Progress Report

HP: Big Data Analytics
Group 13

Oregon State University
CS 462
2017

Prepared By:
Nic Desilets and Nathaniel Whitlock

March 18, 2017

Abstract

This document serves as a recollection of our progress made over Winter break and also the past seven weeks of Winter term. In this document we each discuss our personal experiences and bring them into a collective overview of what has been accomplished as well as the current state of our project. Our individual experiences are also discussed in the video presentation.

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

CONTENTS

1 Introduction

The main purpose of this document is to outline the work our team has done since the end of Fall term. This document will be split into two main sections which will focus on the individual experiences of our two group members. Throughout the term we each focused our individual efforts on a specific topic, so breaking the time line up seems most appropriate. Our experiences in the project to date are summarized and broken down into a week by week digest of problems encountered, progress made, and any intermediate goals between weeks. Finally, a retrospective has been put together of our positive experiences, outstanding issues, and future actions that need to be taken.

2 Purpose and Goals

The primary purpose of this research project is to develop a good understanding of how Oracle database can be setup for analytical workloads that typically involve full table scans. Our clients have been running analytical queries on their instance of Oracle database which by default is not optimized for these types of queries. This resulted in queries taking several hours to run and not fully utilizing the system's available resources. Our ultimate goal is to reduce query execution times by developing an understanding of how we can tune the database to better fit our clients use cases.

During the period of time since fall term, our main goal has remained centered around optimizing the Performance Efficiency Index (PEI), this would ultimately lead to a reduction in query execution time. The index can be calculated by resolving the following formula:

$$PEI = (DB\ Time\ /\ Wall\ Time)$$

Additional goals included generating experimental data that we could use to support our hypothesis and assisting in the preparation of a presentation that covered our research and experimental results. The experimental data will also be used to direct further investigation into the effects of the features available within Oracle 12c.

3 Current State

At this point in time in our capstone project we have finished the bulk of our research and were able to identify key parameters and features that can increase database performance for analytical workloads. In addition to this, we have created a suite of tests that demonstrate the performance gains of modifying certain parameters and enabling features. During evenings, we have been collecting data from our experiments so that we can visualize the results, the visualizations are then analyzed at our weekly meetings to ensure that we are on the right track with our research and experimental design. Some of the final experiment results were formally graphed and used in the presentation that our clients gave at the Oracle HOTSOS convention. Additionally, they were provided with topic specific slides that detailed the features that we experimented with.

4 Timeline of Events

4.1 Nic Desilets

4.1.1 Winter Break

During the first week of Winter break we met with our clients again to discuss what areas of research we should each focus on. We decided that for now we should focus on learning about the different trace events in Oracle database. These trace events can be analyzed to gain insights on how the database operates and why it makes certain decisions when determining how queries are executed. Some of these parameters can include degrees of parallelism, private work area size per parallel process, types of join algorithms to be used, etc.

At the end of the meeting we also discussed the possibility of creating a ramdisk which we could move the temp tablespace into as a temporary solution to alleviate some of our performance problems. The temp tablespace is used when serial or parallel processes working on a query run out of their allotted memory. These processes must swap to the temp table which resides on disk similar to when an operating system must swap to disk. If we were to move the temp tablespace into a ramdisk this would drastically reduce the amount of time spent waiting on disk IO in scenarios when the temp tablespace must be used. I did some testing on my own instance of Oracle database and saw large performance improvements when a ramdisk for the temp tablespace is used.

I also started on a rudimentary web based toolkit that we could use to automate some of our testing. The basic idea of it was to specify what query to run and what performance metrics to keep track of while the querying is executing. Once the query finishes executing a report containing the specified performance information is returned. However, due to time constraints we decided that we should focus on researching the database and finish the toolkit later. The bulk of our research was due at the end of February and so we only had a couple months left.

