A Large Scale Data Warehouse Application Case Study

Dan Pollack – America Online Inc.

ABSTRACT

Large data warehouse applications are beginning to become more necessary as large amounts of data are collected from the day to day interactions of businesses and their customers. Businesses are now using these enormous amounts of data to decide how and where to advertise, what resources projects will require in the future, and to gauge the general direction of the company to keep it on track.

This paper is a case study of the design and implementation of a 1+ terabyte data warehouse for marketing decision support from a systems design and administration perspective. It will include a brief discussion of software selection and a detailed look at the sizing, testing, tuning, and implementation of the data warehouse. The case study will address such issues as sizing, I/O subsystems, I/O bandwidth, backup issues, performance trade-offs, and day to day operations.

Introduction

When the business systems department said they wanted to look into building a data warehouse they came to the database operations department at AOL and asked us to evaluate the options and choose something that would scale well and provide them with the tool they needed to provide marketing decision support. They only knew how much data they already had and how much it might grow based on projections. We set ourselves to the task of building something that might accommodate what we thought they asked for.

Data Warehouse Software

The first thing that needed to be done was to pick the software to run the data warehouse. After a short search two candidates emerged. Sybase IQ and Redbrick. Sybase IQ was recommended by our local Sybase Inc. representatives and Redbrick was recommended by our local Silicon Graphics Inc. representatives. Oracle was contemplated but was dismissed because it would have been more difficult to test given the lack of a local support system since our relationship with Oracle Inc. wasn't as extensive.

Sybase and Silicon Graphics both had representatives on-site to provide information and support. After testing on demonstration systems, Redbrick was chosen for ease of use and implementation reasons as well as generally good performance. Sybase IQ performed a bit better than Redbrick, in general, during data loads. Redbrick wasn't that far behind and it was ultimately chosen for administrative rather performance reasons. Sybase IQ did not have the ability to add user access to the application without stopping and restarting which would be unacceptable in an environment with constantly changing user requirements and maximum availability requirements.

Disks

Since Redbrick was chosen as the data warehousing software, the demonstration platform was also chosen for testing and implementation. The testing platform was a Silicon Graphics Power Challenge XL with 550 Gigabytes of disk space in a mirrored JBOD configuration for a usable storage area of 225 GB. The machine also had 12 MIPS R10000 CPUs and 2 GB of RAM. This size was fine for testing but not large enough to do any real work on.

The disk space was doubled by swapping the 4.3 GB disk drives for 8.5 GB disk drives. The data was preserved by splitting the mirrored data volumes and replacing half the drives with the larger 8.5 GB drives while keeping the data on the small drives. The data was dumped from the preserved mirror sides on the small drives to filesystems on the new disks and then the new larger filesystems were mounted in the place of the old smaller ones. The remaining old smaller drives drives were replaced with new larger ones and those were mirrored together with the previously replaced drives. The whole process required very little downtime and was made possible by using the XLV volume manager [1] from SGI. This is the machine that was initially put into production.

The total disk space was then doubled again by doubling the number of disks about two months after the start of production. This addition of disks caused performance problems. The machine became very slow and we went back to do further testing on a second machine being built as a companion since the data had once again grown and was now too large to be contained on a single machine.

Detail data was separated from aggregate data in order to accommodate the amount of data required to be online. The testing on the second machine showed that there were certain I/O limitations inherent in the Challenge XL and also that the RAID configuration for the recently added disks was not optimal.

The major limitations of the Challenge XL were physical space limitations. It can only accommodate 40 SCSI controller connections on its bulkhead which limits any application to 40 SCSI controllers – which would seem like enough until you need very large data sizes for your application along with high performance. More disks could be connected but we found that performance suffered too greatly.

The new machine was set up in a completely different RAID 0+1 configuration that was chosen for performance as well as data protection. Since the amount of data was so large, backups could not be performed right away. We decided that we would just keep two copies of the data at all times with disk mirroring.

The mirroring and striping of the regular SCSI disks was accomplished with SGI's XLV product. The second time the disk capacity was doubled, hardware RAID units from SGI were added and they were configured as a 0+1 as well. The striping of the disks was done to increase I/O performance. The total size of the mixed regular and RAID disk machine is 2.448 TB with 1.224 TB usable due to mirroring. The database size on this machine is approximately 840 GB with the rest of the 1.224 TB or 384 GB used for staging of the raw data. The total size of the all RAID disk machine is 2.72 TB with 1.36 TB usable due to mirroring. The database size on this machine is approximately 870 GB with the rest of the 1.36 TB or 490 GB used for staging of the raw data from the detail machine.

