Hardware Tuning Based Approach for Data Warehouse Tuning

Hitesh Kumar Sharma[1], Jagdish Chandra Patni[2], Ravi Tomar[3], Md. Ezaz Ahmed[4]
[1,2,3] University of Petroleum and Energy Studies, Dehardun, India
[1]hkshitesh@gmail.com, [2]patnijack@gmail.com, [3]ravitomar7@gmail.com
[4]Saudi Electronic University Al Madinah. KSA
[4]ahmedezaz@rediffmail.com

Abstract
The concept of the data warehouse has been emerging over the last few years, but with the new, faster computers and the reduction in the cost of disk storage, the data warehouse is finally becoming a reality. This paper first looks at what a data warehouse is, then at the characteristics of the data warehouse, and then at the data access patterns seen in the data warehouse.

KEYWORDS
OLTP, DSS, Data Warehouse, OS

1. INTRODUCTION
As its name implies, the data warehouse is a storage depot for corporate data. Enormous amounts of data are concentrated in the data warehouse; sources for the data include the following:

- Customer information databases
- Accounts receivable
- General ledger
- Inventory databases
- Customer credit databases
- Other sources

These sources combine to provide a wealth of data about customers and their buying habits as well as information about the general state of our business.
A data warehousing system is similar to a DSS system in some of its functionality, but the scale and focus are different. A typical DSS system focuses on one type of business function; by using its various data-input sources, the data warehousing system may perform much broader business queries. Data warehousing systems can easily achieve sizes in the hundreds of gigabytes; some systems even break the terabyte barrier. These systems are made possible by the continuing trend of computer hardware to increase in speed while decreasing in price. In the near future, it may not be uncommon to see tables of a terabyte in size [1. 2].

Although the cost of data warehousing hardware is decreasing, it is still out of reach of the main stream for now. As we go further into the information age and the value of information is better understood, We believe data warehousing will become more mainstream.

2. CHARACTERISTICS OF A DATA WAREHOUSE
Here are some of the characteristics of the data warehousing system:

- **Queries against large volumes of data.** The data warehousing system consists of much more data than the typical OLTP or DSS system.
- **Queries exhibit a variety of access patterns.** Queries may be simple or quite complex, with complicated joins and aggregations on large amounts of data.
- **Highly complex queries.** Queries on a data warehouse are typically much more complex than those used in OLTP and DSS systems.
- **Data access stresses the system to its limitations.** The processes stretch the system in terms of both performance and capacity.
- **Intense load activity.** As data from various sources is entered into the data warehouseing system, the load increases dramatically.
- **Is a conglomeration.** A data warehouse is a compilation of various input sources, usually tied into the corporate OLTP databases and other sources.

The load on the data warehousing system is typically very high. As with the DSS system, because users do not typically use a data warehousing system for online processing, it is reasonable to push the system to its limits. It is not uncommon for the decision support queries run against the data warehousing system to take hours or even days to complete. The queries are complex and the amount of data being queried is enormous. These systems are optimized for throughput rather than for response times. By maximizing throughput, some jobs may suffer in terms of response time. If we think that the data warehousing system is just a glorified DSS system, we are partially correct. The data warehouse may just be the next step in the evolution of the DSS system. There are many similarities, but there are many differences as well.

3. DATA ACCESS PATTERNS

The data access patterns seen in a data warehouse are fairly similar to those seen in a DSS system. Based on the types of transactions we generate, we should be able to fairly accurately determine these patterns. Although each system has its own specific data access patterns, the data warehousing system has the following, general characteristics:

Redo log activity is moderate to high. Unlike the DSS system (where the redo log activity is very low), the redo log activity for a data warehouse may be moderate or even high. This is caused, not by the activity of the business transactions, but by the procedures necessary to prepare and load the data. The metadata may be constantly put together from many external sources.