4.1.2 Week One

During our first meeting with our clients of Winter term we determined that we should meet twice a week to move quickly with our research. At this point in time, our priorities were to figure out how the Program Global Area (PGA) works and how the System Global Area (SGA) work within Oracle database. Most of our problems seemed to be coming from the PGA portion of the database. Serial and parallel process memory work areas are defined by parameters relating to PGA and by default are suboptimal for analytical workloads and don't take advantage of our machine's resources. If we could figure out how the PGA works then we could develop an understanding of what parameters to tweak to optimize the database for our query workloads. We came up with a list of parameters that determine private process work area size, cumulative private process work area size, maximum private process work area size for parallel processes, and a couple more. Because these values are too small by default, this causes the database to not make optimum use of the memory available in our server (up to 3TB of RAM). We figured that increasing these values should help with achieving more optimum use of system resources. My goal for the next meeting was to come up with some slides detailing the structure of PGA and how these parameters affect each structure.

4.1.3 Week Two

This week I was focusing on cleaning up the slides that I had created on PGA and the PGA parameters. Another research topic that I looked into was the In-Memory Parallel Execution feature made available in recent versions of Oracle database. Because our queries make heavy use of parallel processing, it made sense to look into how this feature might increase database performance under our specific use cases. Normally when a parallel process needs to fetch records it must go to disk to obtain the data blocks that it needs. However, with In-Memory Parallel Execution enabled these blocks can be cached in main memory and shared between parallel processes. This allows each process to first check if the buffer cache has the records it needs before resorting to going to disk, which can drastically increase query execution times. My goal for next weeks meeting was to start researching the SGA and the components inside of it.

4.1.4 Week Three

I started detailing the components within SGA that seemed relevant to increasing database performance. One critical component within the SGA is the buffer cache which caches blocks of data fetched from disk. This is the same buffer that the In-Memory Parallel Execution feature leverages to help speed up query execution times. Traditionally, only single blocks of data can be in this cache which works well for randomized Online Transaction Processing (OLTP) workloads. However, this does not offer optimal performance for databases that much process entire database objects such as Decision Support System (DSS) queries. Newly introduced in Oracle database 12c is a featured called Automatic Big Table Caching (ABTC) which complements the traditional block cache by caching entire database objects such as tables and indexes. Another new feature that we looked into was the In-Memory Column Store (IMCS) which can be used to speed up (DSS) type workloads. This is a separate structure in memory that caches table data in a columnar format which can improve the efficiency of analytical queries typically found in DSS workloads. In my own testing that Ive done I saw large performance improvements when the IMCS was enabled.

4.1.5 Week Four

I spent more time this week researching the IMCS and how it specifically benefits DSS queries. One interesting feature of the IMCS is that the column format in memory is specially optimized for vector processing. Most recent Intel and AMD processors have support for AVX and SSE instruction sets which are Single Instruction Multiple Data (SIMD) extensions. These instruction sets allow the processor to perform one instruction on multiple data elements all at once instead of going through each element of data one by one as normally done. When AVX instructions are used, Oracle database can process 8 data values at once per CPU core. This allows each CPU core within a processor to scan through billions of rows per second according to Oracle. At this point we felt that we had covered all the parameters of interest that we would like to test. We decided to switch gears from research to designing our testing suite to evaluate the parameters and features that we had researched in the past few weeks. My focus for the next week was to assist Nathaniel with writing the testing scripts for our test suite.

4.1.6 Week Five

We had finished some initial test scripts to test out some PGA related parameters and the ABTC feature. The results for the PGA tests were promising and showed system resources were being used more effectively and the temp tablespace was not being used as much, thus greatly improving query response times. The ABTC test results were generally promising but had some weird results that we were not expecting. We spent some time looking into what might have caused some of the odd results we were seeing from the ABTC tests. We concluded that part of it was the fact that having too large of ABTC cache relative the traditional block cache can degrade performance. Later on we created the IMCS test script and also spent some time redesigning our old tests so that they were in a more modular format. We ran into some issues with the IMCS test due to background processes filling up the column store in the background. These background processes are resource intensive and will have a negative impact on any queries that are running while the column store is still being populated. To remedy this issue I created a PL/SQL script that blocked the IMCS script from running until the column store was fully populated and the background processes were no longer running. This was done by the monitoring some internal database performance views that monitored the status of the column store.

