PERFORMANCE TUNING FOR EFFECTIVE SQOOP TO TRANSFORM BETWEEN SQL & NOSQL

A Major II Project Report

Submitted in partial fulfillment of
the requirements for the award of the degree
of
Master of Technology
in
Computer Science and Engineering
by

Pranav Sharma

under the guidance of

Dr. Rajni Jindal
Associate Professor
CSE Department

DEPARMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY, NEW DELHI

DELHI-110 042 (INDIA)

JULY 2017

CERTIFICATE

This is to certify that Project Report entitled "Performance tuning for effective sqoop to transform between Sql & noSql" Submitted by Pranav Sharma (roll no. 2K15/CSE/13) in partial fulfillment of the requirement for the award of degree Master of Technology (Computer Science and Engineering) is a record of the original work carried out by him under my supervision.

Dr. Rajni Jindal
Associate Professor
Department of Computer Science & Engineering
Delhi Technological University

DECLARATION

I hereby declare that the Major Project-II work entitled "Performance tuning for effective sqoop to transform between Sql & noSql" which is being submitted to Delhi Technological University, in partial fulfillment of requirements for the award of the degree of Master Of Technology(Computer Science and Engineering) is a bonafide report of Major Project-II carried out by me. I have not submitted the matter embodied in this dissertation for the award of any other Degree or Diploma.

(Pranav Sharma)
2K15/CSE/13

ACKNOWLEDGEMENT

I am highly indebted to Dr. Rajni Jindal and obliged for giving me the autonomy of working and exploring different avenues regarding thoughts. The sustaining and blooming of the present work is primarily because of her significant direction, proposals, insightful judgment, valuable feedback and an eye for flawlessness. My mentor always answered the myriad of my doubts with smiling graciousness and immense patience, never letting me feel that I am novices by always lending an ear to my views, appreciating and improving them and by giving me a free hand in my project. It's only because of her overwhelming interest and helpful attitude; the present work has attained the stage it has.

I would like to express my sincere gratitude towards family for their blessings, encouragement and moral support. In the end, I would like to thank my peer group for providing me with a conductive and competitive environment. Without light and fun moments shared with them, this research work would have been difficult.

(Pranav Sharma)

ABSTRACT

Transferring data to and from relational databases is challenging and laborious. Because data transfer requires careful handling, Apache Sqoop, short for "SQL to Hadoop," was created to perform bidirectional data transfer between Hadoop and almost any external structured data store. Taking advantage of MapReduce, Hadoop's execution engine, Sqoop performs the transfers in a parallel manner. It is very challenging task to transfer data with maximum performance and efficient manner.

The variety of data sources and analytic targets presents a challenge in setting up effective data transfer pipelines. Data sources can have a variety of subtle inconsistencies: different DBMS providers may use different dialects of SQL, treat data types differently, or use distinct techniques to offer optimal transfer speeds. Depending on whether you're importing to Hive, Pig, Impala, or your own MapReduce pipeline, you may want to use a different file format or compression algorithm when writing data to HDFS. Sqoop helps the data engineer tasked with scripting such transfers by providing a compact but powerful tool that flexibly negotiates the boundaries between these systems and their data layouts.

In order to enhance the performance of Sqoop import and export operation, different parameters are configured in this project. Basic information regarding Sqoop import and export tool are presented in the different chapter. Analysis of configured Sqoop parameters is documented in the result and analysis chapter.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Overview

Regardless of whether moving a little accumulation of individual get-away data between applications or, then again moving petabytes of information between corporate distribution center frameworks, incorporating information from different sources remains a battle. Information stockpiling is more available because of the accessibility of various generally utilized capacity frameworks and going with apparatuses. Center to that are relational databases (e.g., Oracle, MySQL, SQL Server,) that have been utilized for a considerable length of time to serve and store immense measures of information over all businesses.

So Big Data can be described by its 5 characteristics:

1. Volume: Big Data is a huge data set and it is not a sample.
2. Velocity: often available in real time and speed in which data is generated.
3. Variety: type and nature of data which draws from text, images, audio and video.
4. Variability: inconsistency of data set create problem to handle and manage it.
5. Veracity: the quality of captured data can vary and affecting analysis.

So processing of data sets of Big Data there is an open-source software framework **Apache** Hadoop. It used for distributed storage and data processing using the MapReduce programming model. Apache Hadoop has 2 cores, for storage part known as **Hadoop Distributed File System** (HDFS) and for processing part **MapReduce** programming model. Hadoop distribute the data on several nodes by splitting the file and node will process the data in parallel.

Hadoop cluster includes a single master and multiple slave nodes. The master node consists of 2 Tracker: 1st Job Tracker and 2nd Task Tracker and 2 Node: 1st NameNode, and 2nd DataNode. A slave node acts as both a DataNode and Task Tracker, though it is possible to have data-only and compute-only slave nodes.

Basic requirement for Hadoop is Java Runtime Environment (JRE) 1.6 or higher and standard startup and shutdown script require Secure Shell (SSH) to setup between nodes. Hadoop is mainly written in java and its command line utility written as Shell Script.

Hadoop support many software packages to extend its basic capabilities:

1. Apache Sqoop: a tool which is used to transfer bulk amount of data between HDFS and RDBMS.
2. Apache Hive: a data warehouse which provides data analysis, summarization and query.
3. Apache Pig: a high-level programming platform to create program which run on Hadoop.
4. Apache Spark: a fast engine for Sql, machine learning and graph processing.

So this thesis only focuses on Apache Sqoop for its performance tuning in import and export data between HDFS and RDBMS.

1.2 What is Sqoop

Sqoop is a tool which is used to exchange data between Hadoop Distributed File System and relational databases Management System. Using Sqoop you can import data from a RDBMS (MySQL / Oracle) into the Hadoop Distributed File System (HDFS) and then export the data back into an RDBMS.

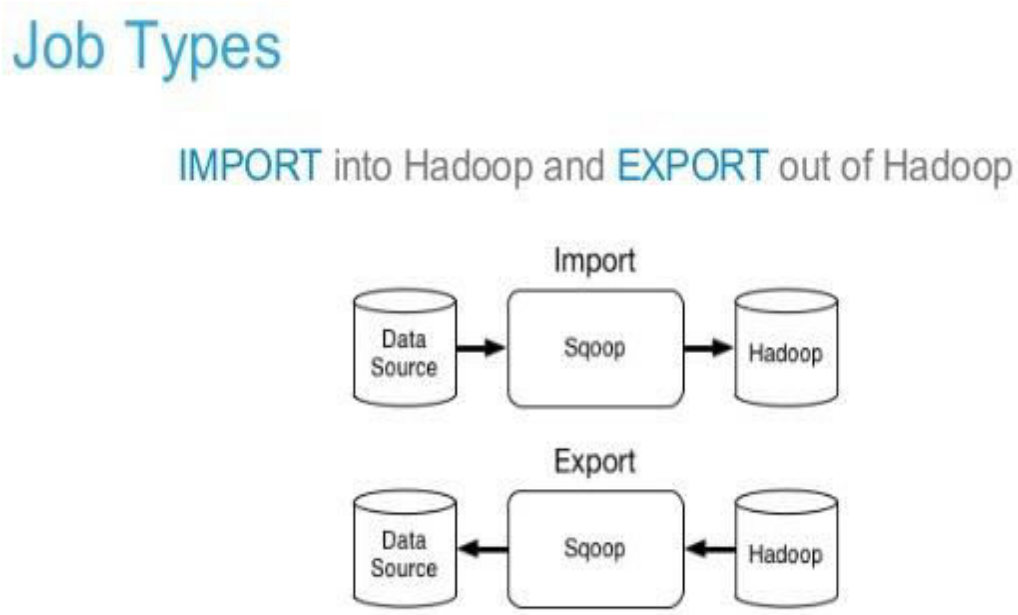

Figure 1-1 Sqoop Import & Export

1.2.1 Sqoop Import

The import tool of Sqoop will import individual tables from Relational Database to the Hadoop distributed File System (HDFS). Each row in a table is treated as a record in HDFS and all records are stored as text in file or as binary data or as sequence file.

The following syntax is used for the import command:

$ sqoop tool (generic-arguments) (import-arguments)
$ sqoop-tool (generic-arguments) (import-arguments)
Here tool indicates the "import" operation.

1.2.2 Sqoop Export

The export tool exports a set of files from HDFS back to an RDBMS. The files given as input to Sqoop contain records, which are called as rows in table and the target table must exist in the target database. Each row of table is created based on the delimiter provided by the user.

The following syntax is used for the export command:

$ sqoop tool (generic-arguments) (export-arguments)
$ sqoop-tool (generic-arguments) (export-arguments)
Here stool indicates the “export” operation.

The default operation in export command is insert all the record from the input file to the target table or in the update mode UPDATE statement is replace the all the record of existing database.

1.3 Working of Sqoop

Sqoop do most of the process internally based on the database to describe the schema for the import of data.

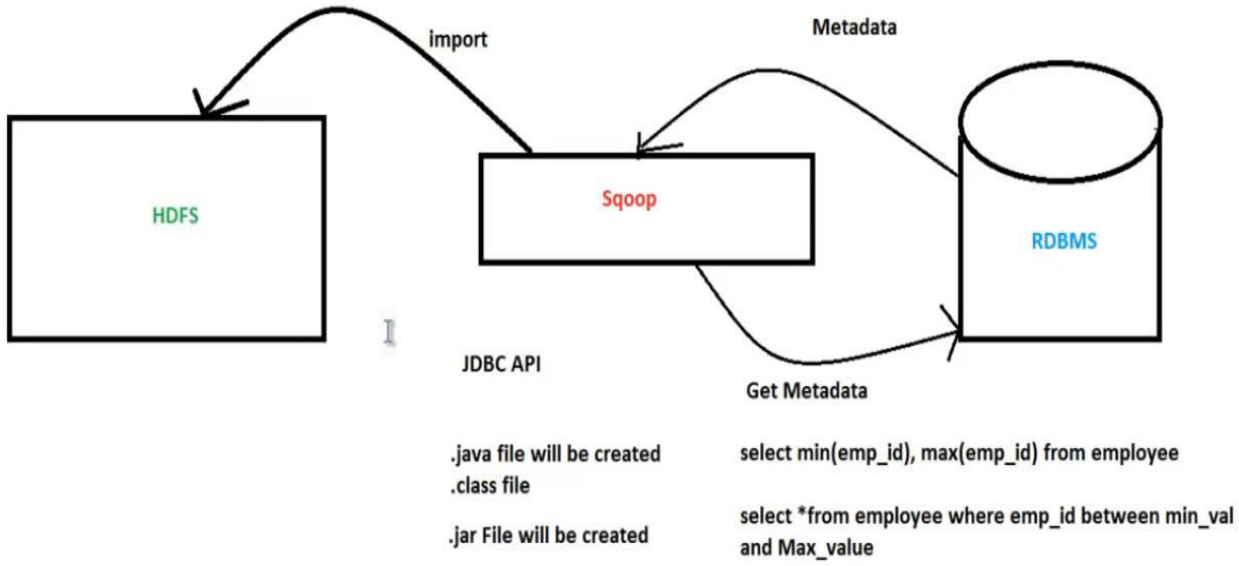

Figure 1-2 Sqoop internal working

1.3.1 Internal working

Step 1: First sqoop will communicate to the RDBMS tool for metadata. This metadata include the information about database schema.

Step 2: Using this metadata sqoop will generate a java class file and it usages JDBC API to generate the data.

Step 3: Now java compiler compile the class file to creates the .jar file (generate table structure).

Step 4: Sqoop will again try to get some information about split column from table which is primary key column. Sqoop will internally run SQL command to fetch the record of table which tries to import. Sqoop get the MIN & MAX value of the import table.

