Optimization of massive data retrieval performance based B-tree index

Lihua Geng [1, a]

[1] Changchun University of Science and Technology, Changchun, China

[a]glhchangchun@163.com

Keywords: Oracle 11g; B-tree index scan; massive data retrieval; performance optimization; the average transaction response time;

Abstract: With the rise of the Internet Web 2.0, the paper studies a use of B-tree index technology supported by Oracle 11gin mass data retrieval. B-tree index scanning mode use the index selectivity, histogram etc to select the best execution path to improve query performance. Put forward the solutions to avoid the useless index resulted by non-equality operation, Null operation and functional operation. After the comparison and analysis of before and after optimization of Oracle Database Performance ,the result shows that the average response time of the optimized Oracle database is better than before.

Preface

With the growing size of the database, the database, storing massive data, will take up a lot of disk space, so the query time will be extended. In the current research on query strategy optimization is mainly focused on the design for the access mode on the physical layer, relying on hardware technology and performance to achieve the aim of improving the efficiency of the query. Today many researchers have already transferred efficiency optimization to the software side, which means the huge technology space and potential of mass data query strategy.

Timing and indexed data columns are the important factors affecting the index's reasonableness. Height of almost all the B-tree indexes is 2 or 3, so to find a key in the index usually takes only two or three times I /O [1]. The one kind of index scan mode which has pertinence includes index uniqueness scan, index range scan, index full scan and index fast full scan [2], so you can choose different indexes timing depending on the various situations [3]. From an efficiency point of view, if you access, in the table, a small part of the data, the use of index-scan method will be more efficient; if you access the most or even full of the table, using full-table-scan approach is more successful [4]. Let index inherits storage attributes of table space .when you create the table space, you do not need to manage the storage properties of each index structure, effectively reducing the workload of the database administrator [5]. Before the implementation of Oracle SQL, first determine the execution plan for SQL statement. Execution plan includes two aspects: to determine the data scan path and determine the data collection and solution approach [6]. In this paper, we will achieve the purpose of optimizing the performance of massive data retrieval through the study of the Oracle database indexing technology.

An index scan basic theory

First, query the indexed object based on access conditions, then return the result to the upper operating level, and finally get the data you want by accessing the data table. Data scanning method is "INDEX UNIQUE SCAN", namely through the index to retrieve a row of data and return a single Row id. The benefit of this approach is to reduce SQL execution cost and play a significant role in optimization on mass data retrieval performance.

The way to achieve an index unique scan is: create a unique index or a primary key index on the data sheet. Because this query returns only one row of data, the execution is an index unique scan. Obviously, this is an efficient way to index call. Index range scan is used when accessing the index, traversing the indexes, and storing a number of lines of data. The most representative case is querying data in some range. If the index is a combinational unique index type ,it just uses part of the indexed column and even if the query result set is multiple lines, always, the index range scan will occur. Full index scan is firstly directly reading the index blocks in, then following the order of the data to continuously read the data. Cases of a full index scan, there are two: SQL statement requiring order; index containing columns SQL statements need to query. Index fast full scan mode is to scan data blocks of all the leaf nodes of indexes and does not require to sort the query result. If you need to sort, select the full index scan; if not, just count the amount of data , the index fast full scan is better.

Selection of data scan paths

Under appropriate conditions, creating a reasonable index can help improve the efficiency of massive data queries. Data scanning path selection should be decided by the data storage density ,the data storage continuity, as well as the index dispersion degree of the data table. The database must allocate storage space to the index because of the index structure, while it is required to maintain the index, accommodate index to database tables, make sure whether the using time of indexes is satisfactory and whether the index column has the appropriate decisions on the index after the DML operation on the database table. It is found from comparing Oracle Data scanning that two cases can use the index: using the B-tree index is the best choice to improve query efficiency and reduce the consumption of system resources while referring to accessing a small percentage of the rows , or even a line in the table; It is not necessary to use the data table if the index contains enough information to answer this query while it is required to obtain or access many rows in the table, and even the entire data table.

