Oregon State University
College of Engineering

CS Capstone System Comparison

February 8, 2019

# Impala performance tuning on HDFS

Prepared for

Hewlett Packard

Andy Weiss _____ _____
*Signature* *Date*

Prepared by

Group 38

Team Name


Caitlyn Cook _____ _____
*Signature* *Date*

Iliana Javier _____ _____
*Signature* *Date*

Nicholas Skinner _____ _____
*Signature* *Date*

Amy Tang _____ _____
*Signature* *Date*

**Abstract**

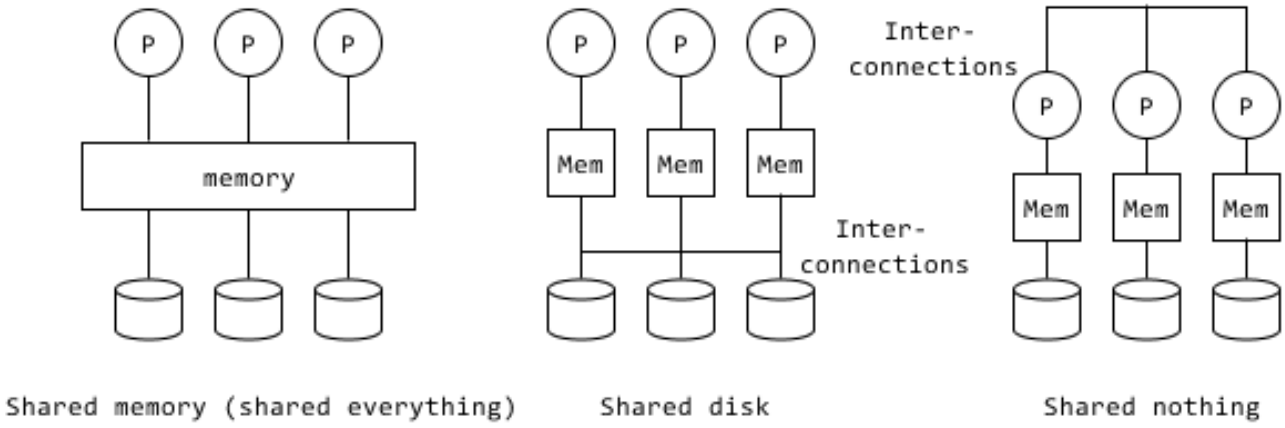
Hewlett Packard, Inc (henceforth HP Inc.) currently uses an Oracle database to store industrial IoT data gathered from large printing presses. The Big Data team is interested in moving this centralized database to a distributed system, with special interest in moving from the current shared-everything architecture to a shared-nothing architecture. From previous research done by the Big Data team, Clouderas Impala has been selected as the primary candidate for implementing the new system. To begin investigating the possible new systems, our team first needed to understand HP Incs current system. In the first half of this report, we present our research on shared-everything architectures and their specific implementation in Oracle databases. In the second half, we introduce shared-nothing architecture as a whole and its specific implementation in Cloudera Impala databases.

## CONTENTS

# 1 Shared Everything

## 1.1 Concept

In a shared everything environment, all servers and workload can access to the same shared store. Meanwhile, they can read and write any portion of data they want easily. In contrast to shared-nothing environment, the main feature is that clusters are able to exchange data with other during computing. This is one of the solutions for parallel database system and when process huge data set. To make clusters interactive, there are multiple ways of how clusters access data from other.




Above fig.1 is a basic diagram shows the different architecture between shared memory system and shared disk system. In share memory system database, each node or process is working with the same memory. If a node needs the calculation done by other node, it can access the updated data easily from the shared memory. In shared storage system, all connected system share all disk. Each cluster still has it own local memory and can access to disks through interconnection.

Shared everything focuses on maximizing resource utilization [1]. Since data may be cached in multiple places and accessed by any node in the system at any time, data consistency and contention may become an issue. Although shared everything enables high scalability and performance, they are limited when lots of data are traveling through the interconnection at the same time. However, shared disk provides fault tolerant ability, which mean if one node is fail then other node can complete the task as well.