Step 5: By using the min & max value, it will fetch all the record of the table and this select command is generated by each mapper that is trying to fetch the data.

Step 6: After fetching Meta data information and getting some data, sqoop will try to import the data from RDBMS to HDFS. Instead of HDFS, sqoop can transfer the data to the Hive and Hbase also.

1.3.2 Available commands

Sqoop is a command line tool, and type "sqoop help" to see all the available command.

```
Available commands:
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table
  help               List available commands
  import             Import a table from a database to HDFS
  import-all-tables  Import tables from a database to HDFS
  import-mainframe   Import datasets from a mainframe server to HDFS
  job                Work with saved jobs
  list-databases     List available databases on a server
  list-tables        List available tables in a database
  merge              Merge results of incremental imports
  metastore          Run a standalone Sqoop metastore
  version            Display version information

See 'sqoop help COMMAND' for information on a specific command.
pranav@pranav:~$
```

Figure 1-3 available commands

The command line of sqoop has the below structure:

"Sqoop TOOL PROPERTY_ARGUMENTS SQOOP_ARGUMENTS [-- EXTRA_ARGS]"

TOOL indicates the operation you want to perform like import or export. PROPERTY_ARGUMENTS are the parameter set for java database connectivity. SQOOP_ARGUMENTS contains all the various Sqoop parameters like database, tables, user, password etc. EXTRA_ARGS specifies the additional parameter for each connector and this must be separated from Sqoop arguments with a "--".

1.3.3 Specialized connector

In the Sqoop distribution connectors are not the part of it. To get the benefit of these optimizations, particular connector should be downloaded and install those specialized connectors. On the node where sqoop is running, you can install particular connector and the suitable JAR file will be promulgating to the Sqoop file system.

A major power of Sqoop is to work with all kind of database systems and data warehouses. To extract the distinctive conduct of every framework, Sqoop presented the idea of connectors: all database-particular operations are assigned from Sqoop to the specific connectors. Sqoop itself

groups numerous such connectors; you don't have to download anything additional with a specific end goal to run Sqoop. The broadest connector packaged with Sqoop is the Generic JDBC Connector that uses just the JDBC interface. This will work with each JDBC consistent database framework. Despite this non specific connector, Sqoop additionally dispatches with specific connectors for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, and DB2, which use uncommon properties of every specific database framework. You don't have to expressly choose the coveted connector, as Sqoop will consequently do as such in light of your JDBC URL.

Most of the connectors rely on upon the fundamental JDBC drivers with a specific end goal to make the connection with the remote database server. It's basic to introduce both the particular connector and the proper JDBC driver. It's additionally imperative to recognize the connector from the JDBC driver. The connector is a Sqoop particular pluggable piece that is utilized to give a portion of the utility that may be done speedier when utilizing database meticulous changes. The JDBC driver is additionally a pluggable part. Be that as it may, it is in area of Sqoop and uncovered database interfaces in a convenient way for all Java applications.

“Sqoop always requires both the connector and the JDBC driver.”

1.4 Problem statement

Today`s world, immense measure of information is produced as often as possible, so to perform operation on that information, Relational Database System will take additional time. Apache Hadoop gives the Mapping system to perform operation on information parallel. Utilizing Apache Hadoop, single node or multi-node cluster, MapReduce structure will do operation in less time contrast with RDBMS.

So to perform operation on that RDBMS data, it should bring to HDFS or hive or Hbase. Apache Sqoop provides setting to transport data from RDBMS to HDFS. Sqoop tool is a command line program and using its command we can transfer the records of table. In the Sqoop command line it has many arguments in JDBC connection or sqoop mapping to **optimize performance**.

You can tune the following Sqoop program argument in the **JDBC connection** or **mapping** to optimize performance of import and export tool:

- batch
- boundary-query
- compress or z
- direct
- Dsqoop.export.records.per.statement
- Enable primary key
- fetch-size
- num-mapper
- split-by

Tune above Sqoop parameter for the better performance in the importing and exporting the data between traditional data store (RDBMS) to Hadoop cluster.

Before moving enormous measures of information, it is astute to test and tweak each piece of the information flow to accomplish most ideal execution. Any non-ideal piece of the flow will add to expanded cost of data exchange.

With the end goal of this thesis, I will be doing test data migration from a MySQL instance to a Hadoop cluster. Likewise, I will be concentrating on the greater part of the Migration tool`s performance. However, we will see that tuning source, sqoop and Hadoop yields a huge change in data exchange times.

1.5 Scope of work

With the happening to cutting edge age, endeavors need to keep up with data heightening impact. Social database does not have the ability to manage such measure of information for continuous framework, e.g., weather forecast system.

The prerequisite for Sqoop foundation is Java and Hadoop. These two must be preinstalled in the structure. As Sqoop is a sub-wander of Hadoop, it can simply take a shot at Linux working framework. For the test, we will move data from a MySQL table of various sizes with different parameters at Hadoop and Sqoop level.

For the improvement in the performance of importing and exporting the data using sqoop tool, we need to establish the environment. So we have an Oracle VM Virtual Box, which is a free and open source programming at present being created by Oracle Corporation. The source machine for the test has provisioned IOPS with 500 GB of internal storage. It is vital to note here that the I/O speed of the source framework would be basic in general speed of the data migration. In the event that this machine is acting moderate, regardless of how quick different parts of the information stream are, things would remain slow.

In the fundamental operation of sqoop tool, it will give the diverse outcomes when the arguments will tune. To perform tuning of execution we have the setup for Java, Hadoop, Sqoop, MySQL database server and the distinctive size of tables to import and export.

Information relocation into Hadoop can wind up plainly precarious and testing due to:

- Size of data
- Machine’s execution time
- Network delay
- Data transfer tool’s delay (Sqoop)
- Hadoop nodes execution time

In the past research on the performance tuning of the sqoop, "correlation aware technique for Sql to NoSql transformation" proposed and give a calculation to the better locality and query efficiency. Relational database usually has a log file to store operation including configuration, action, modification and query. The log record is generally used to screen the database. What's more, the logged operations can be dealt with as the inquiries getting to the database. Consequently, once the log record is dissected, we can know the tables which are as often as possible use by inquiries.

The plan of distributed file system gives the capacity to execute jobs in parallel while information is part and imported to node randomly. Nonetheless, this conduct may not be useful for preparing a few data which is as often as possible utilized. To enhance the data placement may upgrade the performance as far as database utilization and this is the inspiration of my thesis.

In this research I analyze the diverse scenario of importing the different size of table of relational database. Tuning the argument of migration tool, different time stamp is noted down for data relocation into Hadoop. By utilize that time stamps different graphs are prepared. Using that graph analyze the performance with the default nature of sqoop arguments.

Thesis organization:

Chapter 2: literature review, this chapter previously tuned sqoop tool described.

Chapter 3: Importing data, this chapter describes the importing tool parameter which will be configured.

Chapter 4: Exporting data, this chapter describes the exporting tool parameter which will be configured.

Chapter 5: Implementation, this chapter describes the implementation required for experiment for performance tuning.

Chapter 6: Result and Analysis, this chapter describes the result of performance tuned parameter.

Chapter 7: Conclusion and Future work, this chapter concludes the result and describes future work.

2 Literature Review

In this chapter, I describe the previously existing sqoop tool and CA_Sqoop tool model. At the end to this chapter comparative analysis between these models is also presented.

In the Apache Sqoop tool each table separated into four parts and randomly distributed on Hadoop node. In any case, there is still a database execution worry with this data transformation mechanism. Correlation -Aware technique on Sqoop (CA_Sqoop) to enhance the data locality. By network occasion related information as nearer as it could be to diminish the information change taken a toll on the system and enhance the performance regarding database use. The CA_Sqoop additionally considers the table relationship and size for better data region and query productivity. Simulation comes about demonstrate that data locality of CA_Sqoop is two times superior to that of unique Apache Sqoop.

2.1 Correlation aware Technique

Relational database usually has a log file which is store the operation related to configuration, modification and query. This log file is mainly used to inspect the database activity. By utilizing that log document we can discover what table is as often as possible utilized by the queries.

Queries are utilized to get to databases and demonstrated the data with various importances. JOIN is an as often as possible utilized operation which combines two table as indicated by particular column. On the off chance that the information of two tables is dispersed on eight data nodes, information change on system can't be maintained a strategic distance from to play out the JOIN operation. The speed of data getting to on system is clearly slower than that in neighborhood circles. Thus, data locality should be improved to evade data transformation on network and improve the performance of JOIN process.

Table Correlation (TC)	Table 1	Table 2	Table 3	Table 4	Table 5
Table 1		15	25	63	21
Table 2			82	24	34
Table 3				72	12
Table 4					45
Table 5					

Table 2-1 Example of Table Correlation

Table 2-I exhibits a lattice demonstrating level of Table Correlation (TC) by dissecting questions in the log record. Taking Table 1 and Table 2 as a case, the degree is 15 which imply there were 15 inquiries getting to these two tables at the same time as indicated by history records. So also, there were 45 questions getting to Table 4 and table 5 as indicated by the log. Sometimes, it is

most certainly not important to put two relative little tables together even in spite of the fact that the level of table connection is high. Hadoop doles out an errand to one of the hubs, which possesses more information to decrease the cost information change. Such idea is additionally connected in this work to enhance the execution as per table sizes.

2.2 CA_Sqoop comparison

The ranges of parameters for CA_Sqoop tuning model are given as follows.

- Table Size : 1 to 10 GB
- Table Correlation : 1~1000
- Node Capacity : 2~50
- Number of Tables : 20~300
- Number of Nodes : 60~100

Figure 2-1 is the aftereffect of various number of hub limit while bringing in tables into the bunch. It gives the improvement on information territory with little cluster.

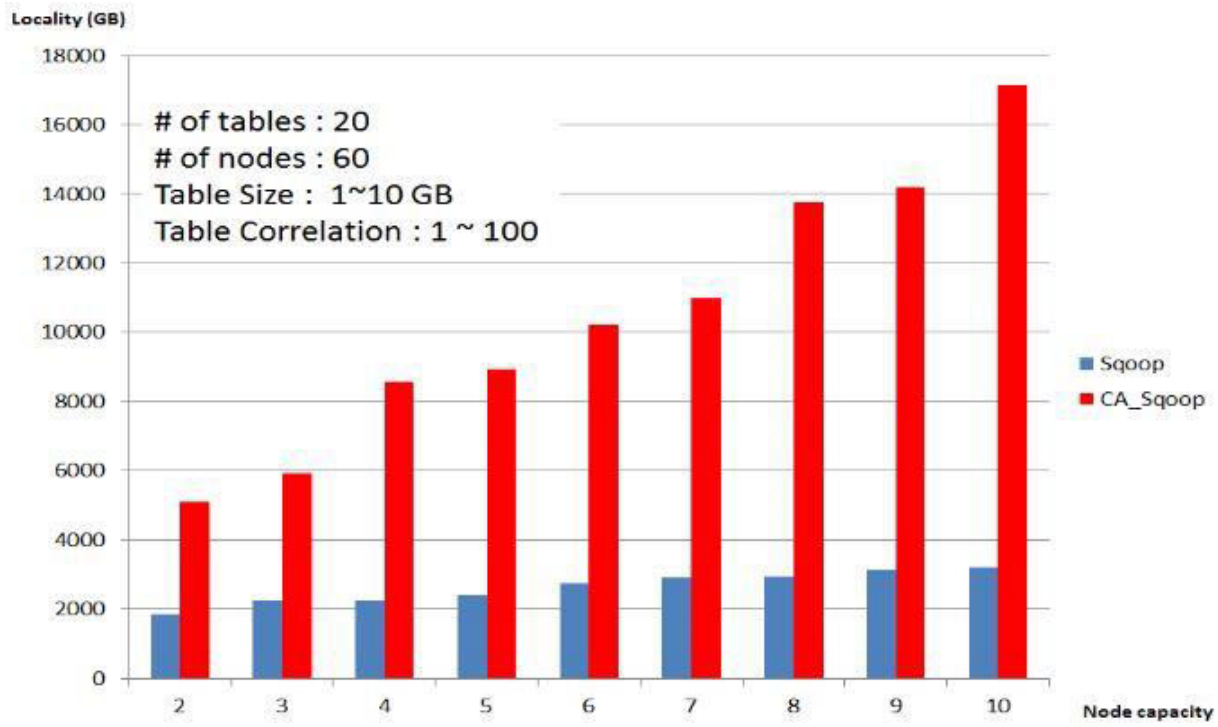

Figure 2-1 Importing twenty table to a small cluster

CA_Sqoop defeats Sqoop with improved data locality even the node limit is expanded. In this simulation tables and nodes are variable and CA_Sqoop gives far better improvement in the correlation. In this result, when node capacity increased, Sqoop gives constant locality with minor improvement but CA_Sqoop give extremely better result.

