Characterizing Cloudera Impala Workloads with BigDataBench on InfiniBand Clusters [*]

Kunal Kulkarni Xiaoyi Lu Dhabaleswar K. (DK) Panda
Department of Computer Science and Engineering, The Ohio State University
{kulkarni.120, lu.932, panda.2}@osu.edu

Abstract

Cloudera Impala is an open source massively parallel processing (MPP) SQL query engine for data stored in Hadoop Distributed File System (HDFS) and HBase. The High Performance Computing (HPC) domain has exploited high performance networks such as InfiniBand for many years. InfiniBand network provides high bandwidth and low latency. BigDataBench is a well known benchmarking suite for Big Data applications that provides real workload scenarios. In this paper we first characterize BigDataBench query workloads in Impala running in IPoIB mode on an InfiniBand cluster and determine the time spent in I/O, communication and computation. With full remote mode operations, we see that BigDataBench Queries run faster on InfiniBand QDR (IPoIB, 32Gbps) compared to 10Gb Ethernet - Scan by 21%, Aggregate by 27%, and Inner Join by 6%. Interestingly, although Join is most Communication intensive, we did not see much difference in the overall runtime of the query between InfiniBand QDR (IPoIB) and 10Gb Ethernet. This is because the computation dominates the Join query performance. Even though the network communication latency and throughput on InfiniBand QDR was better than 10Gb Ethernet by 19%, there was no significant difference in the overall runtime of the query. In the scalability study by increasing the number of compute nodes in the cluster, we observed that Scan and Aggregate queries scale linearly but the improvement in Join execution time is negligible due to increased Computation. From these experiments we see that although InfiniBand improves the Communication part of Join query, the Join computation time in Impala needs to be optimized further so that we get more benefit in the overall query execution time with high performance networks/protocols.

Keywords Cloudera Impala, InfiniBand, BigDataBench, Workload Characterization

1. Introduction

Big Data is fundamentally changing the way decisions are being made in a wide range of domains including biomedical research, Internet search, finance and business informatics, scientific computing, and others. Big data can be analyzed with software tools commonly used as part of advanced analytics disciplines such as predictive analytics, data mining, text analytics and statistical analysis. During the last decade, the Apache Hadoop [4] platform has become one of the most prominent open-source frameworks to handle Big Data analytics. The most recent IDC report [6], “Trends in Enterprise Hadoop Deployments”, found that 32% of the companies had already deployed Hadoop. Not only for Enterprise Computing, Hadoop has also been steadily gaining momentum in the HPC community.

Cloudera Impala brings scalable parallel database technology to Hadoop [19], enabling users to issue low-latency SQL queries to data stored in HDFS and Apache HBase without requiring data movement or transformation [3]. Impala is integrated with Hadoop to use the same file and data formats, metadata, security and resource management frameworks. The fast response for queries enables interactive exploration and fine-tuning of analytic queries, rather than long batch jobs traditionally associated with SQL-on-Hadoop technologies such as Hive [27]. Impala is promoted [1] for analysts and data scientists to perform analytics on data stored in Hadoop via SQL or business intelligence tools. The result is that large-scale data processing (via MapReduce) and interactive queries can be done on the same system using the same data and metadata removing the need to migrate datasets into specialized systems and/or proprietary formats simply to perform analysis [11]. Big Data Benchmark tests by Berkeley AMPLab shows that Impala performs faster than Hive by 15X and Tez by 10X [5].

Impala is best suited for running Ad Hoc queries over a subset of data to get fast results. But if we are analyzing huge amount of data in a batch processing manner, then Hive is best suited for this than Impala. For ETL type of jobs where failure of one job would be expensive then Hive is the best option. Figure 1 shows how Hive and Impala fit in the Hadoop ecosystem meant for serving different use cases.

In this paper we evaluate the performance of Cloudera Impala on two clusters with two High-Performance Networks InfiniBand QDR (32 Gbps) / FDR (56 Gbps) and 10Gigabit Ethernet. On InfiniBand clusters, Cloudera Impala runs with IPoIB mode. We run BigDataBench Suite which provides benchmarking software specific to Impala containing the query workloads [29]. BigDataBench provides data generation feature where we can specify the amount of data in GB to be generated in HDFS.

An application/workload is said to be I/O intensive if it spends most of the time doing I/O operations. The application is said to be Compute intensive if it spends more time doing Computations using the CPU. The application can be said to be Communication intensive if the network time dominates both I/O and Compute time. The Big Data applications can benefit from high performance networks if the application is Communication intensive [25]. In this paper we address the following questions:

[*] This research is supported in part by National Science Foundation grants #CNS-1419123, #IIS-1447804, and #ACI-1450440.

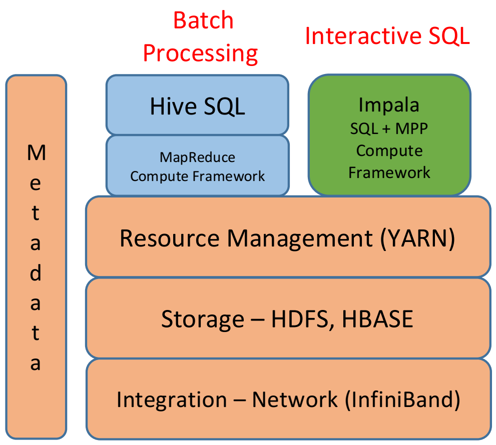

Batch Processing
Interactive SQL
Metadata
Hive SQL
MapReduce Compute Framework
Impala SQL + MPP Compute Framework
Resource Management (YARN)
Storage – HDFS, HBASE
Integration – Network (InfiniBand)

Figure 1. Impala and Hive in Hadoop ecosystem

1. How to profile and characterize the BigDataBench workloads as I/O, Communication, Compute intensive? Through this characterization, can we identify the query workloads that are Communication intensive?
2. In full remote mode setup of Impala which is more Communication intensive, how much benefit does InfiniBand QDR/FDR provide compared to 10Gb Ethernet for each of the queries?
3. Increasing the number of compute nodes and fixing the datasize, how much does the execution time of each of the queries become better due to increased parallelism?
4. Increasing the datasize and fixing the number of compute nodes, how does the execution time of each of the queries change?

