# **Table of Contents**

[Table of Contents 1](#_Toc481554624)

[1. Last sprint 7](#_Toc481554625)

[1.1 Data Ingestion 7](#_Toc481554626)

[1.1.1 Sqoop Import delimiter 8](#_Toc481554627)

[1.1.2 Sqoop Free From Query 8](#_Toc481554628)

[1.1.3 Sqoop Import Into Hive Table 9](#_Toc481554629)

[1.1.4 Sqoop Export 10](#_Toc481554630)

[1.1.5 Sqoop Export (update/Insert) 11](#_Toc481554631)

[1.1.6 Start Flume Agent 12](#_Toc481554632)

[1.2 Data Transformation 13](#_Toc481554633)

[1.2.1 Handling Null's 13](#_Toc481554634)

[1.2.2 Tokenize 13](#_Toc481554635)

[1.2.3 Start Flume Agent 14](#_Toc481554636)

[1.2.4 Edukerka 15](#_Toc481554637)

[1.2.5 Projections in PIG 18](#_Toc481554638)

[1.2.6 Data Items 18](#_Toc481554639)

[1.2.7 Example 1 20](#_Toc481554640)

[1.2.8 Write and Execute Pig Script 20](#_Toc481554641)

[1.2.9 Load Data into Pig Relation 21](#_Toc481554642)

[1.2.10 NULL Operator 21](#_Toc481554643)

[1.2.11 Sort the Output 23](#_Toc481554644)

[1.2.12 Remove Duplicates 23](#_Toc481554645)

[1.2.13 Number of Reduce Task in MapReduce 24](#_Toc481554646)

[1.2.14 Join (inner/outer/replicated) 24](#_Toc481554647)

[1.2.15 Transform Data into Specified format 26](#_Toc481554648)

[1.2.16 Run Pig Job in TEZ 26](#_Toc481554649)

[1.2.17 Storage in HDFS 27](#_Toc481554650)

[1.2.18 Storage in Hive 27](#_Toc481554651)

[1.3 Data Analysis 27](#_Toc481554652)

[1.3.1 Hive Table that Uses ORC File Format 27](#_Toc481554653)

[1.3.2 Hive Table Storage Format and Delimiters 28](#_Toc481554654)

[1.3.3 Hive DML 29](#_Toc481554655)

[1.3.4 Hive SubQuery (in From and Where) 30](#_Toc481554656)

[1.3.5 Hive Compressed File 30](#_Toc481554657)

[1.3.6 Hive Through Vectorization 31](#_Toc481554658)

[1.3.7 Hive Explain Plan 31](#_Toc481554659)

[1.3.8 Hive Property file Configuration 32](#_Toc481554660)

[1.3.9 Hive Query Ordered across Multiple Reducers 32](#_Toc481554661)

[1.3.10 Hive Load Data using Select Statement 32](#_Toc481554662)

[1.4 Hive 32](#_Toc481554663)

[1.4.1 Partition 32](#_Toc481554664)

[1.4.2 Static Partitioning 33](#_Toc481554665)

[1.4.3 Dynamic Partitioning 34](#_Toc481554666)

[1.4.4 Alter Partitioning 36](#_Toc481554667)

[2. HDFS Processing Layer -Map Reduce 37](#_Toc481554668)

[2.1 Hadoop Components 38](#_Toc481554669)

[2.1.1 Map Reduce 39](#_Toc481554670)

[2.1.2 Map Stage 39](#_Toc481554671)

[2.1.3 Reduce Stage 39](#_Toc481554672)

[2.1.4 Input Split 40](#_Toc481554673)

[2.2 Hadoop 1 Processing Daemons 40](#_Toc481554674)

[2.2.1 Name Node 40](#_Toc481554675)

[2.2.2 Job Tracker 41](#_Toc481554676)

[2.2.3 Task Tracker 41](#_Toc481554677)

[2.2.4 How Does MapReduce Work 41](#_Toc481554678)

[2.2.5 MapReduce is Stable 42](#_Toc481554679)

[2.2.6 MapReduce Uses Functional PRogramming 42](#_Toc481554680)

[2.2.7 MapReduce Optimizes network Traffic 43](#_Toc481554681)

[2.2.8 Limitation of Haddop 1 43](#_Toc481554682)

[2.2.9 Haddop Ecosystem & Others 44](#_Toc481554683)

[2.3 Hadoop 2 - YARN 48](#_Toc481554684)

[2.3.1 Architectural Changes 48](#_Toc481554685)

[2.3.2 Yarn Components 48](#_Toc481554686)

[2.3.3 Resource Manager 48](#_Toc481554687)

[2.3.4 Scheduler 49](#_Toc481554688)

[2.3.5 Application Manager 49](#_Toc481554689)

[2.3.6 Node Manager 49](#_Toc481554690)

[2.3.7 Application Master 49](#_Toc481554691)

[2.3.8 Container 49](#_Toc481554692)

[2.3.9 YARN Process Flow 50](#_Toc481554693)

[2.3.10 UBER Mode 50](#_Toc481554694)

[3. HDFS File Read operation 51](#_Toc481554695)

[4. Hadoop - Data Life Cycle Management 52](#_Toc481554696)

[4.1 Sqoop Hive Import 52](#_Toc481554697)

[4.1.1 PArameter-Sqoop Hive Import 53](#_Toc481554698)

[4.1.2 Pitfalls -Sqoop Hive Import 53](#_Toc481554699)

[4.2 Hive Export To MySQL through SQOOP 54](#_Toc481554700)

[4.2.1 PArameters For Sqoop Export 54](#_Toc481554701)

[4.2.2 Sqoop Export –case 1 55](#_Toc481554702)

[4.2.3 Date Data Type 55](#_Toc481554703)

[4.2.4 Import - Date Data Type 56](#_Toc481554704)

[4.2.5 EXPORT - Date Data Type 56](#_Toc481554705)

[5. Hive 58](#_Toc481554706)

[5.1 Setup of Hive 58](#_Toc481554707)

[5.1.1 Hive Site XML Changes 58](#_Toc481554708)

[5.2 Hive 59](#_Toc481554709)

[5.2.1 Key Attributes 59](#_Toc481554710)

[5.2.2 SQL Features 59](#_Toc481554711)

[5.2.3 Data Units Hierarchy 59](#_Toc481554712)

[5.2.4 Data Types 59](#_Toc481554713)

[5.2.5 hIVE oPERATORS 61](#_Toc481554714)

[5.3 . Hive SQL 61](#_Toc481554715)

[5.3.1 Create 61](#_Toc481554716)

[5.3.2 Relationship between HIVe,HDFS and Metadata 63](#_Toc481554717)

[5.3.3 Hive Script 64](#_Toc481554718)

[5.4 Partitions 64](#_Toc481554719)

[5.4.1 Static Partition 64](#_Toc481554720)

[5.4.2 Dynamic Partition 64](#_Toc481554721)

[5.5 BUCKETS 66](#_Toc481554722)

[5.6 Serialization and De Serialization(SERDE) 68](#_Toc481554723)

[5.7 Hive vz Learning 69](#_Toc481554724)

[5.7.1 Join 69](#_Toc481554725)

[5.7.2 PArtitions /Bucket 69](#_Toc481554726)

[5.8 Performance Tunning 70](#_Toc481554727)

[5.8.1 Sort By 70](#_Toc481554728)

[5.8.2 Order By 71](#_Toc481554729)

[5.8.3 Distribute By 71](#_Toc481554730)

[5.8.4 Cluster By 71](#_Toc481554731)

[5.8.5 Cluster By 71](#_Toc481554732)

[5.8.6 Using Explain 71](#_Toc481554733)

[5.8.7 Optimized Joins 71](#_Toc481554734)

[5.8.8 Local Mode 72](#_Toc481554735)

[5.8.9 Parallel Execution 72](#_Toc481554736)

[5.8.10 Strict Mode 72](#_Toc481554737)

[6. PIG 73](#_Toc481554738)

[6.1 PIG Concepts 73](#_Toc481554739)

[6.1.1 Advantages 73](#_Toc481554740)

[6.1.2 History 73](#_Toc481554741)

[6.1.3 Pig Philosophy 74](#_Toc481554742)

[6.1.4 Application of PIG 74](#_Toc481554743)

[6.1.5 Pig Setup 75](#_Toc481554744)

[6.1.6 Architecture 75](#_Toc481554745)

[6.1.7 Interface 75](#_Toc481554746)

[6.1.8 PIG Comes Commands 75](#_Toc481554747)

[6.1.9 Execution Mode 76](#_Toc481554748)

[6.1.10 Data Types 76](#_Toc481554749)

[6.1.11 Pig and SQL Comparison 76](#_Toc481554750)

[6.2 PIG Workouts 77](#_Toc481554751)

[6.2.1 Pig Example 77](#_Toc481554752)

[6.2.2 Pig Scripts 79](#_Toc481554753)

[6.2.3 Pig Scripts 1 79](#_Toc481554754)

[6.2.4 Pig Scripts 2 80](#_Toc481554755)

[6.2.5 Pig Scripts 3 80](#_Toc481554756)

[6.2.6 Pig Scripts 4 80](#_Toc481554757)

[6.2.7 Pig Scripts 5 81](#_Toc481554758)

[6.2.8 Pig Functions 81](#_Toc481554759)

[Pig UDF 81](#_Toc481554760)

[6.3 PIG BOOk 81](#_Toc481554761)

[7. SQOOP 87](#_Toc481554762)

[7.1.1 Pig Google Cloud Workouts 88](#_Toc481554763)

[7.2 Sqoop Concepts 92](#_Toc481554764)

[7.2.1 Tidbits 93](#_Toc481554765)

[7.2.2 Advantages 94](#_Toc481554766)

[7.2.3 Features of Sqoop 95](#_Toc481554767)

[7.2.1 Sqoop Properties 95](#_Toc481554768)

[7.3 Sqoop Import Architecture 96](#_Toc481554769)

[7.3.1 Sqoop Import – Vz Learn 96](#_Toc481554770)

[7.3.2 readFIELDS() & write() Methods 97](#_Toc481554771)

[7.3.3 Splitting Column 98](#_Toc481554772)

[7.3.4 Text and Binary File Formats 98](#_Toc481554773)

[7.3.5 Controlling the Import 99](#_Toc481554774)

[7.3.6 Imports and Consistency 99](#_Toc481554775)

[7.3.7 Incremental Imports 99](#_Toc481554776)

[7.3.8 Direct Mode Import 99](#_Toc481554777)

[7.4 Sqoop Export Architecture 100](#_Toc481554778)

[7.4.1 Export VZ learn 100](#_Toc481554779)

[7.4.2 Import/Export into HBase VZ learn 100](#_Toc481554780)

[7.4.3 Sqoop CLI error Log-- VZ learn 101](#_Toc481554781)

[7.4.4 Export Architecture 101](#_Toc481554782)

[7.4.5 Export and Transactionality 102](#_Toc481554783)

[7.5 Sqoop Best PRactices and PErformance Tunning 102](#_Toc481554784)

[7.5.1 Import- Definite Number of Mappers 102](#_Toc481554785)

[7.5.2 Import - USe Direct Mode Where Ever Possible 103](#_Toc481554786)

[7.5.3 Import - Splitting Data / Boundry Query 104](#_Toc481554787)

[7.5.4 Import - Using $CONDITIONS 104](#_Toc481554788)

[7.5.5 Export- BATCH Mode 104](#_Toc481554789)

[7.5.6 Export- Specify Number of Records to Export 105](#_Toc481554790)

[7.5.7 Export- Specify Number of Records Per Transaction 105](#_Toc481554791)

[8. Flume 114](#_Toc481554792)

[8.1 Flume Concepts 114](#_Toc481554793)

[8.1.1 Website ClickStream Analytics 114](#_Toc481554794)

[8.1.2 Flume Data 115](#_Toc481554795)

[8.1.3 Problem Solved by Flume 115](#_Toc481554796)

[8.1.4 Intorducing CURL 115](#_Toc481554797)

[8.1.5 Flume Vs Hadoop vs POSIX 116](#_Toc481554798)

[8.1.6 Available Solutions 117](#_Toc481554799)

[8.1.7 FLUME Configuration files 117](#_Toc481554800)

[8.1.8 Installing Flume 118](#_Toc481554801)

[8.1.9 Functional parts of Flume 118](#_Toc481554802)

[8.1.10 Creating Flume Agents 118](#_Toc481554803)

[8.1.11 First Agent for Flume (capturing network data) 118](#_Toc481554804)

[8.1.12 Second Agent for Flume 120](#_Toc481554805)

[8.1.13 Flume Source 121](#_Toc481554806)

[8.1.14 Flume Sink 121](#_Toc481554807)

[8.1.15 Flume Channels 122](#_Toc481554808)

[8.1.16 File Channel to HDFS 123](#_Toc481554809)

[8.1.17 Avro Serialization 123](#_Toc481554810)

[8.1.18 Flume Multi Hop 124](#_Toc481554811)

[8.2 Multiplex Agents for Flume 124](#_Toc481554812)

[8.2.1 Interceptors 124](#_Toc481554813)

[8.2.2 TimeStamp Interceptors 125](#_Toc481554814)

[8.2.3 Multiple Flow and Sources with Flume 125](#_Toc481554815)

[8.2.4 Multiple Sink 127](#_Toc481554816)

[8.2.5 Flume Troubleshooting 128](#_Toc481554817)

[8.2.6 Flume Events 128](#_Toc481554818)

[8.2.7 Flume Components 128](#_Toc481554819)

[8.2.8 Agent 129](#_Toc481554820)

[8.2.9 Source 129](#_Toc481554821)

[8.2.10 Channels 129](#_Toc481554822)

[8.2.11 Sink 130](#_Toc481554823)

[8.2.12 Additional Components of Flume Agent 130](#_Toc481554824)

[8.2.13 Interceptors 130](#_Toc481554825)

[8.2.14 Channel Selectors 130](#_Toc481554826)

[8.2.15 Sink Processors 131](#_Toc481554827)

[8.2.16 Flume - DataFlow 131](#_Toc481554828)

[8.2.17 Failure Handling 131](#_Toc481554829)

[8.2.18 End to End Flow 131](#_Toc481554830)

[8.2.19 Summary 132](#_Toc481554831)

[9. Kumar Preparation 133](#_Toc481554832)

[9.1 Kumar Preparation 133](#_Toc481554833)

[9.1.1 Gcloud 133](#_Toc481554834)

[9.1.2 Hive 134](#_Toc481554835)

[9.1.3 Pig 135](#_Toc481554836)

[10. Interview Questions 137](#_Toc481554837)

[10.1 HDFS/ Hadoop – Interview Questions 137](#_Toc481554838)

[10.1.1 encrypted files in HDFS 137](#_Toc481554839)

[10.1.2 Read in Hdfs files in HDFS 137](#_Toc481554840)

[10.1.3 Limit Directory Usage 137](#_Toc481554841)

[10.1.4 Data Across Cluster 138](#_Toc481554842)

[10.1.5 Rebalance Operations 138](#_Toc481554843)

[10.1.6 SafeMode 138](#_Toc481554844)

[10.1.7 Counters In Haddop 138](#_Toc481554845)

[10.1.8 Block 139](#_Toc481554846)

[10.1.9 InputSplit and Block 139](#_Toc481554847)

[10.1.10 Speculative Execution 139](#_Toc481554848)

[10.1.11 Combiner 140](#_Toc481554849)

[10.1.12 Replica Placement Strategy 141](#_Toc481554850)

[10.1.13 Limit Directory Usage 141](#_Toc481554851)

[10.1.14 Limit Directory Usage 141](#_Toc481554852)

[10.1.15 Limit Directory Usage 141](#_Toc481554853)

[10.1.16 Limit Directory Usage 141](#_Toc481554854)

[10.2 Flume – Interview Questions 141](#_Toc481554855)

[10.2.1 Multiplexing 141](#_Toc481554856)

[10.2.2 Replication 142](#_Toc481554857)

[10.2.3 Components 142](#_Toc481554858)

[10.3 Pig/Hive/Sqoop 143](#_Toc481554859)

[11. Inceptize Running Notes 154](#_Toc481554860)

[11.1 Hadoop-Hive 159](#_Toc481554861)

[11.2 Hadoop-INTERVIEW QUESTIONS 163](#_Toc481554862)

[11.3 Hadoop Admin Interview Questions 171](#_Toc481554863)

[11.3.1 Pig Interview Questions 173](#_Toc481554864)

[11.3.2 Hive Interview Questions 173](#_Toc481554865)

[11.3.3 How do you debug a performance issue or a long running job? 190](#_Toc481554866)

[11.3.4 Assume you have Research, Marketing and Finance teams funding 60%, 30% and 10% respectively of your Hadoop Cluster. How will you assign only 60% of cluster resources to Research, 30% to Marketing and 10% to Finance during peak load? 191](#_Toc481554867)

[11.3.5 How do you benchmark your Hadoop cluster with tools that come with Hadoop? 192](#_Toc481554868)

[11.3.6 Assume you are doing a join and you notice that all but one reducer is running for a long time how do you address the problem in Pig? 193](#_Toc481554869)

[11.3.7 What is the difference between SORT BY and ORDER BY in Hive? 194](#_Toc481554870)

[11.3.8 What is Speculative execution? 195](#_Toc481554871)

[11.3.9 What is the benefit of using counters in Hadoop? 195](#_Toc481554872)

[11.3.10 What is the difference between an InputSplit and a Block? 196](#_Toc481554873)

[11.3.11 Can you change the number of mappers to be created for a job in Hadoop? 196](#_Toc481554874)

[11.3.12 How do you do a file system check in HDFS? 196](#_Toc481554875)

[11.3.13 What are the parameters of mappers and reducers function? 196](#_Toc481554876)

[11.3.14 How do you overwrite replication factor? 197](#_Toc481554877)

[11.3.15 What are the functions of InputFormat? 197](#_Toc481554878)

[11.3.16 What is a Record Reader? 197](#_Toc481554879)

[11.3.17 What is a sequence file in Hadoop? 197](#_Toc481554880)

[11.3.18 1) What is Hadoop? 205](#_Toc481554881)

[11.3.19 2) What platform and Java version is required to run Hadoop? 205](#_Toc481554882)

[11.3.20 3) What kind of Hardware is best for Hadoop? 205](#_Toc481554883)

[11.3.21 4) What are the most common input formats defined in Hadoop? 206](#_Toc481554884)

[11.3.22 5) What is InputSplit in Hadoop? Explain. 206](#_Toc481554885)

[11.3.23 6) How many InputSplits is made by a Hadoop Framework? 206](#_Toc481554886)

[11.3.24 7) What is the use of RecordReader in Hadoop? 206](#_Toc481554887)

[11.3.25 8) What is JobTracer in Hadoop? 207](#_Toc481554888)

[11.3.26 9) What are the functionalities of JobTracer? 207](#_Toc481554889)

[11.3.27 10) Define TaskTracker. 207](#_Toc481554890)

[11.3.28 11) What is Map/Reduce job in Hadoop? 207](#_Toc481554891)

[11.3.29 12) What is Hadoop Streaming? 208](#_Toc481554892)

[11.3.30 13) What is a combiner in Hadoop? 208](#_Toc481554893)

[11.3.31 14) Is it necessary to know java to learn Hadoop? 208](#_Toc481554894)

[11.3.32 15) How to debug Hadoop code? 208](#_Toc481554895)

[11.3.33 16) Is it possible to provide multiple inputs to Hadoop? If yes, explain. 208](#_Toc481554896)

[11.3.34 17) What is the relation between job and task in Hadoop? 209](#_Toc481554897)

[11.3.35 18) What is distributed cache in Hadoop? 209](#_Toc481554898)

[11.3.36 19) What commands are used to see all jobs running in the Hadoop cluster and kill a job in LINUX? 209](#_Toc481554899)

[11.3.37 20) What is the functionality of JobTracker in Hadoop? How many instances of a JobTracker run on Hadoop cluster? 209](#_Toc481554900)

[11.3.38 21) How JobTracker assign tasks to the TaskTracker? 210](#_Toc481554901)

[11.3.39 22) Is it necessary to write jobs for Hadoop in Java language? 210](#_Toc481554902)

# Last sprint

## Data Ingestion

### Sqoop Import delimiter

If the data itself has comma (also field delimiter is also comma) it causes data quality issue hence we use ( --enclosed-by \”) enclosed by property which will enclose the field with given calues

*$ sqoop import \*

*--query 'SELECT a.\*, b.\* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' \*

*--split-by a.id --target-dir /user/foo/joinresults \*

*--enclosed-by \";*

has comma (also field There will cases where we need to escape certain characters, and its done by --escaped-by property

*$ sqoop import \*

*--query 'SELECT a.\*, b.\* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' \*

*--split-by a.id --target-dir /user/foo/joinresults \*

*--enclosed-by \"*

*--fields-terminated-by \| \*

*--lines-terminated-by \: ;*

*Sqoop import*

*--connect jdbc:mysql://hdcentos/CRMDB \*

*--username sqoopuser –password Mayyam%123 \*

*--table invoices \*

*--hive –import \*

*--hive –table pet.invoices \*

*--num –mappers 1*

*describe formatted pet.invoices;*

You will find that field delimiter is 'n\0000' and also you can cat the file in hive directory.

you can also sqoop import into hive driectory without hive import since

*Sqoop import*

*--connect jdbc:mysql://hdcentos/CRMDB \*

*--username sqoopuser –password Mayyam%123 \*

*--table invoices \*

*--target-dir /user/hive/warehouse/database1/invoices \*

*--append ;*

But you will find all the values are NULL (assumption is describe formatted invoces have null field delimited). Sqoop as comma delimiter , due to data mismatch all the values will be displayed as NULL when you run select \* from invoices;

### Sqoop Free From Query

Sqoop can also import the result set of an arbitrary SQL query. Instead of using the --table, --columns and --where arguments, you can specify a SQL statement with the --query argument.When importing a free-form query, you must specify a destination directory with --target-dir.

If you want to import the results of a query in parallel, then each map task will need to execute a copy of the query, with results partitioned by bounding conditions inferred by Sqoop. Your query must include the token $CONDITIONS which each Sqoop process will replace with a unique condition expression. You must also select a splitting column with --split-by. For example:

*$ sqoop import \*

*--query 'SELECT a.\*, b.\* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' \*

*--split-by a.id --target-dir /user/foo/joinresults*

*$ sqoop import \*

*--query 'SELECT a.\*, b.\* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' \*

*-m 1 --target-dir /user/foo/joinresults*

### Sqoop Import Into Hive Table

Sqoop can also import the data into Hive by generating and executing a CREATE TABLE statement to define the data’s layout in Hive you can specify the --hive-overwrite option to indicate that existing table in hive must be replaced.Sqoop will generate a Hive script containing a CREATE TABLE operation defining your columns using Hive’s types, and a LOAD DATA INPATH statement to move the data files into Hive’s warehouse directory.

Note:-This function is incompatible with --as-avrodatafile and --as-sequencefile.

Even though Hive supports escaping characters, it does not handle escaping of new-line character. Also, it does not support the notion of enclosing characters that may include field delimiters in the enclosed string. It is therefore recommended that you choose unambiguous field and record-terminating delimiters without the help of escaping and enclosing characters when working with Hive; this is due to limitations of Hive’s input parsing abilities.

So if you do use --escaped-by, --enclosed-by, or --optionally-enclosed-by when importing data into Hive, Sqoop will print a warning message.

**--hive-drop-import-delims** :- Hive will have problems using Sqoop-imported data if your database’s rows contain string fields that have Hive’s default row delimiters (\n and \r characters) or column

delimiters (\01 characters) present in them. You can use the --hive-drop-import-delims

Alternatively, you can use the --hive-delims-replacement option to replace those characters with a user-defined string on import to give Hive-compatible text data.

**(--input-null-string, --input-null-non-string & --input-null-string, --input-null-non-string):-** Sqoop will by default import NULL values as string null. Hive is however using string \N to denote NULL

values and therefore predicates dealing with NULL (like IS NULL) will not work correctly. You should append parameters --null-string and --null-non-string in case of import job or --input-null-string

and --input-null-non-string in case of an export job if you wish to properly preserve NULL values.

$ sqoop import ... --null-string '\\N' --null-non-string '\\N'

**--compress & --compression-codec:-** You can import compressed tables into Hive using the --compress and --compression-codec options. One downside to compressing tables imported into Hive is that many

codecs cannot be split for processing by parallel map tasks. The lzop codec, however, does support splitting. When importing tables with this codec, Sqoop will automatically index the files for splitting

and configuring a new Hive table with the correct InputFormat. This feature currently requires that all partitions of a table be compressed with the lzop codec.

|  |  |
| --- | --- |
| **Argument** | **Description** |
| --hive-home <dir> | Override $HIVE\_HOME |
| --hive-import | Import tables into Hive (Uses Hive’s default delimiters if none are set.) |
| --hive-overwrite | Overwrite existing data in the Hive table. |
| --create-hive-table | If set, then the job will fail if the target hive |
|  | table exits. By default this property is false. |
| --hive-table <table-name> | Sets the table name to use when importing to Hive. |
| --hive-drop-import-delims | Drops *\n*, *\r*, and *\01* from string fields when importing to Hive. |
| --hive-delims-replacement | Replace *\n*, *\r*, and *\01* from string fields with user defined string when importing to Hive. |
| --hive-partition-key | Name of a hive field to partition are sharded on |
| --hive-partition-value <v> | String-value that serves as partition key for this imported into hive in this job. |
| --map-column-hive <map> | Override default mapping from SQL type to Hive type for configured columns. |

### Sqoop Export

The export tool exports a set of files from HDFS back to an RDBMS. The target table must already exist in the database. The input files are read and parsed into a set of records according to the user-specified delimiters.

The default operation is to transform these into a set of INSERT statements that inject the records into the database. In "update mode," Sqoop will generate UPDATE statements that replace existing records in the database, and in "call mode" Sqoop will make a stored procedure call for each record.

|  |  |
| --- | --- |
| **Common Argument** | **Description** |
| --connect <jdbc-uri> | Specify JDBC connect string |
| --connection-manager <class-name> | Specify connection manager class to use |
| --driver <class-name> | Manually specify JDBC driver class to use |
| --hadoop-mapred-home <dir> | Override $HADOOP\_MAPRED\_HOME |
| --help | Print usage instructions |
| --password-file | Set path for a file containing the authentication password |
| -P | Read password from console |
| --password <password> | Set authentication password |
| --username <username> | Set authentication username |
| --verbose | Print more information while working |
| --connection-param-file <filename> | Optional properties file that provides connection parameters |
| --relaxed-isolation | Set connection transaction isolation to read uncommitted for the mappers. |

|  |  |
| --- | --- |
| **Argument** | **Description** |
| --columns <col,col,col…> | Columns to export to table |
| --direct | Use direct export fast path |
| --export-dir <dir> | HDFS source path for the export |
| -m,--num-mappers <n> | Use *n* map tasks to export in parallel |
| --table <table-name> | Table to populate |
| --call <stored-proc-name> | Stored Procedure to call |
| --update-key <col-name> | Anchor column to use for updates. Use a comma separated list of columns if there are more than one column. |
| --update-mode <mode> | Specify how updates are performed when new rows are found with non-matching keys in database. |
|  | Legal values for mode include updateonly (default) and allowinsert. |
| --input-null-string <null-string> | The string to be interpreted as null for string columns |
| --input-null-non-string <null-string> | The string to be interpreted as null for non-string columns |
| --staging-table <staging-table-name> | The table in which data will be staged before being inserted into the destination table. |
| --clear-staging-table | Indicates that any data present in the staging table can be deleted. |
| --batch | Use batch mode for underlying statement execution. |

### Sqoop Export (update/Insert)

*create table emp (*

*id int not null primary key,*

*name varchar(50));*

vi emp --> create file with below contents and Move the file to hdfs

1,Thiru

2,Vikram

3,Brij

4,Sugesh

hadoop fs -put emp <dir>

*sqoop export --connect <jdbc connection> \*

*--username sqoop \*

*--password sqoop \*

*--table emp \*

*--export-dir <dir> \*

*--input-fields-terminated-by ',';*

update the emp file & move the updated file into hdfs. contents of the updated file

1,Thiru

2,Vikram

3,Sugesh

4,Brij

5,Sagar

*sqoop export --connect <jdbc connection> \*

*--username sqoop \*

*--password sqoop \*

*--table emp \*

*--update-mode allowinsert \*

*--update-key id \*

*--export-dir <dir> \*

*--input-fields-terminated-by ',';*

--update-mode updateonly \ --> for updates

--update-mode allowinsert \ --> for upsert

### Start Flume Agent

bin/flume-ng agent -n $agent\_name -c conf -f conf/flume-conf.properties.template

-n is agent name

-c is direectory

-f is file name

if the Flume is installed there will be a directory call /etc/flume/conf

*flume-ng agent -n agent -f /etc.../flume-conf.properties.template*

**Configure the memory channel.**

1. type (memory) -> needs to be Memory

2. capacity (100) -> Maximum number of events stored in the channel

3. transactionCapacity (100)-> Maximum number of events the channels will take from a source or give to a sink per transaction.

4. keep-alive (3) -> Timeout in Seconds for adding or removing an event

5. byteCapacityBufferPercentage (20) -> Defines the percent of buffer between byteCapacity and estimated total size of all events in the channel, to account for data in headers. See below

6. byteCapacity -> Maximum total bytes of memory allowed as a sum of all events in this channel. The implementation only counts the Event body, which is hte reason for providing the byteCapacityBufferPercentage configuration parameteras well.

## Data Transformation

*SET default\_parallel 20;*

### Handling Null's

Null values should be filtered first. The general rule of thumb in Pig is to "filter early and often" to minimize the amount of data that gets shuffled and sorted, so before the foreach:

For example:

*a = "some Pig relation"*

*b = filter a by $1 is not null;* //filter out tuples where the $1 field is null

*c = foreach b generate* ... //no need to worry about $1 being null

The term "empty" refers to bags typically, and in particular you can use the isEmpty function to check if a bag is empty. You normally do this after a GROUP command:

*a = "some Pig relation"*

*b = group a by $3;*

*c = filter b by not IsEmpty(group);*

### Tokenize

Use the TOKENIZE function to split a string of words (all words in a single tuple) into a bag of words (each word in a single tuple). The following characters are considered to be word separators: space, double quote("), coma(,) parenthesis(()), star(\*).

*A = LOAD 'data' AS (f1:chararray);*

DUMP A;

(Here is the first string.)

(Here is the second string.)

(Here is the third string.)

*X = FOREACH A GENERATE TOKENIZE(f1);*

DUMP X;

({(Here),(is),(the),(first),(string.)})

({(Here),(is),(the),(second),(string.)})

({(Here),(is),(the),(third),(string.)})

### Example 2

A tuple is a fixed-length, ordered collection of PIG data element.