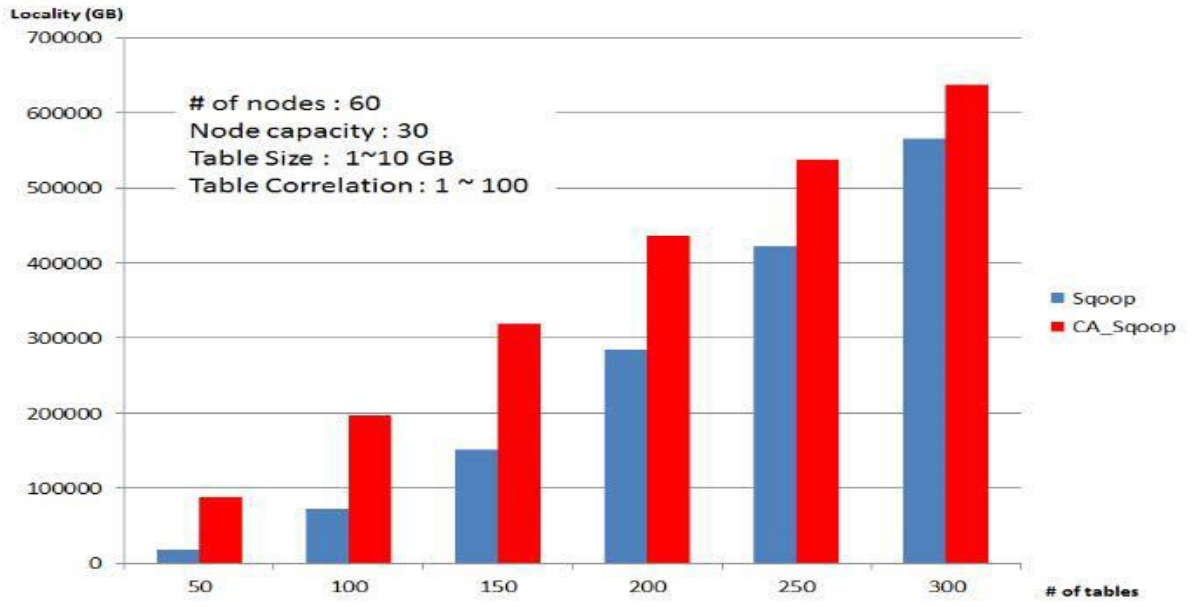

Figure 2-2 Importing different number of tables to a cluster

In this result minor difference are presented in data locality of both Sqoop.

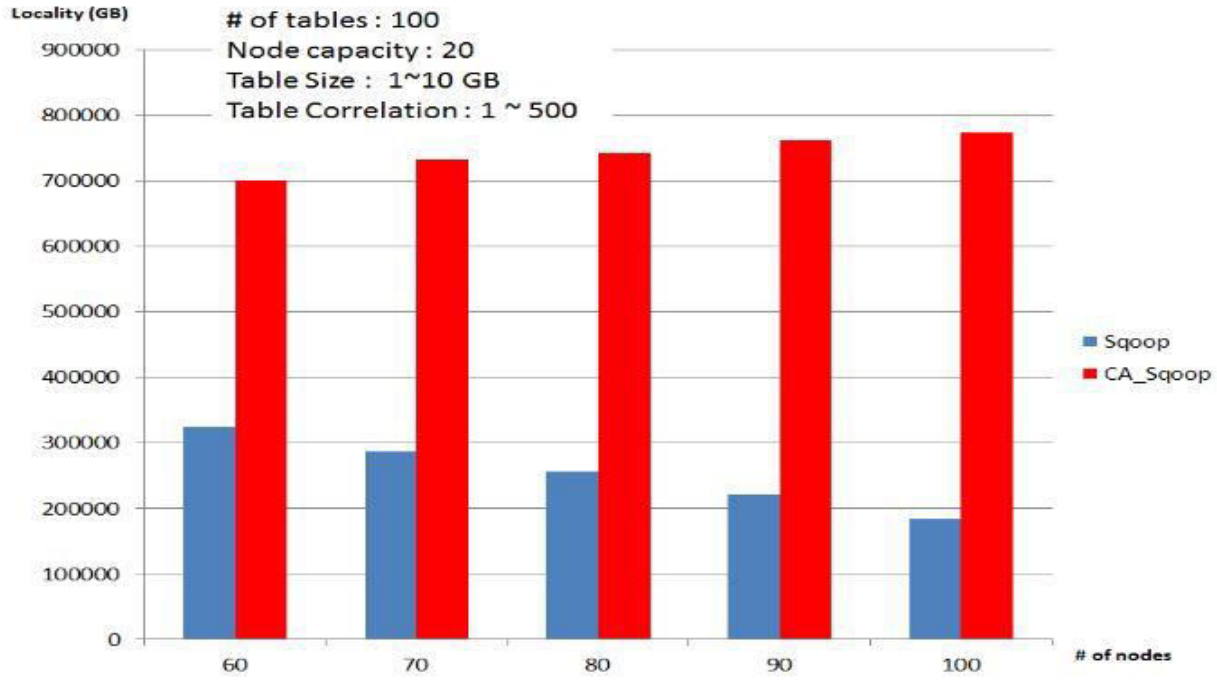

Figure 2-3 Impact on different size of cluster

When number of node is increased, performance decreased but in CA_Sqoop results are good.

Those results demonstrate that approach of CA_Sqoop can get a larger number of data localities than normal Sqoop without tuning. So this locality can lessen data transmission by arrange and enhance MapReduce join execution.

JOIN is one of the as often as possible utilized operations to database and requires much asset. While performing JOIN on a circulated record framework, it is sensible to execute jobs on a few nodes. On the off chance that information is not circulated in these nodes, information changed from different node through system is essential and will influence the execution time of JOIN. The plan of the proposed technique, CA_Sqoop, is to first break down the log to know which tables are much of the time utilized for JOIN. At that point create TCS and circulate above tables on a similar node if conceivable. Simulation comes about demonstrate that CA_Sqoop can move forward the information area in all situations notwithstanding bringing in 300 tables to the conveyed document framework. With CA_Sqoop, the time of information change and occupation execution can be fundamentally progressed.

2.3 Differences: Sqoop1 & Sqoop2

Feature	Sqoop1	Sqoop2
Connectors for all major RDBMS	Supported.	Not supported. Workaround: Use the non specific JDBC Connector which has been tried on the accompanying databases: Microsoft SQL Server, PostgreSQL, MySQL and Oracle. This connector ought to take a shot at some other JDBC agreeable database. Execution is equivalent to that of particular connectors in Sqoop.
Kerberos Security Integration	Supported.	Supported.
Data transfer from RDBMS to Hive or HBase	Supported.	Not supported. Workaround: Follow this two-stage approach: 1. Import information from RDBMS into HDFS 2. Stack information into Hive or HBase utilizing suitable instruments, for example, the LOAD DATA statement in Hive.

Data transfer from Hive or HBase to RDBMS	Not supported. Workaround: follow this two-stage approach: 1. Extract data from Hive or HBase into HDFS (either as a text or Avro file) 2. Use Sqoop to export output of previous step to RDBMS	Not supported. Same approach as Sqoop1.

Table 2-2 Comparison of Sqoop1 & Sqoop2

Apache Sqoop utilizes a customer model where the client needs to the introduce Sqoop alongside connectors/drivers on the customer. Sqoop2 (the above rendition of Sqoop) uses an administration based model, where the connectors/drivers are introduced on the Sqoop2 server. Likewise, every one of the designs should be done on the Sqoop2 server.

From a MR point of view another distinction is that Sqoop presents a Map just employment, while Sqoop2 presents a MapReduce work where the Mapper would be transporting the information from the source, while the Reducers would be changing the information as indicated by the source determined. This gives a spotless deliberation. In Sqoop, both the migration and the changes were given by Mapper as it were.

Another significant contrast in Sqoop2 is from a security point of view. The chairman would be setting up the associations with the source and the objectives, while the administrator client utilizes the officially settled associations, so the administrator client require not know the insights about the associations. Also, administrators will be offered access to just a portion of the connectors as required.

Alongside the continuation of the CLI, Web UI can likewise be utilized with Sqoop2. The CLI and the Web UI expend the REST administrations given by the Sqoop Server. One thing to note is that the Web UI is a piece of (HUE-1214) and not some portion of the ASF. The Sqoop2 REST interface likewise makes it simple to coordinate with different systems like Oozie to characterize a work process including Sqoop2.

So in CA_Sqoop and Sqoop2, few changes are made and tune some of the argument for better performance. But in Sqoop1 we have to tune all the argument according to the data and framework.

3 Importing Data

The Sqoop execution may fluctuate in view of individual situations and different parameters notwithstanding when you utilize similar data. When importing data using sqoop, we will tune the parameter which will give the better performance or may it perform worst than the original sqoop importing argument setting.

3.1 Basic import

Using basic import tool we have to import a relational database table records to the HDFS. Importing a single table is very easy to command, because you have to provide only the import command, specify the database related credential, and table which will import and the connector.

For the help of sqoop import tool type “sqoop import --help” in terminal:

```
17/06/20 11:21:35 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
usage: sqoop import [GENERIC-ARGS] [TOOL-ARGS]

Common arguments:
   --connect <jdbc-uri>                             Specify JDBC connect
                                                    string
   --connection-manager <class-name>                Specify connection manager
                                                    class name
   --connection-param-file <properties-file>        Specify connection
                                                    parameters file
   --driver <class-name>                            Manually specify JDBC
                                                    driver class to use
   --hadoop-home <hdir>                             Override
                                                    $HADOOP_MAPRED_HOME_ARG
   --hadoop-mapred-home <dir>                       Override
                                                    $HADOOP_MAPRED_HOME_ARG
   --help                                           Print usage instructions
-P                                                  Read password from console
   --password <password>                            Set authentication
                                                    password
   --password-alias <password-alias>                Credential provider
                                                    password alias
   --password-file <password-file>                  Set authentication
                                                    password file path
   --relaxed-isolation                              Use read-uncommitted
                                                    isolation for imports
   --skip-dist-cache                                Skip copying jars to
                                                    distributed cache
   --username <username>                            Set authentication
                                                    username
   --verbose                                        Print more information
                                                    while working
```

Figure 3-1 Sqoop import help

```
pranav@pranav:~$ sqoop import --connect jdbc:mysql://localhost/mysql --username
root --password need --table user -m 1
```

Figure 3-2 Basic import command

The primary parameter after the sqoop executable is import, which determines the fitting apparatus. The import apparatus is utilized when you need to exchange data from the relational database into Hadoop. The parameter also contains the JDBC URL to your database. The punctuation of the URL is particular for every database, so you have to counsel your DB manual for the correct organization.