We perform detailed profiling and analysis of BigDataBench workloads on Impala and characterize them as I/O, Communication or Compute intensive. We then analyze the query performance on two network interconnects InfiniBand QDR/FDR and 10Gb Ethernet. We also perform scalability study of Impala with different workloads. We observe that Scan and Aggregate workloads scale linearly but Inner Join workload scales poorly. To the best of our knowledge, this is the first paper for characterizing Cloudera Impala workloads with BigDataBench on InfiniBand Clusters.

The rest of the paper is organized as follows. In Section II, we present background about Impala and how the distributed query execution takes place. In Section III, we describe the schemes used for evaluation and in Section IV, we present the evaluation results. Related works are discussed in Section V. In Section VI, we present conclusions and future work.

2. Background

In this section, we provide an overview of Cloudera Impala, InfiniBand and BigDataBench. We show how Impala is integrated into the Hadoop environment and utilizes a number of standard Hadoop components such as Metastore, HDFS, HBase, and YARN in order to deliver an RDBMS-like experience.

2.1 Cloudera Impala

Impala is an open-source, fully-integrated, massively parallel processing (MPP) SQL query engine designed specifically to leverage the flexibility and scalability of Hadoop. Impala's goal is to combine the familiar SQL support and multi-user performance of a traditional analytic database with the scalability and flexibility of Apache Hadoop. Impala works directly on data stored in HDFS and Apache HBase. Impala keeps its table definitions in a traditional MySQL or PostgreSQL database known as the metastore, the same database where Hive keeps this type of data. Thus, Impala can access tables defined or loaded by Hive. Impala supports several familiar file formats used in Apache Hadoop. Impala can load and query data files produced by other Hadoop components such as Pig or MapReduce, and data files produced by Impala can be used by other components also.

To reduce latency such as that incurred from utilizing MapReduce [24], Impala implements a distributed architecture based on daemon processes that are responsible for all aspects of query execution and that run on the same machines as the rest of the Hadoop infrastructure [28]. Impala's high-level architecture is shown in Figure 2.

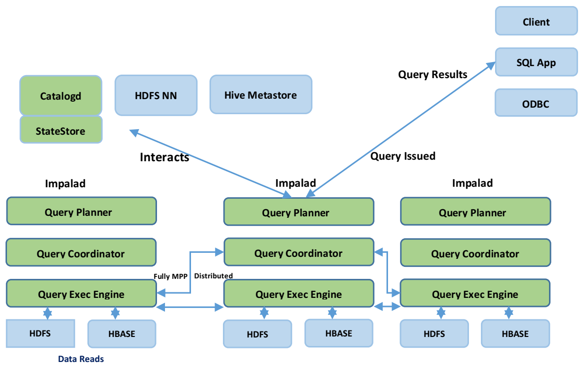


Figure 2. Impala Architecture

Impala deployment is comprised of three services - impalad, statestored and catalogd [15]. The Impala daemon (impalad) service is dually responsible for accepting queries from client processes and orchestrating their execution across the cluster, and for executing individual query fragments on behalf of other Impala daemons. When an Impala daemon operates in the first role by managing query execution, it is said to be the coordinator for that query. In local mode,one Impala daemon is deployed on every machine in the cluster that is also running a datanode process - the block server for the underlying HDFS deployment [13]. In remote mode, HDFS and Impala daemons run in different nodes.

The Impala component known as the statestore checks on the health of Impala daemons on all the nodes in a cluster, and continuously relays its findings to each of those daemons. It is physically represented by a daemon process named statestored and only needs one such process on one node in the cluster. If an Impala node goes offline due to hardware failure, network error, software issue, or other reason, the statestore informs all the other nodes so that future queries can avoid making requests to the unreachable node [19].

The Impala component known as the catalog service relays the metadata changes from Impala SQL statements to all the nodes in a cluster. It is physically represented by a daemon process named catalogd and only need one such process on one node in the cluster. Because the requests are passed through the statestore daemon, typically both the statestored and catalogd services run on the same node in the cluster.

The following is a brief account of the query execution procedure in Impala:

1. The user selects a certain impalad in the cluster, and registers a query by using impala shell and ODBC.
2. The impalad that received a query from the user carries out the following pre-tasks:

(a) It brings Table Schema from the Hive metastore and judges the appropriateness of the query statement.

(b) It collects data blocks and location information required to execute the query from the HDFS namenode.

(c) Based on the latest update of Impala metadata, it sends the information required to perform the query to all impalads in the cluster.

3. All the impalads that received the query and metadata read the data block they should process from the local directory and execute the query.

4. If all the impalads complete the task, the impalad that received the query from the user collects the result and delivers it to the user.

2.2 InfiniBand

InfiniBand is a high-performance networking interconnect that is widely used for high performance computing. The latest TOP500 [9] rankings released in Nov. 2015 indicate that more than 47% of the top 500 supercomputers are using InfiniBand as their primary interconnect. Remote Direct Memory Access (RDMA), one of the main features of InfiniBand [20], allows a node to directly access the memory of another remote node without any involvement from the remote node. It features very high throughput and very low latency [12]. Internet Protocol (IP) packets can be sent via an InfiniBand interface by encapsulating the IP packets in an InfiniBand packet via a network interface. This is known as IP over IB (IPoIB). As long as the InfiniBand network has the necessary driver installed, it creates an interface for each port and can then transport IP packets across the InfiniBand.

