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

Oregon State University
Senior Design: 2016-2017

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

October 14, 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 RAM 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

Each and every day, HPs PageWide Web Presses generate roughly 250GB of data. 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 current database configuration only utilizes a small portion of the available 72 processor cores and 3TB of RAM it has available. There are plenty of physical resources available, yet the current database configuration tends to under allocate memory and CPU usage for large queries. When the Programming Global Area (PGA) reaches a memory threshold the process much reach out to disk in order to complete. Reading or writing to disk is one of the most costly operations for a program. Additionally, there are instances where a query process is allocated a large amount of cores to work with but when the processor use ratio is evaluated it shows that only a couple of cores were used on average.

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 responding very slowly, despite having idle resources. Our goal is to help improve resource usage of the server, and to improve the performance efficiency index. To do this we will develop a toolkit that will allow users to examine information pertaining to the performance specs of the query. This toolkit will consist of PL/SQL statements which will query the dynamic performance view to generate a report. Each report will represent the performance results of a query of the database. Every report will serve as the unique profile of an experimental design that was set up in order to test the performance of the physical hardware. These reports will help users to adjust settings on the server and in queries to improve resource usage and runtime. Our metric for success will be to improve the performance efficiency index, which is the database run time / real time. Specifically we would like the queries to be able to run fast enough to run in real time, the client specified 5 seconds or less. The results we will take away from this will be data showing the improvement gained. The data will show how much we increased the performance efficiency index, as well as the actual runtime 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