The URL is trailed by two parameters, -- username what's more, - -password word, which are the certifications that Sqoop ought to utilize while interfacing with the database.

At last, the last parameter, -- table, contains the name of the table to import.

3.1.1 Importing subset of data

Rather than bringing in a whole table, you have to exchange just a subset of the columns based on different conditions that you can express as a SQL proclamation with a WHERE condition.

Utilize the summon line parameter – where to indicate a SQL condition that the imported data ought to meet. Sqoop will engender the substance of the -- where parameter as is to all produced inquiries that get information. This gives a capable capacity by which to express any condition that your specific database server can handle. Any uncommon capacities, transformations, or even client characterized capacities can be utilized. Since the SQL section will be proliferated into created queries with no Sqoop preparing, any invalid parts may bring about non-intuitive exemptions that are difficult to investigate. This parameter can be mistaking for new Sqoop clients.

```
pranav@pranav:~$ sqoop import --connect jdbc://localhost/mysql --username root
--password need --table cities --where "country='USA'"
```

Figure 3-3 importing subset of data

When utilizing the -- where parameter, remember the parallel way of Sqoop exchanges. Data will be moved in a few simultaneous undertakings. Any costly capacity call will put a noteworthy execution load on your database server. Propelled capacities could bolt certain tables, keeping Sqoop from moving information in parallel. This will antagonistically influence exchange execution. For effective progressed separating, run the sifting enquiry on your database preceding import, spare its yield to a transitory table and run Sqoop to import the transitory table into Hadoop without the -- where parameter.

3.2 Compressing imported data

You need to diminish the general size involved on HDFS by utilizing compression for created documents.

```
pranav@pranav:~$ sqoop import --connect jdbc://localhost/mysql --username root
--password need --table cities --compress
```

Figure 3-4 Enable compression

When you configure the compress argument, you can compress the data around by 60% and diminish the measure of circle space required in the objective. You can design compress when the objective storage is constrained.

As Sqoop representatives compress to the MapReduce motor, you require to ensure the compress guide yield is permitted in your Hadoop setup. For instance, if in the mapred-site.xml record, the property mapred.output.compress is set to false with the last banner, at that point Sqoop won't have the capacity to compress the yield records notwithstanding when you call it with the –compress parameter.

The result shows that performance impact after compress argument configuration is not good. Without compress argument configuration, sqoop gives 35% better performance against the configured compress import tool.

So to reduce the execution time, **do not configure the compress argument.**

3.3 Speeding up transfer using "--direct"

Specifies the --direct, when you are using the MySQL data store. Instead of utilizing the JDBC interface for exchanging information, the immediate mode delegates the occupation of exchanging information to the local utilities given by the database seller. In the instance of MySQL, the mysqldump furthermore, mysqlimport will be utilized for recovering information from the database server or moving information back. On account of PostgreSQL, Sqoop will exploit the pg_dump utility to import information.

```
pranav@pranav:~$ sqoop import --connect jdbc://localhost/mysql --username root
--password need --table cities --direct
```

Figure 3-5 –direct for speeding up transfer

Utilizing local utilities will extraordinarily enhance execution, as they are streamlined to give the most ideal exchange speed while putting less weight on the database server. There are a few confinements that come with this speedier import. For one, not all databases have accessible local utilities. This mode is not accessible for each upheld database. Out of the crate, Sqoop has coordinate bolster just for MySQL and PostgreSQL.

After –direct configuration sqoop gives better performance than the usual sqoop import. But –direct connector available only for few databases management software.

3.4 Custom Boundary Queries

You observed freestyle question import to be extremely valuable for your utilization case. Shockingly, preceding beginning any information move in MapReduce, Sqoop sets aside a long opportunity to recover the minimum and maximum extreme estimations of the value determined in the --split-by parameter that are required for breaking the data into various autonomous assignments

```
pranav@pranav:~$ sqoop import --connect jdbc://localhost/mysql --username root --password need \
> query "select normcities.id, countries.country, normcities.city form normcities \
> join countries using (country_id)\
> where $condition \
> --split-by id \
> --boundary-query "select mid(id), max(id) from normcities"
```

Figure 3-6 Boundary query

Keeping in mind the end goal to parcel information into numerous free cuts that will be moved in a parallel way, Sqoop needs to locate the base and greatest estimation of the column determined in the -- split-by parameter. In a table-based import, Sqoop utilizes the table's essential key as a matter of course and produces the inquiry select min(col), max(col) from tbl (for table tbl what's more, part column col). On account of the freestyle inquiry import, there is no table that Sqoop can use for getting those qualities; rather, it will utilize the whole question determined on the order line as a sub query set up of the table name, coming about in a question "select min(col), max(col) from ($YOUR_QUERY) ". Such a question is profoundly wasteful, as it requires emergence of the yield result set before moving any information only with the end goal of getting the import limits.

Without understanding your question and the hidden information, there aren't numerous improvements that Sqoop can naturally apply. Sqoop offers the parameter – boundary-query, with which a custom inquiry can abrogate the produced question. The requirement for this inquiry is to return precisely one line with precisely two columns. The principal column will be viewed as the lower bound, while the second column will be the upper bound. Both esteems are comprehensive and will be transported in. The kind of both columns must be the same as the kind of the column utilized as a part of the – split-by parameter. Knowing your information and the reason for your question enables you to effortlessly distinguish the fundamental table, if there is one, and select the boundary from this table with no extra join or data transformation.

By using –boundary-query parameter configuration, we retrieve only the information we need at the Hadoop cluster. So it will also give better performance in import operation.

3.5 Configure –fetch-size

Using this parameter we can specify that how many entries sqoop can import at a time.

```
pranav@pranav:~$ sqoop import --connect jdbc://localhost/mysql --username root --password need --table
cities --fetch-size=10
```

Figure 3-7 –fetch-size of import

You can expand the estimation of the fetch size contention in light of the volume of data that you need to peruse. Set the value in light of the accessible memory and data transfer capacity.

The default size of –fetch-size parameter is 1000. So you can update it according to the network parameter or memory.

According to the simulation result it also gives some better performance against the ordinary sqoop import tool.

3.6 Enable primary key

When you import data you have to enable the primary key if it is not enabled. Reading data from the primary key enabled table is gives better performance than non-primary key relation. So we can enable the primary key using the following syntax:

“alter table <table_name> add constraint <table_name_pk primary key {ID}>;”

Keep in mind that you have to remove the primary key or foreign key constrain before you insert data into the table.

3.7 Controlling parallelism

Sqoop as a matter of course uses four concurrent map tasks to exchange data to Hadoop. Exchanging greater tables with more concurrent tasks ought to diminish the time require to exchange all information. You need the adaptability to change the quantity of guide undertakings utilized on a pre-job basis.

```
pranav@pranav:~$ sqoop import --connect jdbc://localhost/mysql --username root
 --password need --table cities --num-mappers 10
```

Figure 3-8 controlling parallelism using mapper

The parameter --num-mappers fills in as an insight. Much of the time, you will get the predetermined number of mappers, yet it's not ensured. On the off chance that your informational collection is little, Sqoop may turn to utilizing fewer mappers. For instance, in case you're exchanging just 4 pushes yet set --num-mappers to 10 mappers, just 4 mappers will be utilized, as the other 6 mappers would not have any information to exchange.

Controlling the measure of parallelism that Sqoop will use to exchange information is the fundamental approach to control the heap on your database. Utilizing more mappers will prompt a higher number of simultaneous information exchange assignments, which can bring about quicker occupation fulfillment.

Be that as it may, it will likewise build the heap on the database as Sqoop will execute more simultaneous inquiries. Doing as such may influence different questions running on your server, unfavorably influencing your generation condition. Expanding the quantity of mappers won't generally prompt quicker occupation consummation. While expanding the quantity of mappers, there is a time when you will completely immerse your database. Expanding the quantity of

mappers past this indicates won't lead quicker occupation finish; indeed, it will have the inverse impact as your database server invests more energy doing setting exchanging instead of serving information.

The ideal number of mappers relies on upon numerous factors: you have to consider your database sort, the equipment that is utilized for your database server, and the effect to different solicitations that your database needs to serve. There is no ideal number of mappers that works for all situations. Rather, you're urged to trial to locate the ideal level of parallelism for your condition and utilize case. It's a smart thought to begin with few mappers, gradually increase, as opposed to begin with an expansive number of mappers, working your way down.

To configure the –num-mappers argument, performance will vary according to the database manager you use. Using MySQL server in the Linux operating system, 10 mappers gives better performance than the usual 4 mapper.

3.8 Split data for parallel task

You have to import one fundamental table; notwithstanding, this table is standardized. The essential qualities are put away in the referenced word reference tables, and the fundamental table contains just numeric foreign keys indicating the qualities in the lexicons instead of to characteristic keys as in the first reference dictionary table. You would want to determine the qualities preceding running Sqoop and import the genuine esteems instead of the numerical keys for the countries.

```
pranav@pranav:~$ sqoop import --connect jdbc://localhost/mysql --username root --password need \
> query "select normcities.id, countries.country, normcities.city form normcities \
> join countries using (country_id)\
> where $condition \
> --split-by id \
```

Figure 3-9 split data for parallel task

Rather than utilizing table import, utilize freestyle query import. In this mode, Sqoop will enable you to determine any inquiry for bringing in information. Rather than the parameter --table, use the parameter --query with the whole inquiry for getting the information you might want to exchange.

There is a great deal to know about when utilizing freestyle query imports. By utilizing query imports, Sqoop can't utilize the database inventory to bring the metadata. This is one of the reasons why utilizing table import may be speedier than the equal freestyle query import. Likewise, you need to physically determine some extra parameters that would generally be populated naturally. Notwithstanding the –query parameter, you have to indicate the --split-by parameter with the segment that ought to be utilized for cutting your information into different parallel assignments. This parameter normally naturally defaults to the primary key of the main table.

If there is not any specific column name provided by you, than sqoop will split the table according to the primary key column.

3.9 Analysis for importing data

By configure the different parameter of importing tool; we determine that performance of the sqoop will increase. But there is some argument which will not gives the better performance.

Following argument gives the better performance when you configure them:

- Import subset of data
- direct argument
- Custom Boundary Query
- Fetch size
- Enable primary key
- Controlling parallelism
- split data

Above arguments are gives different scale of performance with the different configuration values.

- Compress

This argument is fast the import operation but the manipulation of data took more time than without compressed data.

4 Exporting Data

You have a work process of different Hive and MapReduce occupations that are creating data on a Hadoop bunch. You have to exchange this information to your social database for simple queries.