Choosing columns to create the index is a key factor in determining the performance of the database. Rules for selecting data columns to create the index are as follows:

First, create primary key constraints for each data table;

Second, create a unique constraint on data columns with the unique and non-empty constraints;

Third, create an index on foreign key columns;

Finally, create the index on predicate columns.

For massive data tables, when you create an index, you’d better allocate a separate table space to the index structure, and stores them with it to simplify the management of the index structure. If the index of database has a variety of needs for storage space, you can get different sizes of the table space based on the index structure’s demand on storage space, then allocate these to the appropriate index structure to achieve a reasonable and effective use of table space.

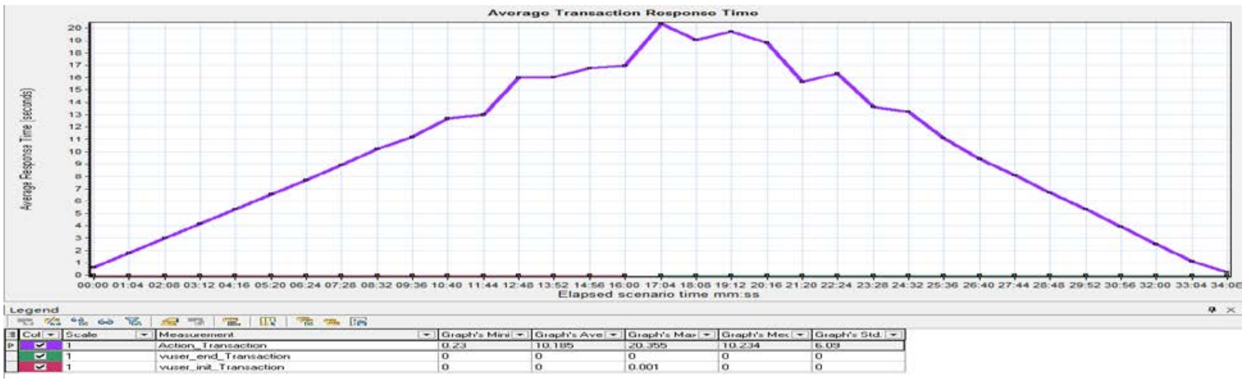
Massive data retrieval performance optimization

From the perspective of index properties, how to create an index and what kind of an index is created make retrieval performance of huge amounts of data improved and optimized. The ratio ranges between 0-1. While the number of index key value gets closer to the number of rows in a database table, the selectivity of the index gets closer to 1,which means higher selectivity; Conversely, closer to 0, the selectivity is worse.

In the case of the determined data table structure ,different set objects makes different SQL execution plans and the optimization work is reflected on the choice of index objects. In massive database systems, histogram can be applied to get data distribution of Oracle database table, and then to optimize according to the histogram statistics and choose the best execution plan to achieve histogram statistics. When data distribution of the table is uneven, revise the execution plan in the optimizer based on the statistics of the histogram, and ultimately achieve optimal performance.

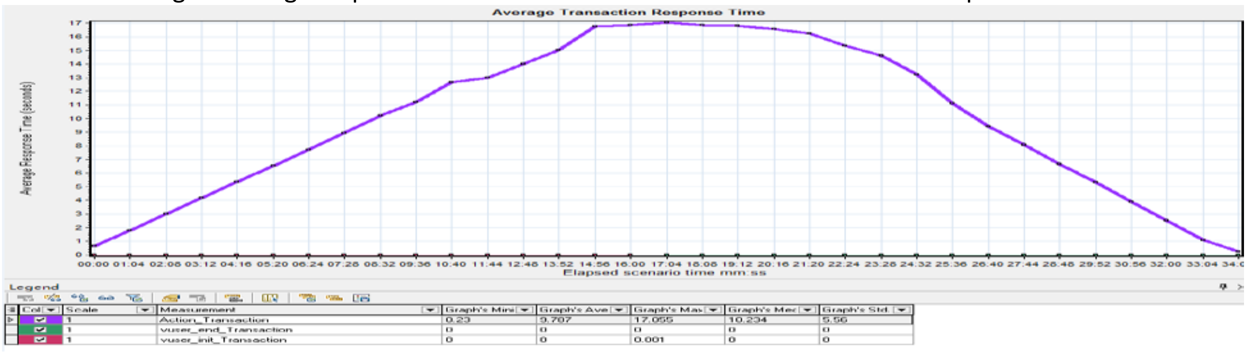
Performance Testing and Result Analysis

Fig. 1 and Fig. 2 show that at the same time and the same number of concurrent cases, the average response time of transaction before and after optimization of Oracle database .



