Optimizing The Locking Methods in Distributed Database Systems

Mehdi Assefi
Dept. of Computer Engineering
Islamic Azad University, Neyshabur Branch
mhd_asefi@msn.com

Abstract

The main objective of this paper is to analyze the results of evaluation on different capabilities of concurrency control in Distributed databases, and examine different choices for locking and required settings on a sample database.

We have chosen the Oracle database of a paperless filing system as the basis for examining. In this distributed database, different transactions could access to a database through different stations and implement some basic operation on it. As we could assume, using a proper mechanism for concurrency control seems to be necessary. By evaluating possible choices and effect of their performance on the throughput of the system, and mean response time of transactions, we will try to evaluate pro and con for each method.

1. INTRODUCTION

As mentioned earlier, Oracle enables the developers to change the architecture of the database using the programming, and at the background. Host languages will give this chance to developers, by making connection with Oracle. using a proper mechanism for concurrency control seems to be necessary. By evaluating possible choices and effect of their performance on the throughput of the system, and mean response time of transactions, we will try to evaluate pro and con for each method.

2. Manual setting at the background

2.1. Using PL/SQL and Java

As you know, you can use a structural control for grouping SQL commands in a PL/SL block and then send the block to the Oracle.
it is also possible to use PL/SQL and Java procedures and reduce calling of the database in the program. For example, for running one SQL command, 10 calling will be needed, but for running a micro program containing 10 SQL commands, only one calling will be needed. This will reduce the network traffic. We can compile PL/SQL and Java procedures separately and put in a database. These procedures will be passed to PL/SQL motor after calling. Moreover, only one copy of these procedures must be loaded in the memory for being used by multiple user.

PL/SQL can also cooperate with Oracle developing tools such as Oracle Forums and Oracle Reports. By adding processing power, performance will increase. Using PL/SQL will cause calculation to be run more efficient and fast and without being called within Oracle. Result will be reduction of response time and network traffic.

2.2. Using advantages of row locking

Oracle enables databases to lock row of data and avoid locking the whole table. Row locking will make increasing of concurrency possible and many users will be able to access to the rows of a table. By this, performance will increase sharply.

2.3. Configuring parameters of a database

Using the host languages and applicable parameters, we are able to optimize the structure of a database. We hereby mention some of these parameters.

db_block_size

This parameter defines the size of blocks in the database. As we know, block is the scale for transferring information from the disk. By increasing the size of the block, we will have this chance to read all items that are at the same class. This will reduce the mean time needed by disk for accessing a unit of data. This parameter will be defined when we create the database and will be permanent. Precise identifying the size will be useful for preventing the information to become scrappy.`

db_cache_size

This parameter will identify the size of data buffer. Data will be read from disk and will be placed in this buffer in order to minimize access request to the disk. Results show that irregular increase of this buffer will drastically reduce the performance, which would be due to the extra burden posed on the operating system for buffer management and reducing the memory (result of occupying the buffer by data).

db_file_multiblock_read_count

This parameter identifies number of block which will be read in any referral to the disk.

Java_pool_size

In case if a database uses Java codes, proper set up for this section will have positive effect on the performance. This place creates a location in the buffer for putting Aplet Java codes.

large_pool_size

If database is supposed to transfer huge blocks, proper set up for this section will be very helpful. These objects will be considered when online backup is undertaken and object is loaded in the memory. Hereby list of these parameters for our database follows. We will change the parameters to evaluate the results.