4.1.7 Week Six

We ran into some problems this week with the IMCS tests and the PL/SQL script that I had written earlier. The main issue was that I tried to make the script modular by having the PL/SQL script as its own script and including it into the IMCS tests. However, there was a path issue and so the PL/SQL script never got called and the queries continued to run while the background processes were still populating the column store. We didnt really have enough time to look into the issue and so our solution was just to copy the contents of the contents of the PL/SQL script into each IMCS test script. This fixed the issue and allowed the IMCS tests to run properly. Once that issue was resolved we felt that our testing suite was complete and so we decided to finalize all the test scripts. Once finalized, we ran the IMCS test scripts and the results showed that the IMCS feature increases performance substantially, even more so compared to the other features and parameter tweaks that we tested prior.

4.1.8 Week Seven

At this point in time our research and test suites were more or less complete, or as complete as it was going to get, since the presentation was due next week. This didnt leave us much time to research any further topics but we felt what we had so far was sufficient. We were able to substantially increase the performance of our database using our test suites over our baseline from a couple months ago. However, we still felt that we had to research one more topic to bring everything together. This topic was Statement Queueing, which if enabled can queue queries to wait until sufficient system resources are available. All of our tests were done in a serial fashion which is not representative of a real world scenario. To help simulate a real world scenario we started many instances of our test scripts concurrently to simulate multiple users with high workloads. Our idea was to test this real world scenario with Statement Queueing enabled and then again with Statement Queueing disabled. When Statement Queueing is enabled, the database first checks to see if the query can have an optimal amount of resources allocated to it. If there is a high system load and not enough resources can be given to the query, it simply puts the query in line to run when enough resources are freed up. Otherwise the query is allowed to immediately run with an optimal amount of resources. In contrast of this, if Statement

Queueing is disabled then the database tries to run as many queries as possible even if it means allocating a suboptimal amount of resources to each query. Our results found that enabling Statement Queueing led to huge decreases in overall query processing time (thus increasing overall throughput) versus Statement Queueing being disabled.

4.2 Nathaniel Whitlock

4.2.1 Winter Break

During winter break our group only had one scheduled meeting with the clients. Kirby, James and Nic traveled for the holidays. Nic put together a front end GUI for the toolkit application early on, after learning a bit about Angular JS I was able to add some functionality. A fair amount of time was spent implementing trace functionality, this included adding some logic to the back-end of our application and set up an external tablespace within the database. In doing this, we were able to query flat files in order to display the content to the front-end.

Before winter break started our client had mentioned the likelihood that they would be invited to speak at an Oracle convention in Austin Texas called HOTSOS. Going into break, we discussed that fact that we would all need to continue pushing forward in case our clients we invited to present. Nic and I were in communication many times throughout the break and each independently researched topics such as memory management, parallel processing, initialization parameters and trace files. Also, Nic was able to set up a tmpfs linux RAM disk on his laptop and direct the temporary tablespace pointer from withing the DBMS to the RAM disk. What this does is redirect work area overflow from sorts, hashes and merges to the RAM disk, effectively bypassing the need to swap to disk. During break I implemented this same experiment on the HPCapstone server and was able to see similar benefits as we had seen in preliminary queries with _PGA_MAX_SIZE set at 4G. This seemed like a feasible option to help with query performance, however, it is not really the solution the client was looking for.

Some issues that I ran into during this period included: data availability with asynchronous Javascript callbacks, developing a method to move all trace files from one session to a specified directory while giving a unique stamp to the file name, and getting the external tablespace configured so it was accessible from the toolkit GUI.

All of the above issues were resolved by the end of winter break. Additionally, we had the toolkit GUI up and running. Moving forward, we plan on halting development on the toolkit app. Our clients were invited to present at HOTSOS, so the pressure is no officially on and we have a true deadline of February 27th, 2017.

4.2.2 Week One

During the meeting on Wednesday this week our group agreed that we should meet two times a week this term for three hours each session. Our clients asked the three of us to individually focus on an area of interest, my task was to learn more about trace events and figure out a way to extract clear diagnostic information from a query execution.

