Paper 4105

DataReplication for HP-UX and Oracle DBMS

How does it work, what are the problems, and what are the alternatives?

Karl Kacerek
Eyal Aronoff
Quest Software, Inc.
610 Newport Center Drive
Suite 1400
Newport Beach, California 92660
(800) 306-9329

Introduction

In 1996 for the first time there were more Oracle-based applications in production than in development. Once an application is deployed to production, the organization becomes dependent on it. That causes Oracle-based production application to become more "mission critical" than ever before. Production applications have the unwieldy profile of rapid growth - more than double in database size between 1996 and 1997. Data Warehouses, Executive Information Systems (EIS) and other data consolidation projects require data integration from multiple sources. Similarly data produced by one part of the organization may be used throughout.

Oracle replication features address some of data replication challenges. However many users of Oracle replication are expressing concerns mainly with issues of performance and manageability. This paper explains the main issues with data replication, how Oracle addresses them, and proposes an alternative way for data replication.

The Environment

Where in the '60s and '70s, you might find 20 people in an organization responsible for one or two IBM mainframes, today there are one or two people responsible for 20 (or more) machines. The administrator in one of the approximately 1,500 IBM mainframe shops had an average of 5 years of experience. In the rapidly expanding open systems market, the local administrator in one of the approximately 100,000 open systems shops might have an average of 1-2 years of experience. The result is that data and applications are stored on many machines and often maintained by people with a lower level of experience. Whatever talent the organization might have is consolidated to a few pockets of expertise. These experts are responsible for the production viability of a number of diverse systems located throughout the organization.

The Problem

As the organization becomes more dependent on data and applications, the importance of data distribution increases. The two most common purposes for data distribution are data sharing and off-site backup/standby. One of the most important aspects of data distribution is data replication. With the speed of database growth and the event of hundreds of users hitting the database, the workload on existing hardware is already close to peak capacity. To combat this effect, the investment in increased capacity often leads to shops with large machines that perform hundreds of database modifications per second. In environments like this, the intrinsic overhead of replication pushes the already overworked machines over the top increasing response time to unacceptable levels The speed of replication is generally not sufficient to support the volume of changes required. For these reasons organizations could not fully deploy the required replication scheme.

The Solution

When considering data replication scheme in the ORACLE environment the following issues must be taken into consideration and addressed to insure a successful replication scheme is implemented:

- **Minimize the overhead on the source machine.**
- **Minimize the impact to the processes that generate database modification.**
- **Minimize the exposure to network failure**
- **Shorten the time delay between modifications on the source and their application on the destination machine (especially important for standby machines).**
- **Enforce read consistency across both source and destinations.**
- **Easily scale to a large number of replicas, some of which many not be accessible at all times.**

Most if not all of these issues are not handled effectively using the replication scheme provided within Oracle. Only a replication scheme based on the data stored in the Oracle log files can address each of these issues effectively.

The Oracle replication scheme contains many configuration options. For simplicity, I use in my examples only the two most frequently used configurations: incremental snapshots and updateable masters. Many of the points, though, apply for most configurations.

Performance Is Everything

Lets look at an example. A job that manipulated data in three tables took five hours to run. Once one of the tables was turned into a replicated master, that same job took 12 hours to complete - blowing through the nightly batch window. Although results vary significantly between different production examples, there is always a significant overhead to transactions on the master table. The reason is that trigger-based replication causes insert, update and delete operations (DML) to be added to your own transactions. These operations are executed as part of the user (or batch job) session. So where initially the job had only 100,000 operations to complete, with trigger based replication, it might have the equivalent of 200,000 operations. Thus, transactions that manipulate replicated tables will take longer (and some times MUCH longer) to complete.

One of the goals of effective data distribution is to create a replication scheme that has minimal impact on the source system (or what we call the “master” system). Log replication is done independent of the Oracle transaction. Hence, it does not make a difference from the user perspective whether a table is replicated or not. The response time of both online and batch transactions is minimally impacted by the replication.