- **Archiving activity is moderate to high.** As with the redo logs, there may be significant activity due to the conglomeration of data stored in the data warehouse.
- **Data access for each query is mostly sequential.** Because the queries usually extract large amounts of data from the tables, full-table scans are not uncommon.
- **Data reads can (and frequently do) take advantage of multiblock reads.** We can expect that many of the disk accesses are the size of multi block reads.
- **Access to the data files is somewhat random.** As with the DSS system, this random access is caused by contention with other transactions and join and indexing operations that result in fairly random access across the data volumes. However, these random reads from the disk drives access the data with a much larger data request.
- **Data access may be sparse.** Because of the massive amounts of data being stored, there may be pockets of data infrequently accessed and other pockets that see much more frequent access.
- **Heavy access to the temporary tables.** Because of the typical size of many of the join and sort operations, the temporary tables are hit hard. Remember that only the sorts that use less memory than SORT_AREA_SIZE are in memory.
- **Data access may not be distributed evenly.** Because of the massive amount of data stored in the warehouse, it is not uncommon for queries to use only isolated pieces of data.

Although these patterns vary depending on how our system operates, the general principles are the same. The access patterns to our tables vary based on how often and how much is done to each table.

4. SYSTEM LOAD

Like the DSS system, the CPUs in a data warehousing system are usually 100 percent active during the large business queries. Where OLTP systems have many users with small queries, the data warehousing system has relatively few users and massive queries (as does a DSS system).These queries should be able to take advantage of the capabilities of the CPUs and memory as long as the system does not become disk bound. By tuning the server using some of the concepts, we can avoid becoming disk bound. The typical OLTP, batch, or DSS system may have an insufficient number of disk drives, which causes an I/O bottleneck. The data warehousing system may not have this problem because so much disk space is needed for the hundreds of gigabytes of historical and current business data, that, with careful planning, there are plenty of disk spindles to distribute the I/O load. Following is a list of some of the load characteristics of a data warehousing system:

- **Relatively few processes on the system.** If we take advantage of the Parallel Query option, we add more processes and subsequently more process switches.
- **Minimal network traffic.** Network traffic is low during the transaction processing phases but may be significant during the data loading and updating phases.
- **Heavy I/O usage.** The decision support queries associated with the data warehouse usually generate large amounts of I/O to the data files. This I/O is somewhat random if multiple decision support queries are active simultaneously; but the I/Os are larger in size because of multiblock reads.
- **Moderate to high redo log activity.** Unlike the DSS system (where the redo log activity is very low), the redo log activity for a data warehouse may be moderate or even high. This is caused, not by the activity of the business transactions, but by the procedures necessary to prepare and load the data. The metadata may be constantly put together from many external sources.
- **Moderate to heavy use of rollback segments**. During the decision support queries, rollback segments will not be used heavily, but during the data-creation or conversion phase, rollback segment activity can be significant.
- **Large amounts of memory.** The memory is used not only for the SGA but for each of the server processes required for sort and join operations.

Defining and understanding these characteristics can help we design and tune our data warehouse [3] for optimal performance. The first sep in this design process is to set goals for what we want to achieve.

5. GOALS

The goal in tuning the data warehouse is to achieve a system that has certain characteristics. Here is a list of the characteristics of an optimally tuned data warehouse:

- **The system is CPU bound during decision support queries.** By removing all other bottlenecks, the system should be able to process as fast as possible, which is the speed of the CPUs.
- **The system is not drive bound.** Any disk bottleneck degrades performance. If this is the case, we should add more or faster disks.
- **Memory is sufficient.** If the machine pages or swaps, performance is severely degraded. The best solution is to add more memory; if that is not possible, reduce the size of the SGA or the number of users until the system no longer pages or swaps.
- **The system meets any additional requirements we might have.** With some data warehousing machines, we must keep current with the OLTP systems by updating on a nightly basis. With the data warehouse, we may have to update the data within a certain time frame.

By setting goals for how we expect the system to perform, we can determine whether we are successful. We can also determine earlier whether we will be able to achieve our specified goals.