A bag is an unordered collection of tuples.

*bascischema = LOAD 'seismic';*

*dump basicschema;*

*basicschema1 = LOAD 'seismic' as (doubleRow:DOUBLE);*

*describe basicschema1;*

*splt = FOREACH basicsschema1 generate STRSPLIT(' ');*

*describe splt;*

*splt: {()} // a set of rows called tuples*

*dump splt;*

*bagSplits = FOREACH splt generate $0 as id, FLATTEN(TOBAG(\*)) as vlaue;*

*dump bagSplits;* // we have unflattened

Note is data is of the form as given below (obviously many records), also delimiter is different

and its not tab i:e, two tab delimiter,

20170430 -0.01038 0.00031 0.02319 0.02247

20170431 -0.01098 0.00021 0.02419 0.03247

*basicschema1 = LOAD 'seismic' as (doubleRow:DOUBLE);*

*describe basicschema1;*

*bascischema = LOAD 'seismic';*

*dump basicschema;*

If you dump the data loaded without schema it will work as the load will happen because default datatype chararray. However if the schema is added as double as given above, dump will not work. Dump will not work with double.

Here you will see converted to tuple of elements which is not useful for us

(20170430 -0.01038 0.00031 0.02319 0.02247)

(20170431 -0.01098 0.00021 0.02419 0.03247)

Tuples can have many tokens and needn't have same datatype, in our example a tuple has 5 tokens and

it's not necessary to have all the tokens have same datatype. Where as bag is a unordered collection of tuples

**We work with tuples and bag using the below schematic:-**

GENERATE FLATTEN LOOPS

Flatten does (1,(1,2) => (1,1), (1,2) nested structure is lost, We going to generate 5 tokens for each tuples which as 2 tab delimiter,

//splt = FOREACH basicschema1 GENERATE STRSPLIT(' ');

*basicshema1 = LOAD 'seismic' AS (doubleRow:Chararray);*

*dump basicschema1;*

*splt = FOREACH basicschema1 GENERATE TOKENIZE(doubleRow);*

*dump splt;*

({(20170430),(-0.01038),(0.00031),(0.02319),(0.02247)})

Now we got 5 tuple inside a bag which is again inside a tuple. We going to unflatten(nested) this so we can explain flatten,

*bagsplits = FOREACH splt generate $0 as id, FLATTEN(TOBAG(\*)) as value;*

*dump bagsplits;*

({(20170430),(-0.01038),(0.00031),(0.02319),(0.02247)},{(20170430),(-0.01038),(0.00031),(0.02319),(0.02247)})

Now we have nested structure with two copies, now we will see the effect of flatten,

*bagsplits2 = FOREACH bagsplits GENERATE FLATTEN (value)*

*Flatten does (1,(1,2) => (1,1), (1,2) nested structure is lost*

(20170430)

(-0.01038)

(0.00031)

(0.02319)

(0.02247)

*bagsplits3 = FOREACH bagsplits GENERATE FLATTEN (id)*

### Edukerka

Data Model(structure ?) of Pig as four types,

1. Atom

2. Tuple

3. Bag

4. Map

A tuple is an ordered set of fields which may contain different data types for each field. you can think of row has record in table, tuple is not arrays, arrays are of same data type.

eg: (1,Linkin Park,7,California)

A Bag is a collection of a set of tuples and these tuples are subset of rows or entire rows of a table.

eg: (Linkin Park,7,California),(Metallica,8),(Mega Death,LosAngeles)

A MAP is a set of key value pairs, a map is [band#Linkin Park,members#7] which can't contain duplicate keys.

key must be a string and value can be complex data type or atom.

eg, [band#Linkin Park,members#7], [band#Metallica, members#8]

A Atom are basic datatypes which are used in all the languages like string,int,float,long,double,char[]

Note: Date dataytpe is not available we represent it with long data type, timestamp is also in long

Pig Latin file Loaders:

BinStorage

PigStorage

TextLoaader

CSVLoader

XMLLoader

Note: Even when you load the data without schema you can reference it through $0,$1..

A.txt

0,1,2

1,3,4

B.txt

0,5,2

1,7,2

*a= LOAD '/A.txt' using PigStorage(',') as (a1:int,a2:int,a3:int);*

*b= LOAD '/B.txt' using PigStorage(',') as (b1:int,b2:int,b3:int);*

*c= UNION a,b;*

//now split the data

*SPLIT c INTO d IF $0 ==0 , e IF $0 == 1;*

*SPLIT c INTO d IF a1 ==0 , e IF b1 == 1;*

//Filter the data

*f = FILTER c BY $1 >3;*

// understand the underlying process.

*illustrate f;*

g= GROUP c by $2;

(2,{(0,1,2),(0,5,7),(1,7,2)})

(4,{(1,3,4)})

*k=FOREACH c GENERATE a2,a2.a3*

//word count

*A = LOAD '/wordcount.txt' USING PigStorage;*

*B = FOREACH A GENERATE FLATTEN(TOKENIZE((chararray) $0)) as word;*

(day)

(java)

(learn)

(day)

(java)

(learn)

*C = GROUP B BY word;*

*D = FOREACH c GENERATE COUNT(B), GROUP;*

28 day

28 java

28 learn

**Process the XML file:**

<property>

<name>Kumar</name>

<name>bharathi</name>

</property>

*REGISTER '/home/../piggybank.jar'*

*xml = LOAD 'user/../xmlinput.txt' USING org.apache.pig.piggybank.storage.XMLLoader('name') AS (doc:chararray);*

*value = FOREACH xml GENERATE FLATTEN (REGEX\_EXTRACT\_ALL(doc,'<name>(.\*)</name>')) AS name:chararray;*

*STORE value INTO '/user/.../output1';*

*dump value;*

### Projections in PIG

https://wiki.apache.org/pig/PigLatin

### Data Items

Data can be referred to in various powerful and convenient ways in Pig. Any data referred to is called a Data Item. We will illustrate all these ways by using the following example tuple.

t = < 1, {<2,3>,<4,6>,<5,7>}, ['apache':'search']>

Thus, t has 3 fields. Let these fields have names f1, f2, f3. Field f1 is an atom with value 1. Field f2 is a bag having 3 tuples. Field f3 is a data map having 1 key. The following table lists the various methods of referring to data.

|  |  |  |  |
| --- | --- | --- | --- |
| Method of Referring to Data | Example | Value for example tuple t | Notes |
| **Constant** | **'1.0**', or **'apache.org**', or **'blah**' | Value constant irrespective of t |  |
| **Field referred to by position** | **$0** | Data Atom '1' | **In Pig, positions start at 0 and not 1** |
| **Field referred to by name** | **f2** | Bag {<2,3>,<4,6>,<5,7>} |  |
| **Projection** of another data item | **f2.$0** | Bag {<2>,<4>,<5>} - the bag f2 projected to the first field |  |
| **Map Lookup** against another data item | **f3#'apache**' | Data Atom 'search' | User's responsibility to ensure that a lookup is written only against a data map, otherwise a runtime error is thrown. If the key being looked up does not exist, a Data Atom with an empty string is returned. |
| **Function** applied to another data item | **SUM(f2.$0)** | 2+4+5 = 11 | SUM is a builtin Pig function. See [PigFunctions](https://wiki.apache.org/pig/PigFunctions) for how to write your own functions |
| **Infix Expression** of other data items | **COUNT(f2) + f1 / '2.0**' | 3 + 1 / 2.0 = 3.5 |  |
| **Bincond**, i.e., the value of the data item is chosen according to some condition | **(f1 = = '1' ? '2' : COUNT(f2))** | '2' since f1=='1' is true. If f1 were != '1', then the value of this data item for t would be COUNT(f2)=3 | See [Conditions](https://wiki.apache.org/pig/PigLatin#Specifying_Conditions) for what the format of the condition in the bincond can be |

### Example 1

I have the following data with schema (t0: chararray,t1: int,t2: int)

(B,4,2)

(A,2,3)

(A,3,2)

(B,2,2)

(A,1,2)

(B,1,2)

I'd like to generate the following results (group by t0, and ordered by t1)

(A, ((1,2),(2,3),(3,2)))

(B, ((1,2),(2,2),(4,2)))

Please note I want only tuples in the second component, not bags. Please help.

Solution:

-- A: (t0: chararray,t1: int,t2: int)

B = GROUP A BY t0 ;

C = FOREACH B {

-- Project out the first column of A.

projected = FOREACH A GENERATE t1, t2 ;

-- Now you can order the projection.

ordered = ORDER projected BY t1 ;

GENERATE group AS t0, ordered AS vals ;

}

NOTE/UPDATE: It seems when I answered this question originally I missed the part where the asker asked for output to be in tuple form. Tuples should only be used when you know the exact number and position of the fields in the tuple. Otherwise then your schema will not be defined and it will be very difficult in order to access the fields. This is because the entire tuple will be treated as a bytearray, and so you will manually have to find and cast everything.

If you must do it this way you can not do this in pure pig. You'll have to use some sort of UDF to do this. I would recommend Python.

### Write and Execute Pig Script

/\* myscript.pig

My script is simple.

It includes three Pig Latin statements.

\*/

*A = LOAD 'student' USING PigStorage() AS (name:chararray, age:int, gpa:float);* -- loading data

*B = FOREACH A GENERATE name;* -- transforming data

*DUMP B;* -- retrieving results

Pig supports running scripts (and Jar files) that are stored in HDFS, Amazon S3, and other distributed file systems. The script's full location URI is required (see REGISTER for information about Jar files). For example, to run a Pig script on HDFS, do the following:

*$ pig hdfs://nn.mydomain.com:9020/myscripts/script.pig*

### Load Data into Pig Relation

With Schema,

*A = LOAD 'myfile.txt';*

*A = LOAD 'myfile.txt' USING PigStorage('\t');*

Without Schema,

*A = LOAD 'myfile.txt' AS (f1:int, f2:int, f3:int);*

*A = LOAD 'myfile.txt' USING PigStorage('\t') AS (f1:int, f2:int, f3:int);*

From Hive Table,

The HCatLoader and HCatStorer interfaces are used with Pig scripts to read and write data in HCatalog-managed tables. No HCatalog-specific setup is required for these interfaces.

To bring in the appropriate jars for working with HCatalog, simply include the following flag / parameters when running Pig from the shell, Hue, or other applications,

*pig –useHCatalog*

**HCatLoader**

HCatLoader is used with Pig scripts to read data from HCatalog-managed tables. HCatLoader is accessed via a Pig load statemen

*A = LOAD 'tablename' USING org.apache.hive.hcatalog.pig.HCatLoader();*

**HCatStorer**

HCatStorer is used with Pig scripts to write data to HCatalog-managed tables. HCatStorer is accessed via a Pig store statement.

*STORE my\_processed\_data INTO 'tablename' USING org.apache.hive.hcatalog.pig.HCatStorer();*

### NULL Operator

In Pig Latin, nulls are implemented using the SQL definition of null as unknown or non-existent. Nulls can occur naturally in data or can be the result of an operation.Functions like (AVG, MIN, MAX, SUM, COUNT) ignore nulls. In most cases it will be null.Operators like COGROUP, GROUP, JOIN these operators handle nulls differently.

*X = FILTER A BY f1 is not null;*

*X = FILTER A BY f1 is null;*

**Nulls and Constants:**

*A = LOAD 'data' AS (a, b, c).*

*B = FOREACH A GENERATE a, null;*

//In this example of an outer join, if the join key is missing from a table it is replaced by null.

*A = LOAD 'student' AS (name: chararray, age: int, gpa: float);*

*B = LOAD 'votertab10k' AS (name: chararray, age: int, registration: chararray, donation: float);*

*C = COGROUP A BY name, B BY name;*

*D = FOREACH C GENERATE FLATTEN((IsEmpty(A) ? null : A)), FLATTEN((IsEmpty(B) ? null : B));*

//In this example both a and null will be implicitly cast to double.

*A = LOAD 'data' AS (a, b, c).*

*B = FOREACH A GENERATE a + (int)null;*

//In this example both a and null will be cast to int, a implicitly, and null explicitly.

*A = LOAD 'data' AS (a, b, c).*

*B = FOREACH A GENERATE a + (int)null;*

**Nulls and GROUP/COGROUP Operators:**

When using the GROUP operator with a single relation, records with a null group key are grouped together.

*A = load 'student' as (name:chararray, age:int, gpa:float);*

*dump A;*

(joe,18,2.5)

(sam,,3.0)

(bob,,3.5)

*X = group A by age;*

*dump X;*

(18,{(joe,18,2.5)})

(,{(sam,,3.0),(bob,,3.5)})

When using the GROUP (COGROUP) operator with multiple relations, records with a null group key from different relations are considered different and are grouped separately. In the example below note that there are two tuples in the output corresponding to the null group key: one that contains tuples from relation A

(but not relation B) and one that contains tuples from relation B (but not relation A).

*A = load 'student' as (name:chararray, age:int, gpa:float);*

*B = load 'student' as (name:chararray, age:int, gpa:float);*

dump B;

(joe,18,2.5)

(sam,,3.0)

(bob,,3.5)

*X = cogroup A by age, B by age;*

*dump X;*

(18,{(joe,18,2.5)},{(joe,18,2.5)})

(,{(sam,,3.0),(bob,,3.5)},{})

(,{},{(sam,,3.0),(bob,,3.5)})

**Nulls and JOIN Operator:**

The JOIN operator - when performing inner joins - adheres to the SQL standard and disregards (filters out) null values. (See also Drop Nulls Before a Join.)

*A = load 'student' as (name:chararray, age:int, gpa:float);*

*B = load 'student' as (name:chararray, age:int, gpa:float);*

*dump B;*

(joe,18,2.5)

(sam,,3.0)

(bob,,3.5)

*X = join A by age, B by age;*

*dump X;*

(joe,18,2.5,joe,18,2.5)

### Sort the Output

alias = ORDER alias BY { \* [ASC|DESC] | field\_alias [ASC|DESC] [, field\_alias [ASC|DESC] …] } [PARALLEL n];

\* -> The designator for a tuple.

Pig currently supports ordering on fields with simple types or by tuple designator (\*). You cannot order on fields with complex types or by expressions.

*A = LOAD 'mydata' AS (x: int, y: map[]);*

*B = ORDER A BY x;* -- this is allowed because x is a simple type

*B = ORDER A BY y;* -- this is not allowed because y is a complex type

*B = ORDER A BY y#'id';* -- this is not allowed because y#'id' is an expression

### Remove Duplicates

alias = DISTINCT alias [PARTITION BY partitioner] [PARALLEL n];

Use the DISTINCT operator to remove duplicate tuples in a relation. DISTINCT does not preserve the original order of the contents (to eliminate duplicates, Pig must first sort the data). You cannot use DISTINCT on a subset of fields; to do this, use FOREACH and a nested block to first select the fields and then apply DISTINCT (see Example: Nested Block).

*A = LOAD 'data' AS (a1:int,a2:int,a3:int);*

*DUMP A;*

(8,3,4)

(1,2,3)

(4,3,3)

(4,3,3)

(1,2,3)

*X = DISTINCT A;*

*DUMP X;*

(1,2,3)

(4,3,3)

(8,3,4)

### Number of Reduce Task in MapReduce

Use the Parallel Features

You can set the number of reduce tasks for the MapReduce jobs generated by Pig using two parallel features. (The parallel features only affect the number of reduce tasks. Map parallelism is determined by the input file, one map for each HDFS block.)You Set the Number of ReducersUse the set default parallel command to set the number of reducers at the script level.

Alternatively, use the PARALLEL clause to set the number of reducers at the operator level. (In a script, the value set via the PARALLEL clause will override any value set via "set default parallel.") You can include the PARALLEL clause with any operator that starts a reduce phase: COGROUP, CROSS, DISTINCT, GROUP, JOIN (inner), JOIN (outer), and ORDER BY.

The number of reducers you need for a particular construct in Pig that forms a MapReduce boundary depends entirely on (1) your data and the number of intermediate keys you are generating in your mappers and (2) the partitioner and distribution of map (combiner) output keys. In the best cases we have seen that a reducer processing about 1 GB of data behaves efficiently.

Let Pig Set the Number of Reducers, If neither "set default parallel" nor the PARALLEL clause are used, Pig sets the number of reducers using a heuristic based on the size of the input data. You can set the values for these properties:

pig.exec.reducers.bytes.per.reducer - Defines the number of input bytes per reduce; default value is 1000\*1000\*1000 (1GB).pig.exec.reducers.max - Defines the upper bound on the number of reducers; default is 999.The formula, shown below, is very simple and will improve over time. The computed value takes all inputs within the script into account and applies the computed value to all the jobs within Pig script.

#reducers = MIN (pig.exec.reducers.max, total input size (in bytes) / bytes per reducer)

*SET default\_parallel 20;*

*A = LOAD 'myfile.txt' USING PigStorage() AS (t, u, v);*

*B = GROUP A BY t;*

*C = FOREACH B GENERATE group, COUNT(A.t) as mycount;*

*D = ORDER C BY mycount;*

*STORE D INTO 'mysortedcount' USING PigStorage();*

### Join (inner/outer/replicated)

**JOIN (inner)**

Performs an inner join of two or more relations based on common field values.

alias = JOIN alias BY {expression|'('expression [, expression …]')'} (, alias BY {expression|'('expression [, expression …]')'} …) [USING 'replicated' | 'skewed' | 'merge' | 'merge-sparse'] [PARTITION BY partitioner] [PARALLEL n];

Example: X = JOIN A BY fieldA, B BY fieldB, C BY fieldC;

Use the JOIN operator to perform an inner, equijoin join of two or more relations based on common field values. Inner joins ignore null keys, so it makes sense to filter them out before the join.

Note the following about the GROUP/COGROUP and JOIN operators: The GROUP and JOIN operators perform similar functions. GROUP creates a nested set of output tuples while JOIN creates a flat set of output tuples.

The GROUP/COGROUP and JOIN operators handle null values differently (see Nulls and JOIN Operator).

**JOIN (outer)**

Performs an outer join of two relations based on common field values.

alias = JOIN left-alias BY left-alias-column [LEFT|RIGHT|FULL] [OUTER], right-alias BY right-alias-column [USING 'replicated' | 'skewed' | 'merge'] [PARTITION BY partitioner] [PARALLEL n];

This example shows a left outer join.

*A = LOAD 'a.txt' AS (n:chararray, a:int);*

*B = LOAD 'b.txt' AS (n:chararray, m:chararray);*

*C = JOIN A by $0 LEFT OUTER, B BY $0;*

This example shows a full outer join.

*A = LOAD 'a.txt' AS (n:chararray, a:int);*

*B = LOAD 'b.txt' AS (n:chararray, m:chararray);*

*C = JOIN A BY $0 FULL, B BY $0;*

This example shows a replicated left outer join.

*A = LOAD 'large';*

*B = LOAD 'tiny';*

*C= JOIN A BY $0 LEFT, B BY $0 USING 'replicated';*

This example shows a skewed full outer join.

*A = LOAD 'studenttab' as (name, age, gpa);*

*B = LOAD 'votertab' as (name, age, registration, contribution);*

*C = JOIN A BY name FULL, B BY name USING 'skewed';*

**Replicated Joins**

**In Pig joins happen on reduce side.**

Fragment replicate join is a special type of join that works well if one or more relations are small enough to fit into main memory. In such cases, Pig can perform a very efficient join because all of the hadoop work is done on the map side. In this type of join the large relation is followed by one or more small relations. The small relations must be small enough to fit into main memory; if they don't, the process fails and an error is generated.

Usage

Perform a replicated join with the USING clause (see JOIN (inner) and JOIN (outer)). In this example, a large relation is joined with two smaller relations. Note that the large relation comes first followed by the smaller relations; and, all small relations together must fit into main memory, otherwise an error is generated.

*big = LOAD 'big\_data' AS (b1,b2,b3);*

*tiny = LOAD 'tiny\_data' AS (t1,t2,t3);*

*mini = LOAD 'mini\_data' AS (m1,m2,m3);*

*C = JOIN big BY b1, tiny BY t1, mini BY m1 ;*

*EXPLAIN C* // you will have both mapper and reducer

*C1 = JOIN big BY b1, tiny BY t1, mini BY m1 USING 'replicated';*

*EXPLAIN C1* // you will have only mapper

### Transform Data into Specified format

First Check the Hive table by

*hive>describe customer\_details*

// for knowing the path and delimiter (|)

*hive>describe formatted customer\_details*

*pig>-use HCatalog*

*pig>cusotmer\_details\_hive = LOAD 'DBName.Table\_name' USING org.apache.hive.hcatalog.pig.HCatLoader();*

*pig>describe customer\_details\_hive;*

Both Pig and Hive structure should be same.

*C = JOIN big BY b1, tiny BY t1, mini BY m1 ;*

*EXPLAIN C* // you will have both mapper and reducer

*C1 = JOIN big BY b1, tiny BY t1, mini BY m1 USING 'replicated';*

*EXPLAIN C1* // you will have only mapper

### Run Pig Job in TEZ

*$vi pigScript.txt*

*order = LOAD ‘pig.demo.orders’ USING org.apache.hive.hcatalog.pig.HCatloader();*

*order\_grouped = GROUP orders ALL;*

*order\_count = FOREACH order\_grouped GENERATE COUNT\_STAR(order);*

*DUMP order\_count;*

*$pig –f pigScript.txt –useHcatalog* // for map reduce

*$pig –f pigScript.txt –useHcatalog –x tez* //for tez engine

*$pig –f pigScript.txt –useHcatalog –x mapreduce* //for tez engine

$pig –x tez -useHCatalog

### Storage in HDFS

*Pig>store ds INTO ‘/user/../departments’ USING PigStorage(‘|’);*

*Pig>store ds INTO ‘/user/../departments’ USING BinStorage(‘|’);*

*Pig>store ds INTO ‘/user/../departments’ USING JsonStorage( );*

1. PigStorage, you can specify delimiters.
2. BinStorage, stores in binary format , you specify the delimiters.
3. JsonStorage, No need to specify the delimiters.

### Storage in Hive

To load the data into Hive from PIG you need to load the PIG with Schema

*LinuxHost$pig -userHcatalog*

*Pig>departments = LOAD ‘/user/…/departments’ USING PigStore(‘,’) AS (depid: INT, depname: CHARARRY);*

*Pig>store departments INTO ‘databasename.departments’ USING org.apache.hive.hcatalog.pig.HCatstorer();*

*Hive>describe formatted departments*

## Data Analysis

### Hive Table that Uses ORC File Format

The upcoming Hive 0.12 is set to bring some great new advancements in the storage layer in the forms of higher compression and better query performance.

1. HIGHER COMPRESSION-- save 78% of text file size.
2. PREDICATE PUSHDOWN-- where clause condition is pushed to storage layer.

ORCFile breaks rows into row groups and applies columnar compression and indexing within these row groups.

*CREATE TABLE mytable (*

*...*

*) STORED AS orc;*

To convert existing data to ORCFile create a table with the same schema as the source table plus stored as orc, then you can use issue a query like:

*INSERT INTO TABLE orctable SELECT \* FROM oldtable;*

|  |  |  |
| --- | --- | --- |
| **Key** | **Default** | **Notes** |
| **orc.compress** | **ZLIB** | Compression to use in addition to columnar  compression (one of NONE, ZLIB, SNAPPY) |
| **orc.compress.size** | **262,144 (= 256KiB)** | Number of bytes in each compression chunk |
| **orc.stripe.size** | **268,435,456 (=256 MiB)** | Number of bytes in each stripe |
| **orc.row.index.stride** | **10,000** | Number of rows between index entries (must be >= 1,000) |
| **orc.create.index** | **TRUE** | Whether to create inline indexes |

For example let’s say you wanted to use snappy compression instead of zlib compression. Here’s how:

*CREATE TABLE mytable (*

*...*

*) STORED AS orc tblproperties ("orc.compress"="SNAPPY");*

CTAS (Create AS) with ORC format:

*create table orders\_orc*

*row format delimited fields terminated by ':'*

*stored as orc*

*as*

*select \* from retail\_stage.orders;*

*describe formatted orders\_orc;*

*show create table orders\_orc;*

*create table orders\_default\_serde(*

*order\_id int,*

*order\_date bigint,*

*order\_customer\_id int,*

*order\_status strig)*

*stored as orc;*

*show create table orders\_default\_serde*

Except for the text file there is no requirement to specify the "row format"

### Hive Table Storage Format and Delimiters

Hive>nulls defined as -1

### Hive DML

**Insert**

*CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3, 2))*

*CLUSTERED BY (age) INTO 2 BUCKETS STORED AS ORC;*

*INSERT INTO TABLE students*

*VALUES ('fred flintstone', 35, 1.28), ('barney rubble', 32, 2.32);*

*CREATE TABLE pageviews (userid VARCHAR(64), link STRING, came\_from STRING)*

*PARTITIONED BY (datestamp STRING) CLUSTERED BY (userid) INTO 256 BUCKETS STORED AS ORC;*

*INSERT INTO TABLE pageviews PARTITION (datestamp = '2014-09-23')*

*VALUES ('jsmith', 'mail.com', 'sports.com'), ('jdoe', 'mail.com', null);*

*INSERT INTO TABLE pageviews PARTITION (datestamp)*

*VALUES ('tjohnson', 'sports.com', 'finance.com', '2014-09-23'), ('tlee', 'finance.com', null, '2014-09-21');*

**Synopsis**

* Each row listed in the VALUES clause is inserted into table tablename.
* Values must be provided for every column in the table. The standard SQL syntax that allows the user to insert values into only some columns is not yet supported. To mimic the standard SQL, nulls can be provided for columns the user does not wish to assign a value to.
* Dynamic partitioning is supported in the same way as for INSERT...SELECT.
* If the table being inserted into supports ACID and a transaction manager that supports ACID is in use, this operation will be auto-committed upon successful completion.
* Hive does not support literals for complex types (array, map, struct, union), so it is not possible to use them in INSERT INTO...VALUES clauses. This means that the user cannot insert data into a complex datatype column using the INSERT INTO...VALUES clause.

**Update:**

Updates can only be performed on tables that support ACID. See Hive Transactions for details.

Standard Syntax:

*UPDATE tablename SET column = value [, column = value ...] [WHERE expression]*

* The referenced column must be a column of the table being updated.
* The value assigned must be an expression that Hive supports in the select clause. Thus arithmetic operators, UDFs, casts, literals, etc. are supported. Subqueries are not supported.
* Only rows that match the WHERE clause will be updated.
* Partitioning columns cannot be updated.
* Bucketing columns cannot be updated.
* In Hive 0.14, upon successful completion of this operation the changes will be auto-committed.

**Delete:**

Deletes can only be performed on tables that support ACID. See Hive Transactions for details.

Standard Syntax:

*DELETE FROM tablename [WHERE expression]*

* Only rows that match the WHERE clause will be deleted.
* In Hive 0.14, upon successful completion of this operation the changes will be auto-committed.

### Hive SubQuery (in From and Where)

*SELECT col FROM (SELECT a+b AS col FROM t1) t2*

*SELECT t3.col FROM (*

*SELECT a+b AS col FROM t1*

*UNION ALL*

*SELECT c+d AS col FROM t2) t3*

*SELECT \* FROM A*

*WHERE A.a IN (SELECT foo FROM B);*

*SELECT A FROM T1*

*WHERE EXISTS (SELECT B FROM T2 WHERE T1.X = T2.Y)*

### Hive Compressed File

Compression works on top of file foramt, if the file is text than compression is also in text.

How to load the compressed data into Hive table

file name to Load: large.deck.txt.gz

*hive>LOAD DATA LOCAL INPATH '/root/largedeck.txt.gz' INTO TABLE deck\_of\_cards;*

// it will hvae compressed file (with extension s\_01/00000\_0.deflate) and non compressed(if exists)

*hive>desribe formatted deck\_of\_cards*

*set hive.exec.compress.output;*

*set hive.exec.compress.output = true;*

*set io.seqfile.compression.type = BLOCK; --NONE/RECORD/BLOCK*

Note: if the data is pre-transformed than you can use LOAD command because you can't enforce any transformations, INSERT Command is slower than LOAD and uses MapReduce and support transformations.

Compressed Data Storage

Keeping data compressed in Hive tables has, in some cases, been known to give better performance than uncompressed storage; both in terms of disk usage and query performance. You can import text files compressed with Gzip or Bzip2 directly into a table stored as TextFile. The compression will be detected automatically and the file will be decompressed on-the-fly during query execution. For example:

*CREATE TABLE raw (line STRING)*

*ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n';*

*LOAD DATA LOCAL INPATH '/tmp/weblogs/20090603-access.log.gz' INTO TABLE raw;*

The table 'raw' is stored as a TextFile, which is the default storage. However, in this case Hadoop will not be able to split your file into chunks/blocks and run multiple maps in parallel. This can cause underutilization of your cluster's 'mapping' power. The recommended practice is to insert data into another table, which is stored as a SequenceFile. A SequenceFile can be split by Hadoop and distributed across map jobs whereas a GZIP file cannot be. For example:

*CREATE TABLE raw (line STRING)*

*ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n';*

*CREATE TABLE raw\_sequence (line STRING)*

*STORED AS SEQUENCEFILE;*

*LOAD DATA LOCAL INPATH '/tmp/weblogs/20090603-access.log.gz' INTO TABLE raw;*

*SET hive.exec.compress.output=true;*

*SET io.seqfile.compression.type=BLOCK; -- NONE/RECORD/BLOCK (see below)*

*INSERT OVERWRITE TABLE raw\_sequence SELECT \* FROM raw;*

### Hive Through Vectorization

### Hive Explain Plan

*hive>set hive.execution.engine;*

*hive>set hive.execution.engine = tez;*

Join: When you impor the date(order\_date) through sqoop it will be in bigint.

*select from\_unixtime(cast(substr(order\_date,1,10) as bigint)) orderdtae,sum(oi.order\_item\_subtotal) revenue\_per\_day from orders o join order\_items oi*

*on 0.order\_id=oi.order\_item\_order\_id*

*group by from\_unixtime(cast(substr(order\_date,1,10) as bigint))*

*hive>set hive.execution.engine = mr;*

*select from\_unixtime(cast(substr(order\_date,1,10) as bigint)) orderdtae,sum(oi.order\_item\_subtotal) revenue\_per\_day from orders o join order\_items oi*

*on 0.order\_id=oi.order\_item\_order\_id*

*group by from\_unixtime(cast(substr(order\_date,1,10) as bigint))*

# get the explain plan

*Hive>explain select from\_unixtime(cast(substr(order\_date,1,10) as bigint)) orderdtae,sum(oi.order\_item\_subtotal) revenue\_per\_day from orders o join order\_items oi*

*on 0.order\_id=oi.order\_item\_order\_id*

*group by from\_unixtime(cast(substr(order\_date,1,10) as bigint))*

EXPLAIN [EXTENDED|DEPENDENCY|AUTHORIZATION] query

### Hive Property file Configuration

See the vedio

### Hive Query Ordered across Multiple Reducers

Order by is Total Ordering (single reducer), Sort by is not(multiple reducers),

### Hive Load Data using Select Statement

You can’t select the columns in Insert in Hive meaning,

Insert into table table\_name (col1, col2,col5) values (val1,val2,val5); is not poassible.