2.3 BigDataBench

BigDataBench is an open-source benchmark suite for scale-out real-world workloads [29]. BigDataBench provides a benchmark software suite for various Big Data middleware such as Impala, Spark, Hive, HBase, MySQL, etc. In representative big data workloads, BigDataBench focuses on units of computation that are frequently appearing in the domains of OLTP, NoSQL, OLAP, interactive and offline analytics, graph computing, and streaming computing [31]. It considers different kinds of data models with varied types and semantics, which are extracted from real-world data sets. It provides data sets in form of unstructured data, semi-structured data, and structured data [23]. BigDataBench also provides an end-to-end benchmarking framework to allow flexible benchmarking by abstracting data operations and workload patterns. It can be extended to other application domains also. BigDataBench Version 3.1 is used in the experiments in this paper. BigDataBench has a built-in data generation tool [16] where we can specify the amount of data to be generated. For Impala applications, the generated data is stored in HDFS.

3. Schemes for Evaluation

In this section, we discuss the different schemes used for evaluating performance of Impala on InfiniBand networks. Experimental results for these dimensions are presented in Section 4. In all of the schemes, the three BigDataBench Interactive queries (Scan, Aggregate and Inner Join) are executed. Table and Column statistics of the Inner Join tables are available so that Impala tries to optimize the Inner Join Query.

It is important to design good evaluation schemes for BigData software to measure its effectivess [10]. The evaluation strategy involves multiple dimensions in how well Impala is evaluated. The different dimensions evaluated in this paper are: 1) Query workload characterization as I/O, Communication and Compute intensive, 2) Running Impala in local and remote mode setups, ad 3) Studying scalability of Impala on different queries which involves two schemes, one is how well Impala scales out on adding more compute nodes and second is how Impala performs on increasing the datasize linearly keeping the number of compute nodes constant. All these evaluations are done on InfiniBand cluster. Figure 3 shows the different dimensions considered while evaluating Impala.

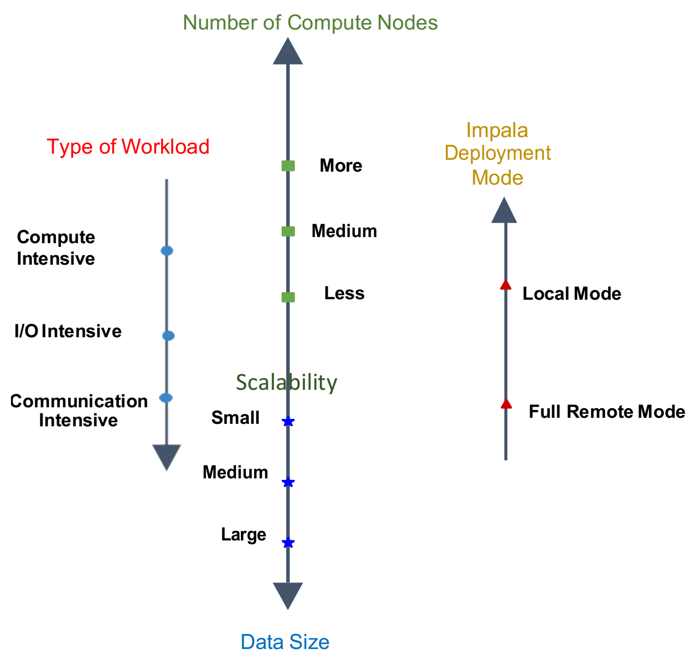


Figure 3. Evaluation Dimensions

3.1 Local Mode Setup

Impala daemons are deployed in the same nodes as the HDFS serving datanodes such that every node running HDFS datanode also has impalad daemon running on it. HDFS short-circuit reads are enabled by default which allows Impala to read local data directly from the file system. This removes the need to communicate through the DataNodes thereby reducing network communication and improving performance. After the query is executed, we run profile command which gives the detailed breakup of query planning and execution. We use this scheme to characterize the BigDataBench workload as I/O, Communication or Compute intensive. In this experiment we deploy both impalad and HDFS datanodes in 8 nodes and run experiments on 32 GB data.

The three main types of BigDataBench interactive queries run are: Scan, Aggregate and Join. Below are the BigDataBench Impala suite interactive queries run:

1. Scan - INSERT INTO TABLE result SELECT goods_price, goods_amount FROM bigdatabench_dw_item3 WHERE goods_amount> 224000;

2. Aggregate - INSERT INTO TABLE result SELECT goods_id, sum(goods_number) FROM bigdatabench_dw_item3 GROUP BY goods_id;

3. Inner Join - INSERT INTO TABLE result SELECT bigdatabench_dw_order3.buyer_id, SUM(bigdatabench_dw_item3.goods_amount) AS total FROM bigdatabench_dw_item3 JOIN bigdatabench_dw_order3 ON

bigdatabench_dw_item3.order_id = bigdatabench_dw_order3.
order_id GROUP BY bigdatabench_dw_order3.buyer_id LIMIT
10;

The BigDataBench Impala suite also provides micro-benchmark queries. Aggregate Avg and UnionAll queries are run from the micro-benchmark suite as follows:

1. Aggregate Avg - Insert Into table result SELECT
   AVG(goods_number) FROM bigdatabench_dw_item2;
2. UnionAll - Insert Into table result SELECT * FROM
   (
   SELECT * FROM bigdatabench_dw_item3 WHERE
   goods_amount> 750000
   UNION ALL
   SELECT * FROM bigdatabench_dw_item3 WHERE
   goods_amount< 5
) temp;

3.2 Full Remote Mode Setup

In this mode, all the Impala daemons are deployed in nodes mutually exclusive to the set of nodes where HDFS datanodes are deployed. This achieves 100% remote mode since no impalad daemon is running on the same node as HDFS datanode. As a result of this setup, all the data has to be first transferred to the impalad nodes across the network. This greatly increases the network communication time. As mentioned in the Impala development paper [19], remote data storage case is common where the data is stored remotely in cloud services such as Amazon S3. The paper also mentions that legacy storage infrastructure based on SANs necessitates a separation of computation and storage nodes, making Impala run in full remote mode. In this mode we evaluate InfiniBand QDR (32 Gbps) and 10Gb Ethernet. InfiniBand is expected to speed up query execution since the communication latency is reduced. In this experiment we deploy 4 impalad nodes and 4 HDFS datanodes in separate nodes, and use 32 GB data. BigDataBench interactive queries are run on this setup.