In addition to the direct impact on the transactions that are involved with replicated tables, there is a constant overhead on systems that host a replicated database. As one can imagine, the overhead of managing the replication process by means of database triggers and insert/update/delete transactions can be very significant. Oracle uses background processes to determine what, when and where to replicate. These background processes scan what could become large tables. If rows are found, they are sent via SQL*Net to the remote sites. Once this is done, the rows are deleted from the local site. With database snapshots, the overhead is relatively low since only the ROWID of the modified rows are inserted and deleted. In symmetrical replication, the entire before and after image of the complete record is manipulated, which many times ends up causing chaining that further increases the overhead. Log replication does not manipulate the database for every change. The modification queues are stored in flat files that are a hundred times faster to scan than database tables. If the workload is moderate, the data flows in and out of the queue without being flushed to disk.

You may be concerned that your site generates hundreds of megabytes of redo log data every day - “Surely log replication will cause a huge amount of network traffic.” What we have found is that only a fraction of the log is user data. The log is written in blocked mode (so if the block is

2K and there were only 124 bytes to write before a commit, there will be 1900 bytes of wasted space). Additionally, Oracle writes index modifications and other "accounting" information to the log such as changes to the high-water-mark, adding and removing blocks from the free list, and more. These modifications do not require replication.

Another advantage of log replication that increases its speed is the ability to use array operations on the remote database. Rather than performing the operation a row at a time, a whole batch of rows may be applied simultaneously.

Timing Is Everything

Many organizations require fast replication, whether for a standby machine or just because their business needs make them concerned about a lag between the time a transaction is completed on the source system and the time it is reflected on the target system. This is specifically important for long running transactions. The reason is that they both generate a lot of data and they are the most difficult to recover. With trigger-based replication, the Oracle background process cannot "see" the data in the snapshot log or the replication queue until the transaction commits. Consider the previous example of that job that took 12 hours. Suppose that job had only one commit in the end. Thus, after 12 hours of work, the Oracle replica would by 12 hours behind. It could take another 5 minutes for the snapshot process to realize that there is data to replicate. The replication process itself and the network traffic might take another 8 hours (if there would be a network problems the entire transfer might have to restart). At that point, the replica would be 20 hours behind. A standby machine that is almost one day behind is not much of a standby machine!

With log replication, transactions are replicated as soon as they hit the log (which is instantly). As long as the replica can keep up with the source, it will not be behind. Once a commit arrives in the log, there is little or no data to send, and the commit just flows through. Additionally, the unit of retry on a network failure is an individual change (or a small group of changes). So a network failure causes the resending of few records at the most, unlike the trigger-based replication that has one commit at the end (which means all or nothing). In our test that same job took only 5.5 hours to run and the replica reflected the changes 2 minutes after completion on the source system.

Another aspect of time delay between the source and destination is the dependency of the replication process on the availability of the network. In Oracle the propagation process makes a direct SQL*Net connection to the remote database. The entire queue of changes is then applied as one transaction to the replica site. At the end of this transaction there is a two-phase commit that ensures all data was replicated to the remote site before deleting it from the queue tables in the source site. If the number of changes to replicate is large this transaction can take a very long time. During this entire time, the network connection has to be available. If at any time during the propagation transaction the network becomes unavailable, the entire propagation is rolled back and Oracle will have to start all over again. In Oracle8, parallel propagation overcomes some of the aspects of this problem by providing multiple simultaneous connections to the remote database. If the transaction is limited by the speed of the network traffic, multiple connections will not get the data any faster to the remote site than a single connection. Even if the network traffic is not the bottleneck, multiple connections only limit the window of exposure but does not solve the base problem. For example: assuming that in Oracle7 the propagation of changes took four hours to run. In Oracle8, with four processes doing the propagation it could take an hour to run. However, if after 40 minutes the network connection is lost, the four propagation transactions will roll back, and the entire data will have to be reapplied to the remote database.