*hive>describe orders;*

*hive>insert into table deck\_cards\_01 select \* from deck\_cards;*

*hive>insert overwrite table deck\_cards\_01 select \* from deck\_cards;*

## Hive

### Partition

http://hadooptutorial.info/partitioning-in-hive/

Let us explore partitions with the help of sample by loading User records into Hive table partitioned\_user, partitioned by Country and State.

**first\_name,last\_name,address,country,city,state,post,phone1,phone2,email,web**

Rebbecca,Didio,171E24thSt,AU,Leith,TA,7315,03-8174-9123,0458-665-290,rebbecca.didio@didio.com.au,http://www.brandtjonathanfesq.com.au

Managed Partitioned Table:

*CREATE TABLE partitioned\_user(*

*firstname VARCHAR(64),*

*lastname  VARCHAR(64),*

*address   STRING,*

*city   VARCHAR(64),*

*post      STRING,*

*phone1    VARCHAR(64),*

*phone2    STRING,*

*email     STRING,*

*web       STRING*

*)*

*PARTITIONED BY (country VARCHAR(64), state VARCHAR(64))*

*STORED AS SEQUENCEFILE;*

*hive> DESCRIBE FORMATTEDpartitioned\_user;*

1. We don’t include country and state columns in table definition but include in partition definition.
2. Partitioned columns country and state can be used in query statements “where” clause and can be treated regular column names even though there is no actual column inside the input file data.

External Partitoned Table:

We can create external partitioned tables as well, just by using EXTERNAL keyword in the CREATE statement, but for creation of creation of external partitioned Tables, we don’t need to mention LOCATION clause as we will mention locations of each partitions separately while inserting data into the table.

Loading Data into External Partitioned Table From HDFS:

As data is already present in HDFS and should be made accessisble by Hive, we just need to mention the location of the HDFS files for each partition.

*ALTER TABLE partitioned\_user\_ext ADD PARTITON (country = ‘US’ , state = ‘CA’) LOCATION '/hive/external/tables/user/country=us/state=ca'*

Similarly we need to repeat the above alter command for all partition files in the directory so that a meta data entry will be created in metastore, mapping the partition and table.

**Inserting Data into Partitioned Tables:**

1. Static Partitioning
2. Dynamic Partitioning

### Static Partitioning

Static Partitioning:

In this mode, input data should contain the columns listed only in table definition but not the columns defined in partitioned by clause (country and state). If we have separate file for each combination of Country and State values, then there files can be easily loaded into partitioned tables with below syntax.

**Sample Data:**

|  |  |
| --- | --- |
|  | first\_name,last\_name,address,city,post,phone1,phone2,email,web  Rebbecca,Didio,171 E 24th St,Leith,7315,03-8174-9123,0458-665-290,rebbecca.didio@didio.com.au,http://www.brandtjonathanfesq.com.au  Stevie,Hallo,22222 Acoma St,Proston,4613,07-9997-3366,0497-622-620,stevie.hallo@hotmail.com,http://www.landrumtemporaryservices.com.au  Mariko,Stayer,534 Schoenborn St #51,Hamel,6215,08-5558-9019,0427-885-282,mariko\_stayer@hotmail.com,http://www.inabinetmacreesq.com.au  *LOAD DATA LOCAL INPATH ‘…/staticinput.txt’ INTO TABLE partitioned\_user PARTITION (country =’US’, state=’CA’)* |
|  | This will create separate directory under the default warehouse directory in HDFS  *Hadoop fs –ls /user/hive/warehouse/partitioned\_user/country=US/state=CA* |

We can load or add partitions with query results from another table as shown below.

*INSERT OVERWRITE TABLE partitioned\_user*

*PARTITION (country=’US’, state=’AL’)*

*SELECT \* FROM another\_user au*

*WHERE au.country = ‘US’ AND au.store=’AL’*

We can Overwrite existing partition with help of “OVERWRITE INTO TABLE partitioned\_table” clause.

### Dynamic Partitioning

Instead of loading each partition with single SQL statement as shown above, which will result in writing lot of SQL statements for huge number of partitions, Hive supports dynamic partitioning with which we can add any number of partitions with single SQL execution. Hive will automatically splits our data into separate partition files based on the values of partition keys present in the input files.

*INSERT INTO TABLE partitioned\_user*

*PARTITION (country,state)*

*SELECT  firstname,*

*lastname  ,*

*address   ,*

*city      ,*

*post      ,*

*phone1    ,*

*phone2    ,*

*email     ,*

*web       ,*

*country   ,*

*state*

*FROM temp\_user;*

We can also mix dynamic and static partitions by specifying it as PARTITION(country=’US’,state). But static partition keys must come before dynamic partition keys.

*set hive.exec.dynamic.partition=true;*

*set hive.exec.dynamic.partition.mode=nonstrict;*

*set hive.exec.max.dynamic.partitions=1000;*

*set hive.exec.max.dynamic.partitions.pernode=1000;*

Use Case for Dynamic partitioning

*set hive.exec.dynamic.partition=true;*

*set hive.exec.dynamic.partition.mode=nonstrict;*

*set hive.exec.max.dynamic.partitions.pernode=1000;*

*DROP TABLE IF EXISTS partitioned\_user;*

*CREATE TEMPORARY TABLE temp\_user(*

*firstname VARCHAR(64),*

*lastname  VARCHAR(64),*

*address   STRING,*

*country   VARCHAR(64),*

*city      VARCHAR(64),*

*state     VARCHAR(64),*

*post      STRING,*

*phone1    VARCHAR(64),*

*phone2    STRING,*

*email     STRING,*

*web       STRING*

*)*

*ROW FORMAT DELIMITED*

*FIELDS TERMINATED BY ','*

*LINES TERMINATED BY '\n'*

*STORED AS TEXTFILE;*

*LOAD DATA LOCAL INPATH '/home/siva/UserRecords.txt' INTO TABLE temp\_user;*

*SELECT firstname,phone1,city*

*FROM temp\_user*

*WHERE country='US' AND state='CA'*

*ORDER BY city*

*LIMIT 5;*

*CREATE TABLE partitioned\_user(*

*firstname VARCHAR(64),*

*lastname  VARCHAR(64),*

*address   STRING,*

*city   VARCHAR(64),*

*post      STRING,*

*phone1    VARCHAR(64),*

*phone2    STRING,*

*email     STRING,*

*web       STRING*

*)*

*PARTITIONED BY (country VARCHAR(64),state VARCHAR(64))*

*STORED AS SEQUENCEFILE;*

*INSERT INTO TABLE partitioned\_user*

*PARTITION (country,state)*

*SELECT  firstname,*

*lastname  ,*

*address   ,*

*city      ,*

*post      ,*

*phone1    ,*

*phone2    ,*

*email     ,*

*web       ,*

*country   ,*

*state*

*FROM temp\_user;*

*SELECT firstname,phone1,city*

*FROM partitioned\_user*

*WHERE country='US' AND state='CA'*

*ORDER BY city*

*LIMIT 5;*

We can see the partitions of a partitioned table with show command.

*SHOW PARTITIONS partitioned\_users;*

*SHOW PARTITIONS partitioned\_users PARTITION(country=’US’)*

*DESCRIBE FORMATTED partitioned\_user PARTITION (country=’US’,state=’CA’);*

### Alter Partitioning

We can alter/change partitions(add/change/drop) with the help of below commands.

Adding Partitions:

We can add partitions to an existing table with ADD PARTITION clause as shown below,

*ALTER TABLE partitioned\_user ADD IF NOT EXISTS*

*PARTITION (country=’US’, state =’xy’) LOCATION ‘/hdfs/external/file/path1’*

*PARTITION (country=’CA’, state =’yZ’) LOCATION ‘/hdfs/external/file/path2’*

Changing Partitions:

We can change a partition location with commands like below. This command does not move the data from the old but the reference to old data file will be lost.

*ALTER TABLE partitioned\_user*

*PARTITION (country=’US’, state =’xy’) SET LOCATION ‘/hdfs/external/file/newpath’*

Droping Partitions:

We can drop partitions ofa table with DROP IF EXISTS PARTITION clause as shown below,

*ALTER TABLE partitioned\_user DROP IF EXISTS*

*PARTITION(country=’US’, state=’CA’);*

# HDFS Processing Layer -Map Reduce

In Hadoop, storage is provided by Hadoop Distributed File System (HDFS) which is modeled after Google's GFS. Compute is provided by MapReduce, which is modeled after Google's MapReduce

HDFS is scalable , fault tolerant ,distributed storage system that works closely with a wide variety of concurrent data access applications, coordinated by YARN.

HDFS Cluster is comprised of NameNode, which manages the cluster metadata, and DataNode that store the data. Files and directories are represented on the NameNode by inodes. Inodes record attributes like permissions,modification and access times, or namespace and disk space quotas. Since each machine is part of the 'storage', we will have a 'daemon' running on each machine to manage storage for that machine. These daemons will talk to each other to exchange data.

HDFS is a Java based file system that provides scalable and reliable data storage and it was designed to span large cluster of commodity servers. HDFS is now extended to support heterogeneous storage media within the HDFS Cluster.

HDFS is better suited for large files;Generic file systems, say like Linux EXT file systems, will store files of varying size, from a few bytes to few gigabytes. HDFS, however, is designed to store large files. Large as in a few hundred megabytes to a few gigabytes. HDFS was built to work with mechanical disk drives, whose capacity has gone up in recent years. However, seek times haven't improved all that much. So Hadoop tries to minimize disk seeks.

Files are write-once only (not updateable);HDFS supports writing files once (they cannot be updated). This is a stark difference between HDFS and a generic file system (like a Linux file system). A generic file system allows files to be modified. However appending to a file is supported. Appending is supported to enable applications like HBase.

HBase, the database for Big Data;Not properly an animal, HBase is nevertheless very powerful. It is currently denoted by the letter H with a base clef. If you think this is not so great, you are right, and the HBase people are thinking of changing the logo. HBase is a database for Big Data, up to millions of columns and billions of rows. Another feature of HBase is that it is a key-value database, not a relational database. We will get into the pros and cons of these two approaches to databases later, but for now let's just note that key-value databases are considered as more fitting for Big Data. Why? Because they don't store nulls! This gives them the appellation of "sparse,".

Hive - data warehousing; "I am Hive, I let you in and out of the HDFS cages, and you can talk SQL to me!"

Hive is a way for you to get all the honey, and to leave all the work to the bees. You can do a lot of data analysis with Hadoop, but you will also have to write MapReduce tasks. Hive takes that task upon itself. Hive defines a simple SQL-like query language, called QL, that enables users familiar with SQL to query the data. At the same time, if your Hive program does almost what you need, but not quite, you can call on

Pig - Big Data manipulation;"I am Pig, I let you move HDFS cages around, and I speak Pig Latin."

Pig is called pig not because it eats a lot, although you can imagine a pig pushing around and consuming big volumes of information. Rather, it is called pig because it speaks Pig Latin. Others who also speak this language are the kids (the programmers) who visit the Hadoop zoo.

So what is Pig Latin that Apache Pig speaks? As a rough analogy, if Hive is the SQL of Big Data, then Pig Latin is the language of the stored procedures of Big Data. It allows you to manipulate large volumes of information, analyze them, and create new derivative data sets. Internally it creates a sequence of MapReduce jobs, and thus you, the programmer-kid, can use this simple language to solve pretty sophisticated large-scale problems.

your MapReduce skill. Hive allows you to write custom mappers and reducers to extend the QL capabilities.

ZooKeeper - Every zoo has a zoo keeper, and the Hadoop zoo is no exception. When all the Hadoop animals want to do something together, it is the ZooKeeper who helps them do it. They all know him and listen and obey his commands. Thus, the ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

ZooKeeper is also fault-tolerant. In your development environment, you can put the zookeeper on one node, but in production you usually run it on an odd number of servers, such as 3 or 5.

## Hadoop Components

1. Hadoop Common:
   1. Contains libraries and utilities needed by other Hadoop Modules.
2. HDFS (Hadoop Distributed File System):
   1. Distributed file system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
3. Hadoop Yarn:
   1. Resource management platform responsible for managing computing resources in clusters and using them for scheduling of users applications.
4. Hadoop MapReduce:
   1. Programming model for large scale data processing.

### Map Reduce

<http://hortonworks.com/hadoop-tutorial/hello-world-an-introduction-to-hadoop-hcatalog-hive-and-pig/#section_3>

<http://hadoopilluminated.com/hadoop_illuminated/HDFS_Intro.html>

<http://hadoopilluminated.com/hadoop_illuminated/MapReduce_Intro.html>

MapReduce is a programming framework. Its description was [published by Google in 2004](http://research.google.com/archive/mapreduce.html). Much like other frameworks, such as Spring, Struts, or MFC, the MapReduce framework does some things for you, and provides a place for you to fill in the blanks. What MapReduce does for you is to organize your multiple computers in a cluster in order to perform the calculations you need. It takes care of distributing the work between computers and of putting together the results of each computer's computation. Just as important, it takes care of hardware and network failures, so that they do not affect the flow of your computation. You, in turn, have to break your problem into separate pieces which can be processed in parallel by multiple machines, and you provide the code to do the actual calculation.

MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. Cluster size doesn't affect a processing job's final results; jobs can be split across almost anynumber of servers.

If a node doesn't respond as expected, the master node re-assigns that piece of the job to other available nodes in the cluster.MR Provides automatic parallelization and distribution, fault tolerant, I/O operation and status updates.

### Map Stage

Map or Mapper’s job Task is to process the input data.Generally the input data is in the form of file or directory and is stored in the (HDFS). Mapper tasks are assigned, where possible, to a slave node where the input split is stored (data locality).

Input file is passed to the mapper function line by line.Mapper processes the data and creates several small chunks of data that is in the form of key value pairs.

### Reduce Stage

Its combination of the Shuffle stage and the Reduce stage.Reducer’s job is to process the data that comes from the mapper such as consolidation, aggregation or sorting.Produces a new set of output, which will be stored in the HDFS.

### Input Split

Hadoop uses a logical representation of the data stored in file blocks, known as input splits. HDFS breaks down very large files into large blocks (for example, measuring 128MB), and stores three copies of these blocks on different nodes in the cluster.

Number of input splits that are calculated for a specific application determines the number of mapper tasks.MapReduce job client calculates the input splits, it figures out where the first whole record in a block begins and where the last record in the block ends.

In cases where the last record in a block is incomplete, the input split includes location information for the next block and the byte offset of the data needed to complete the record.

## Hadoop 1 Processing Daemons

### Name Node

The file content is split into large blocks (typically 128 megabytes), and each block of the file is independently at multiple DataNodes. The blocks are stored on the local file system on the DataNodes.

The NameNode actively monitors the number of replicas of a block. When a replica of a block is lost due to a DataNode failure or disk failure, the NameNode creates another replica of the block.

The NameNode maintains the namespace tree and mapping of blocks to DataNodes, holding the entire namespace image in RAM.The NameNode does not directly send request to DataNodes. It sends instruction to the DataNodes by replying to heartbeats sent by those DataNodes. The instructions include commands to ,

* Replicate blocks to other nodes,
* Remove local block replicas
* Re-Register and send an immediate block report, or
* Shut down the node.

### Job Tracker

Job Tracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop. There is only One Job Tracker process run on any hadoop cluster. Job Tracker runs on its own JVM process. In a typical production cluster its run on a separate machine. Each slave node is configured with job tracker node location

Every 5 seconds Task tracker send signal to reassure the JobTracker that it is still alive. JobTracker is single point of failure for the Hadoop MapReduce service.

1. Client applications submit jobs to the Job tracker.
2. The JobTracker talks to the NameNode to determine the location of the data.
3. The JobTracker locates TaskTracker nodes with available slots at or near the data.
4. The JobTracker submits the work to the chosen TaskTracker nodes.
5. The TaskTracker nodes are monitored. If they do not submit heartbeat signals often enough,

they are deemed to have failed and the work is scheduled on a different TaskTracker.

1. A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do

then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid,and it may even blacklist the TaskTracker as unreliable.

1. When the work is completed, the JobTracker updates its status.
2. Client applications can poll the JobTracker for information.

### Task Tracker

1. A TaskTracker is a slave node daemon in the cluster that accepts tasks (Map, Reduce and Shuffle operations) from a JobTracker.
2. There is only One Task Tracker process run on any Hadoop data node.
3. Task Tracker runs on its own JVM process.
4. Every TaskTracker is configured with a set of slots; these indicate the number of tasks that it canaccept. The TaskTracker starts a separate JVM processes to do the actual work (called as TaskInstance) this is to ensure that process failure does not take down the task tracker.
5. The TaskTracker monitors these task instances, capturing the output and exit codes. When the Task instances finish, successfully or not, the task tracker notifies the JobTracker.
6. The TaskTrackers also send out heartbeat messages to the JobTracker, usually every 5 seconds, toreassure the JobTracker that it is still alive. These messages also inform the JobTracker of the numberof available slots, so the JobTracker can stay up to date with where in the cluster work can bedelegated.

### How Does MapReduce Work

1. MapReduce has a master and workers, but it is not all push or pull, rather, the work is a collaborative effort between them.
2. The master assigns a work portion to the next available worker; thus, no work portion is forgotten or left unfinished.
3. Workers send periodic heartbeats to the master. If the worker is silent for a period of time (usually 10 minutes), then the master presumes this worker crashed and assigns its work to another worker. The master also cleans up the unfinished portion of the crashed worker.
4. All of the data resides in HDFS, which avoids the central server concept, with its limitations on concurrent access and on size. MapReduce never updates data, rather, it writes new output instead. This is one of the features of functional programming, and it avoids update lockups.
5. MapReduce is network and rack aware, and it optimizes the network traffic.

Masters and slaves

MapReduce has a master and slaves, and they collaborate on getting the work done. The master is listed in the "masters" configuration file, and the slaves are listed in the "slaves", and in this way they know about each other. Furthermore, to be a real "master", the node must run a daemon called the "Job Tracker" daemon. The slave, to be able to do its work, must run another daemon, called the "Tasktracker" daemon.

The master does not divide all the work beforehand, but has an algorithm on how to assign the next portion of the work. Thus, no time is spent up front, and the job can begin right away. This division of labor, how much to give to the next Tasktracker, is called "split", and you have control over it. By default, the input file is split into chunks of about 64MB in size. About, because complete lines in the input file have to be preserved.

### MapReduce is Stable

Recall that in my system I gave the responsibility for selecting the next piece of work to the workers. This created two kinds of problems.

1. When a worker crashed, nobody knew about it. Of course, the worker would mark the work as "done" after it was completed, but when it crashed, there was nobody to do this for him, so it kept hanging. You needed watchers over watchers, and so on.
2. Another problem would be created when two overzealous workers wanted the same portion. There was a need to somehow coordinate this effort. My solution was a flag in the database, but then this database was becoming the real-time coordinator for multiple processes, and it is not very good at that. You can image multiple scenarios when this would fail.

By contrast, in MapReduce the Job Tracker doles out the work. There is no contention: it takes the next split and assigns it to the next available Tasktracker. If a Tasktracker crashes, it stops sending heartbeats to the Job Tracker.

### MapReduce Uses Functional PRogramming

MapReduce works on data that resides in HDFS. As described in the previous section, HDFS (Hadoop Distributed File System) is unlimited and linearly scalable, that is, it grows by adding servers. Thus the problem of a central files server, with its limited capacity, is eliminated.

Moreover, MapReduce never updates data, rather, it writes a new output instead. This is one of the principles of [functional programming](http://en.wikipedia.org/wiki/Functional_programming) , and it avoids update lockups. It also avoids the need to coordinate multiple processes writing to the same file; instead, each Reducer writes to its own output file in an HDFS directory, designated as output for the given job. The Reducer's output file is named using the Reducer ID, which is unique. In further processing, MapReduce will treat all of the files in the input directory as its input, and thus having multiple files either in the input or the output directory is no problem.

### MapReduce Optimizes network Traffic

As it turns out, network bandwidth is probably the most precious and scarce resource in a distributed system and should be used with care. It is a problem which I have not seen even in my eDiscovery application, because it needs to be correct and stable before optimizing, and getting there is not an easy task.

MapReduce, however, notes where the data is (by using the IP address of the block of data that needs to be processed) and it also knows where the Task Tracker is (by using its IP address). If it can, MapReduce assigns the computation to the server which has the data locally, that is, whose IP address is the same as that of the data. Every Task Tracker has a copy of the code that does the computation (the job's jar, in the case of Java code), and thus the computation can begin.

If local computation is not possible, MapReduce can select the server that is at least closest to the data, so that the network traffic will go through the least number of hops. It does it by comparing the IPs, which have the distance information encoded. Naturally, servers in the same rack are considered closer to each other than servers on different racks. This property of MapReduce to follow the network configuration is called "rack awareness". You set the rack information in the configuration files and reap the benefits.

### Limitation of Haddop 1

Job tracker is overburden

JobTracker runs on single machine doing several tasks like

Resource allocation and administration.

Job and task scheduling, performing a speculative execution of tasks

Monitoring, although there are so many machines (Data Node) available; they are not getting used. This limitsscalability of JobTracker.

Job tracker is single point of failure

Fixed slots for Map & Reduce tasks

There is concept of predefined number of map slots and reduce slots for each TaskTrackers.

MR is only Framework supported

Job tracker was tightly integrated with MapReduce and only supporting application that obeys MapReduce programming framework can run on Hadoop.

Problem in performing real-time analysis:

Apache Storm which can work better in this case. But in Hadoop 1.0, due to tight coupling these engines cannot run independently.

Problem in running Message-Passing approach:

It is a stateful process that runs on each node of a distributed network. The processes communicate with each other by sending messages, and alter their state based on the messages they receive. This is not possible in MapReduce.

Problem in running Ad-hoc query:

Many users like to query their big data using SQL. Apache Hive can execute a SQL query as a series of MapReduce jobs, but it has shortcomings in terms of performance. Recently, some new approaches such as Apache Tajo , Facebook's Presto and Cloudera's Impala drastically improve the performance, but they require to run services in other form than MapReduceform.Such jobs have to "disguise" themselves as mappers and reducers in order to be able to run on Hadoop 1.0.

Backward compatibility issue

Older version of Map reduces MR1 could not run in the latest MR1 versions, whereas YARN is backward compatible to run older versions.

### Haddop Ecosystem & Others

Hadoop Compute Frameworks

| Tool | Remarks |
| --- | --- |
| [Map reduce](http://hadoop.apache.org/docs/stable/mapred_tutorial.html) | Original distributed compute framework of Hadoop |
| [YARN](http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html) | Next generation MapReduce, available in Hadoop version 2.0 |
| [Weave](https://github.com/continuuity/weave) | Simplified YARN programming |
| Cloudera SDK | Simplified MapReduce programming |

Most of the big data originates outside the Hadoop cluster. These tools will help you get data into HDFS.

Tools for Getting Data into HDFS

| Tool | Remarks |
| --- | --- |
| [Flume](http://flume.apache.org/) | Gathers data from multiple sources and gets it into HDFS. |
| [Scribe](https://github.com/facebook/scribe) | Distributed log gatherer, originally developed by Facebook. It hasn't been updated recently. |
| [Chukwa](http://incubator.apache.org/chukwa/) | Data collection system. |
| [Sqoop](http://sqoop.apache.org) | Transfers data between Hadoop and Relational Databases (RDBMS) |
| [Kafka](http://kafka.apache.org/) | Distributed publish-subscribe system. |

Querying Data in HDFS

| Tool | Remarks |
| --- | --- |
| [Java MapReduce](http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html) | Native mapreduce in Java |
| [Hadoop Streaming](http://hadoop.apache.org/docs/r1.1.2/streaming.html) | Map Reduce in other languages (Ruby, Python) |
| [Pig](http://pig.apache.org/) | Pig provides a higher level data flow language to process data. Pig scripts are much more compact than Java Map Reduce code. |
| [Hive](http://hive.apache.org) | Hive provides an SQL layer on top of HDFS. The data can be queried using SQL rather than writing Java Map Reduce code. |
| [Cascading Lingual](http://www.cascading.org/lingual/) | Executes ANSI SQL queries as Cascading applications on Apache Hadoop clusters. |
| [Stinger](http://hortonworks.com/blog/100x-faster-hive/) / [Tez](http://hortonworks.com/blog/introducing-tez-faster-hadoop-processing/) | Next generation Hive. |
| [Hadapt](http://hadapt.com/) | Provides SQL support for Hadoop.  (commercial product) |
| [Greenplum HAWQ](http://www.greenplum.com/blog/topics/hadoop/introducing-pivotal-hd) | Relational database with SQL support on top of Hadoop HDFS.  (commercial product) |
| [Cloudera Search](http://www.cloudera.com/content/cloudera/en/campaign/introducing-search.html) | Text search on HDFS |
| [Impala](http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/) | Provides real time queries over Big Data. Developed by Cloudera. |
| [Presto](http://prestodb.io/) | Developed by Facebook, provides fast SQL querying over Hadoop |

Work flow Tools

| Tool | Remarks |
| --- | --- |
| [Oozie](http://oozie.apache.org/) | Orchestrates map reduce jobs. |
| [Azkaban](http://data.linkedin.com/opensource/azkaban) |  |
| [Cascading](http://www.cascading.org/) | Application framework for Java developers to develop robust Data Analytics and Data Management applications on Apache Hadoop. |
| [Scalding](https://github.com/twitter/scalding) | Scala library that makes it easy to specify Hadoop MapReduce jobs. Scalding is built on top of Cascading. |
| [Lipstick](https://github.com/Netflix/Lipstick) | Pig work flow visualization |

Serialization Frameworks

| Tool | Remarks |
| --- | --- |
| [Avro](http://avro.apache.org/) | Data serialization system. |
| [Trevni](https://github.com/cutting/trevni) | Column file format. |
| [Protobuf](https://code.google.com/p/protobuf/) | Popular serialization library (not a Hadoop project). |
| [Parquet](http://parquet.io/) | columnar storage format for Hadoop |

NoSQL stores for Big Data

| Tool | Remarks |
| --- | --- |
| [HBase](http://hbase.apache.org/) | NoSQL built on top of Hadoop. |
| [Cassandra](http://cassandra.apache.org/) | NoSQL store (does not use Hadoop). |
| [Redis](http://redis.io/) | Key value store. |
| [Amazon SimpleDB](http://aws.amazon.com/simpledb/) | Offered by Amazon on their environment. |
| [Voldermort](http://www.project-voldemort.com/voldemort/) | Distributed key value store developed by LinkedIn. |
| [Accumulo](http://accumulo.apache.org/) | A NoSQL store developed by NSA (yes, that agency!). |

Tools for Monitoring Hadoop

| Tool | Remarks |
| --- | --- |
| [Hue](http://cloudera.github.com/hue/) | Developed by Cloudera. |
| [Ganglia](http://ganglia.sourceforge.net/) | Overall host monitoring system. Hadoop can publish metrics to Ganglia. |
| [Open TSDB](http://opentsdb.net/) | Metrics collector and visualizer. |
| [Nagios](http://www.nagios.org/) | IT infrastructure monitoring. |

Data Analytics on Hadoop

| Tool | Remarks |
| --- | --- |
| [R language](http://www.r-project.org/) | Software environment for statistical computing and graphics. |
| [RHIPE](http://www.datadr.org/) | Integrates R and Hadoop. |

## Hadoop 2 - YARN

YARN supports multiple processing models in addition to MapReduce. One of the most significant

benefits of this is that we are no longer limited to working the often I/O intensive, high latency

MapReduce framework.

### Architectural Changes

Job tracker and Task tracker are replaced by Resource manager, Application Master, Node manager and Containers.

Job Tracker is split into two different daemons called Resource Manager and Node Manager Task Tracker is split into Appmaster and container.

The resource manager only manages the allocation of resources to the different applications

submitted and maintains the scheduler which just takes care of the scheduling jobs without worrying about any monitoring or status updates.

### Yarn Components

1. ResourceManager (RM)
2. ApplicationsManager
3. NodeManager (NM)
4. ApplicationMaster (AM)
5. Container
6. ResourceManager (RM)

### Resource Manager

The ResourceManager (RM) is the key service offered in YARN. Clients can interact with the frameworkusing RM. RM is the master for all other daemons available in theframework.RM has two major components,

* Scheduler
* ApplicationsManager

### Scheduler

1. The Scheduler is responsible for allocating resources to the various running applications subject tofamiliar constraints of capacities, queues etc.
2. It is a pure scheduler since it does not monitor or track the status of the application instead it purelyperforms its scheduling function based the resource requirements of the applications
3. It schedules the resources depending on the resource “Container”
4. The Scheduler has the pluggable policy plug-in which is responsible for partitioning the cluster
5. resources among the various queues, applications etc, for example a) CapacityScheduler, b)FairScheduler

### Application Manager

1. Applications Manager is responsible for accepting job-submissions.
2. Assigning the first container for executing the application specific Application-Master.
3. Provides the service for restarting the Application-Master container on failure.

### Node Manager

1. Node-Manager is responsible for Container management.
2. Monitoring container resource usage (like cpu, memory, disk, network).
3. Reporting to the Resource-Manager/Scheduler.

### Application Master

1. Application Master is responsible for negotiating resources with the ResourceManager and forworking with the NodeManagers to start the containers.
2. Application-Master is responsible for negotiating appropriate resource containers from the Scheduler.
3. Tracking the status and monitoring progress for applications running in each containers under theApplication-Master.

### Container

Resource Container incorporates elements such as memory, cpu, disk, network, Command line to launchthe process within the container, Environment variables and Local resources necessary on the machineprior to launch, such as jars, shared-objects, auxiliary data files, Security-related tokens etc.

### YARN Process Flow

Let’s walk through an application execution sequence (steps are illustrated in the diagram):

1. A client program submits the application, including the necessary specifications to launch the application-specific ApplicationMaster itself.
2. The ResourceManager (Indeed ApplicationManger) assumes the responsibility to negotiate a specified container in which to startthe ApplicationMaster and then launches the ApplicationMaster.
3. The ApplicationMaster, on boot-up, registers with the ResourceManager – the registration allowsclient program to query the ResourceManager for details, which allow it to directly communicate with its own ApplicationMaster.
4. During normal operation the ApplicationMaster negotiates appropriate resource containers to RM.
5. On successful container allocations, the ApplicationMaster launches the container by providing thecontainer launch specification to the NodeManager. The launch specification, typically, includes thenecessary information to allow the container to communicate with the ApplicationMaster itself.
6. The application code executing within the container then provides necessary information (progress,status etc.) to its ApplicationMaster via an application-specific protocol.
7. During the application execution, the client that submitted the program communicates directly withthe ApplicationMaster to get status, progress updates etc. via an application-specific protocol.
8. Once the application is complete, and all necessary work has been finished, the ApplicationMaster deregisters with the ResourceManager and shuts down, allowing its own container to berepurposed.