3.3 Increasing the Number of Compute Nodes and Fixing the Datasize

The purpose of this scheme is to evaluate how scalable [17] Impala is when the number of Compute Nodes is doubled. BigDataBench interactive queries are run. We want to find out which queries scale linearly, i.e. their query execution time is halved when the number of Compute Nodes is doubled. In this experiment we fix the datasize and vary the number of nodes. By Impala architecture, it is expected to scale out but we can see interesting results as it is dependent on the nature of the query. We fix the datasize to 64 GB and vary the number of Compute Nodes from 4, 8 and 16.

3.4 Increasing the Datasize and Fixing the Number of Compute Nodes

In this mode we fix the number of Compute Nodes to 8 and vary the datasize from 16 to 64 GB. BigDataBench interactive queries are run. We evaluate how the query execution time changes. For certain queries we can expect the running time to double when the datasize is doubled. However, the change in running time is also dependent on the nature of query.

4. Performance Evaluation

In our evaluation, we used Cloudera Impala verion cdh5-2.3.0_5.5.0. The source code is built from source and deployed in the cluster. PostgreSQL is used as a database to store the Hive metadata. Hadoop version used is 2.6.0 and Hive version is 1.1.0 which comes along with the Impala codebase. The experiments were run on two clusters:

1. Cluster A - Each node in the cluster has two 4-core 2.53 GHz Intel Xeon E5630 (Westmere) processors and 24 GB main memory. The nodes support 16x PCI Express Gen2 interfaces and are equipped with Mellanox ConnectX QDR HCAs with PCI Express Gen2 interfaces. The operating system used was RedHat Enterprise Linux Server release 6.4 (Santiago). Each DataNode has a single 1TB HDD, single 300GB SSD, and 12GB of RAM disk. For HDFS storage all three storage medium RAMDISK, SSD and DISK are used. InfiniBand network attached is QDR (32 Gbps). HDFS uses RAMDISK, SSD and DISK all three storage medium. The InfiniBand network card is QDR (32Gbps) and Ethernet used is 10Gb.
2. Cluster B - SDSC Comet Super Computer which is an XSEDE cluster designed by Dell and San Diego SuperComputer. Each compute node in this cluster has two twelve-core Intel Xeon E5-2680v3 processors, 128GB DDR4 DRAM, and 320GB of local SSD with CentOS operating system. Each node has 64GB of RAM disk capacity. The network topology in this cluster is 56Gbps FDR InfiniBand with rack-level full bisection bandwidth and 4:1 oversubscription cross-rack bandwidth. The Ethernet is 10Gb. Comet nodes have 7 petabytes of 200 GB/second performance storage and 6 petabytes of 100 GB/second durable storage. HDFS uses RAMDISK and SSD as storage medium.

Impala is run with HDFS Short-Circuit reads enabled. Impala by default does a Broadcast Join. In all the results below Inner Join means Inner Broadcast Join.

4.1 Local Mode Setup

In this section, we characterize the BigDataBench workloads - Scan, UnionAll, Aggregate Sum, Aggregate Avg, Inner Broadcast Join and Inner Shuffle Join as I/O, Computive or Network intensive. Impala is deployed in local mode where both the impalad daemon and HDFS datanode run on the same node. The experiments are run on Cluster A on InfiniBand in IPoIB mode.

After we run the query, we issue the profile command in Impala which gives the detailed breakdown of query planning [2], execution and resource utilization. To measure I/O intensiveness of a query, we see the stats of HDFS_SCAN_NODE in profile which gives detailed stats such as BytesRead, BytesReadLocal, PerReadThreadRawHdfsThroughput, and I/O time. To measure the Communication intensiveness of a query, we see the stats in DataStreamSender in profile which gives detailed stats such as BytesSent, NetworkThroughput, TotalNetworkReceiveTime, and TotalNetworkSendTime. To measure the Compute intensiveness of a query, we see the stats of the query operator such as AGGREGATION_NODE and HASH_JOIN_NODE which gives further detailed stats such as BuildPartitionTime, ProbeTime, BuildRowsPartitioned, and HashBuckets.

Figure 4 shows the workload characterization of different queries. Impala breaks down the query into small fragments and these distributed fragments are executed in parallel in the compute nodes. The workload characterization graphs show how much time was spent for I/O (HDFS Scan to read the data), Communication and Compute (Aggregate, Hash Join) for the fragments. For example Aggregate Sum graph shows that in fragment 4, 147 seconds were spent in Computation (Aggregation), 47 seconds spent in HDFS Scan to read the read from HDFS and about 13 seconds spent in communication in sending out the pre-aggregated results to the co-ordinator node. This way we sum up the time spent on each fragment and we can conclude if the query is I/O, Communication or Compute intensive. From this graph we see that the time spend on Computation is more than Communication and I/O, i.e. Computation time dominates both Communication and I/O. Hence we categorize Aggreagte Sum query as Compute intensive. This makes sense because Aggregate Sum does not involve broadcast of data across the network and only the pre-aggregated results are sent out to the co-ordinator node across the network which is not high volume data.

Table 1 shows the characterization of BigDataBench workloads on Impala. BigDataBench provides software package for Impala and the same queries in it are run here.