Using the sqoop export tool, you can transfer the data back to the relational database management. Export works similar to the import tool, except export tool transfer the data in opposite direction.

```
pranav@pranav:~$ sqoop export --connect jdbc://localhost/mysql --username root
 --password need --table cities
```

Figure 4-1 exporting data back to RDBMS

For sqoop export help type "sqoop export --help" in terminal:

```
17/06/20 14:52:00 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
usage: sqoop export [GENERIC-ARGS] [TOOL-ARGS]

Common arguments:
   --connect <jdbc-uri>                             Specify JDBC connect
                                                    string
   --connection-manager <class-name>                Specify connection manager
                                                    class name
   --connection-param-file <properties-file>        Specify connection
                                                    parameters file
   --driver <class-name>                            Manually specify JDBC
                                                    driver class to use
   --hadoop-home <hdir>                             Override
                                                    $HADOOP_MAPRED_HOME_ARG
   --hadoop-mapred-home <dir>                       Override
                                                    $HADOOP_MAPRED_HOME_ARG
   --help                                           Print usage instructions
-P                                                  Read password from console
   --password <password>                            Set authentication
                                                    password
   --password-alias <password-alias>                Credential provider
                                                    password alias
   --password-file <password-file>                  Set authentication
                                                    password file path
   --relaxed-isolation                              Use read-uncommitted
                                                    isolation for imports
   --skip-dist-cache                                Skip copying jars to
                                                    distributed cache
   --username <username>                            Set authentication
                                                    username
   --verbose                                        Print more information
                                                    while working
```

Figure 4-2 sqoop export help

Rather than exchanging information from the social database utilizing SELECT query, Sqoop will exchange the information to the social database utilizing INSERT query. Sqoop fare work process coordinates the import case with slight contrasts. After you execute the Sqoop command, Sqoop will associate with your database to get different metadata about your table, including the rundown of all columns with their fitting types. Utilizing this metadata, Sqoop will produce and order the Java class. The produced class will be utilized as a part of the submitted MapReduce work that will trade your information. Like the import mode, no information is being exchanged through the Sqoop client itself. All moves are done in the MapReduce work, with Sqoop managing the procedure from your system.

Sqoop gets the table's metadata in the fare: the goal table (determined with the --table parameter) must exist preceding running Sqoop. The table does not need to be vacant, and you can even fare new information from Hadoop to your database on an iterative premise. The main necessity is that there not is any requirement infringement when playing out the INSERT articulations (for instance, you can twice send out a similar esteem for any essential or one of a kind key).

4.1 Inserting data in Batches

While Sqoop's fare highlight fits your necessities, it's too moderate. It appears that each column is embedded in a different INSERT statement. Custom fitted for different databases and utilize cases, Sqoop offers numerous choices for inserting more than one row at any given moment.

```
pranav@pranav:~$ sqoop export --connect jdbc://localhost/mysql --username root -
-password need --table cities --batch
```

Figure 4-3 –batch argument

Using enable the batch argument of JDBC connection it will give better performance than the usual transfer of the sqoop. Simulation result of the –batch argument shows that after enabling the batch, exporting of data is very fast than ordinary transfer.

The JDBC interface uncovered an API for doing bunches in a readied proclamation with different arrangements of qualities. With the –batch parameter, Sqoop can exploit this. This API is available in all JDBC drivers since it is required by the JDBC interface. The usage may change from database to database. Though some database drivers utilize the capacity to send different lines to remote databases inside one demand to accomplish better execution, others may essentially send each inquiry independently. A few drivers cause far and away more terrible execution when running in clump mode because of the additional overhead presented by serializing the column in inward reserves before sending it push by line to the database server.

Result shows that 98% fast exporting of data back to the relational database than the normal JDBC transfer.

4.2 Record per statement

Another property of export tool is specifying the number of record that will be used in each insert statement.

```
pranav@pranav:~$ sqoop export --Dsqoop.export.records.per.statement=10 --connect
 jdbc://localhost/mysql --username root --password need --table cities
```

Figure 4-4 records per statement

The second method of batching multiple rows into the same query is by specifying multiple rows inside one single insert statement. When setting the property Dsqoop.export.records.per.statement to a value of two or more, Sqoop will create the following query:

"INSERT INTO table VALUES (...), (...), (...)..."

As the queries are totally produced by Sqoop, the JDBC driver doesn't adjust it, finishing it to the remote database as may be. Lamentably, not all databases support different columns in a solitary embed articulation. Basic social databases like MySQL, Oracle, and PostgreSQL do support this; however a few information stockrooms may not. Here is likewise one extra disadvantage that you have to remember when utilizing expansive quantities of lines embedded with a solitary insert statement: most databases have confines on the greatest query measure. The Sqoop fare will come up short if the remote database server does not acknowledge the produced queries.

The other batching component improves execution by assembling various rows as the past two choices did. The insert determined in Dsqoop.export.statements.per.transaction decides what number of insert statement will be issued on the database preceding submitting the exchange and beginning another one.

Higher estimations of this property prompt longer-lived exchanges and evacuate the overhead presented by making and completing the exchange. Utilizing higher esteems for the most part enhances execution. Notwithstanding, the correct conduct relies on upon the basic database and its usefulness. In the event that your database requires an uncommon table-level compose bolt to insert lines into a table, utilizing a higher incentive for explanations per exchange may prompt fundamentally diminished execution.

This is also a kind of batch transformation where user will decide the number of rows for the insert statement, and this will also gives the better performance.

4.3 Updating existing data set

You have detained your database from a past export, yet now you have to engender refreshes from Hadoop. Sadly, you can't utilize the refresh mode, as you have an impressive number of new columns and you have to export them also.

You can exploit the refresh highlight that will issue UPDATE rather than INSERT articulations. The refresh mode is enacted by utilizing the parameter –UPDATE-key that contains the name of a section that can recognize a changed row--usually often than not the essential key of a table.

```
pranav@pranav:~$ sqoop export --connect jdbc://localhost/mysql --username root -
-password need --table cities --update-key id
```

Figure 4-5 updating existing data set

The parameter –update-key is utilized to educate Sqoop to update existing lines rather than insert new ones. This parameter requires a comma-isolated rundown of sections that ought to be utilized to particularly distinguish a column. Those sections will be utilized as a part of the WHERE clause of the created UPDATE query. All other table sections will be utilized as a part of the SET part of the query.

It's critical to comprehend the structure of the queries to perceive how the update mode will send out data from Hadoop. As a matter of first importance, the sections used to recognize the column will never be updated on the grounds that they are not some portion of the SET statement. Likewise, if your data in Hadoop contains some new columns, the "WHERE" clause won't coordinate any columns on the database side. Such an operation on the database side is completely legitimate, yet it brings about no update columns. In this manner, new columns are not sent out in update mode by any means.

4.4 Analysis of exporting data

Configuring the export tool parameter, transfer data from HDFS to relational database management, performance is very good than usual transfer.

- Inserting data in batches
- Records per statement
- Update existing data

These three configurations of export tool will give better performance for the different parameter values. Performance may decrease when the parameter values are not properly configured.

Mostly all the above export tool parameter gives above 90% better performance than usual transformation in the best configuration.

5 Implementation

In this chapter I discuss the technical implementation of sqoop tool and the result of all the tuned argument performance. In this chapter all the import and export tool argument are tuned and some of them performed very good than ordinary tool.

5.1 Technical requirement

Basic requirement of the implementation of sqoop is below:

1. Linux operating system
2. Java development kit (JDK 6) or greater
3. Secure Shell (SSH)
4. Hadoop 1.0 or greater
5. MySQL server
6. Sqoop 1.0 or greater

For the setup of sqoop we have to change the ~/.bashrc file, which is a shell script that Bash runs whenever it is started interactively.

In this file we have to setup the home path of JAVA, HADOOP and SQOOP.

```
#JAVA_HOME  directory setup
export JAVA_HOME="/usr/lib/jvm/java-7-openjdk-amd64"
set PATH="$PATH:$JAVA_HOME/bin"

#HADOOP_HOME directory setup
export HADOOP_HOME="/usr/local/hadoop"
PATH="$PATH:$HADOOP_HOME/bin"

#SQOOP_HOME directory setup
export SQOOP_HOME="/usr/local/sqoop"
PATH="$PATH:$SQOOP_HOME/bin"

export PATH
```

Figure 5-1 Bashrc file update

So after these implementation first of all we have to check that setup is successfully installed or not.

For check the Hadoop and Sqoop installation, check the version of both software using command line argument version.

```
pranav@pranav:~$ hadoop version
Hadoop 1.2.1
Subversion https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r 1503152
Compiled by mattf on Mon Jul 22 15:23:09 PDT 2013
From source with checksum 6923c86528809c4e7e6f493b6b413a9a
This command was run using /usr/local/hadoop/hadoop-core-1.2.1.jar
```

Figure 5-2 Hadoop version

```
pranav@pranav:~$ sqoop version
17/05/25 17:57:26 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
Sqoop 1.4.6
git commit id c0c5a81723759fa575844a0a1eae8f510fa32c25
Compiled by root on Mon Apr 27 14:26:52 CST 2015
```

Figure 5-3 Sqoop version

After checking the installation of Hadoop and sqoop, start the secure shell for the secure transfer of data and run Hadoop application in the safe mode.

5.2 Starting secure shell

To start the secure shell on your local machine, type “ssh localhost” in the terminal.

```
pranav@pranav:~$ ssh localhost
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.13.0-117-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

0 packages can be updated.
0 updates are security updates.

New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Your Hardware Enablement Stack (HWE) is supported until April 2017.

This Ubuntu 12.04 LTS system is past its End of Life, and is no longer
receiving security updates.  To protect the integrity of this system, it's
critical that you enable Extended Security Maintenance updates:
 * https://www.ubuntu.com/esm

Last login: Thu May 25 17:57:33 2017 from localhost
```

Figure 5-4 start secure shell

Secure Shell (SSH) is a cryptographic system convention for working system benefits safely over an unsecured network. The best known case application is for remote login to PC frameworks by clients. SSH gives a safe channel over an unsecured system in a customer server engineering, associating a SSH customer application with a SSH server. Common applications

incorporate remote charge line login and remote order execution, yet any system administration can be secured with SSH. The convention detail recognizes two noteworthy renditions, alluded to as SSH-1 and SSH-2.

5.3 Starting Hadoop

After establish the secure shell, you have to start all the node and tracker of Hadoop using the command line “start-all.sh”.

```
pranav@pranav:~$ start-all.sh
starting namenode, logging to /usr/local/hadoop/libexec/../logs/hadoop-pranav-na
menode-pranav.out
localhost: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoo
p-pranav-datanode-pranav.out
localhost: starting secondarynamenode, logging to /usr/local/hadoop/libexec/../l
ogs/hadoop-pranav-secondarynamenode-pranav.out
starting jobtracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-pranav-
jobtracker-pranav.out
localhost: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/ha
doop-pranav-tasktracker-pranav.out
```

Figure 5-5 starting Hadoop nodes and trackers

The NameNode is the centerpiece of a HDFS document framework. It keeps the registry tree of all documents in the record framework, and tracks where over the bunch the record information is kept. It doesn't store the information of these documents itself.

Customer applications converse with the NameNode at whatever point they wish to find a record, or when they need to add/copy/move/delete a document. The NameNode reacts the fruitful demands by restoring a rundown of pertinent DataNode servers where the data lives.

The NameNode is a Single Point of Failure for the HDFS Cluster. HDFS is not at present a High Availability framework. At the point when the NameNode goes down, the record framework goes disconnected. There is a discretionary SecondaryNameNode that can be facilitated on a different machine

A TaskTracker is a hub in the group that acknowledges tasks - Map, Reduce and Shuffle operations - from a JobTracker.

Each TaskTracker is arranged with an arrangement of openings, these show the quantity of errands that it can acknowledge. At the point when the JobTracker tries to discover some place to plan an undertaking inside the MapReduce operations, it initially searches for a vacant space on a similar server that has the DataNode containing the information, and if not, it searches for a vacant opening on a machine in a similar rack.

The JobTracker is the administration inside Hadoop that ranches out MapReduce undertakings to particular cluster in the bunch, in a perfect world the node that has the data, or possibly is in a similar rack.

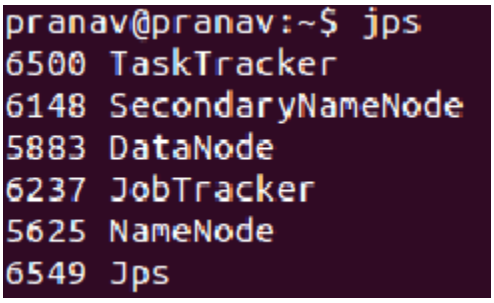

Figure 5-6 jps status

To check that all the trackers and nodes are started or not, type “jps” in the terminal. It will show all the process id of the jobs running by the Hadoop application.