### UBER Mode

In normally mappers and reducers will run by ResourceManager (RM), RM will create separate containerfor mapper and reducer. uber configuration, will allow to run mapper and reducers in the same processas the ApplicationMaster (AM) runs in the same container. Uber jobs are jobs that are executed withinthe MapReduce ApplicationMaster. Rather then communicate with RM to create the mapper andreducer containers. The AM runs the map and reduce tasks within its own process and avoided theoverhead of launching and communicate with remote containers. If you have a small dataset, want torun MapReduce on small amount of data. Uber configuration will help you out, by reducing additionaltime that MapReduce normally spends mapper and reducers phase.

# HDFS File Read operation

http://hadooptutorials.co.in/tutorials/hadoop/internals-of-hdfs-file-read-operations.html

Since HDFS plays with large sets of data it is interesting to know how it manages File I/O operations over it when requested by client. In the following section we are going to understand the flow of HDFS file read operation. Let's have a look at pictorial representation of the data read operation and components involved in it.

Before going to understand read operation let's take a walkthrough of few important components.

HDFS Client: HDFS client interacts with Namenode and Datanode on behalf of user to fulfil user request. User establishes communication with HDFS through File System API and normal I/O operations, processing of user request and providing response over it is carried out by File System API processes.

Namenode: Namenode is the masternode of HDFS cluster. It stores metadata information and edit log in it. Metadata information contains addresses of block locations of Datanodes, this information is used for file read and write operation to access the blocks in a HDFS cluster.

Datanode: Datanodes also known as slave nodes holds the actual data. Datanode only stores block, a block is what is used to store and process the data. Data resides within blocks of Datanode. Datanode gives periodic heartbeat signals to Masternode to indicate that it is alive and can be used to store and retrieve data

Packet: A Packet is a small chunk of data which is used during transmission; packet is a subset of block. The default size of a block is around 64 MB or 128 MB, it will create a huge network overload if we transfer data of the size of blocks, hence client API transfers this block in small chunks known as packets.

Next we will go through data flow in HDFS file read operation.

HDFS File Read workflow

1. To start the file read operation, client opens the required file by calling open() on Filesystem object which is an instance of DistributedFileSystem. Open method initiate HDFS client for the read request.

2. DistributedFileSystem interacts with Namenode to get the block locations of file to be read. Block locations are stored in metadata of namenode. For each block,Namenode returns the sorted address of Datanode that holds the copy of that block.Here sorting is done based on the proximity of Datanode with respect to Namenode, picking up the nearest Datanode first.

3. DistributedFileSystem returns an FSDataInputStream, which is an input stream to support file seeks to the client. FSDataInputStream uses a wrapper DFSInputStream to manage I/O operations over Namenode and Datanode. Following steps are performed in read operation.

a) Client calls read() on DFSInputStream. DFSInputStream holds the list of address of block locations on Datanode for the first few blocks of the file. It then locates the first block on closest Datanode and connects to it.

b) Block reader gets initialized on target Block/Datanode along with below information:

* Block ID.
* Data start offset to read from.
* Length of data to read.
* Client name.

c) Data is streamed from the Datanode back to the client in form of packets, this data is copied directly to input buffer provided by client.DFS client is reading and performing checksum operation and updating the client buffer

d) Read () is called repeatedly on stream till the end of block is reached. When end of block is reached DFSInputStream will close the connection to Datanode and search next closest Datanode to read the block from it.

4. Blocks are read in order, once DFSInputStream done through reading of the first few blocks, it calls the Namenode to retrieve Datanode locations for the next batch of blocks.

5. When client has finished reading it will call Close() on FSDataInputStream to close the connection.

6. If Datanode is down during reading or DFSInputStream encounters an error during communication, DFSInputStream will switch to next available Datanode where replica can be found. DFSInputStream remembers the Datanode which encountered an error so that it does not retry them for later blocks.

As you can see that client with the help of Namenode gets the list of best Datanode for each block and communicates directly with Datanode to retrieve the data. Here Namenode serves the address of block location on Datanode rather than serving data itself which could become the bottleneck as the number of clients grows. This design allows HDFS to scale up to a large numbers of clients since the data traffic is spread across all the Datanodes of clusters.

# Hadoop - Data Life Cycle Management

Hadoop as a part of data Life cycle Management.

* Data infrastructure (collect multiple, unstructured data )
* Data preprocessing
* Data creation (Fuse of new data, returns the data to RDBMS)
* Data Archival

## Sqoop Hive Import

Steps for importing data

* Create the table in Hive
* Import the data into Hive Table (MYSQL to Hive)
* Validate the data with Hive Query.

### PArameter-Sqoop Hive Import

MySQL:

* --table
* --columns
* --where

Hive:

* --hive – import
* --hive – table
* --map – column –hive
* --hive – overwrite

### Pitfalls -Sqoop Hive Import

* Field Terminators
* Line Terminators
* Numeric Data types
* Date time Data types

Note: Direct import to the Hive Partitions are possible and we need to write individual select statement.

// Create a customer table in MySQL

*>CREATE TABLE invoices(*

*InvNum INT (10) primary key not null,*

*OrderDate Date,*

*Region VARCHAR(15),*

*SaleRep VARCHAR(15),*

*Item VARCHAR(20),*

*Units INT,*

*UnitPrice Decimal(10,2),*

*Total Decimal(10,2));*

*LOAD DATA*

*LOCAL INFILE ‘path../petshopinvoices.csv’*

*INTO TABLE invoices*

*FIELDS TERMINATED BY ‘,’*

*LINES TERMINATED BY ‘\n’;*

// create new database in Hive

*>show databases;*

*>create database pets;*

*>use pets;*

//lets see what as happened

*>hdfs dfs –ls /user/hive/warehouse*

//new directory called pets.db will be created.

// Use Sqoop to import invoice table

*Sqoop import*

*--connect jdbc:mysql://hdcentos/CRMDB \*

*--username sqoopuser –password Mayyam%123 \*

*--table invoices \*

*--hive –import \*

*--hive –table pet.invoices \*

*--num –mappers 1*

// Use Sqoop to import data of sales rep where sales is greater than 500$

*Sqoop import*

*--connect jdbc:mysql://hdcentos/CRMDB \*

*--username sqoopuser –password Mayyam%123 \*

*--table invoices \*

*--columns “InvNum,OrderDate,SalesRep,Total” \*

*--where “Total >= 500” \*

*--hive –import \*

*--hive –overwrite \*

*--hive –table pet.repstotals \*

*--num –mappers 1*

## Hive Export To MySQL through SQOOP

You cann’t export the whole table from hive into MySQL, because Hive is just a meta data of the file distributed in Hadoop , So Hive table is not actually table, but a file. So Sqoop export requires attention to detail when exporting Hive table into SQL database. Details like Delimiter(Field and numeric), Numeric data types and Date and time data types.

### PArameters For Sqoop Export

**MySQL:**

*--table*

*--update-key*

*--update-mode*

**Hive:**

*--export-dir*

*--input-fields-terminated-by*

*--input-lines-terminated-by*

**Two Optional parameters:**

Parameters for Updating Records

You can use the parameter to control either to be a insert statement or update statement.

*--update-key // primary key on the Sqoop table.*

*--update-mode //control update or insert.*

### Sqoop Export –case 1

// They cann't create the relational table in MySQL. Table must be predefined as a best practice copy the scehema of the table and then load the data and rename.

// In Mysql

*create table hiveinvoices like invoices;*

*describe hiveinvoices;*

*select \* from hiveinvoices;*

//In hdfs

*hdfs dfs -ls /user/hive/warehouse/pets.db*

//invoices directory

*hdfs dfs -ls /user/hive/warehouse/pets.db/invoices*

//remove the \_SUCCESS file before export

*hdfs dfs -rm /user/hive/warehouse/pets.db/invoices/\_SUCCESS*

//In SQOOP

*sqoop export \*

*--connect 'jdbc:mysql://hdcentos/CRMDB' \*

*--username uname --password pword \*

*--table cleaninvoices \*

*--export-dir /user/hive/warehouse/pets.db/invoices \*

*--input-field-terminated-by ‘\001’ \*

*--input-line-terminated-by ‘\n’ \*

### Date Data Type

Limitations of Hadoop with Date Data type.

Data types for Date and Time:

1. Date

2. DateTime

3. TimeStamps

System Datetime Stamps

yyyy-mm-dd hh:mm:ss

2014-03-17 03:23:36

### Import - Date Data Type

HIVE Import From MYSQL THROUGH SQOOP :

Import Date data into Hive

1.Hive Cast using map-column-hive

2.Cast in select statement

*sqoop import \*

*--connect jdbc:mysql://hdcentos/CRMDB \*

*--username sqoopuser --password Mayyam%123 \*

*--table invoices \*

*--hive-import \*

*--hive--overwrite \*

*--hive-table pets.invoices \*

*--map-column-hive OrderDate=timestamp \ # cast*

*--num-mappers 1*

//We receive the null for the values.

*sqoop import \*

*--connect jdbc:mysql://hdcentos/CRMDB \*

*--username sqoopuser --password Mayyam%123 \*

*--target-dir invoices \*

*--query 'select InvNum,timestamp(OrderDate) as OrderDate,SalesRep,Item,Units,UnitPrice*

*,Total from invoices where $CONDITIONS' \*

*--hive-import \*

*--hive--overwrite \*

*--hive-table pets.invoices \*

*--map-column-hive OrderDate=timestamp \ # cast*

*--num-mappers 1*

//Now the right data is available, cast at SQL and Hive level.

//We receive the null for the values

### EXPORT - Date Data Type

HIVE EXPORT TO MYSQL THROUGH SQOOP

*sqoop export \*

*--connect 'jdbc:mysql://hdcentos/CRMDB' \*

*--username uname --password pword \*

*--table timeinvoices \*

*--export-dir /user/hive/warehouse/pets.db/timeinvoices \*

*--input-field-terminated-by ‘\001’ \*

*--input-line-terminated-by ‘\n’ \*

// you will get the java error.

//In Mysql

*ALTER TABLE timeinvoices MODIFY COLUMN OrderDate TIMESTAMP;*

// we have OrderDate type as DATE and now changed to TIMESTAMP

// Run the export script again and you will see export completed.

# Hive

HiveQL is a declarative language where users issue declarative queries and Hive figures out how to translate them into MapReduce jobs. While the sophisticated process of query parsing, planning, optimization and execution are all taken care in the back ground.

## Setup of Hive

Configuration files

* + 1. hive-env.sh, stores environmental variables
    2. hive-log4j.properties, stores log file properties
    3. hive-site.xml, stores parameters

### Hive Site XML Changes

1. hive.metastore.warehouse.dir, sets the location to Hive warehouse
2. javax.jdo.option.ConnectionURL, JDBC connection URL tells where the database for metadata is located.
3. javax.jdo.option.ConnectionDriverName, Class name of the JDBC driver
4. javax.jdo.option.ConnectionUserName, Username for logging into the database.
5. javax.jdo.option.ConnectionPassword, Password for logging into the database.

## Hive

### Key Attributes

1. Batch Processing
2. HiveQL
3. Unstructured Data, doesn’t require data to be structured.
4. Metadata, Creates and stores metadata in a RDBMS.
5. Interfaces, Hive Shell and Web UI

### SQL Features

1. Select certain columns from the table using a select clause.
2. Filter rows from a table using a where clause.
3. Do equi-joins between two tables.
4. Evaluate aggregations on multiple “Group BY” columns for the data stored ina table.
5. Manage tables and partitions(Create, Drop and Alter)
6. Store the results of a query into another table.
7. Download the contents of a table to a local directory.
8. Download the contents of a Query into a HDFS directory.
9. Plug in custom scripts for custom MapReduce Jobs.

### Data Units Hierarchy

1. Databases: Namespaces that separate tables and other data units from naming confliction.
2. Tables: Homogeneous units of data which have the same schema.
3. Partitions: is the subdivision of a single table into multiple segments, each of which holds a subset of values. Each Table can have one or more partition Keys which determines how the data is stored.Partitions - apart from being storage units - also allow the user to efficiently identify the rows that satisfy a certain criteria.Sorting Key which determines how the data is stored.
4. Buckets (or Clusters): Data in each partition may in turn be divided into Buckets based on the value of a hash function of some column of the Table. **In hive a partition is a directory but a bucket is a file.**

*set hive.enforce.bucketing=true;*

Note that it is not necessary for tables to be partitioned or bucketed, but these abstractions allow thesystem to prune large quantities of data during query processing, resulting in faster query execution

### Data Types

Primitive Data Types

Integral Types: Integer type data can be specified using integral data types, INT. When the data range exceeds therange of INT, you need to use BIGINT and if the data range is smaller than the INT, you use SMALLINT, TINYINT is smaller than SMALLINT.

TINYINT 1 byte int, SMALLINT 2 byte int, INT 4 byte int, BIGINT 8 byte int

String Type:String type data types can be specified using single quotes (' ') or double quotes (" "). It contains twodata types: VARCHAR and CHAR. Hive follows C-types escape characters.The following table depicts various CHAR data types:VARCHAR 1 to 65355; CHAR 255

Boolean: TRUE/FALSE

Timestamp:It supports traditional UNIX timestamp with optional nanosecond precision. It supportsjava.sql.Timestamp format “YYYY-MM-DD HH:MM:SS.fffffffff” and format “yyyy-mm-ddhh:mm:ss.ffffffffff”.

Dates: DATE values are described in year/month/day format in the form {{YYYY-MM-DD}}.

Decimals: The DECIMAL type in Hive is as same as Big Decimal format of Java. It is used for representing immutable arbitrary precision. The syntax and example is as follows:

DECIMAL(precision, scale)

*decimal(10,0)*

Union Types: Union is a collection of heterogeneous data types. You can create an instance using create union. Thesyntax and example is as follows:

UNIONTYPE <int, double, array<string>, struct<a:int,b:string>>

{0:1}

{1:2.0}

{2:["three","four"]}

{3:{"a":5,"b":"five"}}

{2:["six","seven"]}

{3:{"a":8,"b":"eight"}}

{0:9}

{1:10.0}

Literals: The following literals are used in Hive:

1. Floating Point Types, are nothing but numbers with decimal points. Generally, this type of data iscomposed of DOUBLE data type.
2. Decimal Type, data is nothing but floating point value with higher range than DOUBLE data type. Therange of decimal type is approximately -10-308 to 10308.

Null Value: Missing values are represented by the special value NULL.

Complex Types

The Hive complex data types are as follows:

Arrays: Arrays are ordered sequence of similar data elements indexed using numeric values.

Syntax: *ARRAY (‘a’,’b’,’c’)*

the second value can be accessed using ARRAY[1]

Structs: Structs in Hive is similar to using complex data with comment.

Syntax: STRUCT <col\_name:data\_type , col\_name:data\_type>

example: STRUCT {a:int, b:char} which can be referred using ‘.’ Notation like column.a

Maps: Maps are collection of key value pairs.

Syntax: MAP<key, value>,

*map(‘first’,’a’,’second’,’b’,’last’,’c’),* here map[‘first’] can retrieve ‘a’

Delimiters:

1. New Line for the row -> \n
2. ASCII 0001 for field -> Ctrl-A ; Note Comma is not field delimiter
3. Tab character -> \t

File Types:

1. Text File
2. Sequence File; for compressing data.

Reaching the CLI: for the sub shell

1. hive> ! ls

/hive/warehouse

.

### hIVE oPERATORS

1. OVERWRITES, overwrites data if overwrite is not used then it will be append.
2. LOCAL, takes from the Local file system if Local is not used then by default takes from HDFS.
3. INSERT, To Load into another table.
4. Standard, Arithematic, Logical, equal operators
5. Standard functions like ,
   1. Concat(string a, string b)
   2. rand() // random number
   3. upper(string s) //returns string s in uppercase

## . Hive SQL

### Create

Note , best practice, Database name -> all Caps, Table Name -> lower Case, Column Name-> Title Case Variable Name -> Camel Case

// Create a customer table

*>CREATE TABLE customer(*

*CustID INT,*

*Company STRING,*

*ADDRESS STRING,*

*CITY STRING,*

*STATE STRING,*

*ZIP STRING,*

*INDUSTRY STRING,*

*REPID INT,*

*FIRSTYEAR INT)*

*ROW FORMAT DELIMITED*

*FIELDS TERMINATED BY ‘\t’ // Loading the file with tab delimiter*

*STORED AS TEXTFILE;*

// Create a Sales Rep table

*>CREATE TABLE Saelsforce(*

*REPID INT,*

*FirstName STRING,*

*SirName STRING,*

*Territory INT,*

*Commission STRING,*

*Salary INT,*

*Startdate STRING,*

*Empid INT)*

*ROW FORMAT DELIMITED*

*FIELDS TERMINATED BY ‘,’ // Loading the file with ‘,’ delimiter its .csv*

*STORED AS TEXTFILE;*

// Check schema of the customer table

*>DESCRIBE customer*

// Rename table Salesforce to Salesreps

*ALTER TABLE salesforce RENAME TO salesresps;*

// Rename column SirName to LastName

*ALTER TABLE salesreps CHANGE COLUMN SirName Lastname STRING AFTER FirstName;*

//Load the data into Customer table

*>LOAD DATA LOCAL INPATH ‘/home../customer.csv’*

*OVERWRITE INTO TABLE customers;*

//Select the customer.

>SELECT \* FROM customer LIMIT 10;

//Load the data into Salesrep table

*>LOAD DATA LOCAL INPATH ‘/home../x*

*OVERWRITE INTO TABLE salesrep;*

//different ways to use Load commands

// leave the OVERWRITE

*>LOAD DATA LOCAL INPATH ‘/home../salesforce.csv’*

*INTO TABLE salesrep;*

//Select the customer.

*>SELECT \* FROM salesrep LIMIT 10;*

// Truncate the table

*>TRUNCATE TABLE salesrep;*

//Select Statement

*>SELECT State,COUNT(DISTINCT Company) FROM customers WHERE State = “NY” or State= “TX”*

*GROUP BY State*

*ORDER BY State DESC;*

### Relationship between HIVe,HDFS and Metadata

In the hdfs prompt, relationship between HIVE and HDFS

*>hdfs dfs –ls /user/hive/warehouse*

// you will find directory for very table

*> hdfs dfs –ls /user/hive/warehouse/table\_name*

// .CSV file is loaded inside the directory

Lets look at the Metadata, login to the metastore

*>mysql –u hiveuser –p password*

*> SHOW tables*

// look for TBLS & COLUMNS\_V2

*>SELECT Owner, Tbl\_name FROM tbls;*

*> SELECT \* FROM columns\_v2;*

Lets login to the HIVE Shell

>*DROP TABLE salesreps;*

*>SHOW TABLES;*

// look for file in the warehouse directory

*>hdfs dfs –ls /user/hive/warehouse*

//salesrep folder has been dropped

//rebuilt the salesrep table

//Load the salesrep from HDFS

*>LOAD DATA INPATH*

*‘/user/hduser/crm/salesrep.csv’*

*INTO TABLE salesrep;*

Lets Export data from Hive to file in HDFS tmp Rows are directory. Note terminated with \n and fields are terminated with ”Ctrl A” or use contact or row format delimited

>*SELECT \* FROM Customers LIMIT 100;*

// Let us export

*>INSERT OVERWRITE DIRECTORY “customer\_ca”*

*SELECT concat(Company,”,”,City,”,”,) FROM customer WHERE state=”CA”;*

// look for file in the temp directory

*>hdfs dfs –ls /tmp*

//Customer\_ca folder has been Created

### Hive Script

//Lets create the .hql file (customer.hql)with the below select statement in HDFS

*>SELECT State,COUNT(DISTINCT Company) FROM customers WHERE State = “NY” or State= “TX”*

*GROUP BY State*

*ORDER BY State DESC*

//Lets Execute the script login to hdfs prompt

*>hive –f ./customer.hql*

## Partitions

### Static Partition

* Insert input data files individually into a partition table is Static Partition.
* You “statically” add a partition in table and move the file into the partition of the table.
* We can alter the partition in static partition.
* You should use 'where' clause to use 'limit' in static partition.

Usually when loading files (big files) into Hive tables static partitions are preferred. That saves your timein loading data compared to dynamic partition. You "statically" add a partition in table and move the fileinto the partition of the table. Since the files are big they are usually generated in HDFS. You can get thepartition column value form the filename, day of date etc without reading the whole big file.

If you want to use Static partition in hive you should set property set hive.mapred.mode = strict This property set by default in hive-site.xml

Static partitioning we need to specify the partition column value in each and every LOAD statement.

*hive>LOAD DATA INPATH '/hdfs path of the file' INTO TABLE t1 PARTITION(country="US")*

*hive>LOAD DATA INPATH '/hdfs path of the file' INTO TABLE t1 PARTITION(country="UK")*

### Dynamic Partition

* Single insert to partition table is known as dynamic partition.
* If you want to partition number of column but you don’t know how many columns then also dynamic partition is suitable.
* We can’t perform alter on Dynamic partition.
* You can perform dynamic partition on hive external table and managed table.
* If you want to use Dynamic partition in hive then mode is in nonstrict mode.
* No requirement to use 'where' clause to use 'limit' in static partition.

In case of dynamic partition whole big file i.e. every row of the data is read and data is partitionedthrough a MR job into the destination tables depending on certain field in file. So usually dynamicpartition are useful when you are doing sort of a ETL flow in your data pipeline. e.g. you load a huge filethrough a move command into a Table X. then you run a insert query into a Table Y and partition databased on field in table X say day , country. You may want to further run a ETL step to partition the datain country partition in Table Y into a Table Z where data is partitioned based on cities for a particularcountry only. etc.

To take an example where partitions are commonly used, imagine logfiles where each record includes atimestamp. If we partition by date, then records for the same date will be stored in the same partition.

The advantage to this scheme is that queries that are restricted to a particular date or set of dates canrun much more efficiently, because they only need to scan the files in the partitions that the querypertains to. Notice that partitioning doesn’t preclude more wide-ranging queries: it is still feasible toquery the entire dataset across many partitions.

A table may be partitioned in multiple dimensions. For example, in addition to partitioning logs by date,we might also subpartition each date partition by country to permit efficient queries by location.Partitions are defined at table creation time using the PARTITIONED BY clause, which takes a list ofcolumn definitions. For the hypothetical logfiles example, we might define a table with recordscomprising a timestamp and the log line itself:

*CREATE TABLE logs (ts BIGINT, line STRING)*

*PARTITIONED BY (dt STRING, country STRING);*

When we load data into a partitioned table, the partition values are specified explicitly:

*LOAD DATA LOCAL INPATH 'input/hive/partitions/file1'*

*INTO TABLE logs*

*PARTITION (dt='2001-01-01', country='GB');*

At the filesystem level, partitions are simply nested subdirectories of the table directory.After loading a few more files into the logs table, the directory structure might look likethis:

/user/hive/warehouse/logs

├── dt=2001-01-01/

│ ├── country=GB/

│ │ ├── file1

│ │ └── file2

│ └── country=US/

│ └── file3

└── dt=2001-01-02/

├── country=GB/

│ └── file4

└── country=US/

├── file5

└── file6

The logs table has two date partitions (2001-01-01 and 2001-01-02, corresponding to subdirectoriescalled dt=2001-01-01 and dt=2001-01-02); and two country subpartitions (GB and US, corresponding tonested subdirectories called country=GB and country=US).

The datafiles reside in the leaf directories. We can ask Hive for the partitions in a table usingSHOW PARTITIONS:

*hive> SHOW PARTITIONS logs;*

dt=2001-01-01/country=GB

dt=2001-01-01/country=US

dt=2001-01-02/country=GB

dt=2001-01-02/country=US

One thing to bear in mind is that the column definitions in the PARTITIONED BY clause are full-fledgedtable columns, called partition columns; however, the data files do not contain values for these columns,since they are derived from the directory names.

You can use partition columns in SELECT statements in the usual way. Hive performs input pruning toscan only the relevant partitions.For example:

SELECT ts, dt, line FROM logs WHERE country='GB';

Will only scan file1, file2, and file4. Notice, too, that the query returns the values of the dt partition

column, which Hive reads from the directory names since they are not in the datafiles.

Dynamic partition allow us not to specify partition column value each time. the approach we follows is as below:

1. Create a non-partitioned table t2 and insert data into it.
2. Now create a table t1 partitioned on intended column(say country).
3. Load data in t1 to t2 as below:

*hive> INSERT INTO TABLE t2 PARTITION(country) SELECT \* from T1;*

1. Make sure that partitioned column is always the last one in non-partitioned table(as we are having country column in t2)

## BUCKETS

There are two reasons why you might want to organize your tables (or partitions) into buckets. The firstis to enable more efficient queries. Bucketing imposes extra structure on the table, which Hive can takeadvantage of when performing certain queries. In particular, a join of two tables that are bucketed onthe same columns — which include the join columns — can be efficiently implemented as a map-sidejoin.

The second reason to bucket a table is to make sampling more efficient. When working with largedatasets, it is very convenient to try out queries on a fraction of your dataset while you are in theprocess of developing or refining them. We will see how to do efficient sampling at the end of thissection. First, let’s see how to tell Hive that a table should be bucketed.

We use the CLUSTERED BYclause to specify the columns to bucket on and the number of buckets:

*CREATE TABLE bucketed\_users (id INT, name STRING)*

*CLUSTERED BY (id) INTO 4 BUCKETS;*

Here we are using the user ID to determine the bucket (which Hive does by hashing the value andreducing modulo the number of buckets), so any particular bucket will effectively have a random set ofusers in it. In the map-side join case, where the two tables are bucketed in the same way, a mapperprocessing a bucket of the left table knows that the matching rows in the right table are in itscorresponding bucket, so it need only retrieve that bucket (which is a small fraction of all the data storedin the right table) to effect the join. This optimization also works when the number of buckets in the twotables are multiples of each other; they do not have to have exactly the same number of buckets.

The data within a bucket may additionally be sorted by one or more columns. This allows even moreefficient map-side joins, since the join of each bucket becomes an efficient merge sort. The syntax fordeclaring that a table has sorted buckets is:

*CREATE TABLE bucketed\_users (id INT, name STRING)*

*CLUSTERED BY (id) SORTED BY (id ASC) INTO 4 BUCKETS;*

How can we make sure the data in our table is bucketed? Although it’s possible to load data generatedoutside Hive into a bucketed table, it’s often easier to get Hive to do the bucketing, usually from anexisting table.

WARNING

Hive does not check that the buckets in the datafiles on disk are consistent with the buckets in the tabledefinition (either in number or on the basis of bucketing columns). If there is a mismatch, you may getan error or undefined behavior at query time. For this reason, it is advisable to get Hive to perform thebucketing.Take an unbucketed users table:

*hive> SELECT \* FROM users;*

0 Nat

2 Joe

3 Kay

4 Ann

To populate the bucketed table, we need to set the **hive.enforce.bucketing** property to true so that Hiveknows to create the number of buckets declared in the table definition. Then it is just a matter of usingthe INSERT command:

*INSERT OVERWRITE TABLE bucketed\_users*

*SELECT \* FROM users;*

Physically, each bucket is just a file in the table (or partition) directory. The filename is not important,but bucket n is the nth file when arranged in lexicographic order. In fact, buckets correspond toMapReduce output file partitions: a job will produce as many buckets (output files) as reduce tasks. Wecan see this by looking at the layout of the bucketed\_users table we just created. Running thiscommand:

*hive> dfs -ls /user/hive/warehouse/bucketed\_users;*

shows that four files were created, with the following names (the names are generated by Hive):

000000\_0

000001\_0

000002\_0

000003\_0

The first bucket contains the users with IDs 0 and 4, since for an INT the hash is the integer itself, andthe value is reduced modulo the number of buckets — four, in this case:

*hive> dfs -cat /user/hive/warehouse/bucketed\_users/000000\_0;*

0Nat

4Ann

We can see the same thing by sampling the table using the TABLESAMPLE clause, which restricts the

query to a fraction of the buckets in the table rather than the whole table:

*hive> SELECT \* FROM bucketed\_users TABLESAMPLE (BUCKET 1 OUT OF 4 ON id);*

4 Ann

0 Nat

Bucket numbering is 1-based, so this query retrieves all the users from the first of four buckets. For alarge, evenly distributed dataset, approximately one-quarter of the table’s rows would be returned. It’spossible to sample a number of buckets by specifying a different proportion (which need not be an exactmultiple of the number of buckets, as sampling is not intended to be a precise operation). For example,this query returns half of the buckets:

*hive> SELECT \* FROM bucketed\_users TABLESAMPLE(BUCKET 1 OUT OF 2 ON id);*

4 Ann

0 Nat

2 Joe

Sampling a bucketed table is very efficient because the query only has to read the buckets that matchthe TABLESAMPLE clause. Contrast this with sampling a nonbucketed table using the rand() function,where the whole input dataset is scanned, even if only a very small sample is needed:

*hive> SELECT \* FROM users TABLESAMPLE(BUCKET 1 OUT OF 4 ON rand());*

2 Joe

## Serialization and De Serialization(SERDE)

The SerDe interface is extremely powerful for dealing with data with a complex schema. By utilizing

SerDes, any dataset can be made queryable through Hive.

The SerDe interface allows you to instruct Hive as to how a record should be processed. A SerDe is a

combination of a Serializer and a Deserializer (hence, Ser-De). The Deserializer interface takes a string orbinary representation of a record, and translates it into a Java object that Hive can manipulate.

The Serializer, however, will take a Java object that Hive has been working with, and turn it intosomething that Hive can write to HDFS or another supported system. Commonly, Deserializers are usedat query time to execute SELECT statements, and Serializers are used when writing data, such as throughan INSERT-SELECT statement. Example of serde data for Hive are XML, JSON file etc.

Hive uses SerDe (and FileFormat) to read and write table rows.

* HDFS files --> InputFileFormat --><key, value> --> Deserializer --> Row object
* Row object --> Serializer --><key, value> --> OutputFileFormat --> HDFS files

Registration of Native SerDes

As of Hive 0.14 a registration mechanism has been introduced for native Hive SerDes. This allows

dynamic binding between a "STORED AS" keyword in place of a triplet of {SerDe, InputFormat, and

OutputFormat} specification, in CreateTable statements.

