Oregon State University
College of Engineering

CS Capstone Project Update Fall 2018

December 3, 2018

Impala performance tuning on HDFS

Prepared for

Hewlett Packard

Andy Weiss _____ _____
Signature *Date*

Prepared by

Group 38

Addax

Caitlyn Cook _____ _____
Signature *Date*

Iliana Javier _____ _____
Signature *Date*

Nicholas Skinner _____ _____
Signature *Date*

Amy Tang _____ _____
Signature *Date*

Abstract

The Big Data team at Hewlett Packard currently uses an Oracle database to store industrial IoT data gathered from large printing presses. They are interested in moving from their current centralized database to a distributed architecture, and have asked our team to investigate the operation of their planned system. This report introduces our project, defines our goals, discusses the work completed during Fall Term, and discusses some challenges encountered during the term.

CONTENTS

1 Project Description

1.1 Background

The HP BackOffice database stores data collected from approximately 300 industrial printers, called web presses, in operation globally. The presses print on paper up to 110 in wide at the largest, and are capable of printing at speeds of up to 1000 ft per minute in full color. The presses are used to print books, packaging, and mail at an industrial scale for many major companies. For example, the presses are a part of how Amazon is able to print and ship a brand new copy of a book within 4 hours of ordering it on their website.

Each press records data that is used to diagnose issues, study press operation, and guide the development of new features. Data is collected on alarms, pen health, print jobs, and other aspects of operations. This data is streamed to HP's database, where it is processed and analyzed by teams of engineers. New data is added to the database at a rate of approximately 60 - 100 GB per day; currently, the database holds approximately 30TB of data.

HP's current database is a centralized Oracle 12c database which shares all memory and disk resources between multiple processors. While Oracle databases handle transactions extremely well, HP doesn't have much need for transactions. Their data is static once written; once the data for a day is collected and stored, it is not further modified. Instead, HP mostly performs analytic processing on their data, which suffers unnecessarily from the overhead that makes Oracle so good at transactions. The monolithic architecture of their current database also presents difficulties with scaling. As HP anticipates future data loads, they want a solution that will allow cheaper and more efficient scaling. This is their motivation behind our project; we are to investigate the behavior of a distributed system which will fare better in these key areas. Our research will result in information that will help ease their transition process.

1.2 Purpose

HP's data team has tasked us with gathering information about a distributed database system implementation. The data team is particularly interested in Apache / Cloudera Impala, a SQL engine that runs on the popular Hadoop Distributed File System (HDFS), and would like us to focus on researching this engine's mechanisms.

Our sponsor has a variety of reasons for being interested in distributed database systems. At the core of every business and engineering decision is cost, in terms of both time and money. In terms of time, HP's current exponentially growing database is causing queries to result in long running times. This is due simply to the large amount of data tables that need to be scanned, reduced, or joined by the single server. In addition, there is significant slowdown caused by the overhead Oracle implements for each query. This work could be eased by implementing a system that distributes queries across multiple servers. HP has also spent years understanding and optimizing their SQL queries to best fit their data's behavior, so they would like any new system they adopt to run SQL as well. Apache Impala fits these criteria. Monetarily, HP is interested in moving to an open source solution to cut the cost of Oracle's software licensing fees, which grow more expensive the more data the database houses. Impala, as well as its underlying Hadoop system, is open source. These criteria make Impala a promising solution for HP's data problems.

1.3 Goals

The goal of our project is to produce a document that will guide a person familiar with shared-everything database architectures through the setup and use of a shared-nothing architecture. We will focus on multiple aspects of the system and provide an explanation of their function which will allow the reader to examine and improve the performance of a

distributed system. In particular, this document will examine the tools available for diagnosing and tuning performance problems, the metadata store of the database, the generation of query plans and how parallelization across nodes affects these plans. We will accomplish this through a combination of experimentation and reading existing documentation.