The jps tool records the instrumented Hotspot Java Virtual Machines (JVMs) on the objective framework. The instrument is constrained to revealing data on JVMs for which it has the get to consents.

On the off chance that jps is keep running without determining a hostid, it will search for instrumented JVMs on the neighborhood have. In the event that begun with a hostid, it will search for JVMs on the demonstrated host, utilizing the predetermined convention and port. A jstatd procedure is thought to be running on the objective host.

The jps order will report the nearby VM identifier, or lvmid, for each instrumented JVM found on the objective framework. The lvmid is commonly, however not really, the working framework's procedure identifier for the JVM procedure. Without any alternatives, jps will list every Java application's lvmid trailed by the short type of the application's class name or jug record name. The short type of the class name or JAR document name overlooks the class' bundle data or the JAR records way data.

The jps command utilizes the java launcher to discover the class name and contentions gone to the fundamental strategy. In the event that the objective JVM is begun with a custom launcher, the class name (or JAR record name) and the contentions to the primary strategy won't be accessible. For this situation, the jps command will yield the string Unknown for the class name or JAR record name and for the contentions to the principle technique.

The rundown of JVMs delivered by the jps command might be constrained by the consents conceded to the primary running the summon. The summon will just rundown the JVMs for which the standard approaches rights as controlled by working framework particular get to control instruments.

After all this you can use the sqoop import and export tool for data exchange between HDFS and relational database.

6 Result and Analysis

The complete experiment setup is explained in previous chapter. In this chapter, results are captured and analysis of those results is explained. Results are taken on the single node Hadoop machine. According the results, comparative analysis with the ordinary tool is also described.

6.1 Experiment execution

In this experiment, sqoop import process is described. Several steps are following in the process of importing the data.

In the first step of the execution, import query is process according to the syntax of import command. Sqoop will look at the configuration files of HBase, Hcatalog and acuumulo installation. If the configuration is not done yet, then it will give the warning message to set the HOME environment of these files.

After that MySQL manager preparing the MySQL streaming result set for the code generation. In code generation, MySQL manager will execute the SQL statement.

“Executing SQL statement: SELECT t.* FROM `cities` AS t LIMIT 1 “

Using this Sql statement it will get the MIN & MAX value of the primary key. MapReduce Job client will assign a unique number for the executing the job and then start the mapping and reducing. After job complete it will shows all the results.

Sqoop imports data in parallel and you utilize the –m contention to indicate the level of parallelism, the default esteem is 4. When bringing in information in parallel, a criterion to part the workload is required. The essential key is recognized and utilized as a matter of course. For instance, in the event that we have an essential with least estimation of 1 and a most extreme estimation of 40000 and we have set a parallelism level of 4. Sqoop will run 4 errands each working on a scope of information. One process will work on the information in 1 to 10000 territories, another procedure will work on the 10001 to 20000 territory, another procedure will work on 20001 to 30000 and another procedure will work on the 30001 to 40000 territory.

On the off chance that you import data once and you just need to import new information since the last import Sqoop gives two approaches to do as such. The add mode is utilized when there is an incremental option of lines. A segment that contains the line id is determined utilizing the –check-segment contention. This section is inspected and just columns with a value more noteworthy than --last-value are foreign made.

When you might want to catch changes that outcome in a refresh where update are caught in a timestamp you utilize the last modified mode. You determine the section containing the timestamp as --check-column and just columns with a value later than --last-value are transported in. As a matter of course information is foreign made as delimited content.

```
pranav@pranav:~$ sqoop import --connect jdbc:mysql://localhost/mysql --username
root --password need --table cities
Warning: /usr/local/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/local/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/../accumulo does not exist! Accumulo imports will fail
.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop/../zookeeper does not exist! Accumulo imports will fai
l.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
17/06/20 12:41:36 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
17/06/20 12:41:36 WARN tool.BaseSqoopTool: Setting your password on the command-
line is insecure. Consider using -P instead.
17/06/20 12:41:37 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
17/06/20 12:41:37 INFO tool.CodeGenTool: Beginning code generation
17/06/20 12:41:39 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `cities` AS t LIMIT 1
17/06/20 12:41:39 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `cities` AS t LIMIT 1
17/06/20 12:41:39 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/
hadoop
Note: /tmp/sqoop-pranav/compile/dc4c8317e744b1604790eada97b7840d/cities.java use
s or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
17/06/20 12:41:46 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-pran
av/compile/dc4c8317e744b1604790eada97b7840d/cities.jar
17/06/20 12:41:46 WARN manager.MySQLManager: It looks like you are importing fro
m mysql.
17/06/20 12:41:46 WARN manager.MySQLManager: This transfer can be faster! Use th
e --direct
17/06/20 12:41:46 WARN manager.MySQLManager: option to exercise a MySQL-specific
 fast path.
17/06/20 12:41:46 INFO manager.MySQLManager: Setting zero DATETIME behavior to c
onvertToNull (mysql)
17/06/20 12:41:46 INFO mapreduce.ImportJobBase: Beginning import of cities
17/06/20 12:41:52 INFO db.DBInputFormat: Using read commited transaction isolati
on
```

Figure 6-1 importing data-I

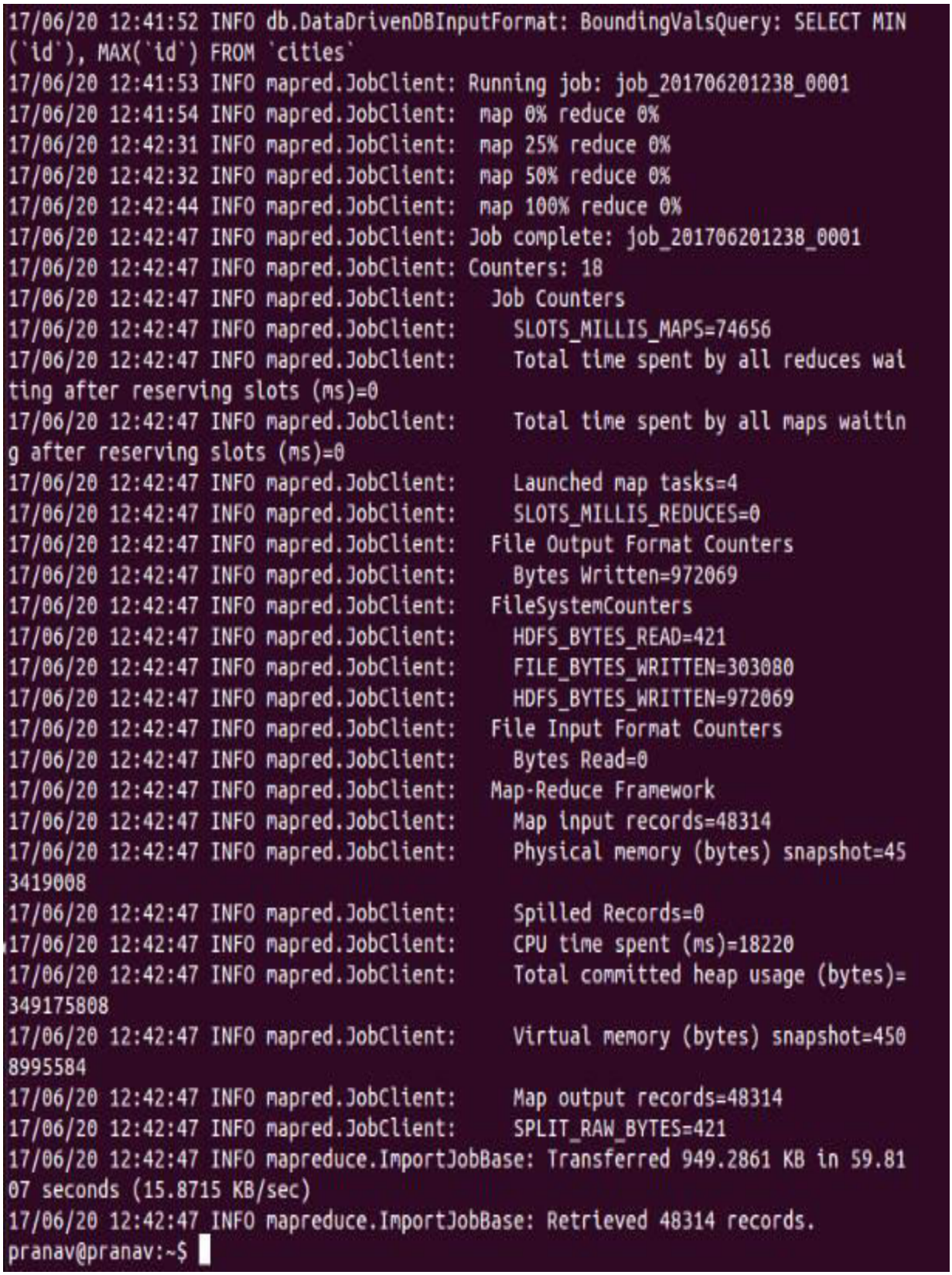

Figure 6-2 Importing data-II

6.2 Experiment Results

6.2.1 Compress argument

Using the –compress argument configuration experiment result shows that it will perform 42 % faster than without configure the compress argument. Using the –compress argument importing process will give better performance but when it imported to HDFS then performing operation on that data will be took more time than non-compressed data.

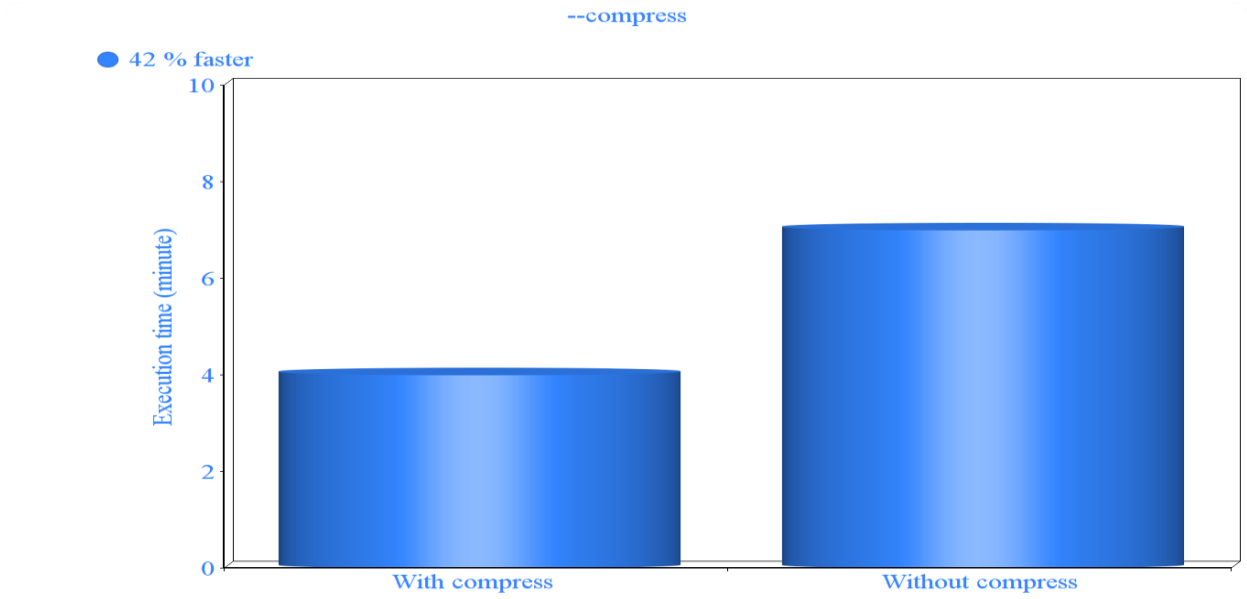

Figure 6-3 –compress argument result

6.2.2 Direct argument

Using the direct argument configuration, result show that it will give 50% better performance than without direct argument configuration.