In order to begin this task, I spent time learning more about the different trace event codes in order to find ones that would be relative to our experimental data collections. The codes that stood out were the following: dump sort statistics (10032), standard trace output (10046), Optimizer Trace Dump (10053), parallel execution slave (10390), parallel execution granules (10391). While I was working through this task I ran into issues getting 10390/391 trace events to generate output files. This is something that I spent considerable time on and had no luck producing any valid output files. I brought this problem up with the group and it was agreed to move forward. The 10390/391 trace event codes had the levels of granularity expressed in hexadecimal while the other trace events that worked appropriately were expressed in decimal. I tried converting the hex values to decimal, but that did not do the trick either.

Moving forward my plans were to finish reading the Oracle white paper on parallel execution in hopes to generate some insights on where to direct our experimental efforts. We were also asked to begin making slides or documenting our research, with an emphasis on noting the sources so that we can cite them later.

4.2.3 Week Two

This week we had our first official meeting with our T.A. Jon Dodge. We discussed with him that our clients have requested that we direct our efforts back towards research in order to gain insights for experimentation with new features of Oracle 12c. Jon requested that we make this official and get an email from the clients that outline what they have requested, this way this request could be clearly communicated to the professors and there would be documented proof of the request. At our Wednesday meeting I spoke with Kirby about emailing the T.A. and professors with detail about what they wanted from us, he was happy to cooperate and sent out an email promptly.

Additional time was spent this week reading through the remaining parts of the Oracle paper on parallelism and partitioning. Nic and I have put together a OneNote document that we shared with James and our clients, the goal being to use it as a medium to clearly document and retain the research that is collected during the remaining process. We wanted to create a repository were we could organize our notes by subject and easily append images or tables of data. Our mentor Andy Weiss requested that we try to use sources from Oracle Ace or Oak Table site, these are sites from individuals with paid or non-paid support from the Oracle company.

During this week I thought I had finally gathered valid trace output from 10390/391 event, however, quickly I figured out that it was not what I was looking for. I used a method that I stumbled across where multiple trace events can be monitored concurrently. Since many of the other trace events were generating output fine I wrapped my target query in a 10046, 10053, and then 10390/391. Though there was resulting output trace files, they were composed of spooled output from the 10046 and 10053 trace events. Another issue that I ran into was setting the initialization parameter MEMORY_MAX to a value larger than the size of available resources. This meant that once the database was restarted, it would not be able to come back online until the parameter file was manually adjusted. Andy was able to walk me through what went wrong and teach me the process of resolving the issue and getting the database back up.

Due on the lack of consistent information online about many of the initialization parameters that govern parallel processing, I manually changed the value of PGA_AGGREGATE_TARGET and monitored the changed in _PGA_MAX_SIZE, _SMM_MAX_SIZE, and _SMM_PX_MAX_SIZE. Based on the values gathered I was able to create a graph of the relationship how the three MAX_SIZE parameters are calculated from the main parameter. Additionally, I was able to use this to derive true formulas for the MAX_SIZE parameter in regards to Oracle 12c. This provided us a mathematical model for our insights into PGA size.

Moving forward our client requested that I write up a baseline file that resets the database parameters to their initial value. This baseline file will be used to ensure the database is in a controlled state before continuing with experimentation. Andy also mentioned the need for a generic file that would use substitution variables to prompt the user for input on what they want the parameter values to be, an idea was also proposed to turn this into a command line tool so it can be streamlined.

4.2.4 Week Three

This week there was nothing major to report. Most of my time was spent reading through an Oracle white paper on In-Memory, this paper goes over some of the new features of Oracle 12c at a technical level. The plan is to try to figure out how we can set up some simple experiments in order to gather data on the effectiveness of some of these techniques.

There were no major problems to report this week. Things with James continue to grow stressful. Since the end of fall term he has missed two mandatory meetings with the client and at least one with Jon Dodge. Since the beginning of fall, the only times that Nic and I heard from James was during scheduled meetings with the client or T.A., or the days before an assignment was due. Additionally, at the end of week two James scheduled a CS class at OSU that runs from 12-1:00pm, this time directly conflicts with our two 3-hour weekly meetings and he now shows up at least an hour late to both sessions.