Cache and I/O

db_block_size=8192

db_cache_size=25165824

db_file_multiblock_read_count=16

Cursors and Library Cache

open_cursors=300

Database Identification

db_domain=""

db_name=ASY_DB

Diagnostics and Statistics

background_dump_dest=G:\oracle\admin\myora\bdump

core_dump_dest=G:\oracle\admin\myora\cdump

timed_statistics=True

user_dump_dest=G:\oracle\admin\myora\dump

File Configuration

Control_files=("G:\oracle\oradata\ASY_DB\ctrl1ASY_DB.ctl",

"G:\oracle\oradata\ASY_DB\ctrl3ASY_DB.ctl")

Instance Identification

Instance_name=ASY_DB

Miscellaneous

Compatible=9.2.0.0.0

Optimizer

Hash_join_enabled=TRUE

Query_rewriter_enabled=FALSE

Star_transformation_enabled=FALSE

Faster_statr_mttr_target=300

Security and Auditing

Remote_login_passwordfile=EXCLUSIVE

Sort, Hash Joins, Bitmap Indexes

Pga_aggregate_target=25162824\sort_area_size=524288

System Managed Undo and Rollback Segments

Undo_management=AUTO

Undo_retention=10800

Undo_tablespace=UNDOTBS1

3. Optimizing I/O parameters related to data involvement

In order to see the change in the events related to I/O, in any system we can use perfmon. By observing counters related to the system parameters, we could have required changes. Following parameters are available:

Disk reads/Sec	**number of reads per second**
Disk write / sec	**number of writes per second**
Disk transfer / sec	**number of transfers per second**
Avg. Disc Sec/Read	**average time spent for reading**
Avg. Disc Sec/Write	**average time spent for writing**
Avg Disc Sec/Transfer	**average time spent for reading and writing**
Avg Disk Queue/Length	**average number of I/O in I/O sub system**

For example, in the system that we analyze, I/O operations will be complete in 20-30 ms. If this amount increase from 6 ms for a disk, system will halt and database must be optimized.

3.1. Using UTLBSTAT and UTLESTAT

To useful parameters within Oracle are UTLBSTAT and UTLESTAT. There are some documents inside ORANT\RDBMS80\ADMIN folder that use internally by Oracle. When the database is going to be created, CATPROC.ORA and CATALOG.ORA will run from this folder. UTLBSTAT and UTLESTAT also run from this folder.

UTLBSTAT sets up internal tables and creates an instant photo from Oracle's internal counter. It will then create another photo after a high loaded operation by UTLESTAT and compares the results. UTLESTAT introduces number of complete counter values and a full statistical briefing during this period. It must be considered that UTLBSTAT.SQL and UTLESTAT.SQL both have a CONNECT INTERNAL string at top. Obviously both of these documents will have problem in NT. This row must be marked as explanation. For doing this, we put Rem at the beginning of them. UTLESTAT and UTLBSTAT could be run using SVRMGR30 application. After being connected to the database as INTERNA or SYS we can run UTLBSTAT as follow:

D:\ORANT\RDBMS80\ADMIN\UTLBSTT;

After a heavy load operation and after a considerable amount of time UTLESTAT must be implemented using the following pattern:

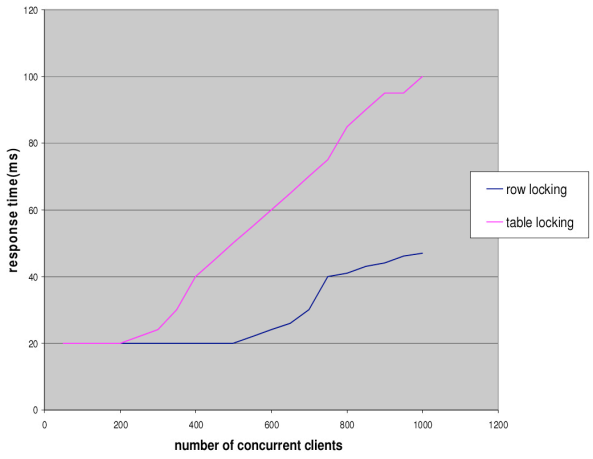
D:\ORANT\RDBMS80\ADMI\UTLESTAT;

