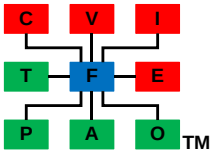

The Data Organization

1251 Yosemite Way
Hayward, CA 94545
(510) 303-8868
rschoenrank@computer.org

Database Performance

By Rainer Schoenrank

Data Warehouse Consultant

April 2012

Biography

Rainer Schoenrank is the senior data warehouse consultant for The Data Organization. He has degrees in physics from the University of Victoria and computer science from the University of Victoria and California State University Hayward. He has built data warehouses for clients such as Pacific Bell, Genentech, GE Leasing, SGI, PPFA, Brobeck, Bank of America, Clorox and Leapfrog. He can be reached at rschoenrank@computer.org.

Table of Contents

1. INTRODUCTION

The first complaint when an application does not perform well is that “the database is too slow.”

The document provides a model of the components of an application shows how these components are connected and identifies the places where poor performance may occur. This model outlines the database performance problem, lists the components of the problem, outlines the impact of each component on the DBMS performance and describes the optimum solution to minimize the impact of the component.

2. DATABASE MANAGEMENT SYSTEM FUNDAMENTALS

The purpose of the Database Management System (DBMS) is to provide its users with the ability to create, read, update and delete individual data records (rows within a set of related tables or database). The DBMS manages many users and many tables simultaneously. The DBMS ensures that the data always conforms to the database implementation and that table row operations by different users do not corrupt each user’s view of the data.

To keep the data correct, the DBMS must execute all the data row operations in the sequence in which they arrive. This ensures that the data in the database is not deleted before it is changed, etc. For each data row operation the DBMS must read and write data to physical disks that store the database and the DBMS uses the physical disks to store the data, the indexes to the data and the DBMS operations log.

3. APPLICATION PERFORMANCE BOTTLENECKS

To identify possible solutions to the problem of poor application performance, a model of the application implementation is required. The model allows measurements to be made to pinpoint where the performance problems exist and to isolate the factors that would have the highest impact on the application performance.

Figure 1 shows a layered stack model of an application with the application layers identified in the first column. The performance issues and their impact identified in remaining columns.

Typical Application Layers	performance issues	performance impact	improvement objective	improvement suggestions
User Interface	program efficiency	low		
network	network latency	medium	reduce network traffic	- only two remote procedure calls per user transaction
Application Server	program efficiency	low		
network	network latency	medium	reduce network traffic	- use stored procedure database interface to minimize network traffic
DBMS Server	non-relational database	medium	reduce extra processing	- normalize the database - one database per DBMS server - one DBMS per hardware server
network	network latency	medium	reduce network traffic	- connect DBMS server directly to disks
Disk Storage	disk I/O	high	increase disk throughput	- configure disks to match DBMS architecture - use solid state drives

Figure 1: Physical Layers of an Application Implementation

This document will focus on the performance of the DBMS, database and disk storage layers of the application model.

In general, the performance of a database depends on how long the DBMS has to wait for other processes to free the CPU, the number of operations the DBMS performs, and how long the disk operations take once the CPU is free. The issues that affect DBMS performance are:

1. The configuration of the DBMS hardware – the DBMS machine configuration can allow disk operations to proceed in parallel rather than sequentially.
2. The congestion of the DBMS instance – when a single DBMS instance contains more than one database, the different databases' sequential processing requirements interfere with each other.
3. The interface of the database – how the database is accessed and used creates extra processing for the DBMS.
4. The implementation of the database
    a. Missing indexes create extra processing for the DBMS when retrieving the data
    b. The size of the columns causes additional disk read operations to retrieve the data
    c. The size of the keys causes additional compare operations by the DBMS to find the data.

4. APPLICATION TO DATABASE INTERFACE

4.1 SQL Language Usage

How the database is accessed can create extra processing for the DBMS. The DBMS presents a command line interface to the world. The interface language is some variant of the standard SQL language. Which variant is used depends on the DBMS vendor. The command line is processed in three steps. First, the command line is parsed for language syntax errors. Then the command line is compiled into machine language. Finally, the command line is executed and the retrieved data is displayed.