The initial research for our PGA experiment has been completed and we were working on finalizing research for Big Table Caching and In-Memory. Going forward, I planned on doing some targeted experimentation to quickly test potential benefits of In-Memory storage and In-Memory parallel execution. Experimentation was also performed on Big Table caching, this was mainly to get a feel for how the features react and if there were any immediate benefits which were observable.

4.2.5 Week Four

While finishing up the Oracle paper on In-Memory, I stumbled across another paper that was titled, "When to use In-Memory". After glancing through the contents, I immediately added it to my list. By the end of the week I had read through the sections that were relative to version 12.1c. The main points of interest from these documents were as follows:

- Store blocks in columnar format
- Ability to scan uncompressed blocks
- Architecture assisted indexing
- Leverage bloom filters for hash join
- Vectorized scans and calculations

During Friday's meeting we were asked to put together a series of shell scripts to facilitate running nightly test suites along with a resource monitor package. During the first 40 minutes of the meeting Andy outlined his initial vision of having a driver script that orchestrated multiple connections to the database so that we can run a monitor loop in the background while SQL queries are run in the foreground. After he was finished I began trying to establish a connection to the HPCapstone database with Bash. There were some initial complications getting the connection to work as intended. The example that I found as a template used a flag for SQL*Plus to allow for login bypass. This proved to be problematic because the tables that we were querying were not accessible without logging into the database. After trying to log into SQL*Plus through the same method from the terminal I quickly noticed that we just needed to remove that flag and use the generic log in credentials from the connect string.

By the tail end of the meeting a working version of the test automation suite was completed and running. The collection of scripts were as follows:

- driver.sh: Prepares the environment and calls two run scripts
- runMonitor.sh: Establishes connection to database and executes resource monitoring loop
- runTest.sh: Establishes connection to database and executes test suite

During the weekend I spent a quite some time cleaning up and commenting the test automation scripts, as well as adding logic to make it more usable. Also, I added the remaining units of work that needed done to GitHub issue tracker in hopes to communicate progress with the group.

Moving forward, I planned on getting feedback early in week five on the current state of the test automation scripts. After coming to an agreement, or finishing up the addition of new features, we will begin using the test automation scripts to gather data from the three experiments we currently have set up which test: PGA, Big Table Cache, and In-Memory.

4.2.6 Week Five

Early in the week we noticed that the original design for the test automation scripts did not work out like we had intended. For the scripts to work correctly there needs to be two independent connections to the database, this is very important. Our issue rooted from the fact that one connection was needed to run the monitoring loop in the background, while the other connection was needed to run the test queries.

Our file system is organized into directories of experiments (ie. PGA, Big Table Cache and In-Memory) and withing each directory there are individual tests which are SQL files. At the beginning of each experiment a parameter file is executed, the final statement in the parameter file terminates running processes and restarts the database. Since we had established two independent connections to the same pluggable database, the first connection was terminated. Therefore, we were able to see the tests running but there was no sign of the monitoring loop and there was no performance data written to our collection tables.

After changing the order of execution within the driver script we were able to incorporate a step before the monitor loop was executed that configured and restarted the database, this resolved the issue we were facing and allowed us to proceed with data collection. By the end of the meeting on Friday we had finished improving and testing the test automation scripts and fired off our three experiments. The plan was that the experiments would finish by Sunday, then we would be able to start graphing some of the data to see if we any of our hypothesis were correct regarding the effects of our chosen system configurations.

4.2.7 Week Six

Over the weekend I monitored the progress of our experiments to ensure that everything was running smoothly. Since I am an HP employee, I have VPN access to the Corvallis site so I did not have to be onsite like Nic or James. Initially it appeared that we were able to get data from each of our three main experiments, a brief summary of the purpose of these experiments is outlined below:

- PGA Experiment: Focused on altering the values of _PGA_MAX_SIZE and monitoring the potential spill over to temp during operations such as sort, hash, and window sort
- Big Table Cache Experiment: Focused on altering the percentage of the buffer cache that caches large objects in memory and monitoring potential read performance
- In-Memory Experiment: Focused on enabling In-Memory column store and monitoring the potential performance improvements from compressed reads and execution

Early in the week we realized that the data collected from the In-Memory experiment was not what we expected. This was traced back to the fact that we were not waiting for the tables to be populated into the In-Memory Column Store (IMCS), therefore, the pool of CPU threads available for parallel processing was decreased due to some being used as worker processes to populated the IMCS. Additionally, the four test tables that we had would not fit in the IMCS when it was set to 192G and labeled as NO_MEMCOMPRESS.