4. Evaluating the performance

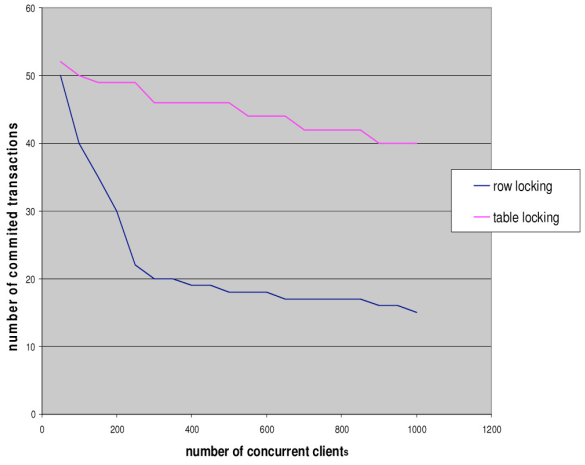
We now want to analyze the implementing time of our database's transaction considering different parameters. For increasing the work content and enhancing the workload, we will run transactions with increasing content on the database. We will then calculate the response time of the transactions, using the introduced calculation tools.

We first apply table locking rather than row locking. Results show that row locking will reduce the time that transactions need to access data, and in result response time will decrease and operational capability will increase. Chart related to the effect of these changes is depicted in the chart below.

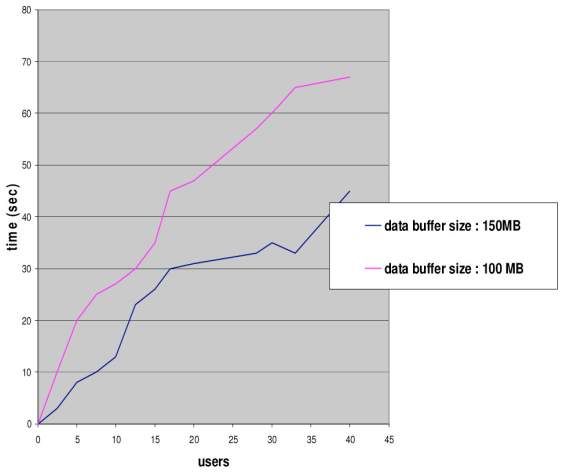
We use 100 MB and 150 MB buffers. Database volumes in both cases were 1.6 GB. We used IMPORT/EXPORT and INSERT commands in this evaluation. Number of clients was vary from 10 to 40. Results show that using a bigger buffer will result in reduction of the time needed by transactions to access the data, wich will reduce the operation time and will increase operational capability. Following charts will depict the effect of these changes.



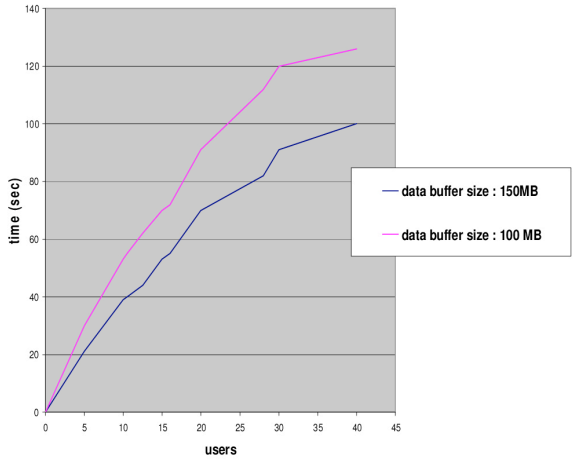
Effect of using row locking on transaction's response time



Effect of using row locking on throughput of system



Effect of using a bigger locking queue on response time of INSERT transaction



Effect of using a bigger locking queue on response time of IMP/EXP transactions

5. Summary

Oracle database is capable of doing manual setting using the programming. Majority of operation related to managing concurrency control is possible to be undertaken manually. The focus will be on increase or decrease on the length of queues and data buffers, and row locking and table locking. By considering system specifications ad proper adjustment for these parameters and other parameters of I/O, we will be able to considerably increase the performance of the database.

REFERENCES

[1] PETER M. G. APERS, ALAN R. HEVNER, S. BING YAO: OPTIMIZATION ALGORITHMS FOR DISTRIBUTED QUERIES. IEEE TRANS. SOFTWARE ENG. 9(1): 57-68(1983).