6. DESIGN CONSIDERATIONS

Looking at data access patterns can give us a good idea how to design the system. Before looking at the design process, consider these important issues, introduced earlier in this book:

- **I/O is typically the limiting factor in the system.** We can do only a fixed number of random I/Os per second per disk drive.
- **I/Os can be reduced by caching data blocks in the SGA.** If the data we want to access is already in the SGA, a disk I/O is not required.
- **Isolate sequential I/Os.** Most of the time spent reading from or writing to the disk is spent seeking to where the data is located. If we can reduce seeks, we can achieve more I/Os per second.
- **Spread out random I/Os.** Random I/Os have a maximum rate per drive. By spreading the I/Os out among many drives, we increase the overall rate.
- **Avoid paging and swapping.** Any time the system pages or swaps, performance is severely degraded. Avoid this at all costs.

All these factors contribute to the optimal data layout of the system. The physical layout along with SGA and shared pool tuning creates an optimally configured server for the decision support tasks usually performed in the data warehouse. In data warehousing systems, the design of the queries is also very important, as we will see in later chapters.

7. PHYSICAL DATA LAYOUT

This section looks at how the data in a data warehouse should be configured. First, it looks at how to lay out the data on traditional disks; then it looks at disk arrays. We recommend using disk arrays if at all possible; the ease of use and performance benefits are worth the cost of the array.

The main goals in designing the physical data layout are to balance the I/O across all the disks that are randomly accessed and to isolate the sequential I/O. The data warehousing system typically involves loading and processing of data, which causes moderate to significant use of the redo logs. By isolating the redo log file in a data warehouse, the majority (if not all) of the data files are accessed in a random fashion but can take advantage of multiblock reads. To take advantage of multiblock reads, stripe the data over as many disks as necessary to achieve I/O rates our disk drives can handle.

8. TRADITIONAL DISKS

The layout for a data warehouse can be large and difficult to manage. A minimal configuration should look something like this:

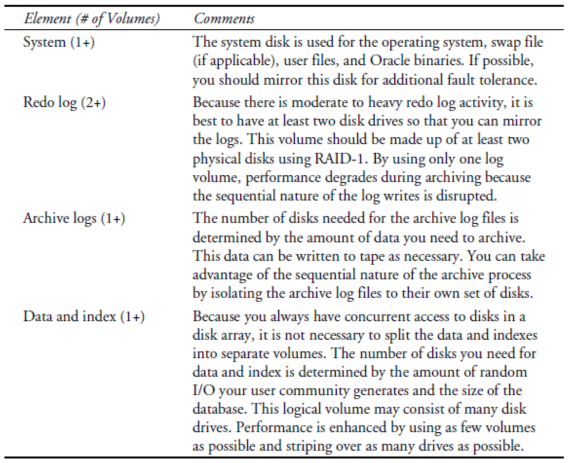
Element (# of Disks)	*Comments*
System (1+)	The system disk is used for the operating system, swap file (if applicable), user files, and Oracle binaries.
Redo log (2+)	Because there is moderate to heavy redo log activity, it is best to have at least two disk drives so that you can mirror the logs. When you factor in archiving, you may be better off with at least four disks.
Archive logs (1+)	You can take advantage of the sequential nature of the archive process by isolating the archive log files to their own set of disks.
Data files	The number of disks you need for data is determined by the amount of random I/O your user community generates and the size of the database. In this type of environment, the number of disks can be significant.
Index files	The number of disks needed for indexes is determined by the size of the indexes and the number of I/Os to the indexes. In this type of environment, the number of index disk drives can also be significant.

Both the data files and the indexes should be striped over as many disk drives as necessary to achieve optimal I/O rates on those disks. remember that we can only push a disk drive to a maximum random I/O rate. The data and indexes can be striped across the disks using Oracle or RAID striping or a combination of the two. With large data warehousing systems, We recommended OS or hardware striping. To take advantage of the Oracle Parallel Query option, we will benefit from having several large extents. An optimal configuration may consist of several data files residing on the same large, striped volume. If we do not use

Oracle striping and build one large extent, we may not see the full benefits of the Parallel Query option. We preferred a hardware disk array to manual Oracle striping primarily because the disk array provides excellent performance and is easy to use. When we use a disk array, the task of distributing I/Os can be greatly simplified.