A sequence of SQL statements can be saved by the DBMS as a stored procedure. A stored procedure is parsed and compiled by the DBMS when it is created. When the stored procedure is called, the DBMS only needs to execute it. The stored procedure can be accessed from the command line, has the ability to accept input parameters and will pass out the data retrieved by the SQL statements.

To improve the performance of the DBMS, the database should be accessed using stored procedures. This will increase the DBMS performance by a factor of three because the DBMS has to do less work for each command.

4.2 Nested Statements

Even when using stored procedures, how the SQL language is used can compromise performance. In an SQL statement, any data value can be replaced by an SQL statement that will provide the data value when executed. These are called nested SQL statements.

When a nested SQL statement is compiled and executed, the order is from the innermost statement to the outmost statement. The steps are:

1. Compile the innermost statement.
2. Execute the innermost statement to get the value of the data.
3. Substitute the data value into the next statement and compile.
4. Execute the next statement to get the value of the data.
5. Repeat steps 3 and 4 until the outermost statement is executed.

So, a stored procedure that contains nested SQL statements is only partially compiled and will require the DBMS to do extra work to complete the nested statement.

Since nested SQL statements perform like a command line statement, they are to be avoided when database performance is an issue.

4.3 Optimum SQL Language Usage

Because each access of the DBMS by the application process is a remote procedure call with the possibility of a data transfer across the network, to get the best performance the application process must:

- Access the database via stored procedures.
- Pass only the required data between the application and the database.
- Maximize what is done in a single call to the DBMS.
- Minimize the number database calls required by the application.

5. DATABASE STRUCTURE

5.1 Database Type

For data updating, the performance of the database is a trade-off between storage space and the amount of processing required to keep the database consistent. To minimize DBMS processing, the structure of the database should be:

- Third normal form tables
- Database is a minimum cover set of the relational algebra defined by the business application
- No redundant relationships in the database so that the DBMS optimizer works correctly

For data reporting, the performance of the database is inversely proportional to the number of joins required for the report (1/number of table joined). To minimize the amount of DBMS processing, the database should be denormalized into first normal form. This would give a reporting performance index of 1. Any other form for the database would have a smaller performance index.

5.2 Data Model Implementation

During implementation, there are a number of choices that can affect the performance of the database. The database implementation tradeoff is disk space versus processing time. The conditions that force the DBMS to do extra processing when accessing the data in the database are:

a. Table keys which are too long for a single comparison.
b. Table columns which are too long for a single disk retrieval.
c. Missing indexes for table columns used in joining data.

5.2.1 Long Keys

When the DBMS is comparing table keys to find a particular record, the comparisons are done at the CPU register level and the amount of data that can be compared depends on the length of the register in the operating system. For a 32 bit operating system, the length of the key that can be compared in one operation is 4 bytes (an integer). For a 64 bit operating system, the comparison length is 8 bytes (a long integer).

Every table will have a natural key. To improve the performance of the DBMS in joining tables when a table has a long or multiple column natural key, the database designer creates a surrogate key that is the same size as the operating system register and makes the natural key into a unique alternate key.

5.2.2 Long Columns

When the DBMS is retrieving data from the physical disk the fetches are done by the physical architecture of the disk. The disk is divided into cylinders and sectors within the cylinder. The sector size is 256 bytes. When a column is longer than 256 bytes, then the data spans more than one disk sector and the disk must do multiple fetches to retrieve the column thus slowing down the DBMS performance.

When designing the physical database, the designer should make every effort to limit the length of the table columns to the sector size or multiples of the sector size.

5.2.3 Missing Indexes

When the DBMS is joining two tables using a column in each table, it finds the rows in each table using an index on the columns. If the indexes do not exist, the DBMS will create temporary indexes. Unfortunately, those indexes are deleted at the end of the statement execution and must be recreated each time the join is executed. This extra processing affects the DBMS performance.

When DBMS performance is analyzed using the performance tools, the tool identifies the missing indexes.

When designing the physical database, the designer should create indexes for each key, alternate key and foreign key on the tables. As the database stored procedures are defined, indexes should be created on the join columns used in the stored procedures. Some DBMS's avoid the cost of the indexing analysis requirements by indexing every column in each table.