Performance Modeling

Multiple RAID configurations were evaluated in the event that backups would become available. In order to test performance, the applications characteristic read and write behavior needed to be modeled. Since Redbrick does large long running I/Os in 8 KB chunks, it was not that difficult to create a general model of the I/O behavior of the application. Using a simple dd command, the read and write I/O behavior could be adequately modeled. The command used to model write behavior was:

```
% dd if=/dev/zero of=/test/file01 \
     bs=8192 count=100000
```

and the command used to model read behavior was:

```
% dd if=/test/file01 of=/dev/null \
     bs=8192 count=100000
```

These were used to simulate single read and write processes. Redbrick also has the ability to perform parallel reads and writes so multiples of these two commands were linked to specific processors using the built-in processor affinity commands in IRIX 6.2 and rerun to approximate the way Redbrick would run them. See Appendix A for the actual scripts.

The RAID [2] performance data was collected for RAID 0+1, RAID 3, and RAID 5. RAID 0+1 is what is known as a stripe of mirrors in which pairs of mirrored drives are striped for performance reasons. RAID 3 is a group of disks with a drive designated as the parity drive. In a RAID 3 configuration, parity is always written to the same drive while data is striped across the rest in the group. RAID 5 is a group of disks with data and parity striped across all of the drives in the groups in a round robin fashion.

I found RAID 0+1 to perform best for both reads and writes as well as providing the best data protection. It is, however, the most costly configuration. The RAID 0+1 filesystem configuration I used was four disk RAID 0+1 LUNS in groups of eight using four controllers with 128 block RAID stripes and 512 block filesystem stripes. A RAID stripe indicates the size of a data stripe placed on each drive in a RAID group or LUN. The filesystem stripe indicated the size of a data stripe placed on each disk if the filesystem is made up of individual disks or in this case the size of a data stripe placed on an individual LUN since our filesystems are made up of RAID LUNS.

Filesystem throughput was was approximately 80 MB/sec or 10 MB/sec per LUN for a single write and 53.33 MB/sec or 6.67 MB/sec per LUN for four parallel writes. Filesystem throughput was approximately 24.33 MB/sec or 3 MB/sec per LUN for a single read and 44.53 MB/sec or 5.56 MB/sec per LUN for four parallel reads.

The RAID 3 filesystem configuration I used was five disk RAID 3 LUNS in groups of eight using four controllers with a RAID 3 stripe of one and a 512 block filesystem stripe. Filesystem throughput was approximately 48.84 MB/sec or 6.11 MB/sec per LUN for a single write and 52.72 MB/sec or 6.59 MB/sec per LUN for four parallel writes. Filesystem throughput was approximately 23.49 MB/sec second or 2.94 MB/sec per LUN for a single read and 35.68 MB/sec or 4.46 MB/sec per LUN for four parallel reads.

RAID LEVEL	1W	4W	1R	4R
RAID 0+1	80 MB/sec	53.33 MB/sec	24.33 MB/sec	44.53 MB/sec
RAID 3	48.84 MB/sec	52.72 MB/sec	23.49 MB/sec	35.68 MB/sec
RAID 5	30.32 MB/sec	32.91 MB/sec	14.39 MB/sec	42.93 MB/sec

Table 1: Performance summary.

The RAID 5 configuration I used was five disk RAID 5 LUNS in groups of eight using four controllers with a RAID 5 stripe of 128 blocks and a filesystem stripe of 512 blocks. Filesystem throughput was approximately 30.32 MB/sec or 3.79 MB/sec per LUN for a single write and 32.91 MB/sec or 4.11 MB/sec per LUN for four parallel writes. Filesystem throughput was approximately 14.39 MB/sec or 1.80 MB/sec per LUN for a single read or 42.93 MB/sec or 5.37 MB/sec per LUN for four parallel writes. The results are summarized in Table 1.

From the tests I found that RAID 3 had comparable write performance to RAID 0+1 but it was approximately 20% slower on reads than RAID 0+1. RAID 5 had comparable read performance to RAID 0+1 but it was approximately 38% slower on writes. I decided RAID 0+1 was the only viable option since the slower writes of RAID 5 would extend load times of the database unacceptably and the slower read times would cause user queries of the data to run longer which could potentially cause problems with service requirements. RAID tests with other configurations were done with these three RAID levels. They are not reported here since they were used to optimize the stripe sizes for each RAID level and cache read/write size distributions since the hardware RAIDs have built in caches for further speed up of reads and writes for general performance improvements.