The choice of data structure is and important decision that influence on the performance, stability and accessibility. Tabular data are inherently rectangular. More strictly, every row has the same set of column nodes and every record share same variables. However, nested data are inherent in tree layout. This structure can be expected to faster than tabular since it enables processes to read less data. Dremel and Solar are two successful distributed system that implement shared everything architecture with nested data structure. According to ”Dremel: Interactive Analysis of Web-Scale Datasets,” it claims that nested data structure help their shared disk system access data faster and reduce CPU cost due to cheaper compression [2]. On the other hand, Solar used the share everything architecture base on tree data set. They increase the performance and scalability by adding storage nodes that are used for data storage and read access [3].

Oracle is a shared everything database. Specially, Oracle uses data dictionary to store their metadata. Data dictionary is formed with tables and views, and is being access frequently by dictionary cache, where Oracle keeps data. The view in the data dictionary allows users to access and read data only. The oracle server accesses these two locations in order to shares data for users access [4].

### 1.2 Implementation

Oracle is an example of a database system that implements the shared-everything model. The data contained within the Oracle system is accessible to all the processing units without limitations [5], meaning that 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.

Oracles shared-everything architecture does not require data partitioning to enable parallelism by default; data is accessible from all processing units without limitations [5]. 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 the storage of the Oracle system, and will do all actions within their block range. Partition-based granules leverage individual partitions of the database to further optimize query resolution speed. Individual granules are capable of processing alongside other granules, enabling parallel execution along the system.