	I/O Intensive	Communication Intensive	Compute Intensive
Scan	√	×	×
UnionAll	√	×	×
Aggregate Sum	×	×	√
Aggregate Avg	×	×	√
Inner Join (Broadcast)	×	√	√
Shuffle Inner Join [1]	×	√	√

Table 1. BigDataBench WorkLoad Characterization

For the Scan and UnionAll queries, it reads all of the data locally and there is no aggregation involved in the query as it is select-based query. Hence for this query, I/O dominates the overall query execution time over Communication and Computation since most of the time is spent in reading the data from HDFS. For the Aggregate query, there is a Group By involved and hence it is Compute intensive. Aggregate Avg is even more Compute intensive since it divides the sum by the count of Group By operation. Inner Join in Impala is by default Broadcast Inner Join where the smaller data table is sent to all the nodes in the cluster across the network making it Communication intensive. Inner Join also involves a lot of Computation to do the matching of tuples based on Join predicates and extract the Inner Join results. This explains why Inner Join is Compute and Communication intensive. Shuffle Inner Join is the most Compute intensive workload and is less Communication intensive compared to Broadcast Inner Join. In Shuffle Inner Join we observed that more number of Send and Receive of data takes place but the amount of data sent or received is small compared to Broadcast Join since Shuffle Join partitions the table data into smaller segments and sends the Hash of it to other nodes. Since Shuffle Inner Join works on the hash of data we see it performs better than Broadcast Inner Join. Figure 5 shows the comparison of Broadcast (Default) and Shuffle Inner Joins in Impala.

4.2 Full Remote Mode Setup

In full remote mode, all the impalad daemons run in different nodes to HDFS datanodes. These experiments are run on Cluster A. Here impalad daemons are launched on 4 nodes and 4 HDFS datanodes are launched in different set of nodes. So every query first needs to fetch the data from remote data storage location across the network to the impala nodes. Hence we see lot of network communication in this case. The cluster has 10Gb Ethernet network interface card and InfiniBand QDR (32 Gbps). Figure 6 shows the comparison of BigDataBench Interactive queries on InfiniBand (IPoIB mode) and Ethernet networks.

From the performance results, we see Scan performs better on InfiniBand by 21% compared to Ethernet. This is because of the network transfer involved in fetching the data stored in remote location. Aggregate Sum benchmark performs better on InfiniBand QDR by 27% compared to 10Gb Ethernet. The benefit is more here because Aggregate involves sending the pre-aggregated results across the network to the co-ordinator node for final merge, which adds to the network communication along with the remote data fetch. Inner Join benchmark on InfiniBand is better only by 8% compared to Ethernet because computation dominates over communication since performing the Inner Join matching with all the tuples of the smaller table is computation intensive. Table and Column statistics of the Join tables were made available so that Impala tries to optimize the Join Query.

Figure 7 shows the network communication time of InfiniBand and 10Gb Ethernet. Interesting thing seen from the results is that even though the Inner Join query network communication latency and throughput on InfiniBand was better than 10Gb Ethernet by 19%, this entire benefit is not reflected in the overall query execution time due to Inner Join computation.

[1] This is a new query that can be added to BigDataBench Impala software package that evaluates Shuffle based Inner Join in Impala.

4.3 Increasing the Number of Compute Nodes and Fixing the Data Size

These experiments are run on Cluster B. Figure 8 shows the Scalability of Impala in IPoIB mode on InfiniBand FDR for 64 GB datasize on 4, 8 and 16 compute nodes.

From the performance results we see that Scan and Aggregate (Sum) workloads scale linearly on increasing the number of compute nodes. We see that the query execution times of Scan and Aggregate Sum reduce by approximately 50% on doubling the number of compute nodes from 4 to 8 and 8 to 16. But the improvement in the execution time of Inner Join is negligible since Impala by default does a broadcast Inner Join where the full smaller table data is sent to all the compute nodes. Hence adding more nodes means so many additional copies of data is transferred across the network and at each node the Inner Join matching needs to be done with all the tuples of the smaller table. Hence adding more nodes adds overhead in network communication and compution, which significantly reduces the benefit gained by increased parallelism on adding nodes.

Figure 9 shows the Scalability of Impala on 10Gb Ethernet for 64 GB datasize on 4, 8 and 16 compute nodes. From the results we see that Inner Join scales poorly on Ethernet as well. In fact from the query execution time, we see Inner Join scales slightly better on InfiniBand compared to Ethernet.

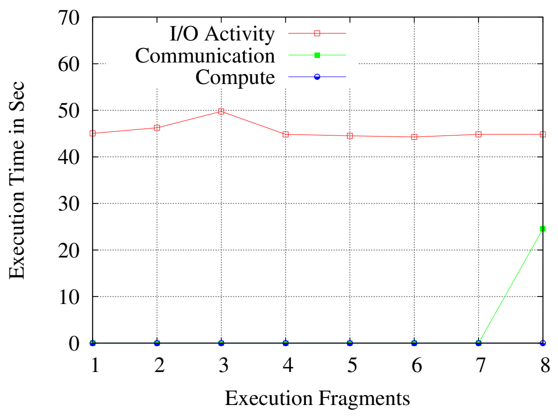
4.4 Increasing the Data Size and Fixing the Number of Compute Nodes

These experiments were run on Cluster B. In this experiment, we want to evaluate how Impala performs on increasing the datasize keeping the number of compute nodes constant. The experiments were run on InfiniBand FDR in IPoIB mode. Figure 10 shows the evaluation results on 8 compute nodes varying the datasize from 16, 32 and 64 GB.

