchronous Snapshot	8804	< 20ms	38.0MB	3.22%
Asynchronous Multiple master	3299	< 20ms	24.3MB	3.96%
Synchronous Multiple master	1360	< 20ms	10.2MB	1.75%

Table 2 summarizes the results obtained from benchmarking under test environment 2. The Windows 2000 Professional OS was used in all of the benchmarks conducted in this environment.

Table 2. Summary of Benchmarking Results for Test Environment 2

Bandwidth Tested	Avg. No. of Transactions per minute	Avg. Replication Latency
9,600bps	60	213.5ms
19,200bps	60	183ms
38,400bps	60	174ms
57,500bps	60	154.5ms
115,200bps	60	90ms

In this environment, the paucity of adequate network bandwidth made it impossible to allow the transaction generator to generate transactions as quickly as possible. This is because the time required to propagate a large number of transactions was prohibitive, even at 115,200bps. Due to the lack of bandwidth, transactions are stored in a deferred queue of transactions to be propagated. To avoid this, the transaction generator was set to generate approximately 60 transactions per minute. However, even at this low level of activity, the replication latency was still quite large. Unfortunately, the amount of data transferred and bandwidth utilization could not be obtained in this environment because the infrared devices used to simulate these bandwidths could not be monitored using LANScan Network Monitor. Strictly speaking, they are not really network devices as TCP/IP runs on top of the IRDA protocol.

Table 3 summarizes the results obtained from benchmarking under test environment 3. The Solaris 7 OS was used in all of the benchmarks conducted in this environment.

The importance of these results is in elucidating the fact that the choice of hardware (along with network bandwidth and the data replication scheme) plays a vital role in overall data replication performance. The enormity of the difference in the number of transactions propagated by the Sun Enterprise 420R versus a Sun SparcServer20 is testament to this fact. As expected, the average replication latency remains less than 20ms due to the presence of a high speed, high bandwidth network.

Table 3. Summary of Benchmarking Results for Test Environment 3

Hardware Transaction Generator executes on / Replication Scheme	Avg. No. of Transactions per minute	Avg. Replication Latency	Total Bytes Transferred	Avg. Bandwidth Utilization
Enterprise 420R Asynchronous Multi-master	4558	< 20ms	30.1MB	2.23%
Enterprise 420R Synchronous Multi-master	3538	< 20ms	26.3MB	2.10%
SparcServer 20 Asynchronous Multi-master	2226	< 20ms	14.1MB	2.03%
SparcServer 20 Synchronous Multi-master	1203	< 20ms	9.8MB	1.91%

6. Conclusions

The benchmarking of data replication performance for the DIMHRS Project was discussed in this paper. Some benchmarks were defined to quantify data replication performance and the benchmarking of several data replication schemes under simulated test

environments was carried out. Analysis of the results advocate that the DIMHRS replicated environment should only use asynchronous replication schemes as they provided better performance than their synchronous counterparts in every test environment. The choice of hardware and bandwidth availability also makes a big difference in overall replication performance. Mobile users should have at least 19,200bps of bandwidth for satisfactory performance. Fully-replicated master servers should have at least 10Mbps of bandwidth and massive I/O capacity to adequately deal with the propagation of updates to replicated data. There is a great deal of work that can be done in addition to that presented in this paper. Replication software from vendors other than Oracle could be used in the benchmarking process. Or, hybrid test environments that combine large numbers of snapshots with several fully-replicated servers could be benchmarked. Designing and constructing such environments would pose a unique challenge because of their inherent complexity.

References

[1] B. Burton, The Real Value of Data Replication, Gartner Group, 1996

[2] International Business Machines Corporation, IBM DB2 Replication Guide and Reference (1st ed.), International Business Machines Corporation, 1999

[3] Oracle Corporation, Oracle8 Server Concepts Release 8, Oracle Corporation, 1997

[4] Lisa Brownsword, Forward, J. Dean, A. Gravel (Eds.), COTS-Based Software Systems, Proceedings of the First International Conference on COTS-Based Software Systems, 2002.

[5] Carnegie Mellon Software Engineering Institute, COTS-based systems initiative, http://www.sei.cmu.edu/cbs