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 2

1.1 Background . . . 2

1.2 Purpose . . . 2

1.3 Goals . . . 2

1.4 Challenges . . . 3

2 Fall Term Work 3

2.1 Review of Papers . . . 3

2.1.1 E. F. Codd - A Relational Model of Data for Large Shared Data Banks . . . 3

2.1.2 The Oracle Optimizer: Explain the Explain Plan . . . 4

2.1.3 Dremel: Interactive Analysis of Web-Scale Datasets . . . 4

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 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 HPs 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.

HPs 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 doesnt 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

HPs 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 engines 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, HPs 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 datas 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 Oracles 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 HPs 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. However, we were not informed about this detour from the assigned documents until week 5 of the class, once our groups 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 paper, 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