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 [5]. 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.

```
Plan hash value: 1928163552

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                 |   144 | 10656 |     7   (0)| 00:00:01 |
|*  1 |  HASH JOIN                   |                 |   144 | 10656 |     7   (0)| 00:00:01 |
|   2 |   MERGE JOIN CARTESIAN       |                 |    23 |  1357 |     6   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS STORAGE FULL | REGIONS         |     1 |    14 |     3   (0)| 00:00:01 |
|   4 |    BUFFER SORT               |                 |    23 |  1035 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS STORAGE FULL| LOCATIONS       |    23 |  1035 |     3   (0)| 00:00:01 |
|   6 |   INDEX FULL SCAN            | COUNTRY_C_ID_PK |    25 |   375 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

For further query optimization, Oracle is capable of operating on the same processing paradigm as a shared-nothing system. If the Oracle partitioning feature is enabled, Oracle will assign each table a smaller result set, minimizing

the amount of data that needs to be scanned. Oracle claims that the partitioning system carries the exact same parallel processing capabilities as a shared-nothing system, however, Oracle is capable of implementing this partitioning without the restrictions of the fixed parallel access encompassed in the data layout [5].

## 2 Shared Nothing

### 2.1 Concept

A shared nothing architecture is a model in which the processors do not share any resources. Each node within the system has its own memory and disk storage, rather than sharing one or both with other processors. Each node would have a portion of the databases stored data, and is the sole control on that portion of the data. The nodes connect through a network to coordinate actions by requesting information from each other.

Shared nothing architectures are primarily used for their easy and relatively cheap scalability. They can be built from commodity hardware, rather than requiring specialized and expensive parts to achieve high performance. One source estimated the cost of a single commodity node and a single specialized node at $700 and $10,000 [6]. This source calculated that in order for specialized hardware to be as cost efficient, it must be at least 14 times as fast, which there is no indication of being true.

Shared nothing architectures are also not subject to the points of contention and bottlenecks that sharing architectures have, such as the need for multiple nodes to read and write to the same data. Rather than moving the data itself between processors, each node asks another node a question about the data it holds. That node returns an answer which the asking node uses to continue with its work, until the complete question - the users query - is answered [7]. This reduction in data transfer means that a higher proportion of time is spent processing the questions, which is a much easier task to speed up.

### 2.2 Implementation

Impala is an SQL engine that implements a shared-nothing, parallel processing architecture over Hadoop. Queries are fragmented and distributed across nodes in the system to correspond with the portion of data stored locally on individual nodes. The system was inspired by Googles Dremel in that it uses columnar storage, executes SQL commands natively instead of building them on top of MapReduce jobs, and is specifically designed for distributed computing [8] [2]. Impala takes it a step further and implements as pure of a shared-nothing architecture as it can.

An Impala system has three main components which work together to implement this shared-nothingness: the Impala daemon, the Statestore, and the Catalog daemon [9]. The impalad is the Impala daemon which runs on every node in the system. An impalad's responsibilities include, but are not limited to, accepting initial queries, reading and writing data files, executing query fragments assigned to it, and transmitting intermediate query results back to the coordinator [10] [9]. The coordinator is simply the first impalad to receive an initial (raw SQL) query, at which point it assumes the coordinator role by managing that querys execution across all other nodes and the assembly of the final query result [10]. This avoids the need for a single node dedicated to coordinating all queries, since any impalad in the system could be assigned the initial query. Since an impalad instance runs on every machine in the system, this "allows Impala to take advantage of data locality", where each node can have its own partition of the data locally in own memory without having to use network communication to read other nodes data [10].

The StateStore further cements Impala as a shared-nothing system. Each node working independently raises the issue of how coordinators can stay up to date on the other nodes in the system, and how they can be alerted if a node

has failed. An Impala cluster has a single daemon called a StateStore to push updates about the system to the nodes who need the information [10]. The StateStore maintains a table of topics, which is any information needed for query scheduling and execution [9]. Processes running on nodes contact the StateStore only once, at startup, to give it a list of topics it needs to stay updated on [10]. An important topic is the list of nodes in the system which are healthy and thus can accept work. For this topic, the StateStore monitors node health through the use of keepalive messages [10]. If a node goes offline (doesnt respond to to the keepalive message), the StateStore broadcasts this change to the other impalads, so they do not assign work to an unreachable node [9].

The Statestore also broadcasts changes in metadata with help from the Catalog daemon. Physically represented as the daemon process ”catalogd”, the catalog daemon ”pulls information from third-party metadata stores (for example, the Hive Metastore or the HDFS Namenode), and aggregates that information into an Impala-compatible catalog structure” [10]. This is to make integration with third-party storage systems easier, while also allowing users to import ”Impala-specific information”, such as user-defined functions [10].

## REFERENCES


[1] C. Claussen, “Shared everything vs. shared nothing: Get the best of both with sap sybase iq 16,” February 2013.

[2] 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

[3] T. Zhu, Z. Zhao, F. Li, W. Qian, A. Zhou, D. Xie, R. Stutsman, H. Li, and H. Hu, “Solar: Towards a shared-everything database on distributed log-structured storage,” in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, 2018.

[4] *Oracle 9i Database Concepts*, Oracle, March 2002.

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

[6] D. J. DeWitt, S. Madden, and M. Stonebraker, “How to build a high-performance data warehouse,” *Available: http://db.lcs.mit.edu/madden/high_perf.pdf [Accessed: June 2011]. BIBLIOGRAPHY BIBLIOGRAPHY*, 2005.

[7] D. DeWitt and J. Gray, “Parallel database systems: the future of high performance database systems,” *Communications of the ACM*, vol. 35, no. 6, pp. 85–98, 1992.

[8] A. Floratou, U. F. Minhas, and F. Özcan, “Sql-on-hadoop: full circle back to shared-nothing database architectures,” *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1295–1306, 2014.

[9] *Apache Impala - Interactive SQL*, Cloudera.

[10] 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.”

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

[12] 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.

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

[14] G. Nico, “Parallel programming - architecture (shared nothing, shared disk, shared memory),” June 2018.

[15] M. G. Norman, T. Zurek, and P. Thanisch, “Much ado about shared-nothing,” *ACM SIGMOD Record*, vol. 25, no. 3, pp. 16–21, 1996.

[16] *HDFS Architecture Guide*, Apache Hadoop, September 2018.

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