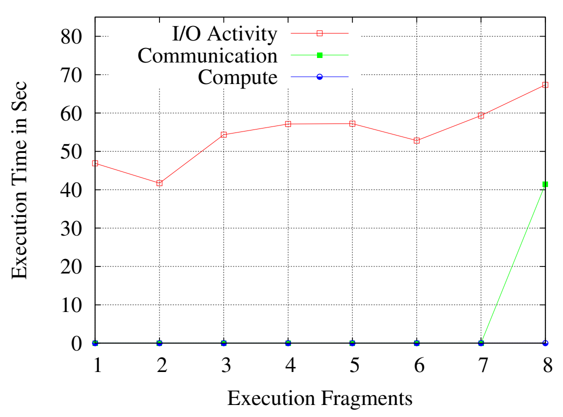
From the evaluation results we see that the execution times of Scan and Aggregate (Sum) workloads increase by average 90% on doubling the datasize from 16 GB to 32 GB and 32 GB to 64 GB. An interesting observation is that the execution time of Inner Join increases by 99% on doubling the datasize from 16 GB to 32 GB, but increases by 200% doubling the datasize from 32 GB to 64 GB. The results show that broadcast Inner join of Impala scales poorly on increasing the datasize since it increases the network data transfer due to broadcast operation and also increases the Join computation since it has to do a matching with all the tuples in the smaller table at each compute node.

5. Related Work

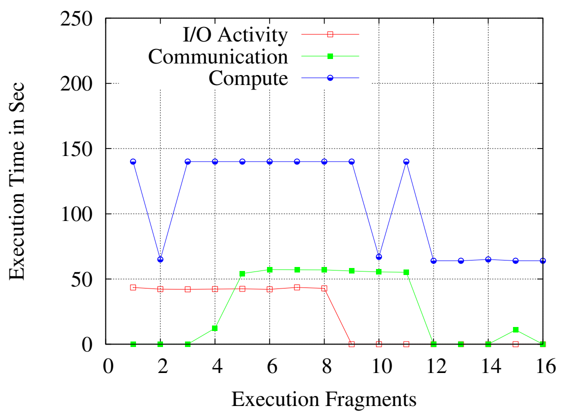
Recent studies in Big Data show that Impala performs faster than MapReduce frameworks such as Hive and Tez. This is because Impala uses massive parallel processing (MPP) engine and not MapReduce and thereby avoids the overheads of two phase execution model of MapReduce. Moreover, in MapReduce, the intermediate map outputs are spilled to disk which results in performance degradation, whereas Impala tries to do most of its computation in memory. Impala is faster than Apache Hive but Impala does not provide one stop SQL solution for all big data problems. Impala is memory intensive and does not run effectively for heavy data