Backups

Backups were initially planned using existing DLT4000 drives in robot tape changers on the network. The size of the data quickly outgrew the capacity of the DLT robots and the network. Alternate solutions were sought while the RAID 0+1 was the only method of data protection. Adding to the problem was the limited number of controllers left for attaching disk devices after disk expansion and the relatively short backup window.

Of the limited total number of SCSI controllers available on the Challenge XL, 36 had already been used for the disks that were attached. Since only four SCSI controllers were available for backup devices and approximately 1 TB needed to be backed up in a six hour window, we needed to find high bandwidth backup devices.

The devices we tested were multiple DLT 7000 robot tape changers with multiple drives on each SCSI controller and multiple DLT 7000 robot tape changers connected to a remote machine with a HIPPI network for connectivity. An AMPEX 812 DST was considered but physical size and lack of software support caused us to rule it out.

The directly attached DLT 7000 robots worked quite well and were able to sustain 7 MB per second throughput per drive when they were directly attached to a machine via fast-wide differential SCSI connections. Three drives per SCSI channel were used to get maximum throughput per SCSI channel. The backups of the smaller machine in this configuration took 3.58 hours. This backup included database and raw data staging areas.

Dedicating three large, expensive tape robots to a single machine where they would be used at most only one third of the time seemed like a bad idea, so we looked for other options to move data off the machines at high speed. We spoke to our local SGI reps and they suggested we investigate HIPPI [3] with an interesting twist called Bulk Data Services [4], which is a high performance NFS add-on used to take advantage of the high bandwidth of HIPPI by bypassing the buffer caches on the local and remote machines.

With BDS, instead of doing a normal NFS transaction, the buffer caches are ignored on both sides and the data is retrieved via direct I/O from the machine where it resides and it is passed via an 'aligned network send' straight into memory on the remote machine where the pages are then flipped to the application. BDS enables long running streaming IO's like the kind we were doing during backups to bypass much of the overhead of NFS, enabling much greater performance.

The filesystems that were to be backed up were exported via NFS on a warehouse machine with BDS installed to a machine that also had BDS installed and several DLT 7000 tape robots as well. A slight software modification was required in the backup software to take advantage of BDS as well. After all of those issues were addressed, backups were run and throughput was measured to be approximately 5 MB/sec per tape drive with 10 drives running simultaneously. The aggregate throughput of the HIPPI link between machines was approximately 50 MB/sec. This allowed the large warehouse machine to be backed up fully in five hours.

The HIPPI/BDS configuration fit in with our backup time requirements and allows us to use the devices for backups of more than one machine which provides a savings in cost and amount of hardware needed. The HIPPI/BDS configuration is the one we will finally be using for backups.

Nearline Storage

In addition to the backups to tape, we decided that some other more near line area was needed to store old but not necessarily unneeded data. Redbrick has the ability to unload tables of data in a format that can be easily reloaded and that also takes up less space than the original tables. An NFS server seemed ideal for seldom accessed data that should be held onto but didn't need to be online. A Sun Ultra 2 was set up with 200 GB of disk space contained in a Sun Storage RSM219. Two 100 GB filesystems were mounted on each of the data warehouse machines for the purpose of holding this seldom used data. The backups and the NFS server allowed us to to have protection from catastrophe as well as an easy to use near line data space for old warehouse data.

Administration

Certain compromises were made to give adequate performance and still keep administrative headaches to a minimum. Redbrick's consultants initially recommended filesystems no larger than 2 GB, since no single Redbrick file can be over 2 GB in size. These 2 GB file systems can be arranged to give increased performance through striping between them but there would have been several hundred filesystems in a 1 TB data warehouse. We decided to compromise by using a few large filesystems but still striping the files themselves across multiple filesystems and carefully avoiding using the same filesystem for two things at the same time during a single operation.

When loads are done, for instance, the loaded data is read out of single filesystem and written out to multiple filesystems consecutively. The consecutive nature of the data writes is a Redbrick limitation but it simplifies the process of ensuring that no SCSI controller bandwidth contention occurs. We can take care to use a filesystem that uses one set of SCSI controllers for reading staged data and a completely different set of SCSI controllers for table writes.