The following mappings have been added through this registration mechanism:Syntax Equivalent

*STORED AS AVRO /STORED AS AVROFILE*

ROW FORMAT SERDE

'org.apache.hadoop.hive.serde2.avro.AvroSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'

*STORED AS ORC /STORED AS ORCFILE*

ROW FORMAT SERDE

'org.apache.hadoop.hive.ql.io.orc.OrcSerde'

STORED AS INPUTFORMAT

'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'

*STORED AS PARQUET /STORED AS PARQUETFILE*

ROW FORMAT SERDE

'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'

*STORED AS TEXTFILE*

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'

## Hive vz Learning

### Join

* Hive must use equi-joins and mandate the use of “ON” clause to ensure an equi-join, it cannot support ?, <, >, or Like.
* Hive can support inner joins, left and right outer joins and left semi joins as long as they are build on equi-joins
* Hive can use function in join
  + LOWER to convert to lower case.
  + SUBSTR to extract a substring
  + LENGTH to determine the length of a string.

Eg:

SELECT a.col1,b.col1,c.col1 from a

join b on (a.col1=b.col1)

join c on (c.col1 = b.col1)

### PArtitions /Bucket

* There is no indexing in Hive.
* Instead partition and bucket
* This is optimized by query, purpose of the Partitions and Bucket is to reduce the lookup time by directly looking for the data in hdfs.
* Partitioning and bucketing should be decided ahead of data loading.
* Don’t confuse MapReduce with Hive Partitioning.

Physically Hive data files are stored in the /user/hive/warehouse on HDFS. We determine the partition schema prior to loading data. When we load the data into Hive we create partitioned directories for the data store. This way data is physically broken up into these partitioned directories. Most partitioned schema is one deep but can be more.

Example, you have millions of data for US sales and you have partitioned by State, you will have directories created for each state.

*/user/hive/warehouse/state=CA*

*/user/hive/warehouse/state=OR*

Based on the bucket size (say 4) on any of the column in the table, each partition will have same number of bucket (4) defined. Using hashing of the defined column data will be distributed.

// you need to set the below parameter in hive(by firing in prompt) to true, else it will produce wrong number of reducers.

*>set hive.enforce.bucketing = TRUE*

*CREATE TABLE buckets (*

*BuckID INT*

*Company STRING,*

*Address STRING,*

*City STRING,*

*ZIP STRING)*

*PARTITIONED BY (State STRING)*

*CLUSTERED BY (BuckID) INTO 4 BUCKETS*

*ROW FORMAT DELIMITED*

*FIELDS TERMINATED BY ‘,’*

*STORED AS TEXTFILE;*

# you can drop the partitions

*ALTER TABLE buckets DROP PARTITION (State=’WI’);*

*INSERT OVERWRITE TABLE T*

*PARTITION (ds='2010-03-03', hr)*

*SELECT key, value, /\*ds,\*/ hr FROM srcpart WHERE ds is not null and hr>10;*

## Performance Tunning

### Sort By

Sort By, Hive uses the columns in the Sort By to sort the rows before feeding the rows to a reducer.

Ordering: It orders data at each of ‘N’ reducers, but each reducers can have overlapping ranges of data.

Outcome: N or more sorted files with overlapping ranges.

### Order By

Order By, Guarantees total ordering of data, but for that it has to be passed on to a single reducer, which is normally unacceptable and therefore in strict mode, hive makes it compulsory to use LIMIT with ORDER BY so that reducer doesn’t get overburdened.

Ordering: Total Ordered data.

Outcome: Single Output

### Distribute By

Distribute By, Hive uses the columns in Distribute By to distribute the rows among reducers. All rows with same Distribute By column values will go to the same reducer. It ensures each of N reducers get non- Overlapping ranges of column, but doesn’t sort the output of each reducer. You end up with N or more unsorted files with non-Overlapping ranges.

### Cluster By

Cluster By, is a short for both Distribute By and Sort By. CLUSTER BY x ensures each of N reducers get non-overlapping ranges, then sorts by those ranges at the reducers.

Ordering: Global ordering between multiple reducers.

Outcome: N or more sorted files with non-overlapping ranges.

Note: Instead of specifying Cluster By, the user can specify Distribute By and Sort By, so the partition columns and sort columns can be different.

### Cluster By

### Using Explain

It helps to learn how HIVE translates queries into MapReduce Jobs. Output as abstract Syntax tree, which shows how HIVE parsed the query into bvtokens and literals. EXPLAIN EXTENDED, produces even more output.

Example:

EXPLAIN SELECT SUM(number) FROM onecol;

### Optimized Joins

Use the /\*Streamtable(table\_name)\*/ directive.

If all but one table is small enough, typically to fit in memory, then Hive can perform a map-side join, eliminating the need for reduce tasks and even some map tasks. Sometimes even large tables that do not fit in memory are good candidates because removing reduce phase outweighs the cost of bringing semi-large tables into each map tasks.

### Local Mode

Many jobs need the full scalability benefits of Hadoop to process large data sets, However, there are times when the input to Hive is very small. In cases, the overhead of launching tasks for queries consumes a significant percentage of the overall job execution time. In many of these cases, Hive can leverage the ligher weight of the local mode to perform all the tasks for the job on a single machine and sometimes in same process.

You can explicitly enable local mode temporarily,

* set mapred.job.tracker =local;
* select \* from people where firstname=bob;

### Parallel Execution

Hive Converts a query into one or more stages. Stages could be a MapReduce stage, sampling Stage, merge Stage, limit Stage, or other possible tasks Hive needs to do. Hive executes these stages one at a time. However, a particular job may consist of some stages that are not dependent on each other and could be executed in parallel. Setting *hive.exec.parallel* to *true* enables parallel execution.

* <property>
* <name>hive.exec.parallel</name>
* <value>true</name>
* </property>

### Strict Mode

Strict Mode is a setting in Hive that prevents users from issuing queries that could have unintended and undesirable effects. Setting *hive.mapred.mode* to *strict* enables parallel execution.

1. Queries on partitioned tables are not permitted unless they include a partition filter in the where clause,

# PIG

Pig provides an engine for executing data flows in parallel on Hadoop. It includes a high level data flow language, Pig Latin. Pig Latin includes operators for many of the traditional data operations (join, sort, filter, etc.), as well as the ability for users to develop their own functions for reading, processing, andwriting data.

* High level data flow language for exploring very large datasets.
* Provides an engine for executing data flows in parallel on Hadoop.
* Core functionality is an optimizer for parallelization
* Compiler that produces sequences of Map Reduce programs
* Structure is amenable to substantial parallelization
* Operates on files in HDFS
* Metadata not required, but used when available

Power of PIG:

* Loading Large volumes of Data
* Processing Multiple Data Sources, create complex data flow with custom code to data from different sources
* Analytic Insight: Able to perform random samplings from large data sets.

## PIG Concepts

Key Properties of Pig:

Ease of programming: Trivial to achieve parallel execution of simple and parallel data analysis tasks

Optimization: Allows the user to focus on semantics rather than efficiency

Extensibility: Users can create their own functions to do special-purpose processing

* Data flow Language (for high level data processing)
* Compiler, (pig Latin)
* Grunt, (shell)
* Case-Sensitive

Interacting With Pig:

### Advantages

* Makes writing hadoop jobs a lot simple - 5% of the code, 5% of time
* You don’t have to be a programmer to write Pig scripts
* Provides major functionality required for DW and Analytics- Load, Filter, Join, Group By, Order, Transform, UDFs, Store.
* User can write custom UDFs (User Defined Function)

### History

Pig started out as a research project in Yahoo Research, where Yahoo! scientists designed it andproduced an initial implementation. Hadoop “is too low-level and rigid, and leads to a great deal ofcustom user code that is hard to maintain and reuse.” At the same time they observed that many

MapReduce users were not comfortable with declarative languages such as SQL. Thus they set out toproduce “a new language called Pig Latin that we have designed to fit in a sweet spot between thedeclarative style of SQL, and the low-level, procedural style of MapReduce.” Yahoo! Hadoop usersstarted to adopt Pig. So, a team of development engineers was assembled to take the researchprototype and build it into a production-quality product. In year 2007, Pig was open sourced via theApache Incubator. The first Pig release came a year later 2008 and graduated from the Incubator andbecame a subproject of Apache Hadoop. Early in 2009 other companies started to use Pig for their dataprocessing. Amazon also added Pig as part of its Elastic MapReduce service. By the end of 2009 abouthalf of Hadoop jobs at Yahoo! were Pig jobs. In 2010, Pig adoption continued to grow, and Pig graduatedfrom a Hadoop subproject, becoming its own top-level Apache project.

### Pig Philosophy

Pig eats Anything:

Pig can operate on data whether it has metadata or not. It can operate on data that is relational, nested, orunstructured. And it can easily be extended to operate on data beyond files, including key/value stores, databases,etc.

Pigs live anywhere:

Pig is intended to be a language for parallel data processing. It is not tied to one particular parallel framework. Ithas been implemented first on Hadoop, but we do not intend that to be only on Hadoop.

Pigs are domestic animals:

Pig is designed to be easily controlled and modified by its users. Pig allows integration of user code whereverpossible, so it currently supports user defined field transformation functions, user defined aggregates, and userdefined conditionals. These functions can be written in Java or in scripting languages that can compile down toJava (e.g., Jython).

Pigs fly:

Pig processes data quickly. We want to consistently improve performance, and not implement features in waysthat weigh Pig down so it can’t fly.

### Application of PIG

Pig Latin use cases tend to fall into three separate categories: traditional extract transform load (ETL)data pipelines, research on raw data, and iterative processing.The largest use case is data pipelines. A common example is web companies bringing in logs from theirweb servers, cleansing the data, and pre computing common aggregates before loading it into their datawarehouse. In this case, the data is loaded onto the grid, and then Pig is used to clean out records frombots and records with corrupt data.

It is also used to join web event data against user databases so that user cookies can be connected withknown user information. Another example of data pipelines is using Pig offline to build behaviorprediction models. Pig is used to scan through all the user interactions with a website and split the usersinto various segments. Then, for each segment, a mathematical model is produced that predicts howmembers of that segment will respond to types of advertisements or news articles. In this way thewebsite can show ads that are more likely to get clicked on, or offer news stories that are more likely toengage users and keep them coming back to the site.

Traditionally, ad-hoc queries are done in languages such as SQL that make it easy to quickly form aquestion for the data to answer. However, for research on raw data, some users prefer Pig Latin.Because Pig can operate in situations where the schema is unknown, incomplete, or inconsistent, andbecause it can easily manage nested data, researchers who want to work on data before it has beencleaned and loaded into the warehouse often prefer Pig. Researchers who work with large data setsoften use scripting languages such as Perl or Python to do their processing. Users with thesebackgrounds often prefer the dataflow paradigm of Pig over the declarative query paradigm of SQL.

PIG Architecture

Interfaces

### Pig Setup

Pig as limited environmental setup,

1. “pig-env.sh”, setting java and Hadoop home in PIG

*export PIG\_HADOOP\_VERSION =20*

*export PIG\_CLASSPATH = $HADOOP\_HOME/conf*

1. “pig.properties”

*log4jconf = /etc/pig/conf/log4j.properties*

*pig.logfile = /var/log/pig/pig.log*

### Architecture

### Interface

* Pig Latin - Submit a script directly
* Grunt - Pig Shell
* Java Interface - Java Class similar to JDBC interface

### PIG Comes Commands

1. It has list of utility commands like
2. help
3. quit
4. kill <jobid>
5. set debug on|off
6. set job.name ‘jobname’
7. With you have access to all standard filesystem LINUX commands like cat, cd, copyFromLocal, copyTOLocal, cp, ls, mkdir, mv, pwd, rm, exec, run
8. Pig can handle large number of expressions, one of the strongest is use of regular expression

Pattern, “x matches regex”

### Execution Mode

Local Mode:

PIG engine runs in a single machine will all installation and config files are run using yourlocal host and file system. Pig can run in a non hadoop environment if runs using localmode. Is invoked by using the -x local flag

>pig -x local

MapReduce Mode:

MapReduce mode is the default mode, Need access to a Hadoop cluster and HDFSinstallation. Can also be invoked by using the -x MapReduce flag or just pig

>pig -x mapreduce

### Data Types

Field: It’s a data element, intersection between column and row or a cell of data. In Pig we deal wit the field.

Tuple, is analogous to the row of data. Map is chararray of data elements

Cross

Steam

Functional Operators:

AVG, CONCAT, COUNT, DIFF, MAX, MIN, SIZ, SUM, TOENIZE, ISEMPTY

Test for Null,

* X is null
* X is not null

### Pig and SQL Comparison

## PIG Workouts

### Pig Example

1. Login to the pig interactive grunt shell,

*>help*

#load the data from local data directory, pig works best when data is directly in hdfs

# create sales directory in hdfs

*>mkdir sales*

*>copyFromLocal /..some local path/sales2014.csv sales*

*>ls sales*

#load the file

*>sales = LOAD ‘sales/sales2014.csv’ using pigStorage(‘,’) as (custid:int, customer:chararray, state:chararray, zip:chararray, industry:chararray, repid:int, sales:int);*

*>DESCRIBE sales;*

*>salesLimit= LIMIT sales 10;*

#dump is an execution command

*>DUMP salesLimit*

*>DUMP sales*

#filter

*>saleOH = FILTER sales by state == ‘OH’;*

*>DUMP salesOH;*

#complex filter using “AND”

*>giftsalesOH = FILTER sales BY (state == ‘OH’ AND industry==’Gift)’;*

*>DUMP giftsalesOH*

*> giftsalesNW = FILTER sales BY (state == ‘ID’ OR state == ‘OR’ OR state == ‘WA’);*

*>DUMP giftsalesNW*

*>bigCustomers = FILTER sales BY (sales > 30000);*

*>DUMP bigCustomers;*

#foreach

# example for “foreach” , is equivalent to select.

*>customerSales = FOREACH sales GENERATE custid, customer, sales;*

*>DUMP customerSales;*

# apply athematic function to foreach

*>customerSales1 = FOREACH sales GENERATE custid, customer, sales/1000;*

*>DUMP customerSales1;*

# create new column names by using key word “as”

*>customerSales2 = FOREACH sales GENERATE custid, customer, sales/1000 as salesK;*

*>DUMP customerSales2;*

# Join

# load zipcode file from local File system

*>copyFromLocal /home/hduser/data/zip.csv sales*

*> zipCity = LOAD ‘sales/ zip.csv’ using pigStorage(‘,’) as (zipcode:chararray, city:chararray);*

*>cityJoin = JOIN sales BY zip, zipCity BY zipcode;*

#we have zipcode twice we need to use foreach to select only specific columns

*>cityTable = FOREACH cityJoin GENERATE custid,customer,city,state,zip;*

#GROUP

*>cityTableGroup = GROUP cityTable All;*

#We will have bag(called all) which as number of tuples



# say you can to find the record count of sales.

*>salesGroup = GROUP sales ALL;*

*>salesCount = FOREACH salesGroup GENERATE group, COUNT(sales);*

*>DUMP salesGROUP*

# o/p will be (all,3811), our bag all has 3811 tuples

#Now you want to group by State.

*>salesCount = FOREACH salesGroup GENERATE group, COUNT(sales);*

*>salesGroup = GROUP sales BY state;*

*>DUMP salesGROUP*

#Group command produces o/p multiple bags for each state

#COGROUP

#cogroup is combining of group with join

# we have already loaded sales and zipcity(has only two column)

*>listcities = GROUP sales BY zip, zipcity by zip*

# In above statement we are grouping sales by zip and joining by grouping zipcity by zip

#o/p will be bags that contain other bags

# for the join column zip 99359, we have one bag empty and other bag with (zipcode and city)



#example of cogroup, count number of customers by sales rep

*>salesReps = LOAD ‘sales/salesreps.csv’ USING pigStorage(‘,’) AS (repnum:int,firstname: chararray, lastname:chararray, region:chararray);*

*>listReps = GROUP sales BY repid, salesReps by repnum;*# cogroup

--DESCRIBE listReps

--DUMP listReps

*>repCount = FOREACH listReps GENERATE group, COUNT(sales.custid);*

#FLATTEN

#Flatten the COGROUP we want to remove the nesting.



*>salesList = FOREACH sales GENERATE customer,state,zip;*

*>listCities = Group salesList by zip, zipCity by zip;* #cogroup

--DESCRIBE *listCities*

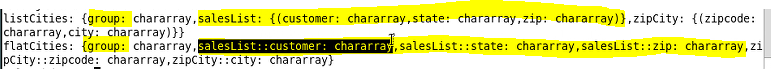
--DUMP *listCities*

#flatten the input file

*>flatCities = FOREACH listcities GENERATE group, FLATTEN(salesList),FLATTEN(zipCity)*

--DESCRIBE *flatCities*

--DUMP *flatCities*





# but this produces duplicates

*>dedupList= FOREACH flatCities GENERATE salesList::customer, zipCity::city, salesList::state, salesList::zip;*

*>deGRP = GROUP dedupList by $0;*

*>finalList = FOREACH deGrp {*

*top\_rec = LIMIT dedupList 1;*

*GENERATE FLATTEN(top\_rec);*

*};*

*>DUMP finalist*

*Note:Flattening doen’t not report out any record which contains empty bag from either data file, cogroup will report out a bag, if either data file has matching tuple*

#Store

#store is an execution command, loading of data back to a set location

*>STORE salesLimit INTO ‘sales/out/saleslimit’;*

*>ls sales/out/saleslimit*

*>cat sales/out/saleslimit/part-r-0000*

### Pig Scripts

* All the pig comments need to end with semicolon
* You normally declare a variable and act on it
* Comment /\*hsdfhg\*/ or --
* Variables can be identified by ‘$’ and can define the variable at the top or you can pass the value to the calling script using “-param”

### Pig Scripts 1

1. Create the weblogs.pig
2. Execute the script

weblogs.pig

--comments

/\* load the file in this script

\*/

*Log =LOAD ‘/data../some.txt’;*

Execute

*>pig scripts/weblogs.pig*

### Pig Scripts 2

myscript.pig

--Loading parameter

*Log =LOAD ‘$input’ AS (user, time, query);*

*Lmt = LIMIT log $size;*

Execute

*>pig –param input=excite-small.log –param size=4 scripts/myscript.pig*

### Pig Scripts 3

Script Parameter file

# parameter file

*input = excite-small.log*

*size = 4*

myscript.pig

--Loading parameter

*Log =LOAD ‘$input’ AS (user, time, query);*

*Lmt = LIMIT log $size;*

Command Line

*pig –param\_file myparams.txt scripts/myscript.pig*

### Pig Scripts 4

--Pig script file FirstScript.pig

*sales = LOAD ‘sales/sales2014.csv’ using pigStorage(‘,’) as (custid:int, customer:chararray, state:chararray, zip:chararray, industry:chararray, repid:int, sales:int);*

*DUMP sales*

*STORE sales INTO ‘sales/out/salesyr’;*

### Pig Scripts 5

--Pig script file sales.pig

*%declare limitSize 1500*

*%declare dollark 2*

*%declare inputFile sales/sales2014.csv*

*sales = LOAD ‘$inputFile’using pigStorage(‘,’) as (custid:int, customer:chararray, state:chararray, zip:chararray, industry:chararray, repid:int, sales:int);*

--DUMP sales

*salesLimit= LIMIT sales $limitSize;*

*customerSales = FOREACH sales GENERATE custid, customer, sales/1000 as salesK;*

*bigDollars = FILTER ciustomerSales BY salesK > $dollarK;*

*DUMP bigDollars;*

-- Store output

*STORE bigDollars INTO ‘sales/out/bigdollars’ using pigStorage(‘,’);*

### Pig Functions

SalesByState.pig

*sales = LOAD ‘sales/sales2014.csv’ using pigStorage(‘,’) as (custid:int, customer:chararray, state:chararray, zip:chararray, industry:chararray, repid:int, sales:int);*

--dump sales

--group by state and display

*stateGroup = Group sales BY state;*

--dump stateGroup

*stateSales = FOREACH stateGroup GENERATE group, SUM(sales.sales);*

--dump stateSales

*STORE stateSales INTO ‘sales/out/stateSales’ using pigStorage(‘,’);*

Pig UDF

* UDF are helper classed to assist data processing.
* Written in JAVA and packaged in jar files.(also be in javascript, groovy, guby and python)
* Must be register first
  + REGISTER recom/java/toupper-1.5.jar;
  + DEFIINE UPPER com.ata.pig.string.UPPER();
  + B = FOREACH a GENERATE UPPER($0);
* PiggyBank is a collection of common UDF’s

## PIG BOOk

1. Pig is a dataflow language, Each processing step results in a new data set or relation. A relation name is referred to as alias.
2. PigStorage and TextLoader are two built in pig load functions that operate on HDFS file, supports globs.
3. If you want to load your data from HBASE, you would use the loader for HBASE,
   1. *Divs = load ‘nyse\_dividends’ using HBaseStorage();*
4. If you don’t specify store function , PigStorage will be used . you can specify a different store fuction with a using clause,
   1. *Store processed into ‘processed’ using HBaseStorage();*
   2. *Store processed into ‘processed’ using PigStorage();*
5. Field references can be preceded by a $ (dollar Sign) and start from 0.
   1. Gain = foreach df generate $6 - $3; //subtract col6-col3
6. You can also refer to range of fields using .. (two periods). This is particularly useful when you have many fields and do not want to repeat.
   1. *baseDF = load ‘NYSE\_daily’ as ( exchange, symbol, date, open, high, low, close, volume, adj\_close);*
   2. *beginning = foreach baseDF generate ..open;* -- exchange, symbol, date, open
   3. *middle = foreach baseDF generate open..close;* --open, high, low, close, volume, adj\_close
   4. *end = reach baseDF generate volume..* --volume, adj\_close
7. Pig also provides a binary condition operator, often referred to as “bincond”. It begins with a Boolean test , followed by a ?.
   1. *2 == 2 ? 1 : 4* –returns 1
   2. *2 == 3 ? 1 : 4* –returns 4
   3. *null == 2 ? 1 : 4*– returns null
   4. *2 == 2 ? 1 : ‘fred’* – type error , both values must be of the same type.
8. To extract data from complex types, use the projection operators. For maps this is ‘#’ (the pound or hash), followed by the name of the Key as a string. Keep in mind that the value associated with a Key may be of any type. If you reference a key that doesn’t exist in the map, the result is a null;
   1. bball = load ‘baseball’ as (name:chararray , team:chararray, position:bag {t:(p:chararray)}, bat:map[]);
   2. avg = foreach bball generate bat#’batting\_average’;
9. Tuple projection is done with ‘.’, the dot operator. As with top-level records, the field can be referenced by name(if you have a schema for the tuple) or by position. Referencing a non-existent positional field in the tuple will return null. Referncing a field name that doesn’t exist in the tuple will produce an error,
   1. A = load ‘input’ as (t:tuple(x:int, y:int));
   2. B = foreach A generate t.x, t.$1;
10. Bag projection is not as straight forward as map and tuple projection. Bags don’t guarantee that their tuples are stored in any order, so allowing a projection of the tuple inside the bag would be meaningful. Instead, when you project field in a bag, you are creating a new bag with only those fields.
    1. A = load ‘input’ as (b:bag{t:(x:int, y:int)});
    2. B = foreach A generate b.x;

This will produce a new bag whose tuples have only the field X in them. You can project multiple fields in a bag by surrounding the fields with parentheses and separating them by commas:

1. A = load ‘input’ as (b:bag{t:(x:int, y:int)});
2. B = foreach A generate b.(x, y);

This seemingly pedantic distinction that b.x is a bag and not a scalar value has consequences. Consider the following Pig Latin, **which will not work:**

1. A = load ‘foo’ as (x:chararray, y:int, z:int);
2. B = group A by x;
3. C = foreach B generate SUM(A.x + A.z)

Its clear what that programmer is trying to do here. But because A.y and B.y are bags and the additional operator is not defined on bags , this will produce an error. The correct way to do this calculating in Pig Latin is:

1. A = load ‘foo’ as (x:chararray, y:int, z:int);
2. A1 = foreach A generate x, y+z as yz;
3. B = group A by x;
4. C = foreach B generate SUM(A1.yz);
5. User Defined Functions(UDFs) can be invoked in ‘FOREACH’. These are called evaluation functions, or eval funcs. Because they are part of a “foreach” statement, these UDF’s take one record at a time and produce one output. Keep in mind that either the input or the output can be bag, so this one record can contain a bag of records:
   1. –udf\_in \_foreach.pig
   2. Divs = load ‘NYSE\_DIVIDENDS’ as (exchange, symbol, date, dividends);
   3. Upped = foreach Divs generate UPPER(symbol) as symbol, dividends;
   4. Grpd = group Upped by symbol; --output a bag upped for each value of symbol
   5. –take a bag of integers, produce one result for each group
   6. Sums = foreach Grpd generate group, SUM(Upped.diviends);

In addition, eval funcs can take ‘\*’ as an argument, which passes the entire record to the function. They can also be invoked with no arguments at all.

1. Naming Fields in foreach:- The result of each ‘foreach’ statement is a new tuple, usually with different schema that the tuple that was in input to foreach.Pig can infer the data types of the fields in this schema from the foreach statement, but it cann’t always infer the names of those fields. For fields that are simple projections with no other operators applied. Pig keeps the same name as before
   1. Divs = load ‘NYSE\_DIVIDENDS’ as (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
   2. Sym = foreach Divs generate symbol;
   3. describe sym;

Sym:{symbol:chararray}

Once any expression beyond simple projection is applied, pig doesn’t assign a name to the field. If you do not explicitly assign a name, the field will be nameless and will be addressable only via a positional parameter, for example ,$0. You can assign a name with the ‘as’ clause:

1. Divs = load ‘NYSE\_DIVIDENDS’ as (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
2. In\_cents = foreach Divs generate dividends \* 100.0 as dividend, dividends\*100.0;
3. describe in\_cents;

in\_cents: {dividend:double, double}

Note: Foreach ‘as’ is attached to each expression. , reason for this will become clear when we discuss flatten.

1. Filter: Pig will short circuit Boolean operations when possible. If the first(left) predicate of an ‘and’ and ‘or’, and you can reverse the outcome of any predicate by using the Boolean not operator. For Boolean operatos, nulls follow the SQL trinary logic. Thus x==null results in a value of null, not true (even when x is null also) or False. Use the “is null” operator whenever value is ‘null’ or ‘not null’
   1. –filter \_matches.pig
   2. Divs = load ‘NYSE\_DIVIDENDS’ as (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
   3. StartWithcm = filter Divs by symbol matches ‘CM.\*’;
   4. –filter\_not\_matches.pig
   5. Divs = load ‘NYSE\_DIVIDENDS’ as (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
   6. notStartWithcm = filter Divs by not symbol matches ‘CM.\*’;
2. Group:- Group statement collects together records with the same key. It is first operator we have looked at the shares its syntax with SQL, but it is important to understand that the grouping operator in Pig Latin is fundamentally different that the one in SQL. In Pig there is no direct connection between group and aggregate functions. Instead group does exactly what it says; collects all records with the same value for the provided key together into a bag. You can pass this to an aggregate function if you want .
   1. –count.pig
   2. Daily = load ‘NYSE\_daily’ as (exchange, stock);
   3. Grpd = group Daily by stock;
   4. Cnt = foreach Grpd generate group, COUNT(daily);

Above example groups records by the key ‘Stock’ and then counts them. Its just legitimate to group them and store them for processing at a later time.

1. –group.pig
2. Daily = load ‘NYSE\_daily’ as (exchange, stock);
3. Grpd = group Daily by stock;
4. Store Grpd into ‘by\_group’;

Record coming out of the ‘group by ‘ statement have two fields, the key and the bag of collected records. Key field is named ‘group’. The bag is named for the alias that was grouped, so in the previous examples it will be named “Daily” and have the same schema as the relation ‘Daily’. If the relation Daily has no schema , the bag Daily will have no Schema. For each record in the group, the entire record (including the key) is in the bag. Changing the last line of the previous script from “store Grpd …” to describe grpd; will produce:

Grpd:{group: bytearray, Daily:{exchange:bytearray, stock:bytearray}}

You can also group on multiple Keys, but the keys must be surrounded by parentheses. The resulting records still have two fields. In this case, the ‘group’ field is a tuple with a field for each key:

--twokey.pig

daily = load ‘NYSE\_daily’ as (exchange, stock, date, dividends);

grpd = group daily by (exchange, stock);

avg = foreach grpd generate group, AVG(daily.dividends);

describe grpd;

grpd:{group: (exchange:bytearray,stock:bytearray),daily:{exchange:bytearray,stock:bytearray,date:bytearray,dividends:bytearray}}

You can also use ‘all’ to group together all of the records in your pipeline,

--countall.pig

daily = load ‘NYSE\_daily’ as (exchange, stock);

grpd = group daily all;

cnt = foreach grpd generate COUNT(daily);

The record coming out of “group all” has the chararray literal “all” as a key. Usually this doesn’t matter because you will pass the bag directly to an aggregate function such as COUNT.But if you plan to store the record or use it for another purpose, you might want to project out the artificial key first.

“group” is the first operator we have looked at that usually will force a reduce phase. Grouping means collecting all records where the key has the same value. If the pipeline is in a map phase, thi will force it to pass through map, shuffle and reduce phases.Pig uses Combiner to remove some Skewness of data. Unfortunately not all calculations can be done using the combiner.

Finally, group handles nulls in the same way that SQL handles them: by collecting all records with a null key into the same group. Note that this is in direct contradiction to the way expressions handle nulls and to the way join handles nulls.

Distinct:

The “distinct” statement is very simple, it removes duplicate records. It works only on entire records not on individual fields.

uniq = distinct df

Limit: sometimes you wanted to see only a limited number of results. Limit allows you do this. Limit causes an additional reduce phase , since it needs to collect the records together to count how many it is returning. It does optimize this phase by limiting the output of each map and then applying the limit again in the reducer. In case where “limit” is combined with “order” the two are done together on the map and reduce. That is, on the map side, the records are sorted by MapReduce and the limit applied in the combiner. They are sorted again by MapReduce as part of the shuffle, and pig applies the limit again in the reducer.

first10 = limit df 10;

#Register UDF

>REGISTER /home/hduser/pig/lib/pig-udf-0.5.jar;

#Define

>DEFINE IntCheck com.ata.pig.IntCheck();

# SQOOP

Apache Sqoop(TM) is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases and vice versa. This processing can be done with MapReduce programs or other higher-level tools such as Hive. (It’s even possible to use Sqoop to move data from a database into HBase.) When the final results of an analytic pipeline are available, Sqoop can export these results back to the data store for consumption by other clients.

Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases. Sqoop imports data from external structured datastores into HDFS or related systems like Hive and HBase.

Sqoop can also be used to export data from Hadoop and export it to external structured datastores such as relational databases and enterprise data warehouses.

Sqoop works with relational databases such as: Teradata, Netezza, Oracle, MySQL, Postgres, and HSQLDB.

SQOOP is an abbreviation for SQL on Hadoop.