MySQL Direct Connector enables quicker import and export to/from MySQL utilizing mysqldump and mysqlimport apparatuses usefulness rather than SQL chooses and embeds.

To utilize the MySQL Direct Connector, determine the --direct contention for your import or export work.

For execution, every essayist will submit the present exchange around each 32 MB of sent out information. You would control be able to this by indicating the accompanying contention before any apparatus particular contentions: - D sqoop.mysql.export.checkpoint.bytes=size, where measure is an incentive in bytes. Set size to 0 to disable middle of the checkpoints, however singular file being sent out will keep on being conferred freely of each other.

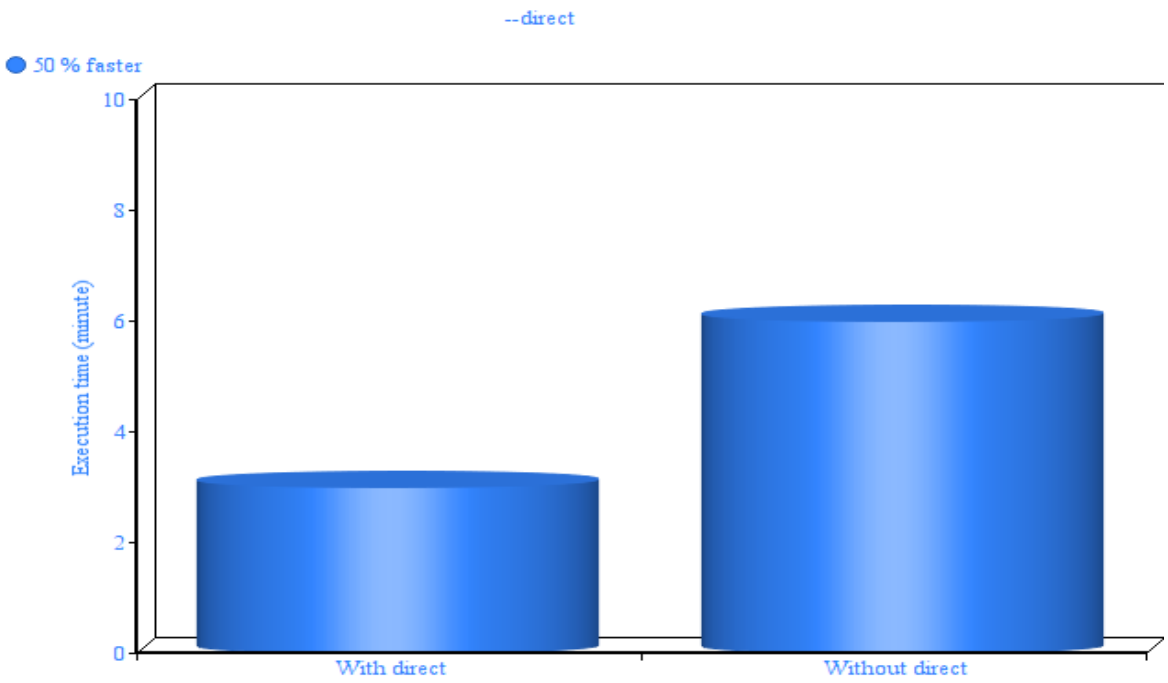

Figure 6-4 direct argument result

6.2.3 Record per statement (Batch mode)

As every strategy utilizes an alternate means for enhancing the fare execution, you can join every one of them together. Every database framework and client condition is distinctive. There aren't best practices that can be comprehensively connected over all utilization cases. Our proposal is to begin with empowering –batch import and determine the quantity of lines per proclamation to generally square with the greatest permitted question measure. From that beginning point, experiment with various values.

The JDBC interface uncovered an API for doing groups in a readied explanation with different arrangements of qualities. With the –batch parameter, Sqoop can exploit this. This API is available in all JDBC drivers since it is required by the JDBC interface. The usage may fluctuate from database to database. Though some database drivers utilize the capacity to send numerous columns to remote databases inside one demand to accomplish better execution, others may just send each inquiry independently. A few drivers cause surprisingly more dreadful execution when running in group mode because of the additional overhead presented by serializing the column in inner stores before sending it push by line to the database server.

Result shows that when batch argument is configured, it will give a huge amount of better performance.

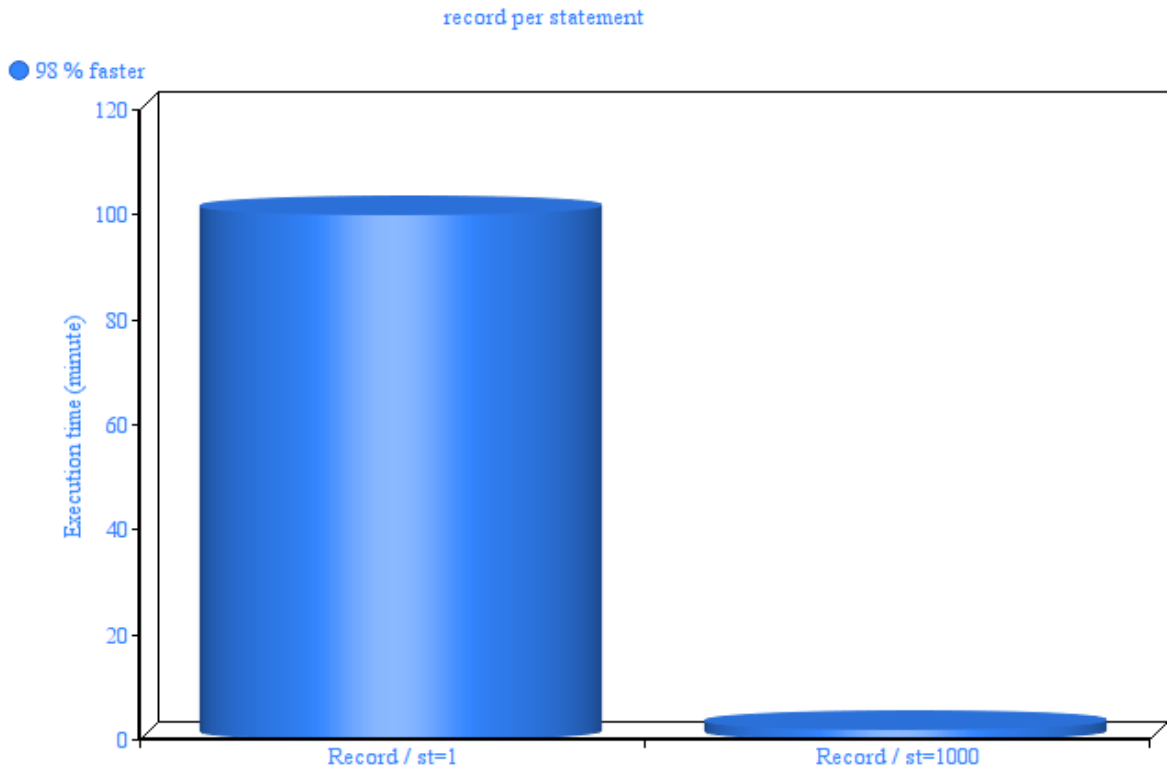

Figure 6-5 record per statement (batch mode) result

6.2.4 Primary key argument

You have to import one primary table; be that as it may, this table is standardized. The essential values are put away in the referenced lexicon tables, and the primary table contains just numeric remote keys indicating the qualities in the word references as opposed to normal keys as in the first urban communities table. You would want to determine the qualities preceding running Sqoop and import the genuine esteems as opposed to the numerical keys for the nations.

This will permit to pick precisely which column to split on, rather than letting Sqoop picking a default one, which will be enhanced.

While picking the column, Sqoop prescribes you not to take a String column (which could be tricky if your SQL server database sorts this column for a situation harsh way). You may likewise decide to use a column which is not a Primary Key. The imperative thing is to have this segment with an even circulation (you need to keep away from skewed Sqoop get assignments) and furthermore attempting to have each parts in a similar plate area in your source table (attempt to get consecutive peruses).

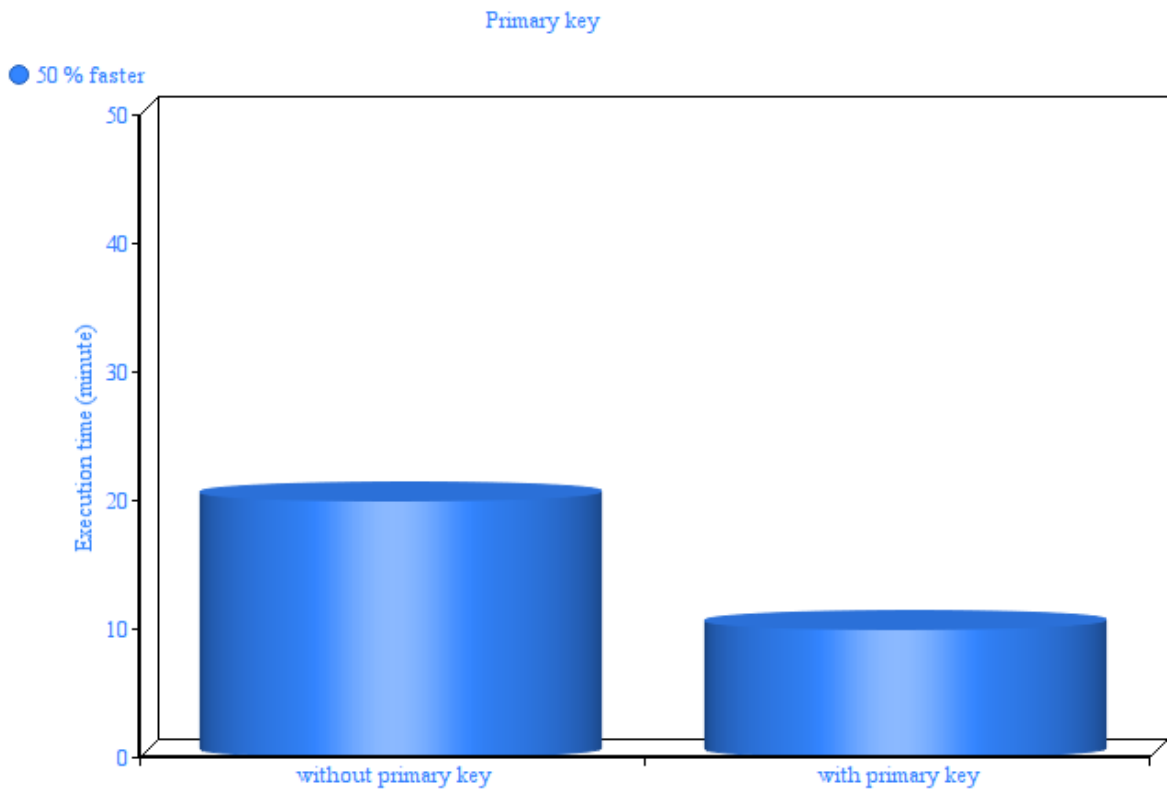

Figure 6-6 enable primary key result

Result show that when there is no primary key in the import process, it will take more time than importing with primary key.