In some situation the reliance on a single transaction to propagate the data to the remote site can bring replication to a halt. For example, lets assume you have a replication schema that replicates an application from the US to Hong-Kong. The network connection to Hong-Kong is not very reliable. You may get one network failure every hour of continuous connection. If the application generates about four hours of network transmission throughout the day, with normal

operation the site in Hong-Kong will be very close behind. The network reliability will not be an issue because the four hours network workload will be spread on a 24 hours span. However, lets assume that one weekend, the Oracle instance in Hong-Kong was shutdown. It stayed shutdown for two days before being restarted on Monday. During this time, the instance in the US kept collecting changes for the Hong-Kong site. On Monday the connection finally got reestablished. The US instance had accumulated enough changes for eight hours of transmission. However, Oracle might never get a strait eight hours span to send the data out without an error that would cause the transmission to restart. As Oracle retries to send the data more data is accumulated at the source. The result is that the Hong-Kong site might never catch up to the US source.

With log based replication the unit of retransmission over the network is a few records at most. This means that if the network is not reliable, the amount of retransmission will be very small. Additionally, the log based replication queues changes on both the source site and the destination. This means that as long as Hong-Kong server is "up" (although the database may be down), it can keep receiving changes and queue them locally. When the database is finally started, the transactions get applied from the local queue with no additional network traffic.

Size Is Everything

As the number of components in a replication scheme increases, so does their interdependency. To resolve this problem, Oracle introduced a new term - "quiesce". This means that while changes are made to the replication scheme, the entire network of replicated machines is locked (or "quiesced") for the duration of the change. However if you have a large number of computers and large volumes of transactions, it could become impossible to get to a quiesced situation. This is especially true if some of the computers that participate in the replication scheme are not connected to the network at all times. It is important to include the configuration changes as part of the data in the data stream. This eliminates the need for a synchronized transaction. Each target machine makes the modifications to the replication scheme when it gets to that 'point in time'. From that time on only data that represent the new scheme will be present in the data stream. Another challenge to overcome is the impact that reorganization of tables on the source system has on the target systems. In snapshots, for example, an import of data will cause a full refresh. Since using the content of the columns (primary key, unique key or the entire record) as an update key rather then the original ROWID, full refresh for data re-organization is not required.

Oracle replication uses SQL*Net and a direct TCP/IP connection between the source and each target machine participating in the replication. This is known as "two tier architecture". When the number of target machines increases, so is the overhead of keeping all these TCP/IP connections. Each new machine means duplicating the data to that machine directly from the source. One hundred machines means sending the same data a 100 times through the network. This strains the network link from the source to the network service provider. Added to that the cost of "accounting" of what machine got what portion of the data that, as shown, involves quite a number of database operations. The result is that the high overhead of adding more target machines limits the size of the replication scheme. An external log based replication paradigm supports the concept of a "multi-tier architecture" with much less overhead than any solution within the Oracle environment. You can define that the data will go through one or more intermediate machines. The data is only duplicated for each directly connected machine. This means that if you have a 100 target machines, the source could be configured to send the data to only 10 machines, and from there the data could be propagated to the remaining 90 machines. The workload on the source and each routing machine would be greatly reduced.

As the replication scheme grows, so increases the need for better control. The current Oracle replication scheme does not provide centralized information on how the entire network of replication is performing. Simple questions such as how behind the replicas are can be very difficult to answer. A central monitoring station should be considered critical in order to provide constant feedback on the status and the control of the replication process. A GUI based tool to

view the performance and status of the entire network of targets is preferred but is not available with Oracle replication.

It Is the Technique

There are many ways to replicate data throughout the organization. Each of these ways has advantages and disadvantages depending on the business requirement it fulfills. We believe that log replication is one of the fastest, most flexible and scaleable way to achieve replication for a variety of applications. Additionally, a generic SQL implementation of log based replication enables you to replicate across databases from multiple vendors, translate table names, column names and even different formats.

About the authors

Karl Kacerek is a Senior Account Executive at Quest Software. Karl has been working in the HP environment since 1979 and after spending 10 years in charge of software development for a large HP VAR, Karl joined the Quest team in 1992. Over the last 6 years has helped hundereds of companies implement local and wide area data replication schemes.

Eyal Aronoff is Vice President of Products at Quest Software. Eyal is the inventor of AdHawk and SQLab database monitoring, administration and tuning tools. He is a certified Oracle DBA and a co-author of the *Advanced ORACLE Tuning and Administration* book published by Oracle Press.