### Pig Google Cloud Workouts

*sudo su –hdfs*

*hadoop fs –mkdir /user/hdfs*

/\* move the file from local file system to hdfs

*>hadoop fs –put testing.txt /user/hdfs*

/\* also move the file from Ambari UI

**Setup retail\_db Database**

* Connect to putty with root/hadoop
* from the github download retail\_db.sql
  + move the file to /root/data/retail\_db.sql
* Connect to mysql

*mysql –u root*

* show databases;

*>create database retail\_db;*

*>create user retail\_dba identified by ‘hadoop’*

*>grant all on retail\_db.\* to retail\_dba;*

*>flush privilages;*

*>show databases;*

*>mysql –u retail\_dba –p*

*>use retail\_db*

*>source /root/data/retail\_db.sql*

*>show tables;*

*Create the new user space “root” under /user in Hadoop FS*

*>sudo –u hdfs Hadoop fs –mkdir /user/root*

/\* change the owner of root to root from hdfs

*>sudo –u hdfs Hadoop fs -chown root:hdfs /user/root*

/\*Move the file\*/

*>hadoop fs –copyFromLocal /root/test /user/root*

*>hadoop fs –cat /user/root/test*

Site.xml

*>/etc/Hadoop/conf*

*>view core-site.xml*

*Has the default path of the “hadoop fs”*

*>hadoop fs –ls hdfs://sandox.hortonworks.com:8020/user/root/test*

*Or*

*>hadoop fs –ls /user/root/test*

Task 2 Create new directory

/\* create multiple level directory

/\* create dir1,dir2,dir3

hadoop fs –mkdir /user/root /dir1/dir2/dir3 /\* throws error

hadoop fs –mkdir –p /user/root/dir1/dir2/dir3

hadoop fs –mkdir –ls –R /user/root

Sqoop:

Code base can be downloaded from github,

Code/Hadoop/edw/hdp/sqoop/sqoop\_demo.txt

Note: use

1. –connect “jdbc:mysql://sanbox.hortonworks.com:3306/retail\_db”
2. Hive warehouse location,
   1. –warehouse-dir=/apps/hive/warehouse/retail\_stage.db
3. For mysql to work with sqoop, you need to have the mysql JDBC connector in sqoop as well.

/usr/share/java/my-connector-java\*.jar

>find / -name “mysql\*.jar”

/\* it should list sqoop here.

Setup:

> pwd

// /home/hdplearning1984/sample\_data\_kum/IZ DataSet/pigdata

hadoop fs -copyFromLocal testdata.txt /user/hdplearning1984

hadoop fs -copyFromLocal tuple.txt /user/hdplearning1984

hadoop fs -copyFromLocal bag.txt /user/hdplearning1984

hadoop fs -copyFromLocal coursedetails.txt /user/hdplearning1984

hadoop fs -copyFromLocal custs /user/hdplearning1984

hadoop fs -copyFromLocal txns /user/hdplearning1984

hadoop fs -copyFromLocal weblogs\_parse.txt /user/hdplearning1984

shadoop fs -ls /user/hdplearning1984

hadoop fs -mkdir /user/hdplearning1984/inputdataset

hadoop fs -mkdir /user/hdplearning1984/outputdataset

hadoop fs -copyFromLocal ~/pig/touch.txt /user/hdplearning1984

hadoop fs -cat /user/hdplearning1984/inputdataset/touch.txt

raw = LOAD 'touch.txt' USING PigStore(':');

raw = LOAD '/user/hdplearning1984/inputdataset/touch.txt' USING PigStore(':');

hadoop fs -copyFromLocal /etc/passwd /user/hdplearning1984

hadoop fs -cat /user/hdplearning1984/inputdataset/passwd

B = load 'passwd' using PigStorage(':');

DUMP B;

STORE B INTO '/user/hdplearning1984/outputdataset/pig1';

hadoop fs -ls /user/hdplearning1984/outputdataset/pig1

hadoop fs -cat /user/hdplearning1984/outputdataset/pig1/part-m-00000

create table txnrecords(userid STRING,Permission STRING,some\_number INT,rest STRING) row

format delimited

fields terminated by ':'

lines terminated by '\n'

stored as textfile;

LOAD DATA LOCAL INPATH '/etc/passwd' INTO TABLE txnrecords;

describe txnrecords;

describe formatted txnrecords;

Warehouse folder:

/apps/hive/warehouse

/apps/hive/warehouse/retail.db

hadoop fs -ls /user/hive

sqoop export

--connect 'jdbc:mysql://hdcentos/CRMDB'

--username uname --password pword

--table salesforce

--export-dir salesteam

--input-field-terminated-by ‘\t’

Hive;

create table tab\_kuma (

name string,

address string,

empid int)

row format delimited

fields terminated by '\t'

stored as textfile;

create table tab\_kumar2 (

name string,

address string,

empid int

)

row format delimited

field terminated by','

stored as textfile;

alter table tab\_kumar2 rename table\_kumar2;

alter table tab\_kumar2 change coulmn name full\_name string before address;

load data local inpath 'file path' overwrite into table table\_kumar2;

load data local inpath 'file path' into table table\_kumar2;

# export the data into tmp directory

inster overwrite directory "file\_name1"

select concat(col1,","col2,",",) from customer where col3="something";

hadoop fs -ls /tmp

# Lets execuet the script

hive -f ./scriptname.hql

# Partitioned table

create table logs(ts BIGINT, line STRING)

partitioned by (dt STRING, country STRING);

# Load data into partitioned data

load data local inpath 'path'

into table logs

partition (df STRING, country STRING)

Pig:

sales = LOAD 'sales/sales2014.csv' using PigStorage(',') as (custid:INT, ...);

DESCRIBE sales;

salesLimit = LIMIT sales 10;

DUMP sales

DUMP salesLimit

#filter

saleOH = FILTER sales by state='OH';

giftsalesNW = FILTER sales BY (state=='ID' or state=='OR' or state == 'WA');

bigCustomer = FILTER sales BY (sales>30000);

#Select

subset1 = FOREACH sales GENERATE custid,customer,city,sales/1000 as salesK;

#Join

cityJoin = Join sales BY zip, zipcity BY zipcode;

#GROUP

salesGRoup = Group sales ALL;

salesCount = FOREACH salesGroup GENERATE group, COUNT(sales);

salesGRoup = Group sales BY state;

## Sqoop Concepts

Relation between Hadoop and RDBMS:

* Hadoop can be an archive server for RDBM’s
* Hadoop can be a preprocessing step, cleaning and normalizing data for the RDBMS
* Hadoop can be data creation tool to populate data for RDBMs from various sources.

### Tidbits

Direct Mode:- means different things for different databases.  
For MySQL or PostgreSQL it relates to bulk loader/unloader utilities (i.e. completetely bypassing JDBC); while for Oracle it relates to "direct path INSERT" i.e. with JDBC but in a non-transactional mode (so you'd better use a temp table, or you might end up with duplicates in a PK and a corrupt table).

|  |  |
| --- | --- |
| **Argument** | **Description** |
| --append | Append data to an existing dataset in HDFS |
| --as-avrodatafile | Imports data to Avro Data Files |
| --as-sequencefile | Imports data to SequenceFiles |
| --as-textfile | Imports data as plain text (default) |
| --boundary-query <statement> | Boundary query to use for creating splits |
| --columns <col,col,col…> | Columns to import from table |
| --direct | Use direct import fast path |
| --direct-split-size <n> | Split the input stream every *n* bytes when importing in direct mode |
| --inline-lob-limit <n> | Set the maximum size for an inline LOB |
| -m,--num-mappers <n> | Use *n* map tasks to import in parallel |
| -e,--query <statement> | Import the results of *statement*. |
| --split-by <column-name> | Column of the table used to split work units |
| --table <table-name> | Table to read |
| --target-dir <dir> | HDFS destination dir |
| --warehouse-dir <dir> | HDFS parent for table destination |
| --where <where clause> | WHERE clause to use during import |
| -z,--compress | Enable compression |
| --compression-codec <c> | Use Hadoop codec (default gzip) |
| --null-string <null-string> | The string to be written for a null value for string columns |
| --null-non-string <null-string> | The string to be written for a null value for non-string columns |

|  |  |
| --- | --- |
| **Argument** | **Description** |
| --connect <jdbc-uri> | Specify JDBC connect string |
| --connection-manager <class-name> | Specify connection manager class to use |
| --driver <class-name> | Manually specify JDBC driver class to use |
| --hadoop-home <dir> | Override $HADOOP\_HOME |
| --help | Print usage instructions |
| -P | Read password from console |
| --password <password> | Set authentication password |
| --username <username> | Set authentication username |
| --verbose | Print more information while working |
| --connection-param-file <filename> | Optional properties file that provides connection parameters |

**$CONDITIONS** : is used by Sqoop process, it will replace with a unique condition expression internally to get the data-set. If you run a parallel import, the map tasks will execute your query with different values substituted in for $CONDITIONS. e.g., one mapper may execute "select bla from foo WHERE (id >=0 AND id < 10000)", and the next mapper may execute "select bla from foo WHERE (id >= 10000 AND id < 20000)" and so on.

**--boundary-query :** By default sqoop will use query select min(), max() from to find out boundaries for creating splits. In some cases this query is not the most optimal so you can specify any arbitrary query returning two numeric columns using --boundary-query argument.

Reason to use : If --split-by is not giving you the optimal performance you can use this to improve the performance further.

Example 1:

Sqoop allows you to import data in parallel and --split-by and --boundary-query allow you more control. If you're just importing a table then it'll use the PRIMARY KEY however if you're doing a more advanced query, you'll need to specify the column to do the parallel split.

sqoop import \

--connect 'jdbc:mysql://.../...' \

--direct \

--username uname --password pword \

--hive-import \

--hive-table query\_import \

--boundary-query 'SELECT 0, MAX(id) FROM a' \

--query 'SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND $CONDITIONS'\

--num-mappers 3

--split-by a.id \

--target-dir /data/import \

--verbose

### Advantages

As more organizations deploy Hadoop to analyse vast streams of information, they may find they need to transfer large amount of data between Hadoop and their existing databases, data warehouses and other data sources.

Loading bulk data into Hadoop from production systems or accessing it from map-reduce applications running on a large cluster is a challenging task since transferring data using scripts is an inefficient and time-consuming task

* Sqoop is basically an ETL Tool used to copy data between HDFS and SQL databases
* Import SQL data to HDFS for archival or analysis
* Export HDFS to SQL ( e.g : summarized data used in a DW fact table )

Sqoop Import

The import tool imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in text files or as binary data in Avro and Sequence files.

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. Those are read and parsed into a set of records and delimited with user-specified delimiter.

### Features of Sqoop

Designed to efficiently transfer bulk data between Apache Hadoop and structured datastores such as relational databases, Apache Sqoop:

* Allows data imports from external datastores and enterprise data warehouses into Hadoop
* Parallelizes data transfer for fast performance and optimal system utilization
* Copies data quickly from external systems to Hadoop
* Makes data analysis more efficient, Mitigates excessive loads to external systems.
* Sqoop was designed to connect to MapReduce and launch MapReduce jobs.
* Can control which columns and/or rows are imported, also support incremental imports.
* It can interact with Hadoop, HBase, Hive, ZooKeper. Environmental file of the Sqoop needs to be configured (sqoop-env.sh).

### Sqoop Properties

* Supports only mappers.
* Default 4 mappers.
* We can define number of mappers with – splitby columns or m 1 or if we have primary key in the table.
* Does not support composite key columns.
* DatadrivenDBInputformat is the input format.
* Imports and Consistency Performs only committed read. (READ COMMITED).
* Incremental Imports can be done.
* Direct-Mode Imports supported by databases such as MySQL, PostgreSQL, Oracle, and Netezza.
* Imports Large Objects as lobfile.
* Export works with combinedfileinputformat.
* Performs merge operation at export.

1. Sqoop is a single client progam.
2. Sqoop creates on to many MapReduce jobs to perform tasks.
3. Sqoop doesn’t have any server processes, it is only a CLI.
4. Sqoop is efficient and effective at determining the number of mappers required to import the data.

Note: it doesn’t have any long running daemons, SQOOP is coming and it is another architectural change. It will have a daemon and will run as a service.

Example,

1. If you were to import 10 million records Sqoop will create 20 mappers.
2. Each Mapper will process about 500000 records.
3. Create 20 SQL statements for each mapper and all will connect to the RDBMS.
4. Each mapper is using the primary key column to pull the data.

Number of mappers is configurable when writing the Sqoop statement, by default its 4 mappers.

## Sqoop Import Architecture

### Sqoop Import – Vz Learn

*sqoop import –connect jdbc:mysql://10.0.0.100/CRMDB –username mysqluser –password Mayyam%123 –table salesforce*

* By default, Sqoop places the output files into the home directory of the user running the command.
* By default, Sqoop writes to a comma delimited text file, binary data can be imported using Avro data files(or sequence file).
* By default, Sqoop uses four mappers.

Steps involved in Sqoop

1. Uses the JDBC connector to query the table.
2. Retrieves columns and data types.
3. Map sql data types and java data types.
4. Conducts column splitting.(by algorithm)
5. Generate the code for import.
6. Launches the determined number of mappers.
7. Each mapper attaches to the RDBMS and extract data rows.
8. Imports into HDFS.

Parallel Access to the Database (even small cluster can initiate multiple mapper and choke DB)

1. To Scale requires sophisticated management of the RDBMS.
2. Requires a complex portioning of the tables into segments and assigning each mapper to its own segment.
3. Large segment will result in long run times, this could result in relaunch of mappers by the application master.

Sqoop is dependent on the primary key’s and will throw error if not found. Works will with the numeric column as primary key. Work around for no primary key (also used when primary key will not produce even splits), set the number of mappers to 1, this allow data to be transferred in sequence and with no parallelization.

Sqoop is written in Java. Java provides an API called Java Database Connectivity, or JDBC, that allows applications to access data stored in an RDBMS as well as to inspect the nature of this data. Most database vendors provide a JDBC driver that implements the JDBC API and contains the necessary code to connect to their database servers.

Before the import can start, Sqoop uses JDBC to examine the table it is to import. It retrieves a list of all the columns and their SQL data types. These SQL types (VARCHAR, INTEGER, etc.) can then be mapped to Java data types (String, Integer, etc.), which will hold the field values in MapReduce applications. Sqoop’s code generator will use this information to create a table-specific class to hold a record extracted from the table.

### readFIELDS() & write() Methods

More critical to the import system’s operation, though, are the serialization methods that form the DBWritable interface, which allow the Widget class to interact with JDBC:

public void readFields(ResultSet \_\_dbResults) throws SQLException;

public void write(PreparedStatement \_\_dbStmt) throws SQLException;

JDBC’s ResultSet interface provides a cursor that retrieves records from a query; the readFields() method here will populate the fields of the Widget object with the columns from one row of the ResultSet’s data.

write() method shown here allows Sqoop to insert new Widget rows into a table, a process called exporting. Exports are discussed in Performing an Export.

### Splitting Column

The MapReduce job launched by Sqoop uses an InputFormat that can read sections of a table from a database via JDBC. The DataDrivenDBInputFormat provided with Hadoop partitions a query’s results over several map tasks. Reading a table is typically done with a simple query such as:

*SELECT col1,col2,col3,... FROM tableName*

But often, better import performance can be gained by dividing this query across multiple nodes. This is done using a splitting column. Using metadata about the table, Sqoop will guess a good column to use for splitting the table (typically the primary key for the table, if one exists). The minimum and maximum values for the primary key column are retrieved, and then these are used in conjunction with a target number of tasks to determine the queries that each map task should issue.

For example, suppose the widgets table had 100,000 entries, with the id column containing values 0 through 99,999. When importing this table, Sqoop would determine that id is the primary key column for the table. When starting the MapReduce job, the DataDrivenDBInputFormat used to perform the import would issue a statement such as SELECT MIN(id), MAX(id) FROM widgets. These values would then be used to interpolate over the entire range of data. Assuming we specified that five map tasks should run in parallel (with -m 5), this would result in each map task executing queries such as

*SELECT id, widget\_name, ... FROM widgets WHERE id >= 0 AND id < 20000,*

*SELECT id, widget\_name, ... FROM widgets WHERE id >= 20000 AND id < 40000,*

and so on.

The choice of splitting column is essential to parallelizing work efficiently. If the id column were not uniformly distributed (perhaps there are no widgets with IDs between 50,000 and 75,000), then some map tasks might have little or no work to perform, whereas others would have a great deal. Users can specify a particular splitting column when running an import job (via the --split-by argument), to tune the job to the data’s actual distribution. If an import job is run as a single (sequential) task with -m 1, this split process is not performed.

After generating the deserialization code and configuring the InputFormat, Sqoop sends the job to the MapReduce cluster. Map tasks execute the queries and deserialize rows from the ResultSet into instances of the generated class, which are either stored directly in SequenceFiles or transformed into delimited text before being written to HDFS.

### Text and Binary File Formats

Sqoop is capable of importing into a few different file formats. Text files (the default) offer a human-readable representation of data, platform independence, and the simplest structure. However, they cannot hold binary fields (such as database columns of type VARBINARY), and distinguishing between null values and String-based fields containing the value "null" can be problematic (although using the --null-string import option allows you to control the representation of null values).

To handle these conditions, Sqoop also supports SequenceFiles, Avro datafiles, and

Parquet files.

These binary formats provide the most precise representation possible of the imported data. They also allow data to be compressed while retaining MapReduce’ s ability to process different sections of the same file in parallel. However, current versions of Sqoop cannot load Avro datafiles or SequenceFiles into Hive (although you can load Avro into Hive manually, and Parquet can be loaded directly into Hive by Sqoop). Another disadvantage of SequenceFiles is that they are Java specific, whereas Avro and Parquet files can be processed by a wide range of languages.

### Controlling the Import

Sqoop does not need to import an entire table at a time. For example, a subset of the table’s columns can be specified for import. Users can also specify a WHERE clause to include in queries via the --where argument, which bounds the rows of the table to import.

For example, if widgets 0 through 99,999 were imported last month, but this month our vendor catalog included 1,000 new types of widget, an import could be configured with the clause WHERE id >= 100000; this will start an import job to retrieve all the new rows added to the source database since the previous import run. User-supplied WHERE clauses are applied before task splitting is performed, and are pushed down into the queries executed by each task.

For more control — to perform column transformations, for example — users can specify

a --query argument.

### Imports and Consistency

When importing data to HDFS, it is important that you ensure access to a consistent snapshot of the source data. (Map tasks reading from a database in parallel are running in separate processes. Thus, they cannot share a single database transaction.) The best way to do this is to ensure that any processes that update existing rows of a table are disabled during the import.

### Incremental Imports

It’s common to run imports on a periodic basis so that the data in HDFS is kept synchronized with the data stored in the database. To do this, there needs to be some way of identifying the new data. Sqoop will import rows that have a column value (for the column specified with --check-column) that is greater than some specified value (set via --last-value).

The value specified as --last-value can be a row ID that is strictly increasing, such as an AUTO\_INCREMENT primary key in MySQL. This is suitable for the case where new rows are added to the database table, but existing rows are not updated. This mode is called append mode, and is activated via --incremental append. Another option is time-based incremental imports (specified by --incremental lastmodified), which is appropriate when existing rows may be updated, and there is a column (the check column) that records the last modified time of the update.

At the end of an incremental import, Sqoop will print out the value to be specified as -- last-value on the next import. This is useful when running incremental imports manually, but for running periodic imports it is better to use Sqoop’s saved job facility, which automatically stores the last value and uses it on the next job run. Type sqoop job --help for usage instructions for saved jobs.

### Direct Mode Import

Sqoop’s architecture allows it to choose from multiple available strategies for performing an import. Most databases will use the DataDrivenDBInputFormat-based approach described earlier. Some databases, however, offer specific tools designed to extract data quickly. For example, MySQL’s mysqldump application can read from a table with greater throughput than a JDBC channel. The use of these external tools is referred to as direct mode in Sqoop’s documentation.

Direct mode must be specifically enabled by the user (via the --direct argument), as it is not as general purpose as the JDBC approach. (For example, MySQL’s direct mode cannot handle large objects, such as CLOB or BLOB columns, and that’s why Sqoop needs to use a JDBC-specific API to load these columns into HDFS.)

For databases that provide such tools, Sqoop can use these to great effect. A direct-mode import from MySQL is usually much more efficient (in terms of map tasks and time required) than a comparable JDBC-based import. Sqoop will still launch multiple map tasks in parallel. These tasks will then spawn instances of the mysqldump program and read its output. Sqoop can also perform direct-mode imports from PostgreSQL, Oracle, and Netezza. Even when direct mode is used to access the contents of a database, the metadata is still queried through JDBC.

## Sqoop Export Architecture

### Export VZ learn

sqoop export

--connect 'jdbc:mysql://hdcentos/CRMDB'

--username uname --password pword

--table salesforce

--export-dir salesteam

--input-field-terminated-by ‘\t’

Sqoop will not create the table and table’s column should line up to the data element. It will only append the record and not override

Exporting data with Reducers:

* Mapping Key Values, reducers receives partitioned data, when setting up the job the developer needs to assign the right key values to each reducer during partitioning.
* Load Management, the same issue of database management applies.
* Health Check, these are long running jobs and they need to be health checked to prevent launching multiply retry reducers.

Writing SQL export files from reducers:

* Use the reducers to write out SQL import files.
* Use a MySQL bulk load tool or even LOAD DATA statements to export the data into MySQL.

Note: Sqoop import has much information about the data it imports, where as sqoop export doesn’t have much understanding of the data types, except for the file location and delimiter used.

### Import/Export into HBase VZ learn

* HBase column families prevent a full table import from the RDBMS into HBase.
* Create the HBase table and column families. Prior to the import.
* Execute a Sqoop import for each data element.
* This means potentially hundreds of Sqoop import statements must identify the HBase column and row and this can be done by using Linux shell script.
* This will override the data while import as we specify the rowid

Critical issue here is Matching the RDBMS RowID and column to the HBase RowID and Column Family.

* RDBMS (import on RDBMS)

--columns “rowid, column-name”

* HBASE

--column-family ColumnFamily

--hbase-row-key ColumnFamily

sqoop import

--connect 'jdbc:mysql://hdcentos/CRMDB'

--username uname --password pword

--table salesforce

--columns “rowid, column-name”

--hbase-table salesreps

--column-family Fname

--hbase-row-key RepID

--num-mappers 1

### Sqoop CLI error Log-- VZ learn

Sqoop doesn’t produce the error log, but we can configure to produce the error log in CLI

### Export Architecture

The Sqoop performs exports is very similar in nature to how Sqoop performs imports Before performing the export, Sqoop picks a strategy based on the database connect string. For most systems, Sqoop uses JDBC. Sqoop then generates a Java class based on the target table definition. This generated class has the ability to parse records from text files and insert values of the appropriate types into a table (in addition to the ability to read the columns from a ResultSet). A MapReduce job is then launched that

reads the source datafiles from HDFS, parses the records using the generated class, and executes the chosen export strategy. The JDBC-based export strategy builds up batch INSERT statements that will each add multiple records to the target table. Inserting many records per statement performs much

better than executing many single-row INSERT statements on most database systems. Separate threads are used to read from HDFS and communicate with the database, to ensure that I/O operations involving different systems are overlapped as much as possible.

For MySQL, Sqoop can employ a direct-mode strategy using mysqlimport. Each map task spawns a mysqlimport process that it communicates with via a named FIFO file on the local filesystem. Data is then streamed into mysqlimport via the FIFO channel, and from there into the database.

Whereas most MapReduce jobs reading from HDFS pick the degree of parallelism (number of map tasks) based on the number and size of the files to process, Sqoop’s export system allows users explicit control over the number of tasks. The performance of the export can be affected by the number of parallel writers to the database, so Sqoop uses the CombineFileInputFormat class to group the input files into a smaller number of map tasks.

### Export and Transactionality

Due to the parallel nature of the process, often an export is not an atomic operation. Sqoop will spawn multiple tasks to export slices of the data in parallel. These tasks can complete at different times, meaning that even though transactions are used inside tasks, results from one task may be visible before the results of another task. Moreover, databases often use fixed-size buffers to store transactions. As a result, one transaction cannot necessarily contain the entire set of operations performed by a task. Sqoop commits results every few thousand rows, to ensure that it does not run out of memory. These intermediate results are visible while the export continues. Applications that will use the results of an export should not be started until the export process is complete, or they may see partial results. To solve this problem, Sqoop can export to a temporary staging table and then, at the end of the job — if the export has succeeded — move the staged data into the destination table

in a single transaction. You can specify a staging table with the --staging-table option.The staging table must already exist and have the same schema as the destination. It must also be empty, unless the --clear-staging-table option is also supplied.

## Sqoop Best PRactices and PErformance Tunning

### Import- Definite Number of Mappers

a. More mappers can lead to faster jobs, but only up to a saturation point. This varies per table, job parameters, time of day and server availability.

b. Too many mappers will increase the number of parallel sessions on the database, hence affect source DB performance affecting the regular workload of the DB.

Example : Definite number of mappers: --num-mappers

### Import - USe Direct Mode Where Ever Possible

a. Rather than using the JDBC interface for transferring data, the direct mode delegatesthe job of transferring data to the native utilities provided by the database vendor. For Eg. In the case of MySQL, the mysqldump and mysqlimport will be used for retrieving data from the database server or moving data back.

b. Escape characters, type mapping, column and row delimiters may not be supported. Binary formats don’t work.

### Import - Splitting Data / Boundry Query

a. By default, the primary key is used. Prior to starting the transfer, Sqoop will retrieve the min/max values for this column. Changed column with the --split-by parameter

b. Boundary Queries - What if your split-by column is skewed, table is not indexed or can be retrieved from another table?

If --split-by is not giving you the optimal performance you can use this to improve the performance further to Use a boundary query to create the splits using the option --boundary-query

Eg. sqoop import \

--connect 'jdbc:mysql://.../...' \

--direct \

--username uname --password pword \

--hive-import \

--hive-table query\_import \

--boundary-query 'SELECT 0, MAX(id) FROM a' \

--query 'SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND $CONDITIONS'\

--num-mappers 3

--split-by a.id \

--target-dir /data/import \

### Import - Using $CONDITIONS

a. $CONDITIONS is used by Sqoop process, it will replace with a unique condition expression internally to get the data-set. If you run a parallel import, the map tasks will execute your query with different values substituted in for $CONDITIONS. For Eg. Above query will execute parallel like this.

SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND a.id BETWEEN 0 AND 10;

SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND a.id BETWEEN 11 AND 20;

SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND a.id BETWEEN 21 AND 30;

Export:

### Export- BATCH Mode

a. Sqoop performs export row by row if we don’t leverage batch mode option.

b. Enabling batch mode will export more than one row at a time as batch of rows.

Eg: BATCH mode --batch

### Export- Specify Number of Records to Export

a. The above option will define how many number of rows should be used in each insert statements.

Eg:Specify the number of records to export -Dsqoop.export.records.per.statement=10

### Export- Specify Number of Records Per Transaction

a. The above option will define how many number of rows should be used in each transactions.

Eg: Specify the number of records per transaction - -Dsqoop.export.statements.per.transaction=10

8. Data Consistency --staging-table

a. In order to provide the consistent data access for the users in end database, using a staging table, Sqoop will first export all data into this staging table instead of the main table that is present in the parameter --table. Sqoop opens a new transaction to move data from the staging table to the final destination, if and only if all

parallel tasks successfully transfer data.

Have you encrypted files in HDFS?

Brief History:

Sqoop successfully graduated from the Incubator in March of 2012 and is now a Top-Level Apache project. Current version 1.4.6.

SQOOP Overview

# Flume

## Flume Concepts

Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and movinglarge amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust andfault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simpleextensible data model that allows for online analytic application.

Flume:

* Regulate Data Flow, flume can regulate data flow or send to temporary staging areas.
* Secondary Preprocessors, flume can send data to secondary preprocessors and then onto the final destination.
* Integrate Scheduling Tools, Flume can integrate into a number of scheduling tools, such as Linux logrotate or Cron.

Note: All this with scaling

History:

ASF released Flume NG on year 2012, Latest stable version is 1.6.0

Application of Flume:

Generally, most of the data that is to be analyzed will be produced by various data sources like applicationsservers, social networking sites, cloud servers, and enterprise servers. This data will be in the form of log files andevents.

Log file: In general, a log file is a file that lists events/actions that occur in an operating system. For example, webservers list every request made to the server in the log files.On harvesting such log data, we can get information about:The application performance and locate various software and hardware failures.The user behavior and derive better business insights.The traditional method of transferring data into the HDFS system is to use the put command. Let us see how to us

### Website ClickStream Analytics

Clickstreams, also known as click-paths, are the route that visitors choose when clicking or navigating through asite. A clickstream is a list of all the pages viewed by a visitor, presented in the order the pages were viewed, alsodefined as the ‘succession of mouse clicks’ that each visitor makes.

A clickstream will show you when and where a person came in to a site, all the pages viewed, the time spent oneach page, and when and where they left.Taken all together, as aggregated statistics, clickstream info will tell you, on average, how long people spend onyour site, and how often they return. It will also tell you which pages are the most frequently viewed.An interactive clickstream is a graphic representation of a clickstream; a list of pages seen in the order in whichthey were visited. The graphic allows you to click on the pages, and see what the visitor saw, hence the label'interactive'.The most obvious reason for examining clickstreams is to extract specific information about what people are doingon your site. Examining individual clickstreams will give you the information you need to make content-relateddecisions without guessing.

There is a wealth of information to be analyzed; you can examine visitor clickstreams in conjunction with any ofthe information provided by a good stats program: visit durations, search terms, ISPs, countries, browsers, etc. Theprocess will give your insight into what your visitors are thinking.

### Flume Data

There are of Two types, how the data flown into data repository,

* File data, Data generated by a system and written to files on same or another system and then is transported as files.
* Network Traffic, Data is generated by a system and transported through the network.

Note: Web logs and system logs be loaded by both method

### Problem Solved by Flume

1. Network performance, Lot of files over the network.
2. IO Performance
3. Adhoc Commands, removes linux commands
4. Security concerns, known standards for moving the data
5. Reliability

### Intorducing CURL

Linux Web Log Commands,

cURL: A Linux client to get and send files to servers. It covers nearly every transport protocol. You will use this along with FLUME for testing connections, download and check datatypes. It supports following protocols HTTP, FTP, TELNET, HTTPS, LDAP, FILE, DICT, GOPHER. Common Usage cURLs for GET,PUT, POST and DELETE files to web server:

Note: FTTP is not supported by Flume

*curl –request GET apache.orgcurl – request GET*

*localhost:8080 > web.txt*

It is designed to be work without user interaction and frequently used in scripts

|  |
| --- |
| -C, --continue-atOFFSET  -G, --get URL  -L, --location  -o, --output FILE  -O, --remote-name  -T, --upload-file FILE  -u, --user USER:PASSWORD  -v, --verbose  -X, --request COMMAND[GET,DELETE,POST,PUT]  -z, --time-cond DATE |

Download Web page using cURL

*>curl* [*http://apache.org*](http://apache.org)*> apache.html*

*>vim apache.html*

Redirect using cURL

*>curl – request GET gnu.org > gnu.html*

*>vim gnu.html*

L takes us to the Redirect web pages

*>curl –L -- request GET gnu.org*

o(brings the opportunity to rename the file as you download)and O (use the file name at the end of the URL)

Note: multiple file can be downloaded as in 3 line.

