HP Big Data Analytics: Effects of Memory Management and Parallelism on Query Performance

Oregon State University
CS 461
2016-2017

Prepared By:
Nic Desilets, James Stallkamp,
and Nathaniel Whitlock

October 26, 2016

Abstract

Computational efficiency and performance are paramount to being able to quickly and effectively analyze data within a reasonable time frame when working with very large data sets. Oracle 12c, a cloud enabled relational database, performs sub-optimally by default when it comes to analyzing the massive amounts of telemetry data generated from HP PageWide Web Press printers. In order to achieve better performance from Oracle 12c, we experimented with various memory management methods in order to maximize memory usage during a query. We also modified database table physical design parameters such as partitioning and measured the effects on parallel processing. Additionally, we developed a toolkit alongside this project in order to accurately measure and compare the efficiency of different methods of memory management and parallelism on database queries. Using this toolkit, we have found the most efficient modifications to the relational database to be ... based on quantitative comparative analysis of partition designs.

Problem Definition

Modern relational database systems are crucial in fields such as business and research. The servers ability to read, write and perform computations on data within a table is not fully utilized until the results are accessible within a reasonable time frame. HPs PageWide Web Press Big Data team manages such relational servers and generate roughly 250GB of data each day. The data generated is normalized and loaded into databases in order to store it so it can be queried for troubleshooting and performing engineering analytics. The machine that the server is on is extremely powerful and should be able to perform operations very quickly, however, the server can be slow when it has not fully utilized the available resources. The core of this problem is to investigate why the server is under utilizing the physical resources of the machine. This involves experimenting with many database configuration settings and table partition designs to help improve the performance of the server.

Currently nightly reports against the database can take up to two hours to complete, the client requested that this process be reduced as much as possible. Ideally this would mean that when engineers ran queries against the database they would have their response in five seconds or less. The under allocation of resources leads to longer periods of wall time during data aggregation and prevents accessibility to real time performance metrics. By experimenting with database configuration settings and table partition designs, we hope to reduce the wall time observed during routine analytics.

Problem Solution

In short, the problem is that the server is taking an overabundance of time processing queries despite having idle resources. Our goal is to increase the efficiency of the server by configuring the memory allocation settings and ensuring the majority of the available processors are utilized during a request. To do this we will develop a toolkit that will allow users to examine information pertaining to the performance specs of the query in process. This toolkit will consist of database query statements which will query the dynamic performance views to generate a report. Each report will represent the performance results of a query on the database. A report will be generated for each unique database setting configuration and partition design, these will allow for a comparative analysis of each designs performance run-time. The results of this analysis will be used to adjust settings on the server and in queries to improve resource usage and run time.

We will be able to isolate the top performing designs by the outcome of the run time analysis. Since there are different analytic jobs run against the database, there will likely be different designs which will be more suitable to specific requests. By comparing the top performing designs we will show how much the system performance increased as well as the actual run time reduction.

PERFORMANCE METRICS

We will primarily be gauging our performance by using the Performance Efficiency Index (PEI) in order to measure query execution times against differing data partitioning strategies, memory management strategies, as well as varying degrees of parallel execution. The PEI is simply database time, the time a database process spends executing code, divided by wall clock time which is time elapsed in the real world. In addition to PEI, we can also measure CPU usage, memory usage, disk input and output, and the paths taken within the database to access data. From a users perspective, wall clock time is going to be one of the most important metrics since it will determine the perceived responsiveness of queries made against the database.

Kirby Sand	Date
Andy Weiss	Date
Nic Desilets	Date
James Stallkamp	Date
Nathaniel Whitlock	Date