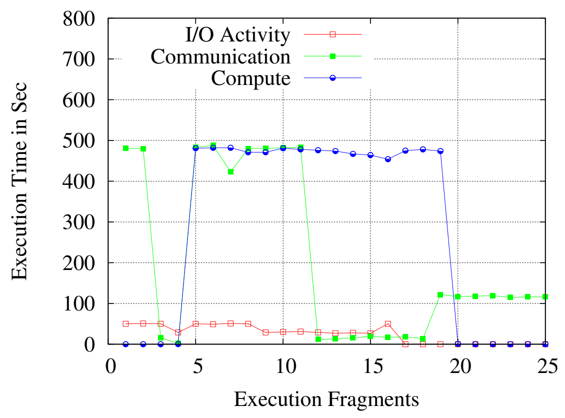
(a) Scan



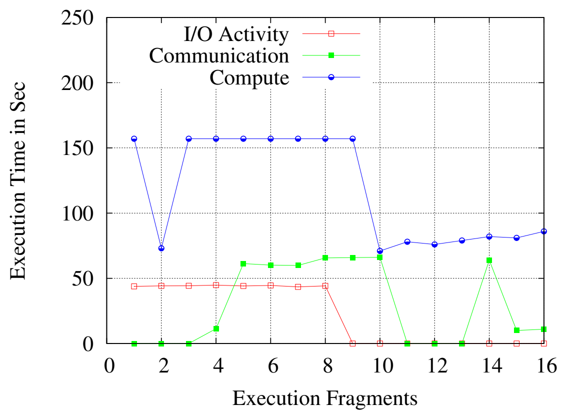
(b) UnionAll



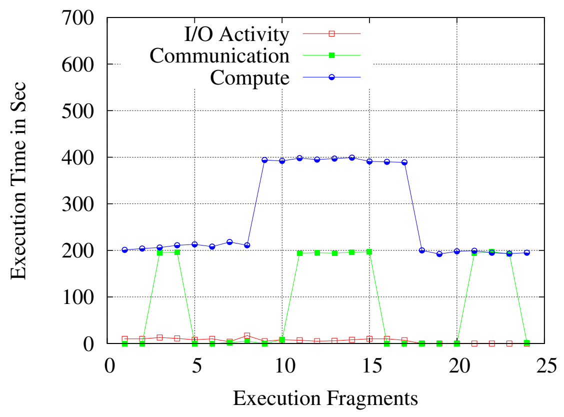
(c) Aggregate Sum



(d) Broadcast Inner Join



(e) Aggregate Avg



(f) Shuffle Inner Join

Figure 4. Workload Characterization of Queries

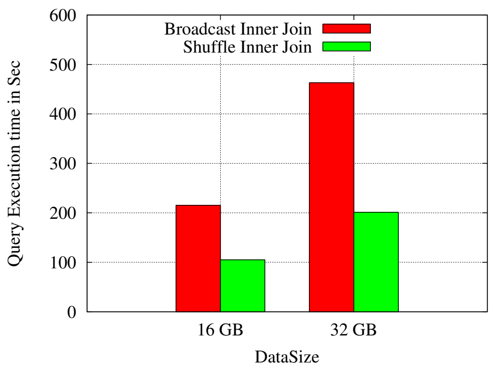


Figure 5. Query Execution of Broadcast and Shuffle Inner Joins

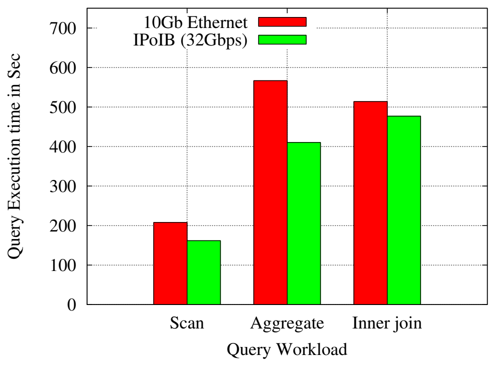


Figure 6. Query Execution on Different Networks

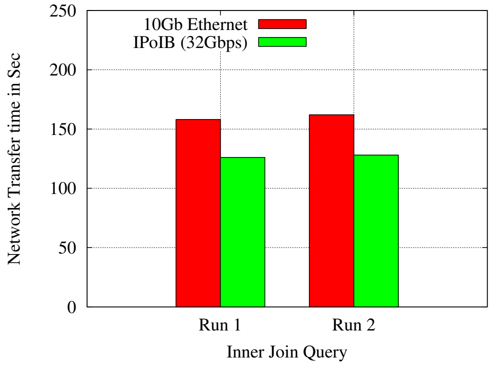


Figure 7. Network Communication Part of Inner Join Query Execution

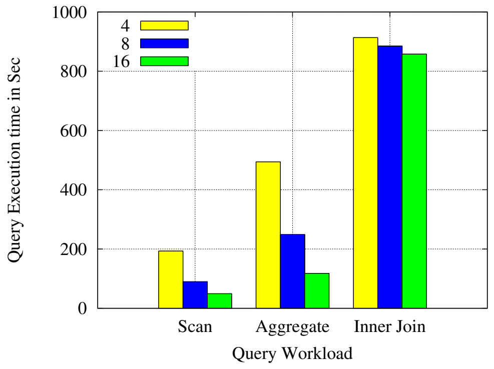


Figure 8. Scalability over Nodes on InfiniBand

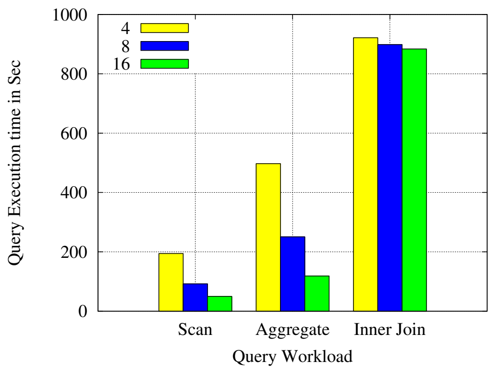


Figure 9. Scalability over Nodes on Ethernet

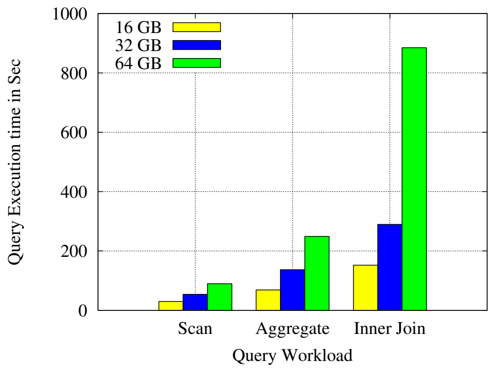


Figure 10. Scalability over Datasize

operations like joins because it is not possible to push in everything into the memory. This is when Hive comes to the rescue. If an application has batch processing kind of needs over big data then organizations must opt for Hive [14]. If real time processing of adhoc queries [26] on subset of data is needed then Impala is a better choice.

Earlier versions of Impala supported pure in-memory queries meaning that the query execution would fail if the data could not fit into memory. Recently Cloudera has added spill to disk feature which enables data to be spilled to disk in case it cannot fit in memory. Impala tries to intelligently avoid disk access by spilling to disk data that may not be referenced again. Impala provides faster query responses by holding as much data as possible in memory.

Recently Apache Software Foundation (ASF) announced that it has taken Cloudera Impala as an incubating project [7]. This means that Impala project development will be moved more and more to the ASF and adopt the open standards of Apache as the manner in which Impala will be next developed.

In recent years, modern interconnects such as InfiniBand, have seen increased usage for HPC and Big Data Systems. InfiniBand provides native support for RDMA (Remote Direct Memory Access) feature. RDMA supports zero-copy networking by enabling the network adapter to transfer data directly to or from application memory, eliminating the need to copy data between application memory and the data buffers in the operating system. The HiBD project [8] at The Ohio State University has made HDFS [18], MapReduce [30] and Hadoop RPC [21] RDMA capable. From the performance results we see RDMA benefits these components. The project has also made Apache Spark [22], another popular Big Data software, RDMA capable and see good benefits in performance. RDMA versions of these software are released in the HiBD website and are free to download. These results show that Big Data applications can benefit from InfiniBand RDMA feature if the Big Data Application is Communication intensive.

6. Conclusion and Future Work

In this paper, we characterize Cloudera Impala workloads with BigDataBench on InfiniBand clusters. BigDataBench queries for Impala are characterized as I/O, Communication or Compute intensive by running them in local mode on an InfiniBand QDR cluster. Experimental results show that Scan and UnionAll queries are I/O intensive, Aggregate Sum and Aggregate Avg queries are Compute intensive, and for Inner Join queries Computation outweights Communication. Remote data storage mode for Impala is common where the data is stored remotely, such as in cloud services (e.g. Amazon S3). In remote data storage setup which is more Communication intensive as the data has to be transferred across the network, we observe that Impala performs better on InfiniBand QDR (IPoIB) network compared to 10Gb Ethernet - Scan by 21%, Aggregate by 27%, and Inner Join by 6%. Communication part of Inner Join Query on InfiniBand QDR is better than 10Gb Ethernet by 19%. Although Inner Join is Communication intensive, we do not see much difference in overall execution runtime of the query on InfiniBand and 10 Gb Ethernet since Computation dominates for doing the Inner Join matching to produce the Join results. We also evaluate scalability of Impala on Comet XSEDE cluster with InfiniBand FDR and observe that Scan and Aggregate query workloads can scale linearly. Results show that Inner Join query benefits are negligible on adding more compute nodes since it increases the Join Computation. From these experiments we can conclude that InfiniBand improves the Communication part of Inner Join queries but the overall execution of the Inner Join query needs to be further enhanced. As future work we plan to change the Communication layer of Impala to InfiniBand RDMA and study the associated benefits.