1.4 Challenges

This project was very unique in comparison to traditional capstone projects. Our group found ourselves in the middle of a transitionary period, where our instructors decided to allow more flexibility in assignments for groups with research projects. The documentation needed to define a research project is very different than the current class assignments, which cater exclusively to product-producing projects. In the past when HP has conducted research-based projects, they have created documents without much practical use. They were not applicable to the project, but were required for the Capstone class, which resulted in much difficulty for the team attempting to write them. However, we were not informed about this detour from the assigned documents until week 5 of the class, once our group's planning for the assigned documents had already begun.

We were tasked with the challenging prospect of creating our own assignments for the entire term. This was made more challenging because our sponsor, instructor, and TA were not sure what kind of documentation we should produce. As the term went on, it became apparent that our sponsor and our instructors had different ideas for what our team should accomplish from this project. Our instructors preferred multiple physical deliverables and set timelines, but our sponsors preferred us to do as little documentation as possible and focusing on reading papers and gaining "hands-on" knowledge. We attempted to combine these two interests to the best of our ability; however, the main result of this attempt was delayed papers due to frequently changing requirements.

We solved this problem by having the instructors and sponsors speak directly to each other, instead of our team relaying information indirectly between the two parties, so they could resolve their differences in expectations. Much time was lost defining these specifics, with the length and quality of our work feeling the brunt of this loss. One of the most disappointing losses that was caused due to this lack of direct communications was the one assignment we did receive guidance on and completed a significant portion of the work for: our Research Proposal. This assignment ended up not being used this term. Our instructor would like us to turn this in at an unspecified future date. In the future we would like to have our stakeholders' expectations more clearly defined and agreed upon much earlier in the term. That way, our time in upcoming terms can be spent productively on deliverables and not spent almost entirely on defining the possible assignments and facilitating communication.

2 Fall Term Work

Once assignments were defined, we begun work by reviewing a selection of relevant papers suggested by our client. The first four of the papers discussed below were the subject of a more detailed analysis and write-up, while the others were used as supplementary sources for an additional assignment. All papers discussed can be found in the bibliography at the end of the report. Below the review of this reading is a week-by-week summary of our progress throughout the term.

2.1 Review of Papers

2.1.1 E. F. Codd - A Relational Model of Data for Large Shared Data Banks

To begin our reading, our client suggested ”A Relational Model of Data for Large Shared Data Banks” by E. F. Codd. Published in June of 1970 in the Communications of the ACM, the paper forms the foundation for relational databases that are in use nearly everywhere today. Codd was concerned with insulating the user from changes in the structure of data storage, which he felt that the user should not need to be aware of. To accomplish this, Codd outlines an application of set and relation theory to databases. The concept of normalization is introduced, including normal form and associated concepts such as primary and foreign keys. Once he has established the concept of normal form, Codd discusses operations and interactions with the table that form the basis for database languages, including SQL.

2.1.2 The Oracle Optimizer: Explain the Explain Plan

To recommend changes, our team needs knowledge of the current system. Our team will be researching and comparing two different systems, but to begin, we require a point of reference. To understand the Oracle 12c system currently in use, HP has recommended reading ”The Oracle Optimizer: Explain the Explain Plan” guide. The Oracle Optimizer is a tool that attempts to resolve the most efficient execution plan for a given query using histogram information in addition to data projection based on indices and dependencies. The optimizer aims to reduce the time and CPU usage to resolve the query. To aid the user in understanding its actions, the Oracle Optimizer can generate a log of the execution plan of a query to display to the user. Logs generated by the optimizer also include information on the resources used to accomplish the task. An execution plan is referred to as an Explain Plan. An Explain Plan from the Query Coordinator takes relevant information about the table structure and dependencies from an executable query and compiles the metadata needed to generate an efficient plan to execute the operation. The Query Coordinator takes in (1) how much CPU is available, (2) the projected time to dispatch and complete individual operations, (3) the composition of the query and (4) external dependencies on the involved tables. With these inputs, the Query Coordinator forms an educated estimation of what methods and what order it should execute the query.