Col	Scale	Measurement	Graph's Mini	Graph's Ave	Graph's Max	Graph's Med	Graph's Std.
🟪 ✓	1	Action_Transaction	0.23	10.185	20.355	10.234	6.09
🟩 ✓	1	vuser_end_Transaction	0	0	0	0	0
🟥 ✓	1	vuser_init_Transaction	0	0	0.001	0	0

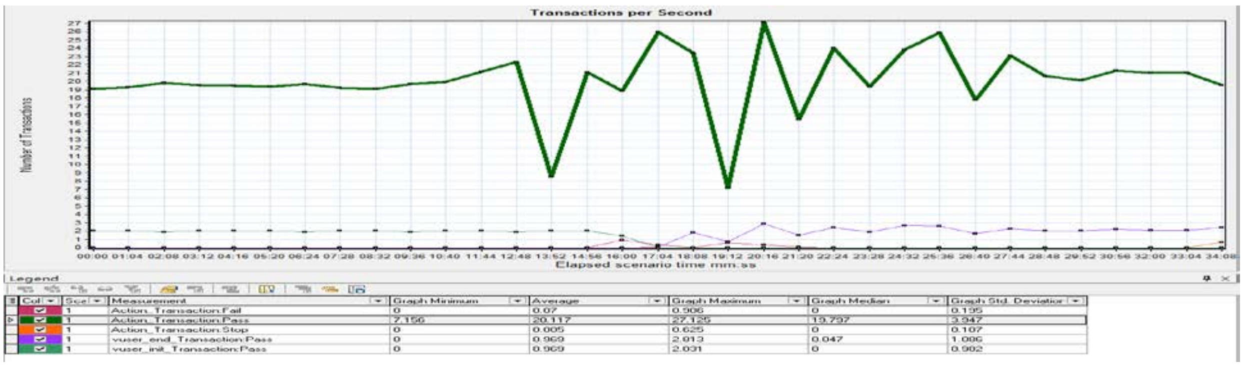
Fig. 1 Average response time of Transaction of Oracle database before optimization



Col	Scale	Measurement	Graph's Mini	Graph's Ave	Graph's Max	Graph's Med	Graph's Std.
🟪 ✓	1	Action_Transaction	0.23	9.707	17.055	10.234	5.56
🟩 ✓	1	vuser_end_Transaction	0	0	0	0	0
🟥 ✓	1	vuser_init_Transaction	0	0	0.001	0	0

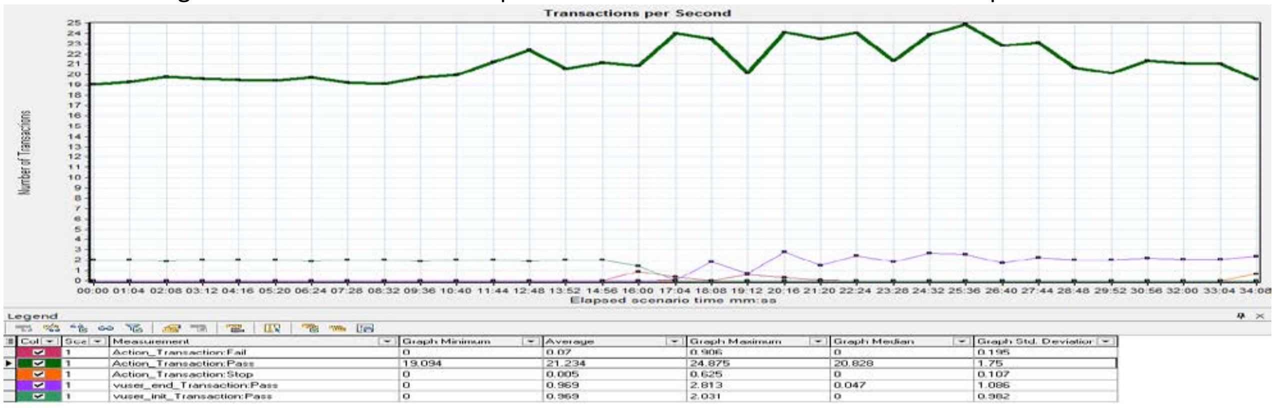
Fig. 2 Average response time of Transaction of optimized Oracle database