Disk Arrays

The layout for the data warehouse on RAID volumes is much simpler than it is on traditional disk drives. A minimal configuration should look something like this:



Element (# of Volumes)	*Comments*
System (1+)	The system disk is used for the operating system, swap file (if applicable), user files, and Oracle binaries. If possible, you should mirror this disk for additional fault tolerance.
Redo log (2+)	Because there is moderate to heavy redo log activity, it is best to have at least two disk drives so that you can mirror the logs. This volume should be made up of at least two physical disks using RAID-1. By using only one log volume, performance degrades during archiving because the sequential nature of the log writes is disrupted.
Archive logs (1+)	The number of disks needed for the archive log files is determined by the amount of data you need to archive. This data can be written to tape as necessary. You can take advantage of the sequential nature of the archive process by isolating the archive log files to their own set of disks.
Data and index (1+)	Because you always have concurrent access to disks in a disk array, it is not necessary to split the data and indexes into separate volumes. The number of disks you need for data and index is determined by the amount of random I/O your user community generates and the size of the database. This logical volume may consist of many disk drives. Performance is enhanced by using as few volumes as possible and striping over as many drives as possible.

Both the data files and the indexes should be striped over as many disk drives as necessary to achieve optimal I/O rates on those disks. we can only push a disk drive to a maximum random I/O rate. Unlike traditional disk drives, when we use a disk array, the data is automatically striped across all the disk drives; therefore, it is necessary to create only one table space and table for all our data. We do not even have to put indexes into another table space although We recommend doing so for other reasons (such as monitoring and maintenance).With traditional disk partitioning, it is difficult to manage hundreds of data files and disks; with a disk array, we can manage hundreds of disks with just a few data files. Of course, Oracle has a 2 gigabyte limitation on the size of a data file, but this is easily resolved by creating a data file for every 2 gigabytes of space we need. The data files can all reside on the same disk array volume. By splitting table spaces into several data files with tables striped across them, all residing on the same logical volume, we can take better advantage of the Parallel Query option. Because the data warehouse may have a large amount of data that is sparsely accessed, it is to our advantage to put many different types of archival and current data on each disk volume. By spreading out the data, the I/O load is more evenly distributed. If we use a disk array, many of the management tasks and load balancing tasks are greatly simplified. With the disk array, we also have the option of using fault tolerance without affecting system performance. Of course, using fault tolerance requires significantly more disks. We recommend that we use a disk array if possible. Software striping is fine, but if our system is under heavy loads (as it is with a typical data warehousing system), we can achieve better performance by offloading the striping overhead to a hardware RAID controller.

9. FAULT TOLERANCE CONSIDERATION

Because the data warehouse contains so much data, we can take one of two approaches to data
protection:

- **Protect everything.** Because there is so much data and so many disks in use, everything must be protected. The large number of disks in use increases the possibility of a disk failure. The massive amount of data increases the time needed for backup and recovery.
- **Conserve cost.** Because there are so many disks involved, it may be cost prohibitive to use RAID-1 or disk mirroring. When we mirror the disks, we double the number of disks.

In a data warehousing system, a good compromise is to use a fault tolerant method such asRAID-5 for the data files. We can be somewhat selective and use RAID-1 on volumes with heavy update activity and RAID-5 on volumes with more read activity. Remember that the performance penalty for RAID-5 is only on writing; we can achieve excellent read performance from RAID-5.

10. HARDWARE CONSIDERATIONS

When choosing hardware to use for a data warehousing system, consider these factors:

- **Low user load.** Not many concurrent processes/threads simultaneously access the system—unless we take advantage of the Parallel Query option.
- **High I/O load.** I/Os are concurrent and heavy, with mostly random I/O.
- **Huge amounts of data.** Data warehousing systems typically involve massive amounts of data. We must make sure that our system can support the high volumes of data we will be using.
- **Low network traffic during runtime, possibly high during load.** During the execution of typical decision support queries against our data warehouse, there is very little network activity. When data is being loaded or updated from other sources (possibly our OLTP systems), the network activity can be quite high.