The ”Explain the Explain Plan” paper is a thorough, yet concise resource on Oracle’s Query Coordinator. The paper goes through detailed information on query planning and coordinating, measuring the weights of different actions and comparing them against the current value of available resources. Cardinality, access methods, join methods, join types, join orders, partition pruning and parallel execution are all vital components to query optimization that will change the cost of a query plan.

2.1.3 Dremel: Interactive Analysis of Web-Scale Datasets

This paper provides fundamental concepts of Dremel, from its architecture to implementation, and explains how Dremel makes more efficient performance than MapReduce computing does. Data in Dremel is organized in a ”columnar” format, which contributes to very fast query speed.

Dremel is a distributed system for interactive analysis of large datasets. It is also a custom, scalable data management solution built from simpler components. Dremel complements the MapReduce paradigm, and provides fault tolerant execution just like MapReduce does. It also includes a flexible data model and in situ data processing capabilities. In situ refers to the ability to access data ’in place’.

Importantly, Dremel uses a columnar nested storage. The architecture of Dremel has the same concept of a serving tree used in distributed search engines. The query gets pushed down the tree and is rewritten at each step. The result

of the query is assembled by aggregating the replies received from lower levels of the tree. Dremel supports a SQL-like syntax, but does not support update or create functions. In contrast to layers such as Pig and Hive, it executes queries natively without translating them into MapReduce jobs. Besides, Dremel uses a column-striped storage representation. The data is represented as a set of columns. The advantage this brings is that it allows users to read less data by retrieving only columns they need, which reduces CPU cost.

Based on the major idea of the project is focusing on switching to distributed system, this paper helps us to have a comprehensive understanding of distributed system that uses columnar nested storage, and clarifies differences between row storage and columnar storage.

2.1.4 Impala: A Modern, Open-Source SQL Engine for Hadoop

This paper served as an introduction to the rest of our research going forward. Impala is an open source SQL engine designed to run with high concurrency and low latency on distributed frameworks such as Hadoop. This paper detailed Impala's SQL engine, while also detailing how it works as a distributed framework- specifically how it distributes data and queries over the Hadoop Distributed File System. This paper gave some explanation from a user's point of view, such as how to create a table and where in the file system that data would be stored. However, most of the paper detailed Impala's architecture: its frontend code, backend code, and data storage. The Impala front end is where the SQL is translated into execution plans, and the backend is where these plans are actually carried out. Query plans in Impala's front end have two stages: single node planning, and plan parallelization and fragmentation– to prepare the query for distribution across all of the system's nodes. When the query plan is optimized, "cost estimation is based on table/partition cardinalities plus distinct value counts for each column". The second part of the frontend planning has two goals for plan distribution: to minimize the amount of data movement and to maximize "scan locality". These two terms which are related, and are placed at such high importance due to the Hadoop File System remote reads (reads not from the current node) being much slower than local (all data found within the node) ones. Impala's backend, the query executor, runs on each node in the system independently. This backend also manages I/O and storage formats. The paper also pointed out the roadmap for where the Impala developers wanted it to be in the future, which also conveniently highlighted certain standard SQL and database features that Impala is currently missing.

This paper served as a starting point for where our research for the rest of the term will go. The questions posed in our project goals were touched on briefly in this paper, but none were discussed in the amount of depth required for our sponsors needs. However, this paper did give us indications for where we can go to find the information needed to answer our questions, such as how we can investigate Impalas frontend more to learn about its query planning, and how we can read more about the backend to learn more about how it handles distributions.

2.1.5 Parallel Execution with Oracle Database 18c Fundamentals

Oracle is an example of a database system that implements shared-everything architecture. The data contained within the Oracle system is accessible to all processing units without limitations. The parallelism implemented within the Oracle system is not limited to the data access that an individual node would possess. Rather, all dispatchable agents associated with the database are capable of accessing all the data contents.