6. DBMS CONGESTION

6.1 One database per DBMS Server

A single machine can support many DBMS servers. The separate DBMS servers can from a single vendor or from different DBMS vendors, for example, ORACLE, SQL SERVER, DB2, My SQL, etc. Each DBMS server can have many separate databases or schemas where each database is the logically connected data for an application or set of processes.

DBMS congestion occurs when a process uses the same DBMS server for its data source database and its target database. During the application processing, when data gets transferred from the source database to the target database, the DBMS instance has to ensure that the operations for each database leave the database in a consistent state before beginning the operations for the other database. This means that the source and target operations must run sequentially. A better solution is to have the two databases on different machines, then the source and target database operations can run simultaneously.

6.2 One DBMS per hardware server

At Denison Mines, the time required for the overnight batch processing was exceeding the available processing time. The hardware architecture was four machines connected to four network attached storage disks. The batch processes had never been configured to maximize throughput. The proposed solutions were to buy another machine to take up part of the processing load or to reconfigure how the processing was distributed on the machines. The OLTP databases were moved to a DBMS on a single machine and the reporting databases were moved to a DBMS on a different machine. The application processes were moved off the DBMS machines onto separate application server machines. Also, the use of the network attached storage was segregated so that the two DBMS servers did not interfere with each other's disk operations. The result of the reconfiguration was to cut the batch processing time in half and a new machine was not required.

7. DISK STORAGE

7.1 Non Interference of Discs with the DBMS

When a new machine is purchased to be used as a DBMS server, it is usually ordered with large amounts of RAM, multiple CPUs, and many hard disks. The large amount of RAM will increase performance because the paging system will be able to keep more data in memory for faster access. The multiple CPUs will not be of value since all the database write operations must be sequential in a single CPU. The machine's hard disks are usually configured as a RAID array with the data striped across several disks to enable data recovery.

When a large disk, RAID array or network attached storage is logically partitioned into several disk drives, the disk drives are implemented as cylinders on the physical device. The DBMS is normally configured to use a logical disk for data, a different logical disk for indexes and so forth. The transfer of data from logical disk to logical disk requires a large disk head movement as it locates the correct cylinder. This physical head movement is the slowest operation that the machine has to perform. The normal configuration of a DBMS across several logical drives entails the movement of the disk head from logical drive to logical drive as the DBMS instance first writes the log, then the data and finally the indexes to the data before completing a data row operation. These disk operations would also have to be done sequentially because only one disk head is available for all the logical disks.

The configuration of hardware for a DBMS server depends on the DBMS that will be installed. Each installed DBMS server will have a certain logical structure for its files. For example, DB2 and MS SQL SERVER use a file system for the data space, a file system for the index space and a file system for the DBMS log file. ORACLE uses a file system for the data space, a file system for the index space, a file system for the roll back log and a file system for the commit log.

For DB2 and SQL Server, the DMBS server should be configured over 3 physically separate disk drives: one drive for data, one drive for indexes and one for the log. The motion of the disk head would be minimized because there is no contention between the log, the data and the index. Each DBMS should be spread over 4 physical drives (DBMS paging system, data, index and log). Also, the three database disk operations could be carried out in parallel thus reducing the total disk operation time.

At Brobeck, Phleger and Harrison, a new machine was purchased for the time management data warehouse. The data warehouse was implemented using SQL Server and the machine was configured with 4 separate mirrored disk drives controlled by a dual ported disk controllers. The mirrored drives allowed for 8 data reads to occur concurrently. The DBMS was configured with a separate disk for the data, indexes and log. This implementation was the optimum choice for Microsoft SQL Server. During production processing, the time management data warehouse machine out performed a much larger DBMS server with the normal RAID configuration by a factor of two.

7.2 Solid State Drives

The highest performance bottleneck is created by the use of physical disk drives. The movement of the read/write head over the spinning disk to access data is measured in milliseconds. Replacing the disk drives with solid state drives reduces data access time to the time required for an application remote procedure call.

8. SUMMARY

In summary, for the best data architecture performance, a single database runs on a single DBMS instance that is installed on a single hardware machine that has four mirrored solid state drives with dual ported disk controllers; one drive each for the paging system, the data, the indexes and the database log. Addressing the architecture issues is most cost effective since no application software changes have to be made. Fixing the data model implementation and interface issues can be time consuming and expensive because software application modifications are involved.