Z helps us look for updated files before(-) or after(+) , look for 4 th line

Telnet for cURL,

*>curl telnet://hdcentos:50070*

wget: A GNU client client for non-interactive download and upload of files from web. Primarily uses HTTP and HTTPS. (for slow and unstable n/w)

### Flume Vs Hadoop vs POSIX

*HDFS put Command*

The main challenge in handling the log data is in moving these logs produced by multiple servers to the Hadoopenvironment.Hadoop File System Shell provides commands to insert data into Hadoop and read from it. You can insert data intoHadoop using the put command as shown below.

*$ Hadoop fs –put /path of the required file /path in HDFS where to save the file*

*Problem with put Command*

We can use the put command of Hadoop to transfer data from these sources to HDFS. But, it suffers from thefollowing drawbacks:

* Using put command, we can transfer only one file at a time while the data generators generate data at amuch higher rate. Since the analysis made on older data is less accurate, we need to have a solution totransfer data in real time.
* If we use put command, the data is needed to be packaged and should be ready for the upload. Since thewebservers generate data continuously, it is a very difficult task.What we need here is a solution that can overcome the drawbacks of put command and transfer the "streamingdata" from data generators to centralized stores (especially HDFS) with less delay.

*Problem with HDFS*

In HDFS, the file exists as a directory entry and the length of the file will be considered as zero till it is closed. Forexample, if a source is writing data into HDFS and the network was interrupted in the middle of the operation(without closing the file), then the data written in the file will be lost.Therefore we need a reliable, configurable, and maintainable system to transfer the log data into HDFS.

Note: In POSIX file system such as Linux, whenever we are accessing a file (say performing write operation), andother programs can still read this file (at least the saved portion of the file). This is because the file exists on thedisc before it is closed.

### Available Solutions

To send streaming data (log files, events etc..,) from various sources to HDFS, we have the following tools availableat our disposal:

* Apache Flume
* Apache Kafka

Apache Flume

Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting largeamounts of streaming data such as log data, events (etc...) from various webservers to a centralized data store.

It is a highly reliable, distributed, and configurable tool that is principally designed to transfer streaming data fromvarious sources to HDFS.

Apache Kafka

Kafka has been developed by Apache Software Foundation. It is an open-source message broker. Using Kafka, wecan handle feeds with high-throughput and low-latency.

### FLUME Configuration files

1. Agent Conf Files: Used for configuring agents. (when naming the agent conf files, use action names explaining what they do)
2. Flume-env.sh: used for setting javahomw, java opts, Ganglia monitoring
3. Log4j.properties: used for setting logging properties

You should have installed below to have FLUME

1. Hadoop
2. HBase
3. Hive

### Installing Flume

1. Install flume and flume agent
   1. Make a symlink
   2. Configure the configuration directory
   3. Change the ownership of the filesystem directories
   4. Configure the working directory
2. Install connector (JDBC connector)
3. Edit the LOG4j.properties to set the log file path.

### Functional parts of Flume

1. Event, single transaction of data, normally a single line of data ending with a new line character.
2. Source, Source of data to be transmitted.(location of the data that needs to be transmitted)
3. Sink, repository location for the data transmitted.
4. Channel, a transmission and retention connection to manage event delivery.
5. Agent, is the combination of Source, Sink and channel.

### Creating Flume Agents

Note: Flume was created for handling weblogs into HDFS and tightly coupled to Hadoop.

* Line- oriented for textual data
* Every line is read as an event
* Do not use to transport binary data (Solution is to use AVRO)
* It doesn’t serialize data.

Syntax:

*flume-ng agent –conf conf\_name –conf-file agent.conf\_name –name agent\_name*

*flume-ng agent -c conf -f agent.conf -n agent*

Best practices for agents files:

1. Define and name the components of the agent.
2. Configure the source for incoming data.
3. Configure the sink for destination data.
4. Configure the channel.
5. Bind the source to the channel and then bind the sink to the same channel.

### First Agent for Flume (capturing network data)

This agent will open and listen port to 1111, Once it receives the input it writes the input to local file system.

1. Changing into Working and Conf directory.

*>cd flume/conf*

*> vim nc2file.conf*

To Define the components for nc2file,

*nc2file.sources = netcat\_source*

# we name the agent (nc2file) and functions (sources) here to identify the source.

To configure the source you have

*nc2file.sources.netcat\_source.type=netcat*

#We name the agent(nc2file) and fuctions (sources) and our name for that fuction(netcat\_source) and then parameter (type)

# Define the components for nc2file.conf

*nc2file.sources = netcat\_source*

*nc2file.sinks = file\_sink*

*nc2file.channels = memory\_channel*

#Configure the source

*nc2file.sources.netcat\_source.type = netcat*

*nc2file.sources.netcat\_source.bind = namenode-demo.c.alpine-canto-161617.internal #hostname*

*nc2file.sources.netcat\_source.port = 1111*

#Configure the sink

*nc2file.sinks.file\_sink.type = file\_roll*

*nc2file.sinks.file\_sink.sink.directory = /home/hdplearning1984/flume/data*

*nc2file.sinks.file\_sink.rollInterval = 0*

#configure the channel

*nc2file.channels.memory\_channel.type = memory*

*nc2file.channels.memory\_channel.capacity = 1000*

*nc2file.channels.memory\_channel.transactionCapacity = 100*

#Bind the source and the sink to the channel

*nc2file.sources.netcat\_source.channels = memory\_channel*

*nc2file.sinks.file\_sink.channel = memory\_channel*

1. Create the agent “nc2file”

*>flume-ng agent \*

*-- conf conf \*

*--conf-file conf/nc2file.conf \*

*--name nc2file*

Note: Run the agent in the same directory “specified configuration file doesn’t exists”, go to the working directory to fix this. Now the agent is listening to the port 1111

1. Lets communicate to this agent. We use cURL and telnet for this

*>curl* [*telnet://hdcentos:1111*](telnet://hdcentos:1111)

I am testing the flume ….

# lets check the data in the data directory

*>cd /flume/data*

*>vim 14272350928-2*

I am testing the flume ….

### Second Agent for Flume

This time we will write to hdfs sink and discuss arguments to parameters

/\*create the directory for sink\*/

*>hdfs dfs –mkdir weblogs*

*>hdfs dfs –ls*

/\*Check the nc2hdfs.conf\*/

*>cd flume/conf*

*>ls*

*>vim nc2hdfs.conf*

*# Define the components for nc2hdfs.conf*

*nc2hdfs.sources = netcat\_source*

*nc2hdfs.sinks = hdfs\_sink*

*nc2hdfs.channels = memory\_channel*

*#Configure the source*

*nc2hdfs.sources.netcat\_source.type = netcat*

*nc2hdfs.sources.netcat\_source.bind = hdcentos*

*nc2hdfs.sources.netcat\_source.port = 2222*

*#Configure the sink*

*nc2hdfs.sinks.hdfs\_sink.type = hdfs*

*nc2hdfs.sinks.hdfs\_sink.sink.hdfs.path = /user/hduser/weblogs*

*nc2hdfs.sinks.hdfs\_sink.sink.hdfs.fileType = DataStream*

*nc2hdfs.sinks.hdfs\_sink.sink.hdfs.writeFormat = Text*

*#configure the channel*

*nc2hdfs.channels.memory\_channel.type = memory*

*nc2hdfs.channels.memory\_channel.capacity = 1000*

*nc2hdfs.channels.memory\_channel.transactionCapacity = 100*

*#Bind the source and the sink to the channel*

*nc2hdfs.sources.netcat\_source.channels = memory\_channel*

*nc2hdfs.sinks.hdfs\_sink.channel = memory\_channel*

/\*run the agent \*/

*>flume-ng agent \*

*-- conf conf \*

*--conf-file conf/nc2hdfs.conf \*

*--name nc2hdfs*

/\*Test the agent\*/

*>curl* [*telnet://hdcentos:2222*](telnet://hdcentos:2222)

I am testing the flume ….

# lets check the data in the data directory

*>cd /flume/data*

*>vim 14272350928-2*

I am testing the flume ….

### Flume Source

1. Avro
2. Exec
3. Netcat
4. TCP
5. Syslog(UDP)

Configuring Source, When the agent receives enough data to create an event it then transmits the data through the channel to the sink. Source has no logic about how the data is stored in the sink. The source may have log for,

* Reading the data
* Translating the events
* Handling Failure

### Flume Sink

Many type of Sink

1. Avro
2. File Roll, is flume tool to sending data to local file system.
3. HDFS
4. HBASE
5. Logger, used for debugging flume agent configuration files, sending the output to log files
6. Null
7. IRC (Internet Ready Chat)

Configuring a Sink:

1. Sink is the import side of the source.
2. Sink waits for events to be received from the configured channels
3. Sink knows nothing of the source or the channels
4. Sink is responsible for (all are configurable)
   1. Output to destination
   2. Managing timeouts
   3. Managing retries
   4. Managing file rotations.

Rotation: The rotation of the sink files is configurable

1. Number of event
2. Size of logs
3. Time

Once a sink file reaches configured parameter the sink will “roll the logs” by opening a new file and closing the previous file. The new file has a.tmp appended to it. This designates it as the file to write data into until full.

### Flume Channels

Channels are the communication and retention mechanism that manage event delivery. Channels connect sources with sinks managing the streaming and buffering (how data is read and wrote) of data flows. Manages the differences in how data is read and written between sources and sinks

* Frequently data is read out faster than the ability to write.
* Sinks may not receive data while rolling data files.

Types of channels:

1. Memory Channel
2. File channel
3. JDBC channel

Memory Channel, it transport the data through memory only.

1. Reads from the source into the memory
2. Reads from memory into the sink upon demand from the sink
3. All data buffer in memory is volatile and will be loas if there is a memory interrupt.

File Channel, uses the disc as the buffer.

1. Reads from the source and writes into buffer file on disk.
2. Acknowledges a successful write to a buffer file.
3. Reads from the buffer file to the sink.
4. When the sink acknowledges a successful write of data, then the file channel deletes the buffer file.

JDBC Channel, Uses database as the buffer.

1. Similar to the file channel except that buffered writes are written to an embedded Derby database to store data.

Performance difference between memory store and disc IO is obvious. The decision to use one or the other is typically made based on the reliability of the source and the sink versus the value of the data being transmitted.

### File Channel to HDFS

We use the file channel to provide the integrity of the data delivered.

1. How to configure for log roll
2. How to configure for file channel.

### Avro Serialization

Serialization is the process of translating data structures into format that can be stored in a files or buffer, forwarded across the network and reconstructed without data loss.

Note: data structures are broken down into bytes and then sequences, it should have architecture independence as they move from one computational environment to another. All programing languages provide internal serialization but issue comes when moving between languages. For the object oriented language complexity is even more because of extensive use of object reference.

Avro is a remote procedure call and serialization framework. Similar to thrift and purposefully created for Hadoop.

1. Many objects have complex data structures, such as binary files, which do not transmit character by character.
2. A serialization framework is used to buffer and transmit such objects to gurarantee their integrity.
3. Process of ensuring a semantically identical clone of the original object is not straight forward.
4. Object compression and splitting are just two of the complexities that must be resolved.

Key attributes of Avro Serialization

* Framework, a data-persistence framework
* Binary Format, binary structured format can be both compressed and split as needed.
* Cross Language
  + Files can be created in any programming language.
  + Files are then processed by Hadoop.
  + Resulting files can be read by any third language.
* Complex Data structures, supports definition of hierarchical data structures.

Flume Uses Avro:

1. Flume can use Avro as a direct source.
2. Flume can use Avro to transmit to another agent.(sink)
3. Flume can also use the Avro standalone client to read a source and to send it to an Avro sink anywhere on the network.

### Flume Multi Hop

1. Avro client transmitting data into Avro source of the first agent1 this agent1 uses Memory channel to transmit the data to Avro sink. Avro Sink of the agent1 send to Avro source of agent2, which uses memory channel to transmit the data to HDFS Sink (in the code to local file system)

## Multiplex Agents for Flume

### Interceptors

Flume as a set of functions called interceptors. A plugin to manipulate and modify events. Inceptors are set of parameters in agent file. They add or remove meta data.

1. TIMESTAMP - Add the time stamp
2. HOST - Add host details
3. STATIC – Add static details
4. REG X FILTERING – remove a pattern

Frequently used to set up directory structure to parse out data sets. We use the interceptor parameter to add “TimestampInterceptor” to the “Netcat” source. This adds Unix timestamp time of the reading the data. Interceptor adds the timestamp at the time of reading the data.

Unix time stamp formats:

1. Year-month-day

filename.%Y-%m-%d

weblogs.2015-03-17

1. Year-month-day hour:minute:second

filename.%Y-%m-%d:%H:%M:%S

weblogs.2015-03-17 : 22 : 43:23

### TimeStamp Interceptors

*>cd /flume/conf*

*>vim nc2hdfs\_time.conf*

#now run the configuration

*>cd ..*

*>flume-ng agent –c conf –f conf/nc2hdfs\_time.conf –nnc2hdfs\_time*

#agent is running

#let us connect with cURL telnet

*>curl* [*telnet://hdcentos:6666*](telnet://hdcentos:6666)

I am testing this exercise

Done

#let us check weblog

*>hdfs dfs –ls weblogs*

#new directory is created with year month and date”nc2hdfs-2015-03-26”

*>hdfs dfs –ls weblogs/nc2hdfs-2015-03-26*

#here we have two files

*>hdfs dfs –cat weblogs/nc2dhfs-2015-03-26/FlumeData.143297237652346*

I am testing this exercise

Done

### Multiple Flow and Sources with Flume

Two key architectures for flume

1. Multiple sources.
2. Multiple Sink.

Use Case: (multisource)

Collecting logs from multiple web servers and consolidating them into HDFS, it is primarily done by Avro.

* First tier of Agents, agent1 will collect the data as Avro sources and then transmitted as Avro sinks normally as memory channel (data is not critical).
* Second tier of Agents, agent4 also as the Avro source and HDFS sink, Source side will consolidate the received events in to single channel. This is called Fan-In.

Writing multifunctional agents:

#Single configuration file, Single agent as two source, two sink, two channel it’s called multi flow

*multiagent.sources = netcat\_source exec\_source*

*multiagent.sink = log\_sink file\_sink*

*multiagent.channels = memory\_channel1 memory\_channel2*

# Single configuration file, two agents, as two source, two sink, two channel it’s called multi source flow

*multiagent1.sources = netcat\_source*

*multiagent1.sink = log\_sink*

*multiagent1.channels = memory\_channel1*

*multiagent2.sources = exec\_source*

*multiagent2.sink = file\_sink*

*multiagent2.channels = memory\_channel2*

Single configuration file, two agents, as two source, two sink, two channel it’s called multi source flow

Multi-Flow Flume Agent:

*>cd flume/conf*

*>vim multiflow.conf*

#let us run the agent

*>cd ..*

*>flume-ng agent –c conf –f conf/multiflow.conf –n multiflow*

#single agent is running two multiple data flow and we can test using cURL command

*>curl* [*telnet://hdcentos:7777*](telnet://hdcentos:7777)

I am tesing here 3

Done

*>ls data*

# transmitted small file into data directory.

*>cat data/18561237793-2*

I am tesing here 3

Done

*> hdfs dfs-ls weblogs*

# you can see another file is transmitted

Multi-Source Flume Agent:

#start the consolidation Agent

*>flume-ng agent –c conf –f conf/multisource.conf –n multisource*

#start the Collector A Agent

*>flume-ng agent –c conf –f conf/collectA.conf –n collectA*

#start the Collector B Agent

*>flume-ng agent –c conf –f conf/collectB.conf –n collect*

#all three agents are online

*>*[*telnet://hdcentos:1111*](telnet://hdcentos:1111)

Checking port 1111 that is Collector A

port 1111 end

*>telnet://hdcentos:3333*

Checking port 3333 that is Collector B

port 3333 end

# check the hdfs

*>hdfs dfs –ls weblogs/multisource*

#multiple output files

*>hdfs dfs –cat weblogs/multisource/weblog.124213491263-2*

Checking port 1111 that is Collector A

Checking port 3333 that is Collector B

port 1111 end

port 3333 end

### Multiple Sink

Sink can receive from only on channel, means one to many relation between Source and Channel but one to one relation between Channel and Sink. Use case will be web log into long term repository and immediate analysis and it’s referred as fan out. Multiple sinks work in either of two way by using “selector” functions.

Note:

* 1. Many source can be configured to one channel- True I guess it’s False
  2. Only one sink can be configured to a channel – True
  3. A single source can be configured to multiple channels – False I guess its True

Note: my understanding A channel can listen to only one source, A Sink can listen to only one Channel

1. Replicating, events are replicated down to all channels
2. Multiplexing, uses logic to determine to which channel an event should be send, its dependent on the value of a specified header field.

### Flume Troubleshooting

Best practices for agents files:

1. Define and name the components of the agent.
2. Configure the source for incoming data.
3. Configure the sink for destination data.
4. Configure the channel.
5. Bind the source to the channel and then bind the sink to the same channel.

Note: Using “channels” when you need “channel” for the sink

File Sink: Records the input data exactly as it was received with no alterations.

Logger Sink: A debug tool that records the information captured by the source while adding metadata and events. Switch to Logger Sink when you face issue with flume working.



### Flume Events

An event is the basic unit of the data transported inside Flume. It contains a payload of byte array that is to betransported from the source to the destination accompanied by optional headers.

### Flume Components

A typical Flume event wouldhave the following structure:

* Agent
* Source
* Channel
* Sink

### Agent

An agent is an independent daemon process (JVM) in Flume. It receives the data (events) from clients or otheragents and forwards it to its next destination (sink or agent). Flume may have more than one agent. Followingdiagram represents a Flume Agent.

### Source

A source is the component of an Agent which receives data from the data generators and transfers it to one ormore channels in the form of Flume events.Apache Flume supports several types of sources and each source receives events from a specified data generator.

Example − Avro source, Thrift source, twitter 1% source etc.

Put simply, Flume sources listen for and consume events. Events can range from newline-terminated stringsin stdout to HTTP POSTs and RPC calls — it all depends on what sources the agent is configured to use. Flumeagents may have more than one source, but must have at least one. Sources require a name and a type; the typethen dictates additional configuration parameters.

On consuming an event, Flume sources write the event to a channel. Importantly, sources write to their channelsas transactions. By dealing in events and transactions, Flume agents maintain end-to-end flow reliability. Eventsare not dropped inside a Flume agent unless the channel is explicitly allowed to discard them due to a full queue.

### Channels

A channel is a transient store which receives the events from the source and buffers them till they are consumedby sinks. It acts as a bridge between the sources and the sinks.

These channels are fully transactional and they can work with any number of sources and sinks.

Example − JDBC channel, File system channel, Memory channel, etc.

Channels are the mechanism by which Flume agents transfer events from their sources to their sinks. Eventswritten to the channel by a source are not removed from the channel until a sink removes that event in atransaction. This allows Flume sinks to retry writes in the event of a failure in the external repository (such as HDFSor an outgoing network connection).

For example, if the network between a Flume agent and a Hadoop clustergoes down, the channel will keep all events queued until the sink can correctly write to the cluster and close itstransactions with the channel.

Channels are typically of two types: in-memory queues and durable disk-backed queues. In-memory channelsprovide high throughput but no recovery if an agent fails. File or database-backed channels, on the other hand, aredurable. They support full recovery and event replay in the case of agent failure.

### Sink

A sink stores the data into centralized stores like HBase and HDFS. It consumes the data (events) from the channelsand delivers it to the destination. The destination of the sink might be another agent or the central stores.

Example − HDFS sink, HBase sink, Avro sink, Kafka sink, etc.

Sinks provide Flume agents pluggable output capability if you need to write to a new type storage, just write a Javaclass that implements the necessary classes. Like sources, sinks correspond to a type of output: writes to HDFS orHBase, remote procedure calls to other agents, or any number of other external repositories. Sinks remove eventsfrom the channel in transactions and write them to output. Transactions close when the event is successfullywritten, ensuring that all events are committed to their final destination.

### Additional Components of Flume Agent

What we have discussed above are the primitive components of the agent. In addition to this, we have a few morecomponents that play a vital role in transferring the events from the data generator to the centralized stores.

### Interceptors

Interceptors are used to alter/inspect flume events which are transferred between source and channel.

### Channel Selectors

These are used to determine which channel is to be opted to transfer the data in case of multiple channels. Thereare two types of channel selectors:

* Default channel selectors: These are also known as replicating channel selectors they replicates all theevents in each channel.
* Multiplexing channel selectors: These decide the channel to send an event based on the address in theheader of that event.

### Sink Processors

These are used to invoke a particular sink from the selected group of sinks. These are used to create failover pathsfor your sinks or load balance events across multiple sinks from a channel.

### Flume - DataFlow

Multi-hop Flow: Within Flume, there can be multiple agents and before reaching the final destination, an event may travel throughmore than one agent. This is known as multi-hop flow.

Fan-out Flow: The dataflow from one source to multiple channels is known as fan-out flow. It is of two types:

* Replicating: The data flow where the data will be replicated in all the configured channels.
* Multiplexing: The data flow where the data will be sent to a selected channel which is mentioned in theheader of the event.

Fan-in Flow: The data flow in which the data will be transferred from many sources to one channel is known as fan-in flow.

### Failure Handling

In Flume, for each event, two transactions take place: one at the sender and one at the receiver. The sender sendsevents to the receiver. Soon after receiving the data, the receiver commits its own transaction and sends a“received” signal to the sender. After receiving the signal, the sender commits its transaction. (Sender will notcommit its transaction till it receives a signal from the receiver.)

### End to End Flow

1. A flow in Flume starts from the Client.
2. The Client transmits the Event to a Source operating within the Agent.
3. The Source receiving this Event then delivers it to one or more Channels.
4. One or more Sinks operating within the same Agent drains these Channels.
5. Channels decouple the ingestion rate from drain rate using the familiar producer-consumer model of dataexchange.
6. When spikes in client side activity cause data to be generated faster than can be handled by theprovisioned destination capacity can handle, the Channel size increases. This allows sources to continuenormal operation for the duration of the spike.
7. The Sink of one Agent can be chained to the Source of another Agent. This chaining enables the creationof complex data flow topologies.

Because Flume’s distributed architecture requires no central coordination point. Each agent runs independently ofothers with no inherent single point of failure, and Flume can easily scale horizontally.

### Summary

* Client send events to agents
* Each agent hosts Flume components such as source, interceptor, channel selector, channels, sinkprocessors and sinks.
* Source & sinks are active components, channels are passive components.
* Source accepts events, passes them through interceptors, and if not filtered, puts them on channelsselected by the configured channel selector.
* Sink processor identifies a sink to invoke, that can take events from a channel and send it to its next hopdestination.
* Channel persistence provides end to end delivery.

# Kumar Preparation

## Kumar Preparation

### Gcloud

hadoop fs -ls /user/hdplearning1984

hadoop fs -mkdir /user/hdplearning1984/inputdataset

hadoop fs -mkdir /user/hdplearning1984/outputdataset

hadoop fs -copyFromLocal ~/pig/touch.txt /user/hdplearning1984

hadoop fs -cat /user/hdplearning1984/inputdataset/touch.txt

raw = LOAD 'touch.txt' USING PigStore(':');

raw = LOAD '/user/hdplearning1984/inputdataset/touch.txt' USING PigStore(':');

hadoop fs -copyFromLocal /etc/passwd /user/hdplearning1984

hadoop fs -cat /user/hdplearning1984/inputdataset/passwd

B = load 'passwd' using PigStorage(':');

DUMP B;

STORE B INTO '/user/hdplearning1984/outputdataset/pig1';

hadoop fs -ls /user/hdplearning1984/outputdataset/pig1

hadoop fs -cat /user/hdplearning1984/outputdataset/pig1/part-m-00000

create table txnrecords(userid STRING,Permission STRING,some\_number INT,rest STRING) row

format delimited

fields terminated by ':'

lines terminated by '\n'

stored as textfile;

LOAD DATA LOCAL INPATH '/etc/passwd' INTO TABLE txnrecords;

describe txnrecords;

describe formatted txnrecords;

Warehouse folder:

/apps/hive/warehouse

/apps/hive/warehouse/retail.db

hadoop fs -ls /user/hive

sqoop export

--connect 'jdbc:mysql://hdcentos/CRMDB'

--username uname --password pword

--table salesforce

--export-dir salesteam

--input-field-terminated-by ‘\t’

### Hive

Hive;

create table tab\_kuma (

name string,

address string,

empid int)

row format delimited

fields terminated by '\t'

stored as textfile;

create table tab\_kumar2 (

name string,

address string,

empid int

)

row format delimited

field terminated by','

stored as textfile;

alter table tab\_kumar2 rename table\_kumar2;

alter table tab\_kumar2 change coulmn name full\_name string before address;

load data local inpath 'file path' overwrite into table table\_kumar2;

load data local inpath 'file path' into table table\_kumar2;

# export the data into tmp directory

inster overwrite directory "file\_name1"

select concat(col1,","col2,",",) from customer where col3="something";

hadoop fs -ls /tmp

# Lets execuet the script

hive -f ./scriptname.hql

# Partitioned table

create table logs(ts BIGINT, line STRING)

partitioned by (dt STRING, country STRING);

# Load data into partitioned data

load data local inpath 'path'

into table logs

partition (df STRING, country STRING)

### Pig

sales = LOAD 'sales/sales2014.csv' using PigStorage(',') as (custid:INT, ...);

DESCRIBE sales;

salesLimit = LIMIT sales 10;

DUMP sales

DUMP salesLimit

#filter

saleOH = FILTER sales by state='OH';

giftsalesNW = FILTER sales BY (state=='ID' or state=='OR' or state == 'WA');

bigCustomer = FILTER sales BY (sales>30000);

#Select

subset1 = FOREACH sales GENERATE custid,customer,city,sales/1000 as salesK;

#Join

cityJoin = Join sales BY zip, zipcity BY zipcode;

#GROUP

salesGRoup = Group sales ALL;

salesCount = FOREACH salesGroup GENERATE group, COUNT(sales);

salesGRoup = Group sales BY state;

# Interview Questions

## HDFS/ Hadoop – Interview Questions

### encrypted files in HDFS

Have you encrypted files in HDFS?

This question is asked to understand the candidate’s knowledge of the encryption functionality in HDFS. HDFS implements transparent, end-to-end encryption.  Transparent - data read from and written to special HDFS directories is transparently encrypted and decrypted without requiring changes to user application code.  End-to-end - means the data can only be encrypted and decrypted by the client. Encryption is supported on both meaning data on persistent media, such as a disk (at-rest encryption) as well as when data is travelling over the network (in-transit encryption).

### Read in Hdfs files in HDFS

During a HDFS Read operation how does a DataNode with a replicated block gets picked up by NameNode for a read operation?

 Client contacts NameNode to determine the locations of the blocks for the first few blocks in the file.

 For each block, the NameNode returns the addresses of the DataNodes (sorted by proximity to the client) that have a copy of that block.

 Client then connects to the first (closest) DataNode for the first block in the file. Data is streamed from the DataNode back to the client.

 When the end of the block is reached the connection to the DataNode is closed and then client will start reading the next block from the best (closest) DataNode for the block.

### Limit Directory Usage

How do you set limits on HDFS usage for a particular directory?

The Hadoop Distributed File System (HDFS) allows the administrator to set quotas for the number of names (Name quota) used and the amount of space (Space quota) used for individual directories. File, directory creations and block allocations will fail if the quota is exceeded.

 Name quota

Sets hard limit on the number of file and directory names under a specific directory.

 Space quota

Sets hard limit on the number of bytes used by files under a specific directory. Each replica of a block counts against the quota.

Illustration

$ dfsadmin -setQuota <N><directory>

Set the name quota to be N for directory.

$ dfsadmin -setSpaceQuota <N><directory>

Set the space quota to be N bytes for directory.

### Data Across Cluster

How do you copy the file from one Hadoop cluster to another?

DistCp (distributed copy) is a tool used for large inter/intra-cluster copying.

Illustration

$ hadoop distcp hdfs://nn1:8020/foo/bar hdfs://nn2:8020/bar/foo

### Rebalance Operations

What is the need for a rebalance operation in HDFS?

Over a period of time when new DataNodes to an existing cluster the data across the cluster will be unbalanced and data might not be uniformly placed across the DataNodes.

HDFS provides a tool called balancer that analyzes block placement and rebalances data across the DataNode.

Illustration

$ hadoop balancer

### SafeMode

What is a Safe mode?

During startup the NameNode waits for DataNodes to report their blocks and the NameNode hold the block locations in memory. During this time, the NameNode will be in Safe mode so that it does not prematurely start replicating the blocks though enough replicas already exist in the cluster. In Safe mode the NameNode is essentially in read-only mode. NameNode will leave Safe mode automatically after the DataNodes have reported that most file system blocks are available.

### Counters In Haddop

What is the benefit of using counters in Hadoop?

Counters are a useful for gathering details about the job in a centralized fashion. Assume you have a 100 node cluster and a job with 100 mappers is running in the cluster on 100 different nodes. Let’s say you would like to know each time you see an invalid record in your Map phase. You could add a log message in your Mapper so that each time you see an invalid line you can make an entry in the log. But consolidating all the log messages from 100 different nodes will be time consuming. You can use a counter instead in your Map program and increment the value of the counter every time you see an invalid record. The nice thing about using counters is that is gives you a consolidate value for the whole job rather than showing 100 separate outputs. In this specific example you would see the number of input lines printed at the end of job.

### Block

What is a ‘block’ in HDFS?

When a file or dataset is uploaded to HDFS, the dataset is divided in to fixed size chunks or blocks and placed across different nodes in the cluster.

The size of block is configurable. In older versions of Hadoop the size of the block was set to 64 MB by default in the later version it is moved up to 128 MB.

Storing the data in blocks help with parallel and distributed access of data and replication for fault tolerance.

### InputSplit and Block

What is InputSplit and how it is different from a Block?

Input splits are a logical division of your records whereas HDFS blocks are a physical division of the input data. You could have a record that started in one block and ends in another block. Block is a physical division of data and does not take in to account the logical boundary of records whereas InputSplit considers the logical boundaries of records as well.