If we can take advantage of the Oracle Parallel Query option, many different processes will use the machine at once; an SMP or MPP machine should scale very well. Because an SMP architecture uses CPUs based on the processes that are available to be run, if we always have a runnable process available for each CPU, we should see good scaling by adding additional processors. With an MPP machine, we see a similar effect but on a much larger scale. Because there is much random access to the disks, we can benefit from a disk array. We prefer hardware striping to OS striping because hardware striping does not incur any additional over head for the operating system and does not take up valuable CPU cycles. If hardware striping is not available, OS striping is adequate. Network traffic may or may not be an issue to our data warehousing system. If necessary, segment the network or add faster network hardware. A network bottleneck is an easy problem to solve: simply add more and faster hardware.

11. TUNING CONSIDERATIONS

The data warehouse is tuned to allow several large processes to run at maximum throughput. There is usually no concern for response times. We may have to tune both Oracle and the server operating system. The following sections look first at Oracle and then at the server operating system [4-12].

Server OS Tuning

We may have to tune the server OS to provide for a large number of processes (if we are using the Parallel Query option) and optimal I/O performance. Some of the things we may have to tune in the server OS are listed here; remember that some OSes may not require any tuning in these areas:

- **Memory.** Tune the system to reduce unnecessary memory usage so that Oracle can use as much of the system’s memory as possible for the SGA and server processes. We may also need significant amounts of memory for sorts.
- **Memory enhancements.** Take advantage of 4M pages and ISM, if they are available. Both features can improve Oracle performance in a data warehouse environment.
- **I/O.** If necessary, tune I/O to allow for optimal performance and use of AIO.
- **Scheduler.** If possible, turn off preemptive scheduling and load balancing. In a data warehousing system, allowing a process to run to completion (that is, so that it is not preempted) is beneficial.
- **Cache affinity.** We may see some benefits from cache affinity in a data warehousing system because the processes tend to run somewhat longer. The server operating system is mainly a host on which Oracle does its job. Any work done by the operating system is essentially overhead for Oracle. By optimizing code paths and reducing OS overhead, we can enhance Oracle performance.

12. HARDWARE ENHANCEMENTS

For a data warehouse, there are several hardware enhancements that can improve performance. These hardware enhancements can be beneficial in the area of CPU, I/O, and network, as described in the following sections.

CPU Enhancements

Enhancing the CPUs on our SMP or MPP system can provide instantaneous performance improvements, assuming that we are not I/O bound. The speed of CPUs is constantly being improved as are new and better cache designs. For SMP or MPP machines, the process of enhancing the CPU may be as simple as adding an additional CPU board. Before we purchase an additional processor of the same type and speed, however, consider upgrading to a faster processor. For example, upgrading from a 66 MHz processor to a 133 MHz processor may provide more benefit than purchasing an additional 66MHz CPU with the added benefit that we now have the option of adding more 133 MHz CPUs. Because of the complexity and run time required by these queries, we can benefit from more and faster CPUs.SMP and MPP computers provide scalable CPU performance enhancements at a fraction of the cost of another computer. When upgrading our processors or adding additional processors, remember that our I/O and memory needs will probably increase along with the CPU performance. Be sure to budget for more memory and disk drives when we add processors.

I/O Enhancements

We can enhance I/O by adding disk drives or purchasing a hardware disk array. The data warehouse can benefit from the disk striping available in both hardware and software disk arrays. Using Oracle data file striping can also help the performance of our data warehouse. If our system performs only one query at a time and we are not taking advantage of the Oracle Parallel Query option, we may not see a benefit from a hardware or software disk array. In this specific case, we do not recommend OS or hardware striping; we should use traditional Oracle striping. Because we are executing only one query at a time without using the Parallel Query option, the I/Os to the data files are purely sequential on the table scans. This scenario is somewhat rare; any variance from “pure table scans” results in degraded performance. Hardware and software disk arrays have the added benefit of optional fault tolerance. We should first choose the correct fault tolerance for our needs and then make sure that we have sufficient I/O capabilities to achieve the required performance level. If we use fault tolerance, we will most likely have to increase the number of disk drives in our system. Another benefit of hardware disk arrays is caching. Most disk arrays on the market today offer some type of write or read/write cache on the controller. The effect of this cache is to improve the speed of writing to the disk; the cache also masks the overhead associated with fault tolerance. If our queries often perform table scans, we may see good improved performance with disk controllers that take advantage of read-ahead features. Read-ahead occurs when the controller detects a sequential access and reads an entire track (or some other large amount of data) and caches the additional data in anticipation of a request from the OS. Unlike an OLTP system in which this is just wasted overhead, in the data warehouse where we are performing DSS queries, it is likely that we will need that data soon; if we do, it will be available very quickly. Enhancements to the I/O subsystem almost always help in a data warehouse environment because large amounts of data are accessed. Be sure that we have a sufficient number of disk drives, properly configured. An I/O bottleneck is usually difficult to work around. As with all types of systems, a well-tuned application is very important.