References

[1] How Impala Works. http://www.slideshare.net/dataera/how-impala-works-38586729?related=3.

[2] Impala Query Compilation. http://www.slideshare.net/cloudera/query-compilation-in-impala, .

[3] Cloudera Impala. http://impala.io/overview.html, .

[4] The Apache Hadoop Project. http://hadoop.apache.org//.

[5] Big Data Bench from AMPLab. https://amplab.cs.berkeley.edu/benchmark/.

[6] IDC: Hadoop Commonly Used with Other Big Data Analytics System. http://www.computerworlduk.com/news/applications/3476789//.

[7] Impala as Apache Incubation Project. http://impala.io/overview.html.

[8] RDMA Hadoop. http://hibd.cse.ohio-state.edu/overview/.

[9] TOP500 Supercomputing Sites. http://www.top500.org/.

[10] C. Baru, M. Bhandarkar, R. Nambiar, M. Poess, and T. Rabl. Benchmarking Big Data Systems and the BigData Top100 List. *Big Data*, 1 (1):60–64, 2013.

[11] M. Beine, A. Boucher, B. Burgoon, M. Crock, J. Gest, M. Hiscox, P. McGovern, H. Rapoport, J. Schaper, and E. Thielemann. Comparing Immigration Policies: An Overview from the IMPALA Database. *International Migration Review*, 2015.

[12] R. Brightwell, D. Doerfler, and K. D. Underwood. A Comparison of 4x InfiniBand and Quadrics Elan-4 Technologies. In *Cluster Computing, 2004 IEEE International Conference on*, pages 193–204. IEEE, 2004.

[13] A. Chauhan. *Learning Cloudera Impala*. Packt Publishing Ltd, 2013.

[14] Y. Chen, X. Qin, H. Bian, J. Chen, Z. Dong, X. Du, Y. Gao, D. Liu, J. Lu, and H. Zhang. A Study of SQL-on-Hadoop Systems. In *Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, pages 154–166. Springer, 2014.

[15] A. Floratou, U. F. Minhas, and F. Özcan. SQL-on-Hadoop: Full Circle Back to Shared-Nothing Database Architectures. *Proceedings of the VLDB Endowment*, 7(12):1295–1306, 2014.

[16] R. Han, S. Zhan, C. Shao, J. Wang, J. Xu, L. K. John, L. Wang, and J. Zhan. BigDataBench-MT: A Benchmark Tool for Generating Realistic Mixed Data Center Workloads. *arXiv preprint arXiv:1504.02205*, 2015.

[17] H. Hu, Y. Wen, T.-S. Chua, and X. Li. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *Access, IEEE*, 2:652–687, 2014.

[18] N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy, and D. K. Panda. High Performance RDMA-based Design of HDFS over InfiniBand. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 35. IEEE Computer Society Press, 2012.

[19] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovytsky, C. Ching, A. Choi, J. Erickson, M. Grund, D. Hecht, M. Jacobs, et al. Impala: A Modern, Open-Source SQL Engine for Hadoop. In *CIDR*, 2015.

[20] J. Liu, J. Wu, and D. K. Panda. High performance RDMA-based MPI Implementation over InfiniBand. *International Journal of Parallel Programming*, 32(3):167–198, 2004.

[21] X. Lu, N. S. Islam, M. Wasi-Ur-Rahman, J. Jose, H. Subramoni, H. Wang, and D. K. Panda. High-performance design of Hadoop RPC with RDMA over InfiniBand. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 641–650. IEEE, 2013.

[22] X. Lu, M. W. U. Rahman, N. Islam, D. Shankar, and D. K. Panda. Accelerating spark with rdma for big data processing: Early experiences. In *High-Performance Interconnects (HOTI), 2014 IEEE 22nd Annual Symposium on*, pages 9–16. IEEE, 2014.

[23] C. Luo, W. Gao, Z. Jia, R. Han, J. Li, X. Lin, L. Wang, Y. Zhu, and J. Zhan. Handbook of BigDataBench (Version 3.1)A Big Data Benchmark Suite.

[24] M. Mammo and S. K. Bansal. Distributed SPARQL over Big RDF Data: A Comparative Analysis Using Presto and MapReduce. In *Big Data (BigData Congress), 2015 IEEE International Congress on*, pages 33–40. IEEE, 2015.

[25] W. Rödiger, T. Mühlbauer, A. Kemper, and T. Neumann. High-Speed Query Processing over High-Speed Networks. *Proceedings of the VLDB Endowment*, 9(4):228–239, 2015.

[26] R. Saltzer, I. Szegedi, and P. De Schacht. Impala in Action: Querying and Mining Big Data. 2015.

[27] S. Wadkar and M. Siddalingaiah. Data Warehousing Using Hadoop. In *Pro Apache Hadoop*, pages 217–239. Springer, 2014.

[28] S. Wanderman-Milne and N. Li. Runtime Code Generation in Cloudera Impala. *IEEE Data Eng. Bull.*, 37(1):31–37, 2014.

[29] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu. BigDataBench: A Big Data Benchmark Suite from Internet Services. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 488–499, Feb 2014.

[30] M. Wasi-ur Rahman, N. S. Islam, X. Lu, J. Jose, H. Subramoni, H. Wang, and D. K. Panda. High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 1908–1917. IEEE, 2013.

[31] J.-M. Zhao, W.-S. Wang, X. Liu, and Y.-F. Chen. Big Data Benchmark - Big DS. In *Advancing Big Data Benchmarks*, pages 49–57. Springer, 2013.