In order to resolve this, we split up the In-Memory experiment into two phases:

- Parent Child Analytics and Parent Child Filter
- Lead Lag and Single Table Analytics

This allowed us to fit the entirety of the tables we would query within the IMCS with no compression. It was also important to note that the blocks within the IMCS could not be scanned until the entire object had been populated. However, you can specify the granularity of the object you would like to flag for IMCS population down to a single column in a table.

During the weekend I finished up my slides on the two IMCS memory pools, I also made a couple of graphs that visualize the observed compression ratio and size of each of our four test tables. Moving forward, Nic and I will be helping our mentors prepare anything else they need to make the presentation at HOTSOS go smoothly. This will likely include making more slides on the topics that we experimented on, as well as some additional experimentation.

4.2.8 Week Seven

Early this week I wrapped up the IMCS slides I was asked to put together and sent them to our clients. During our Wednesday meeting we were asked to set up a statement queuing experiment. Statement queuing is a feature that uses a parameter called PARALLEL_SERVERS_TARGET to govern the execution of queries.

For example, if PARALLEL_SERVERS_TARGET is set to 128 and 8 queries with a degree of parallelism (DOP) of 16 all get fired off at the same time. The first four queries would begin execution, while the last four would begin queuing. This is because the first four queries require all of the available CPU threads:

$$(4_{queries} * 16_{DOP}) = 64_{DOP}$$

Next, we must consider the consumer producer model that is central to Oracle parallel processing. This model states that there will be two sets of processes split off for each degree of parallelism, one to produce the rows (ie. read them from disk) and another to consume the rows (ie. perform computation for output). Therefore, we must multiply the DOP by a factor of two.

$$(64_{DOP} * 2_{producer|consumer}) = 128_{processes}$$

In short, this feature only executes queries when they can have the full amount of CPU threads that are requested. If this feature is not enabled the the actual DOP granted to each query is reduced so that that PARALLEL_SERVERS_TARGET is not surpassed. This means that all queries begin execution concurrently but most begin with suboptimal system resources.

The initial attempt to set up a statement queuing experiment by Nic and I ran into major complications. It turned out that this was due to the test we tried to use, Parent Child Analytics, performed a join between a 1G and 100G table. Launching ten of these queries at the database, even with statement queuing enabled, resulted in the queries failing due to limited memory resources. This issue was resolved by increasing the initialization parameter for PGA_AGGREGATE_TARGET to 192G.

Our plans going forward are to work through our documentation and make any changes that are needed to ensure the documentation reflects that scope of our project. Additionally, once our clients return from the HOTSOS convention we will work with them to implement many of these changes on their production environment and aide with troubleshooting complications which arise.

5 Retrospective

Positives	**Deltas**	**Actions**
Presentation went well	Web based toolkit security	Sanitize input
Promising results on all experiments	Complete web based toolkit	Implement more functionality in web based toolkit
Was able to focus more on project	Roll changes to production	Develop road map with client
Revised design document	-	-
Developed understanding of Oracle database concepts	-	-

6 Conclusion

Overall, winter term went by smoothly. During week six our group member James Stallkamp was fired from our team. Even though we lost a team member, there was no added stress or responsibility passed down to either of the remaining members. To this point we have been able to provide our clients with everything that they have asked for. Their presentation took place on February 28th, though we have not yet had another formal meeting with them, we received a phone call letting us know things went very well and they were invited back for next year. Going forward we will be further fleshing out our testing suite and also upgrading our production database to the newest version of Oracle database. Once the production database has been upgraded we will apply our knowledge gained from our research to best optimize the database to handle Decision Support System (DSS) workloads.