6.2.5 Fetch size

When bringing in information from different RDBMSs, you will have discovered any issue simultaneously. This does not mean your Sqoop Import summon will work consummately constantly. At the point when the extent of brought data is little which can fit into designated memory you won't confront any issues. In any case, when gotten information comes to past the dispensed memory estimate, you will begin discovering issues. Your MapReduce work comes up short and tosses exemption "GC Overhead limit exceed".

With different size of fetch it will give different speed in importing and exporting. At the best it will give 25% faster than the usual fetch size in this experiment.

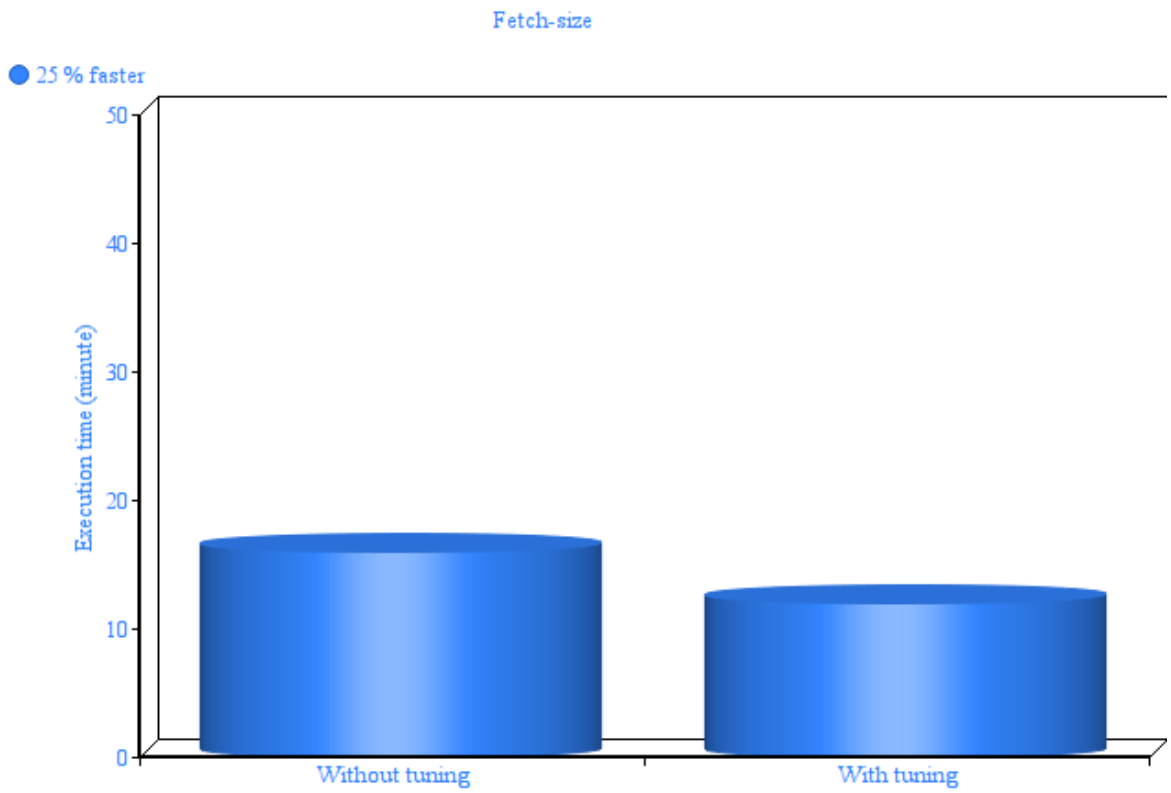

Figure 6-7 fetch size tuning result

6.2.6 Num-mapper argument

With tweaking Hadoop and sqoop parameters we have diminished the run time from 4 odd minutes to 1 minute. Presently this will spare a great deal of CPU and information exchange taken a toll on the more drawn out keep running with enormous information sums.

These circumstances would not remain constant consistently; the test group utilized here was not stacked with some other errands. It additionally relies on upon where on the system source and goal are, for this situation, everything was on AWS framework.

The ideal number of mappers relies on upon numerous factors: you have to consider your database sort, the equipment that is utilized for your database server, and the effect to different solicitations that your database needs to serve. There is no ideal number of mappers that works for all situations. Rather, you're urged to trial to locate the ideal level of parallelism for your condition and utilize case. It's a smart thought to begin with few mappers, gradually increase, as opposed to begin with countless, working your way down

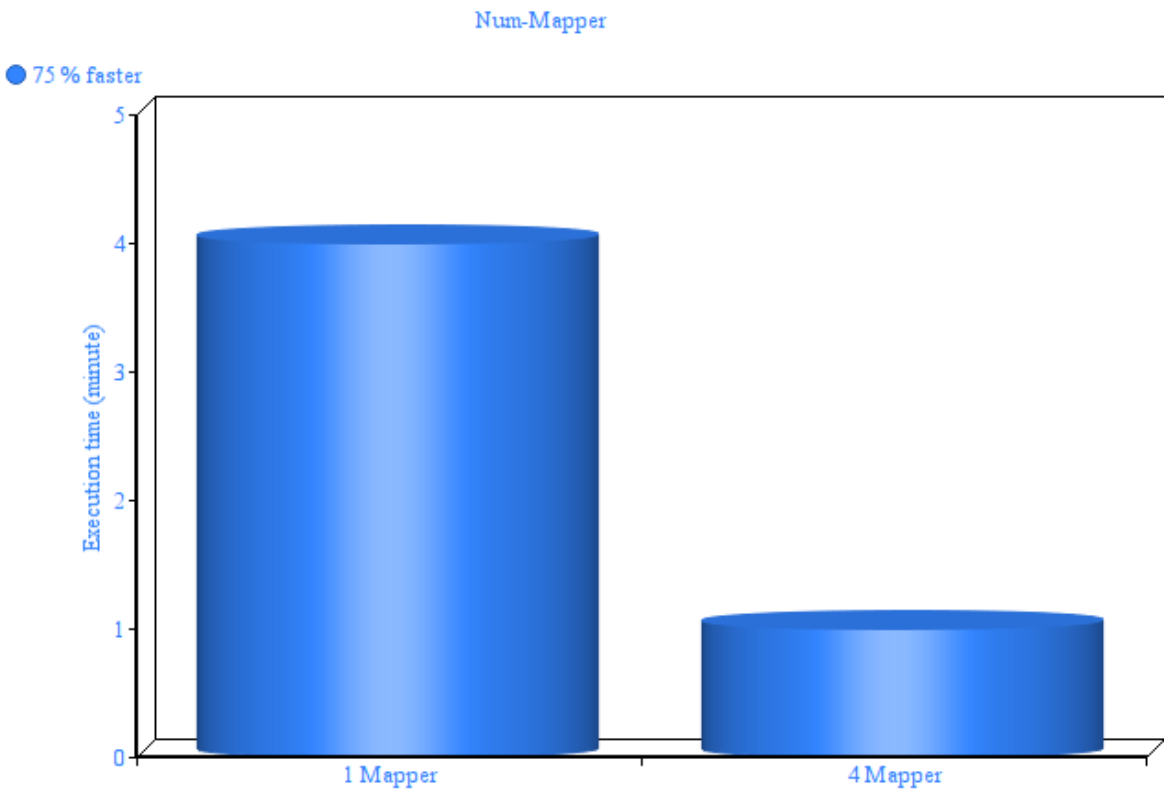

Figure 6-8 num-mapper argument result

In the experiment of variable number of mapper, it gives better performance with the 4 mapper. But when we increase the number of mapper performance was degrades.

In the table given below, the configured parameters are listed with their results respectively.

Parameter configured	**Results (% faster)**
Compress	42%
Direct	50%
Primary key	50%
Batch mode	98%
Fetch size	25%
Num-mapper	75%

Table 6-1 configured parameters and results

7 Conclusion and Future work

7.1 Conclusion

Exchanging data to and from relational databases is testing and arduous. Since information exchange requires cautious dealing with, Apache Sqoop, another way to say "SQL to Hadoop," was made to perform bidirectional information exchange amongst Hadoop and any outside organized data store. The basic design of parameter provides ability to transfer data between relational database and HDFS with the slow speed. To enhance the information position upgrades the execution as far as database utilizations. In order to analyze the factors influencing loading, I repeatedly experimented with various influencing factors.

It is expected that the study will be the basis of the research on how to improve the performance of the whole step of analyzing the formal data in the Lines and Hadoop environment. Previously proposed CA_Sqoop technique is gives better performance than basic sqoop tool. CA_Sqoop improve the data locality in the entire scenario. In this experiment, the parameters of sqoop tool have been configured and this show better results than the default configuration. Data transfer speed of the sqoop tool as now enhanced multiple folds.

7.2 Future work

The experiment has been performed on parameter of sqoop tool and results show that performance of sqoop data transfer is increased. So tuning of migration tool is fully configured and there is more scope in the performance tuning with different kind of database, better network speed and highly configured machines. Relational database frameworks frequently store significant information in an organization. In the event that made accessible, that information can be overseen and handled by Apache Hadoop, which is quick turning into the standard for huge information preparing. A few social database sellers championed creating incorporation with Hadoop inside at least one of their items.

In this experiment, you've perceived how Sqoop streamlines information exchanges amongst Hadoop and databases. Unmistakably it's a device streamlined for control clients. An order line interface giving 60 parameters is both capable and unsolved. So there are many parameters which will give better transfer speed when they will configure.

8 References

1. Masato Asahara, Shinji Nakadai and Takuya Araki, "LoadAtomizer: A Locality and I/O Load aware Task Scheduler for MapReduce, " in 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 317-324, 2012.

2. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt and Andrew Warfield, "Xen and the Art of Virtualization, " SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, vol. 37, Issue 5, pp. 164-177, December 2003.

3. Kasim Selcuk Candan, Jong Wook Kim, Parth Nagarkar, Mithila Nagendra and Ren-wei Yu, "Scalable Multimedia Data Processing in Server Clusters," IEEE Multi Media, pp. 3-5, 2010

4. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C., Hsieh Deborah A., Wallach Mike Burrws, Tushar Chandra, Andrew Fikes, and Robert E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," 7th UENIX Symposium on Operating Systems Design and Implementation, pp. 205-218, 2006.

5. Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters, " Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.

6. Sven Groot, "Jumbo: Beyond MapReduce for Workload Balancing," Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on Cloud Computing Technology and Science, vol. 4, pp. 2675-2678, 2011. July.

7. C. Jin and R. Buyya, "Mapreduce programming model for net-based cloud computing, "in Proceedings of the 15th International Euro-Par Conference on Parallel Processing, Euro-Par (Berlin, Heidelberg), pp. 417-428, 2009.

8. Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz and Ion Stoica, "Improving MapReduce Performance in Heterogeneous Environments, " 8th Symposium on Operating Systems Design and Implementation, pp. 29-42, 2008. Dec.

9. Jenq-Shiou Leu, Yun-Sun Yee, Wa-Lin Chen,"Comparison of Map-Reduce and SQL on Large-scale Data Processing," International Symposium on Parallel and Distributed Processing with Applications, pp. 244-248, 2010.

10. Hung-Ping Lin, "Structured Data Processing on MapReduce in NoSQL Database, "Master Thesis in National Chiao Tung University, 2010.

11. https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html,

12. https://en.wikipedia.org/wiki/Apache_Hadoop

13. https://community.hortonworks.com/articles/70258/sqoop-performance-tuning.html

14. Rubao Lee, Tian Luo, Yin Huai, Fusheng Wang, Yongqiang He, and Xiaodong Zhang, " YSmart: Yet Another SQL-to-MapReduce Translator, " International Conference on Distributed Computing Systems, pp. 25-36, 2011.

15. A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive - a warehousing solution over a Map-Reduce framework, "PVLDB, vol. 2, no. 2, pp. 1626-1629, 2009.