As you can see, the maximum of the average response time of transaction of Oracle database is 20.355s, the minimum is 0.23s, with an average of 10.185s. The maximum of the Average response time of Transaction of optimized Oracle database is 17.055s, the minimum is 0.23s, with an average of 9.787s.



Col	Sce	Measurement	Graph Minimum	Average	Graph Maximum	Graph Median	Graph Std. Deviation
🟥 ✓	1	Action_Transaction:Fail	0	0.07	0.906	0	0.195
🟩 ✓	1	Action_Transaction:Pass	7.156	20.117	27.125	19.797	3.947
🟧 ✓	1	Action_Transaction:Stop	0	0.005	0.625	0	0.107
🟪 ✓	1	vuser_end_Transaction:Pass	0	0.969	2.013	0.047	1.086
🟩 ✓	1	vuser_init_Transaction:Pass	0	0.969	2.031	0	0.902

Fig. 3 Number of transactions per second of the Oracle database before optimization



Col	Sce	Measurement	Graph Minimum	Average	Graph Maximum	Graph Median	Graph Std. Deviation
🟥 ✓	1	Action_Transaction:Fail	0	0.07	0.906	0	0.195
🟩 ✓	1	Action_Transaction:Pass	19.094	21.234	24.875	20.828	1.75
🟧 ✓	1	Action_Transaction:Stop	0	0.005	0.625	0	0.107
🟪 ✓	1	vuser_end_Transaction:Pass	0	0.969	2.813	0.047	1.086
🟩 ✓	1	vuser_init_Transaction:Pass	0	0.969	2.031	0	0.982

Fig. 4 Number of transactions per second of optimized Oracle Database

As you can see, the maximum of number of transactions per second of Oracle database before optimization is 27.125, the minimum is 7.156, with an average of 20.117. The maximum of number of transactions per second of optimized Oracle Database is 24.875, the minimum is 19.094, with an average of 21.234.

Experimental results show that, the change curve of optimized Oracle database is steadier than before optimization. Therefore, Average response time of Transaction of optimized Oracle database is better than before optimization.

Summary

The default B-tree index structure of Oracle database can optimize query performance of massive data .In order to play full role of the index, create the index in the appropriate data table structure, and select a reasonable index column. Full table scan query is inefficient and consumes more system resources, I / O times than the index scan, so the index is a crucial aspect during the database design and development process. However, unreasonable indexes not only make performance degradation, also lead to the collapse of the database. Therefore, the deep exploration of the index structure and the use mechanism is the key to massive data query performance optimization. We can also optimize the query performance of massive data through subregion technology, SQL statements optimization, and operating system, etc.

Reference

[1]Charalambides,Stelios. Introduction to SQLTXPLAIN. Oracle SQL Tuning with Oracle SQLTXPLAIN. Apress, 2013. 1-16.

[2] Jilan Zhang, Jiaju Hao, Yongkang Xu. Performance Optimization Based on ORACLE Database, the National Symposium of Information and Electronic Engineering Academic, 2011: P336-340.

[3] Wang, Peng, Ai-xue Tian, and Chang Guo .Table Partitioning Technology Based on Massive Data. TELKOMNIKA Indonesian Journal of Electrical Engineering 12.3 (2014): 1676-1686.

[4] Jiang Zhang. Oracle Database Performance Optimization Strategy. Modern Computers: the second half Edition 24 (2013): 45-48.

[5] Zhaoyu Liu.Oracle Database Performance Optimization. Grand Week 9 (2011): 121-122.

[6] Belknap, Pete, et al. Oracle Database SQL Tuning Guide, 12c Release 1 (12.1) E15858-15. (2013).