Redbrick is generally simple to get set up and it requires little adjustment to automate its startup and shutdown. The server needs to be started as root which is inconvenient but can be accomplished on start up and shutdown with an init script in the appropriate place. sudo access can be given to the Redbrick install uid for starting and stopping the server at times other than startup and shutdown of the machine. This reduces the total administrative load from Redbrick software. Checkpoint and restart software is also being tested for use with long running jobs to avoid redoing work in the event of an unforeseen machine stoppage such as reboot or filesystem overflow. Initial testing is promising but it is still too soon to tell and our testing continues.

Performance Tuning

Certain kernel parameters require tuning to get better performance from Redbrick. The filesystem cache should be as large as possible and as much RAM as possible is indicated. The fraction of RAM allowed to be allocated to a single process should also be maximized since Redbrick keeps its working data set in memory until it is written out. This can be several hundred megabytes depending on the size of the data set.

The fraction of RAM available to user processes is 75% by default under IRIX 6.2 and we found that making this fraction 90% helped performance without hurting the normal operation of the machine due to the rather large installed memory sizes. In the 4 GB RAM machine, 90% usage still leaves 400 MB of RAM for system usage, which is plenty. The number of open file handles allowed by default was also adjusted upward since Redbrick does many file I/O's and tends to keep file handles open for a long time while doing long queries or loads.

Conclusion

During the implementation of this data warehouse we learned a tremendous amount about I/O performance tuning for large scale data volumes. Additionally, we found many ways to work around and live with the limits of current machines as well as making suggestions for improvements in certain subsystems. This work has also helped us plan for the growth of the application and its supporting hardware.

The RAID benchmarking and I/O tuning has also helped to create a general set of expected behaviors for the disk subsystems of all our machines. This will help us to plan more efficiently in the future for all projects. Growth of the data warehouse continues and that has caused us to look for newer larger more powerful machines for processing the warehouse data. We are hoping to implement a new solution based on fibre channel-arbitrated loop disks attached to an SGI Origin 2000 which will allow the win-win of a larger data space while increasing performance as well mostly due to the advantages of the FC-AL disks and the improved bandwidth of Origin.

Author Information

Dan Pollack was introduced to UNIX in 1988 and has been a System Administrator of one sort or another since 1990. He has worked in the financial, government and online service industries. He currently resides at America Online Incorporated in Reston, Virginia where he is a Senior System Administrator. He can be reached via email at dpollack@aol.net or via US mail at 12100 Sunrise Valley Drive Room 1Q03 Reston VA, 20191

References

[1] "IRIX Admin: Disks and Filesystems," *IRIX 6.2 Reference Manual,* Document Number: 007-2825-001, Silicon Graphics Inc, 1996.

[2] "RAID FAQ," *Frequently Asked Questions about RAID Levels*, http://www.recoverdata.com/raidfaq.htm .

[3] *Gigabit Networking*, Craig Partridge, Addison-Wesley Publishing, 1994, Chapter 7.

[4] "Getting Started with BDSpro," *IRIX 6.2 Reference Manual*, Document Number: 007-3274-001, Silicon Graphics Inc., 1996.

Appendix A

Single Write Script

```
#!/bin/sh
timex dd if=/dev/zero of=/test/fs1/file01 bs=8192 count=100000
```

Single Read Script

```
#!/bin/sh
timex dd if=/test/fs1/file01 of=/dev/null bs=8192 count=100000
```

Four Writer Script

```
#!/bin/sh
runon 1 timex dd if=/dev/zero of=/test/fs1/file01 bs=8192 count=100000 &
runon 2 timex dd if=/dev/zero of=/test/fs2/file01 bs=8192 count=100000 &
runon 3 timex dd if=/dev/zero of=/test/fs3/file01 bs=8192 count=100000 &
runon 4 timex dd if=/dev/zero of=/test/fs4/file01 bs=8192 count=100000 &
```

Four Reader Script

```
#!/bin/sh
runon 1 timex dd if=/test/fs1/file01 of=/dev/null bs=8192 count=100000 &
runon 2 timex dd if=/test/fs2/file01 of=/dev/null bs=8192 count=100000 &
runon 3 timex dd if=/test/fs3/file01 of=/dev/null bs=8192 count=100000 &
runon 4 timex dd if=/test/fs4/file01 of=/dev/null bs=8192 count=100000 &
```