Oracle's shared-everything architecture does not require data partitioning to enable parallelism by default; data is accessible from all processing units without limitations. Parallelism within the Oracle system is implemented through dividing a query into smaller components called granules. Granules represent a fraction of a query, and can be assigned

a specific block range in memory. These block-based granules are assigned a position within the memory or storage of the Oracle system, and will do all actions within their block range. Upon resolving a task, a granule will need to report its result set to another agent. The action it will take to complete this task is known as 'data redistribution', and it is a key component to many non-trivial parallel operations, including parallel aggregations, joins and sorts. The individual granule does not know the broader context of the operation the retrieved data will be used within, so it must pass the result set to a subsequent operation on the result set contents. The query coordinator will dispatch an agent to grab the result set from the previous granule, and assign the new agent a granule of work to process. This series of events is recursively performed until the end query result is reached.

The manual is an additional resource that will be used going into the team's research. The manual presents a fundamental explanation of Oracle's implementation of the shared everything architecture. The manual covers basics of how memory and processing are shared between agents through to the extensions of the Oracle program to accommodate shared-nothing methodologies. It also contains information regarding Oracle's attempt to address the benefits within the shared nothing architecture. Partition based granules are an example of approaching the distributed nodes with dedicated agents within oracle.

2.1.6 The Case for Shared Nothing

"The Case for Shared Nothing" by Michael Stonebraker is a short paper that vaguely describes the merits of shared-nothing architectures. He provides a simple table which ranks shared-nothing, shared-memory, and shared-disk architectures based on performance in terms of "difficulty of concurrency control", "number of messages", and other vague criteria. Several of these criteria are not well defined, and none define clearly how Stonebraker came to his conclusions regarding the rankings the assigned. Although this paper was quite vague, it introduced us to criteria we can use to compare shared-nothing and shared-everything architectures, which our sponsor is very interested in learning the benefits and faults of.

2.2 Weekly Progress

Week	Details	Problem & Solution	Assignment Due
Week 3	We were assigned to groups. We set up the Slack channel and Github repo. We had our first meeting with the client to get details for problem statements, and scheduled future meetings with our client and TA.		Problem statement - individual
Week 4	We had another meeting with our client to discuss the project in more detail.		Problem statement - group final
Week 5	We decided on choosing the Research Proposal over the Requirements Document, and assigned sections to complete.	Problem: We had difficulty brainstorming on how to write a paper that is helpful to our project, and needed more direction. Solution: We met with Dr. Winter, who modeled our writing assignments. We also set up weekly meetings with her.	
Week 6			
Week 7	Dr. Winters gave feedback on our draft Research Proposal, and we planned the small-scale literature review.		Research Proposal
Week 8	We met and discussed the contents of the chosen 4 papers with our clients, and began work on the review paper.		
Week 9	Dr. Winters and our clients spoke directly, and clarified the assignments for the term. We received permission to stop meeting with our TA and meet with Dr. Winters instead.		Review paper
Week 10	We dividied sections for the systems comparison paper.		
Week 11	Finals week		Group system comparison paper, Progress update video and report

REFERENCES

[1] *The Oracle Optimizer - Explain the Explain Plan*, Oracle, April 2017.

[2] *Parallel Execution with Oracle Database 18c Fundamentals*, Oracle, February 2018.

[3] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, "Dremel: Interactive analysis of web-scale datasets," in *Proc. of the 36th Int'l Conf on Very Large Data Bases*, 2010, pp. 330–339. [Online]. Available: http://www.vldb2010.org/accept.htm

[4] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovytsk, C. Ching, A. Choi, J. Erickson, M. Grund, D. Hecht, M. Jacobs, I. Joshi, L. Kuff, D. Kumar, A. Leblang, N. Li, I. Pandis, H. Robinson, D. R. S. Rus, J. Russell, D. Tsirogiannis, S. Wanderman-Milne, and M. Yoder, "Impala: A modern, open-source sql engine for hadoop."

[5] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[6] M. Stonebraker, "The case for shared nothing," *IEEE Database Eng. Bull.*, vol. 9, no. 1, pp. 4–9, 1986.

[7] A. Weiss and N. Whitlock, private communication, November 2018.