[2[Philip A. Bernstein , Nathan Goodman , Eugene Wong , Christopher L. Reeve , James B. Rothnie, Jr., Query processing in a system for distributed databases (SDD-1), ACM Transactions on Database Systems (TODS), v.6 n.4, p.602-625, Dec. 1981 [doi>10.1145/319628.319650]

[3] Philip A. Bernstein , Dah-Ming W. Chiu, Using Semi-Joins to Solve Relational Queries, Journal of the ACM (JACM), v.28 n.1, p.25-40, Jan. 1981 [doi>10.1145/322234.322238]

[4] P.A. Black and W.S. Luk. A New Hueristic for Generating Semi-Join Programs for Distributed Query Processing. Proc. IEEE COMPSAC, December, 1982, pp. 581-588.

[5] A.L.P. Chen and V.O.K. Li. Properties of Optimal Semi-Join Programs for Distributed Query Processing. Proc. IEEE COMPSAC, November, 1983, pp. 476-483.

[6] A.L.P. Chen and V.O.K. Li. Deriving Optimal Semi-Join Programs for Distributed Query Processing. Proc. IEEE INFOCOM, April, 1984.

[7] A.L.P. Chen and V.O.K. Li. Improving Semi-Join Programs for Distributed Guery Processing. To appear in Proc. IEEE COMPSAC, November, 1984.

[8] To-Yat Cheung, A Method for Equijoin Queries in Distributed Relational Databases, IEEE Transactions on Computers, v.31 n.8, p.746-751, August 1982 [doi>10.1109/TC.1982.1676081]

[9] D.M. Chiu. Optimal Query Interpretation for Distributed Databases. Ph.D Th., Harvard University, December 1979.

[10] Dah-Ming Chiu , Philip A. Bernstein , Yu-Chi Ho, Optimizing chain queries in a distributed database system., SIAM Journal on Computing, v.13 n.1, p.116-134, Feb. 1984 [doi>10.1137/0213009]

[11] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM, v.13 n.6, p.377-387, June 1970 [doi>10.1145/362384.362685]

[12] E. F. Codd: Relational Completeness of Data Base Sublanguages. In: R. Rustin (ed.): Database Systems: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California : (1972).

[13] Robert S. Epstein, Michael Stonebraker: Analysis of Distributed Data Base Processing Strategies. VLDB 1980: 92- 101.

[14] G.D. Held, M.R. Stonebraker and E. Wong. INGRES - A Relational Data Base System. Proc. NCC, 1975.

[15] A. HEVNER, QUERY OPTIMIZATION IN DISTRIBUTED DATABASE SYSTEMS. PH.D. TH., U. OF MINNESOTA, 1979.

[16] ALAN R. HEVNER, S. BING YAO: QUERY PROCESSING IN DISTRIBUTED DATABASE SYSTEMS. IEEE TRANS. SOFTWARE ENG. 5(3): 177-187(1979).

[17] K.T. HUANG, QUERY OPTIMIZATION IN DISTRIBUTED DATABASES. PH.D. TH., M.I.T., DECEMBER 1982.

[18] W. S. LUK, LYDIA LUK: OPTIMIZING SEMI-JOIN PROGRAMS FOR DISTRIBUTED QUERY PROCESSING. ICOD 1983: 298-316.

[19] SUGIHARA, K., ET. AL. OPTIMIZATION ALGORITHMS FOR PROCESSING SIMPLE QUERIES IN STAR NETWORKS. PROC. IEEE COMPSAC, NOVEMBER, 1983, PP. 547-554.

[20] CLEMENT T. YU, K. LAM, C. C. CHANG, S. K. CHANG: PROMISING APPROACH TO DISTRIBUTED QUERY PROCESSING. BERKELEY WORKSHOP 1982: 363-390.

[21] C. T. YU , C. C. CHANG, ON THE DESIGN OF A QUERY PROCESSING STRATEGY IN A DISTRIBUTED DATABASE ENVIRONMENT, PROCEEDINGS OF THE 1983 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, MAY 23-26, 1983, SAN JOSE, CALIFORNIA [DOI>10.1145/582192.582203]