13. CONCLUSION

As the price of computer hardware especially disk drives comes down every year, the idea of a data warehouse becomes increasingly more feasible. The amount of hardware that a few short years ago would have cost millions of dollars can now be obtained for much less. The performance of this hardware is also increasing at incredible rates. PC servers are now replacing systems that was considered minicomputers a few years ago with twice the capacity at half the cost. The ongoing reduction in cost and increase in performance will promote the trend of larger and larger databases with better information retrieval. The goal of the data warehouse is to take information from production databases, legacy data, and outside sources and use this information to better our business. As hardware gets faster and faster at less cost, this trend will continue; new applications and ways of looking at our business data will be developed. It's very exciting to look at the way the RDBMS industry has grown in the last few years. It will be exciting to see where it goes in the next few years.

This paper looked at the characteristics of the data warehouse as a business model and from the data access perspective. We can use this information to determine how to configure the system to take advantage of the data access patterns. If we understand how the system operates and what affects the performance of the system, we can use this information to design a system that has well-balanced I/Os and that can take full advantage of the computer's CPU power.

14. REFERENCES

[1] J. Seok Oh, S. Ho Lee, Resource Selection for Autonomic Database Tuning, Korea Research Foundation.

[2] K. P. Brown, M. Metha, M. J. Carey, M. Livny, 1994. Towards Automated Performance Tuning for Complex Workloads, Proceedings of 20th VLDB Conference, Santiago, pp. 72-84.

[3] D. M. Lane, "Hyperstat Online: An Introductory Statistics Textbook and Online Tutorial for Help in Statistic, http://davidmlane.com/hyperstat/index.html

[4] S. Elnaffar, W. Powley, D. Benoit, P. Martin, 2003. Today's DBMSs: How Autonomic are They?, Proceedings of the 14th DEXA Workshop, Prague, pp. 651-654.

[5] D. Menasec, Barbara, and R. Dodge, 2001. Preserving Qos of E-Commerce Sites through Self-Tuning: A Performance Model Approach, Proceedings of 3rd ACM-EC Conference, Florida, pp. 224-234.

[6] D. G. Benoit, Automated Diagnosis and Control of DBMS resources, EDBT Ph.D Workshop, Konstanz, 2000.

[7] B. K. Debnath, 2007. SARD: A Statistical Approach for Ranking Database Tuning Parameters.

[8] K. P. Brown, M. J. Carey, M. Livny, 1996. Goal-Oriented Buffer Management Revisited, Proceedings of ACM SIGMOD Conference, Montrea, pp. 353-364.

[9] P. Martin, H. Y. Li, M. Zheng, K. Romanufa, 2002. W. Poweley, "Dynamic Reconfiguration Algorithm: Dynamically Tuning Multiple Buffer Pools, Proceedings of 11th DEXA conference, London, pp.92-101.

[10] P. Martin, W. Powely, H. Y. Li, and K. Romanufa, 2002. Managing Database Server Performance to Meet Qos Requirements in Electronic Commerce System, International Journal of Digital Libraries, Volume 8, Issue 1, pp. 316-324.

[11] S. Duan, V. Thummala, S. Babu, 2009. Tuning Database Configuration Parameters with iTuned", VLDB '09, Lyon, France.

[12] H. K. Sharma, A. Shastri, R. Biswas, 2002. Architecture of Automated Database Tuning Using SGA Parameters, Database Systems Journal, Volume 3, Issue 1.