### Speculative Execution

A job running on a Hadoop cluster could be divided into many tasks. In a big cluster some of these tasks could be running slow for various reasons, hardware degradation or software misconfiguration etc. Hadoop initiates a replica of a task when it sees a task which is running for some time and failed to make any progress, on average, as the other tasks from the job. This replica or duplicate execution of task is referred to as Speculative Execution.

When a task completes successfully all the duplicate tasks that are running will be killed. So if the original task completes before the speculative task, then the speculative task is killed; on the other hand, if the speculative task finishes first, then the original is killed.

### Combiner

What is a Combiner?

The primary goal of combiners is to optimize/minimize the number of key value pairs that will be shuffled accross the network between mappers and reducers and thus to save as most bandwidth as possible. Usage of the Combiner is optional. If this pass is suitable for your job, instances of the Combiner class are run on every node that has run map tasks. The Combiner will receive as input all data emitted by the Mapper instances on a given node. The output from the Combiner is then sent to the Reducers, instead of the output from the Mappers. The Combiner is a "mini-reduce" process which operates only on data generated by one machine.

### Replica Placement Strategy

What is a Replica Placement Strategy?

Replica placement defines how the blocks are stored in the cluster. By default the 1st replica is placed on the local machine, otherwise a random data node. The 2nd replica is placed on a data node that is on a different rack. The 3rd replica is placed on a data node which is on the same rack as the first replica.

### Limit Directory Usage

### Limit Directory Usage

### Limit Directory Usage

### Limit Directory Usage

## Flume – Interview Questions

### Multiplexing

What is multiplexing in Flume?

A flume event can be sent to a subset of available channels when an event’s attribute matches a preconfigured value.

Illustration

In this illustration, if an event attribute called location is set to newyork, then it should go to memory-channel-newyork and, if it’s chicago then it should go to memory-channel-chicago and memory-channel-default

Below excerpt from the flume configuration file demonstrate that.

agent.sources.avro-collection-source.selector.type = multiplexing

agent.sources.avro-collection-source.selector.header = location

agent.sources.avro-collection-source.selector.mapping.newyork = memory-channel-newyork

agent.sources.avro-collection-source.selector.mapping.chicago = memory-channel-chicago memory-channel-default

### Replication

What is replication in Flume?

In a replication setup, there are multiple channels between the source and the sink. Each Flume event from the source is sent to all the channels. The sink then picks up the events from the channel it is paired with.

Illustration

Below Flume agent configuration is to define a source, 2 channels and 2 sinks (file in local file system and file in HDFS). All the events from the source will be sent to both the channels.

# Flume Components

agent.channels = memory-channel-local memory-channel-hdfs

agent.sources = tail-source

agent.sinks = local-sink hdfs-sink

# Define a memory channel on agent called memory-channel.

agent.channels.memory-channel-local.type = memory

agent.channels.memory-channel-hdfs.type = memory

# Define a source on agent and connect to channel memory-channel.

agent.sources.tail-source.type = exec

agent.sources.tail-source.command = tail -F /home/temp/logfile.log

agent.sources.tail-source.channels = memory-channel-local memory-channel-hdfs

# Define a sink that outputs to local file.

agent.sinks.local-sink.type = file\_roll

agent.sinks.local-sink.sink.directory = flume/local-file

agent.sinks.local-sink.sink.rollInterval = 60

agent.sinks.local-sink.channel = memory-channel-local

# Define a sink that outputs to hdfs.

agent.sinks.hdfs-sink.type = hdfs

agent.sinks.hdfs-sink.hdfs.path = flume/replicate

agent.sinks.hdfs-sink.hdfs.fileType = DataStream

agent.sinks.hdfs-sink.hdfs.rollCount = 5

agent.sinks.hdfs-sink.hdfs.inUseSuffix = .tmp

agent.sinks.hdfs-sink.channel = memory-channel-hdfs

### Components

What are the key components involved in a Flume agent?

 Source - A Flume source consumes events delivered to it by an external source like a web server.

 Channel - When a Flume source receives an event, it stores it into one or more channels. The channel is a passive store that keeps the event until it’s consumed by a Flume sink.

 Sink - The sink removes the event from the channel and puts it into an external repository like HDFS (via Flume HDFS sink) or forwards it to the Flume source of the next Flume agent (next hop) in the flow.

Illustration

Below Flume agent configuration is to define a source (log file), channel and sink (file in local file system)

# Flume Components

agent.channels = memory-channel

agent.sources = tail-source

agent.sinks = local-file-sink

# Define a memory channel on agent called memory-channel.

agent.channels.memory-channel.type = memory

# Define a source on agent and connect to channel memory-channel.

agent.sources.tail-source.type = exec

agent.sources.tail-source.command = tail -F /home/temp/logfile.log

agent.sources.tail-source.channels = memory-channel

# Define a sink that outputs to local file system.

agent.sinks.local-file-sink.type = file\_roll

agent.sinks.local-file-sink.sink.directory = flume/local-file

agent.sinks.local-file-sink.sink.rollInterval = 60

agent.sinks.local-file-sink.channel = memory-channel

## Pig/Hive/Sqoop

Pig

65. How pig scripts are converted in to MapReduce jobs?

Pig creates 3 plans - Logical, Physical and MapReduce plan to transform a Pig Latin script to a set of MapReduce jobs. In each of these plans Pig tries to optimize the script and at the end produce a MapReduce job.

Logical plan

 Describes the logical operators that Pig will use to execute the script.

 Some optimizations are done on this plan. For e.g., filters are pushed up as possible in the logical plan.

Physical plan

 Pig produces a physical plan after optimizing the logical plan.

 This plan describes the physical operators Pig will use to execute the script, without reference to how they will be executed in MapReduce.

 The load, store functions and the actual paths that will be used have been resolved

 Pig identifies 3 important operators. For instance, COGROUP operator will be replaced by Local Rearrange, Global Rearrange, and Package.

 Local Rearrange is the operator Pig uses to prepare data for the shuffle by setting up the key.

 Global Rearrange is for the shuffle.

 Package sits in the reduce phase and directs records to the proper bag

MapReduce Plan

 Pig takes the physical plan and decides how it will place its operators into one or more MapReduce jobs.

 Pig will go through the physical plan looking for all operators that require a new reduce. This occurs anywhere there is a Local Rearrange, Global Rearrange, and Package.

 Pig will look for places to do physical optimizations. For Illustration, it looks for places the combiner can be used, and whether sorts can be avoided by including them as part of the sorting Hadoop does in the shuffle.

 When all the above steps are complete, Pig will produce a MapReduce plan.

66. When do you use pig and when do you use Hive?

Pig is well suited for data transformation jobs as Pig by nature is a Data Transformation Language.

Hive is well suited to be used by Data Analysts and Scientists for data analysis. Hive QL is more aligned with SQL and will be easy to use for anyone with database background.

67. What operations you can't do in pig that you can do in MapReduce?

We need MapReduce when we need very deep level and fine grained control on the way we want to process our data. Sometimes, it is not very convenient to express what we need exactly in terms of Pig and Hive queries.

When your data is hierarchical rather than row-based or if your data is highly unstructured standard MapReduce jobs can be very efficient.

With standard MapReduce you get full control to minimize the number of MapReduce jobs that your data processing flow requires, which translates into performance. But it requires more time to code and introduce changes.

68. What pig operator or clause violate the MapReduce convention that all the key value pairs for the same key must go to same Reducer?

Below 2 operators or clauses violates the MapReduce convention that all the key value pairs for the same key must go the same Reducer.

1. USING SKEWED

2. ORDER BY

69. How do you compare 2 datasets?

Using JOIN operation on the datasets and using the comparison columns as the key columns

COGROUP can also be used giving the columns we are trying to compare

Illustration

grunt> joinbysymboldate = JOIN price\_table by (symbol, date) , dividend\_table by (symbol, date);

grunt> cgrp = COGROUP price\_table BY (symbol, date), dividend\_table by (symbol, date);

Below sample output show 3 rows. Only 2nd row had matching columns on both datasets.

((CSL,2009-05-14),{},{(NYSE,CSL,2009-05-14,0.155)}) ((CSL,2009-08-12),{(NYSE,CSL,2009-08-12,32.65,32.73,32.17,32.54,528900,32.39)},{(NYSE,CSL,2009-08-12,0.16)}) ((CSL,2009-08-13),{(NYSE,CSL,2009-08-13,32.58,33.19,32.49,33.15,447600,32.99)},{})

70. What happens behind the scenes in terms of MapReduce operation when you do a JOIN operation in Pig?

 In the Map phase, the input files are read separately by individual Map function and records from each input are tagged to indicate the input it came from.

 Join key(s) is used in the shuffle so that all records with the same join key goes to the same Reducer

 All records from the left tables are sent to the reducer first and cached in memory

 All records from the rightmost table are streamed to the reducer.

 As each of records from right most records arrives, it is crossed with each record from the left side to produce an output record.

 Reducer execute the join criteria on records as records from the rightmost table come through

71. How do you efficiently join 2 datasets in Pig?

In a regular join, records from the left hand side tables are cached in the reducer. Records from the right most table are streamed to the reducer. So make sure that the table on the right hand side is the biggest table.

72. How do you efficiently join two datasets which are already sorted?

The column which is used to sort the datasets is very important in this case and if your input datasets are already sorted on the join key the join can be done in the map phase. In Pig this is referred to as the merge join and can be implemented using the hint using 'merge’ in your join instruction. This join is very efficient because entire shuffle and reduce phase is avoided because the data is already sorted.

Illustration

jnd = join table1 by cola, table2 by colb using 'merge';

73. How do you efficiently join two datasets if once dataset is big and one dataset is small?

If you have tables which are really smaller in size and can fit in memory, the entire reduce phase can be avoided using the USING REPLICATED hint -

 Entire Join is done at Map side and Reduce phase is ignored

 All right most tables are loaded in to memory on the nodes running the Mapper using DistributedCache

 Left most table is streamed to the mapper

 USING REPLICATED hint supports INNER & LEFT join

 RIGHT JOIN is not supported

74. Assume you are doing a join and you notice that all but one reducer is running for a long time how do you address the problem in Pig?

Pig collects all of the records for a given key together on a single reducer. In many data sets, there are a few keys that have three or more orders of magnitude more records than other keys. This results in one or two reducers that will take much longer than the rest. To deal with this, Pig provides skew join.

 In the first MapReduce job pig scans the second input and identifies keys that have so many records.  In the second MapReduce job, it does the actual join.  For all except the records with the key(s) identified from the first job, pig would do a standard join.  For the records with keys identified by the second job, bases on how many records were seen for a given key, those records will be split across appropriate number of reducers.  The other input to the join that is not split, only the keys in question are then split and then replicated to each reducer that contains that key

Illustration

jnd = join cinfo by city, users by city using 'skewed';

75. How do you achieve a Join without using JOIN operator in Pig?

COGROUP does first half of a join. The keys are collected together, but the cross product is not done. COGROUP combined with a FOREACH and flattening the resulting bag is equivalent to a join.

COGROUP + FOREACH = Join

Illustration

A = load 'input1' as (id:int, val:float);

B = load 'input2' as (id:int, val2:int);

C = cogroup A by id, B by id;

describe C;

C: {group: int,A: {id: int,val: float},B: {id: int,val2: int}}

76. How do you achieve a SEMI join in Pig?

Pig Latin does not have a SEMI JOIN operator. COGROUP can be used to perform a SEMI join. Look for questions under Hive to understand what is a SEMI JOIN.

Illustration

daily = load 'NYSE\_daily' as (exchange:chararray, symbol:chararray, date:chararray, open:float, high:float, low:float, close:float, volume:int, adj\_close:float);

divs = load 'NYSE\_dividends' as (exchange:chararray, symbol:chararray, date:chararray, dividends:float);

grpd = cogroup daily by (exchange, symbol), divs by (exchange, symbol);

sjnd = filter grpd by not IsEmpty(divs);

final = foreach sjnd generate flatten(daily);

77. How do you perform non equi join in PIG?

Pig’s join operator supports only equi-joins, that is, joins on an equality condition. Because general join implementations in MapReduce depend on collecting records with the same join key values onto the same reducer. Where as a Non-equi-join would require collecting records that does not have the same join key values and it is difficult to do.

In Pig you can achieve a non equi join using cross followed by filter.

Illustration

crossed = cross table1, table2 nonequi = filter crossed by table1::date < table2::date;

78. How do you calculate the number of lines in a file in PIG?

Illustration

LOGS= LOAD 'log';

LOGS\_GROUP= GROUP LOGS ALL;

LOG\_COUNT = FOREACH LOGS\_GROUP GENERATE COUNT(LOGS);

79. How do you LOAD a sequence file in PIG? Pig comes with a LOAD function to load Sequence Files – SequenceFileLoader. Illustration

REGISTER /home/ubuntu/pig-0.12.0/contrib/piggybank/java/piggybank.jar; DEFINE SequenceFileLoader org.apache.pig.piggybank.storage.SequenceFileLoader(); seqdataset = LOAD 'input/ sequencefile' USING SequenceFileLoader AS (key:long, data:chararray);

80. How do you register a UDF in Pig?

Use the REGITER instruction.

To locate the jar file, Pig first checks the classpath. If the jar file can't be found in the classpath, Pig assumes that the location is either an absolute path or a path relative to the location from which Pig was invoked. If the jar file can't be found, an error will be printed: java.io.IOException: Can't read jar file: myudfs.jar.

Illustration

REGISTER myudfs.jar;

81. How do you store the output into a Sequence File in Pig?

There is no STORE function natively available in Pig to do so. Elephand Bird (Open Source) initiative from Twitter has functions to store a Sequence File.

<https://github.com/kevinweil/elephant-bird/>

Hive

82. What is a SerDe?

SerDe is a short name for Serializer and Deserializer. The SerDe interface allows users to instruct Hive as to how a record should be processed. The Deserializer takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate. The Serializer, however, will take a Java object that Hive has been working with, and turn it into something that Hive can write to HDFS or another supported system. SerDe is commonly used to map data with complex structures to a row in Hive tables.

Below illustration gives the position of Serializer and Deserializer during write and read operation.

Row object --> Serializer --><key, value> --> OutputFileFormat --> HDFS files

HDFS files --> InputFileFormat --><key, value> --> Deserializer --> Row object

83. What is the functionality of ObjectInspector in Hive?

ObjectInspector is used by SerDe during Input and Output processing of the dataset. When the data is read from the dataset, the object inspector is used by the deserialize() method in SerDe to construct individual fields out of a deserialized record. When the data is written back to HDFS, the object inspector is used by the serialize() method in SerDe to get the individual fields in the record in order to convert the record to the appropriate type.

84. How will you use Hive to process datasets which are unstructured and semi-structured for e.g. dataset with email or JSON messages?

 First step is to use appropriate InputFormat to read the data. For e.g. if the file is a Sequence File use SequenceFileInputFormat

 Next step is to create a SerDe to instruct Hive as to how a record should be processed.

 Once the SerDe is created, create a table with the desired structure and use the SerDe in the table.

Below article from Cloudera has step by step instruction on how to create a SerDe to read a JSON dataset.

<http://blog.cloudera.com/blog/2012/12/how-to-use-a-serde-in-apache-hive/>

85. Explain how do you use case statement in hive select?

Illustration

SELECT exchange, price\_open, price\_close,

CASE

WHEN volume < 200000 THEN 'low'

WHEN volume >= 200000 AND volume < 400000 THEN 'middle'

WHEN volume >= 400000 AND volume < 600000 THEN 'high'

ELSE 'very high'

END AS volume\_level FROM stocks;

86. What is the difference between SORT BY and ORDER BY in Hive?

ORDER BY performs a total ordering of the query result set. This means that all the data is passed through a single reducer, which may take an unacceptably long time to execute for larger data sets.

SORT BY orders the data only within each reducer, thereby performing a local ordering, where each reducer’s output will be sorted. You will not achieve a total ordering on the dataset. Better performance is traded for total ordering.

87. What is DISTRIBUTE BY in Hive?

DISTRIBUTE BY controls how map output is divided among reducers.

DISTRIBUTE BY is used along with SORT BY and it is used to ensure that the records with the same value for the specified column will go to the same reducer, and then use SORT BY to order the data the way we want.

Illustration

With the below query all records with symbol AMZN (for e.g.) will be sent to the same reducer and the records will then be sorted by the ymd column.

hive> SELECT s.ymd, s.symbol, s.price\_close FROM stock\_quotes s DISTRIBUTE BY s.symbol SORT BY s.symbol ASC, s.ymd ASC;

88. How do you perform non equi join in Hive?

Just like Pig, Hive does not support non equi join for the same reason that it is difficult to implement non equi joins in MapReduce.

Follow <https://issues.apache.org/jira/browse/HIVE-3133> for any developments on the non equi join implementation in Hive

89. What is LEFT SEMI JOIN?

A is LEFT SEMI JOIN returns records from the left hand table if records are found in the right hand side table that satisfy the ON predicates. It’s a special, optimized case of the more general inner join. Most SQL dialects support an IN ... EXISTS construct to do the same thing.

90. Why LEFT SEMI JOIN is more efficient than INNER JOIN?

In a LEFT SEMI JOIN, for a given record in the left hand table, Hive can stop looking for matching records in the right hand table as soon as any match is found. At that point, the selected columns from the left hand table record can be projected.

91. How do you perform a LEFT SEMI JOIN in PIG and HIVE?

Illustration - Hive

SELECT s.ymd, s.symbol, s.price\_close FROM stocks s LEFT SEMI JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol;

Illustration - Pig

Pig does not have a SEMI join like Hive. But SEMI join can be achieved by using COGROUP

daily = load 'NYSE\_daily' as (exchange:chararray, symbol:chararray, date:chararray, open:float, high:float, low:float, close:float, volume:int, adj\_close:float); divs = load 'NYSE\_dividends' as (exchange:chararray, symbol:chararray, date:chararray, dividends:float); grpd = cogroup daily by (exchange, symbol), divs by (exchange, symbol); sjnd = filter grpd by not IsEmpty(divs); final = foreach sjnd generate flatten(daily);

92. What is an alternative for using IN clause in Hive?

Below query will not work in Hive.

SELECT s.ymd, s.symbol, s.price\_close FROM stocks s WHERE s.ymd, s.symbol IN (SELECT d.ymd, d.symbol FROM dividends d);

To achieve the above query use LEFT SEMI JOIN

SELECT s.ymd, s.symbol, s.price\_close FROM stocks s LEFT SEMI JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol;

93. How do you do minus operation between 2 datasets?

The Oracle MINUS operator is used to return all rows in the first SELECT statement that are not returned in the second SELECT statement.

There is no direct MINUS operator in Pig or Hive, instead you can achieve this by LEFT JOIN and do a filter out records from the second table which are null.

Illustration

SELECT t1.\* FROM table1 t1 LEFT OUTER JOIN table2 t2 ON t1.col = t2.col WHERE t2.col IS NULL

94. What is a Bucket Map join?

When all the below conditions are met we can perform a Bucket Map join and it will be performed at the map side.

 All join tables are bucketized, and each small table’s number of buckets can be divided by big table’s number of buckets.  Bucket columns and Join columns are same

Below property should be set to true to enable this feature.

set hive.optimize.bucketmapjoin = true;

95. What is a Sort Merge Bucket Map Join?

When all the below conditions are met we can perform a Sort Merge Bucket Map Join and it will be performed at the map side.

 All join tables are bucketized, and each small table’s number of buckets can be divided by big table’s number of buckets.  2.Bucket columns and Join columns are same  Join columns are sorted

Below properties should be set to enable this feature.

set hive.optimize.bucketmapjoin = true; set hive.optimize.bucketmapjoin.sortedmerge = true; set hive.input.format=org.apache.hadoop.hive.ql.io.BucketizedHiveInputFormat; 96. How do you calculate the number of lines in a file in Hive?

hive> select count(\*) from stocks;

97. How does partitioning a table in Hive improve performance?

Partitioning a Hive table with the right partition column has performance benefits. Partitions are used for distributing data horizontally and help in organizing data in a logical fashion. For example if you have a very big USERS table and your users in the table are spread across different states in the country then you can partition the table by country and state.

Partitioning tables changes how Hive structures the data storage and Hive will now create subdirectories reflecting the partitioning structure like .../users/country=ABC/state=XYZ

So next time when a query is issued against the table with country=ABC and state=XYZ the MapReduce job will target only that specific subdirectory.

98. What is the need for Buckets when there is already a way to distribute the data in Hive using Partitions?

Partitioning a Hive table with the right partition column has performance benefits however a table design that creates too many partitions may result in unbalanced partitions that would be result in unbalanced MapReduce execution. Other drawback is many small partitions create an overhead to NameNode since it has to manage the metadata for many files and blocks now.

You can design a table to use country for the top-level partition and state as the bucketing column, the value of this column will be hashed by a user-defined number into buckets. Users from the same state will always be stored in the same bucket. Assuming the number of states is much greater than the number of buckets, each bucket will have users from many states. With bucketing the number of buckets is fixed so it does not fluctuate with data. Bucketing is also used in doing efficient map-side joins.

Illustration

CREATE TABLE usersuser\_id BIGINT, firstname STRING, lastname STRING)

PARTITIONED BY(country STRING)

CLUSTERED BY(state) INTO 25 BUCKETS;

99. If your table is partitioned with 2 partition keys and the no of buckets is set to 5 how many buckets you would see in each partition?

Dataset or records will be bucketed only at the end of the partition tree. First the table will be partitioned by the 2 partition keys and then the data will be divided and stored in 5 buckets.

/partition1 /partition2 /bucket1 /bucket2 /bucket3 /bucket4 /bucket5

100. What is an RCFile?

RCFile (Record Columnar File) is a file format in Hadoop. A table stored in RCFile is first horizontally partitioned into multiple row groups. Then, each row group is vertically partitioned so that each column is stored independently.

Partitioning table horizontally first makes sure that all the columns for a given row are in the same HDFS block. Vertically partition groups all column values of columns together and it gives the advantages of any column oriented store like faster query performance on column level aggregation, efficient compression at column level.

101. Are random writes possible with RCFile?

RCFile does not allow arbitrary data writing operations. RCFile creates and maintains an in-memory column holder for each column. When a record is appended, all its fields will be scattered, and each field will be appended into its corresponding column holder. In addition, RCFile will record corresponding metadata of each field in the metadata header.

RCFile provides two parameters to control how many records can be buffered in memory before they are flushed into the disk. One parameter is the limit of the number of records, and the other parameter is the limit of the size of the memory buffer. 102. How do you register UDF in Hive?

Once your UDF is ready, package it in a jar and add it to the Hive classpath. Once Hive is started up with your jars in the classpath, the final step is to register your function as described in Create Function.

Illustration

CREATE TEMPORARY FUNCTION my\_func AS 'com.example.TempFunc';

To add jar to the classpath for all jobs initiated from that session.

Illustration

hive> add jar /tmp/my\_jar.jar;

Added /tmp/my\_jar.jar to class path

hive> list jars;

my\_jar.jar

103. Assume you have a sales table in a company and it has sales entries from salesman around the globe. How do you rank each salesperson by country based on their sales volume in Hive?

Hive support several analytic functions and one of the functions is RANK() and it is designed to do this operation.

Lookup details on other window and analytic functions - <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics>

Illustration

Hive>SELECT rep\_name, rep\_country, sales\_volume, rank() over (PARTITION BY rep\_country ORDER BY sales\_volume DESC) as rank FROM salesrep;

104. MySQL GROUP\_CONCAT() function returns a string with concatenated non-NULL value from a group. What is a similar function in Hive?

MySQL has a useful function known as GROUP\_CONCAT, which combines all the elements of a group into a single string using a user-specified delimiter.

Hive already has a UDAF called COLLECT\_SET that adds all input but it removes duplicates because it internally uses java.util.Set collection to store the collected values. Set automatically remove duplicate entries.

If we want to keep the duplicate values we need to create a UDAF and replace the instances of Set with instances of ArrayList. The result of the aggregate will be a single array of all values.

105. What are the different types of functions in Hive?

 User Defined Function (UDF)  User Defined Aggregate Function (UDAF). UDAF takes in multiple values, perform aggregate operations and can return arrays or structures. For e.g. SUM (..)  User-Defined Table Generating Functions (UDTF). UDFs cannot return multiple columns or multiple rows. UDTF address this issue and return multiple columns and even multiple rows.

106. How can you do compression at the field level?

In Hive compression can be enabled at intermediate Map Output and Output files. However in HBASE you can enable compression at the column family level.

Illustration

hbase> disable 'test'

hbase> alter 'test', {NAME => 'cf', COMPRESSION => 'GZ'}

hbase> enable 'test'

107. How do you delete records in Hive?

Data in Hive is stored as files behind the scenes so there is no concept of delete in Hive. Instead the dataset behind the table or partitions can be overwritten using INSERT OVERWRITE

108. Since compressed data involves decompression during processing, how does this increase performance?

Compressed files in HDFS help in reducing the amount of needed disk space. But compression does not allow splitting which is a bad thing for MapReduce jobs. However this can be overcome by compression using Sequence files.

But enabling intermediate map output compression can be of huge benefit and will certainly overcome the overhead of decompressing the data at the reduce side.

Assume you have 100 mappers and 1 reducer. Now reducer is trying to bring the data from 100 mappers as they are getting completed. If the size of the data is small the data transfer rate over the network would be higher and will result in a performance boost.

Also, enabling map output compression benefits not only the individual job’s performance but also the overall cluster performance in terms of network utilization when compressed data is transferred between maps and reduce.

109. Let’s say you are deploying a jar and you have a 100 node cluster. Do you have to deploy the jar to all 100 nodes?

No. When you execute a MapReduce program using hadoop jar command the client will upload the jar and needed files into HDFS. Then the jars will be used across all the nodes since all the needed files are on HDFS.

110. If you have only 10 map slots in your cluster, how does a job with 15 mappers run in the cluster?

If all 10 map slots are available the job will execute in 2 waves. In the first wave 10 mappers will run and in the next wave the remaining 5 mappers will run.

Sqoop

111. How is data ingested into your Hadoop cluster?

Data is brought in to the cluster from several data points - Database and files from other systems and files from outside vendors (in some cases).

Sqoop is used if the data is ingested from the database into HDFS In case of files, Shell Scripts can be used to copy the files into HDFS Flume will be used if you are bringing in Application log files into HDFS

112. From where (which node or server) the data gets ingested into the Hadoop cluster?

The question is if you are uploading a file into HDFS from which node you would perform the copyFromLocal or put operation. More specifically will you perform the copyFromLocal or put operation from one of your nodes in the cluster or from a node outside of the cluster.

It is also safe to perform the data ingestion process from a node outside of the cluster. This node is called the EDGE node. This is a good practice for couple of reasons.

 When uploading a file you can make sure the file is good and clean (not a virus etc) before you bring the file in to the cluster  Using a EDGE node will avoid any unnecessary load on the cluster that may incur during the ingestion process

113. How do you control the number of mappers in Sqoop Import?

Using –m <n> or --num-mappers <n> in your Sqoop import command. The default number is 4.

Illustration

sqoop import --connect jdbc:mysql://mysql.address.internal/hadoop --table stocks -m 2 --target-dir /user/temp/stocks

114. How do you import compressed data using Sqoop?

Sqoop import command takes compression arguments to enable the compression and specify the codec.

Use -z,--compress to enable compression and --compression-codec <c> to specify a codec (default gzip)

Illustration

sqoop import --connect jdbc:mysql://mysql.address.internal/hadoop --table stocks --compress

--target-dir /user/temp/stocks

115. How do you bring in only the recent data using Sqoop?

Sqoop incremental import option can be used for this. Incremental import works in 2 modes - append and lastmodified and both relies on the column mentioned in --check-column to be examine when determining which rows to import.

 append - use this when importing a table where new rows are continually being added with increasing row id values.

 lastmodified - use this when rows of the source table may be updated, and each such update will set the value of a last-modified column to the current timestamp.

Illustration

sqoop import --connect jdbc:mysql:// mysql.address.internal/hadoop --table stocks --target-dir /user/temp/stcoks\_increment --incremental append --check-column id

116. How does the state of the last import gets stored for subsequent imports when using incremental import?

--last-value should be mentioned with the value from the previous import so that the current import will pull only the records greater than the value mentioned in --last-value

At the end of an incremental import, the value which should be specified as --last-value for a subsequent import is printed to the screen. When running a subsequent import, you should specify --last-value in this way to ensure you import only the new or updated data. This is handled automatically by creating an incremental import as a saved job, which is the preferred mechanism for performing a recurring incremental import.

Illustration

--To create sqoop job

sqoop job --create incrementalImportJob -- import --connect jdbc:mysql://mysql.address.internal /hadoop --table stocks --target-dir /user/temp/widgets\_job --incremental append --check-column id

--To list sqoop jobs

sqoop job –list

--To show details of sqoop job

sqoop job --show incrementalImportJob

--To execute sqoop job

sqoop job --exec incrementalImportJob

117. How do you only keep the most recent updated data when using incremental load using lastmodified mode?

When you use the lastmodified option in your incremental import, sqoop will import records if the record is updated between imports. This means that sqoop will import the same row during different imports. So how can we make sure we only keep the latest row and ignore the rest.

sqoop-merge allows you to combine two datasets where entries in one dataset should overwrite entries of an older dataset. For Illustration, an incremental import run in last-modified mode will generate multiple datasets in HDFS where successively newer data appears in each dataset. The merge tool will "flatten" two datasets into one, taking the newest available records for each primary key.

118. How can you override the split column used by Sqoop import operation?

You can also explicitly choose a different column with the --split-by argument. If split-by is not mentioned Sqoop will use the primary key (if present) to split data between mappers.

Illustration

sqoop import --query 'SELECT a.\*, b.\* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS'

--split-by a.id --target-dir /user/foo/joinresults

119. How you imported data from AS400 to Sqoop?

The question is to ask what databases are supported by Sqoop. You can use Sqoop with any other JDBC-compliant database. First, download the appropriate JDBC driver for the type of database you want to import, and install the .jar file in the $SQOOP\_HOME/lib directory on your client machine.

# Inceptize Running Notes

http://pig.apache.org/docs/r0.9.1/api/org/apache/pig/piggybank/storage/CSVExcelStorage.html

Hive is a HQL layer on top of hadoop that provides layer for performing OLAP or QL Analysis using MR as processing engine and hdfs as storage layer.

--> ELT Model, varacity of the data is more if you store the data in raw form.

--> HQL is used

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*6feb2016 Hbase

Hbase will provide the capabilites more than- rdbms-especially huge data handling, joining, real time queirs,low latncy response etc.

It can handle 50k column, no need to join,

Example for NoSQL:

Document oriented --Mango DB.

/\*no joins are required \*/

/\* u don't mofiy

Internally its stored as Json or xml.

used for mobile application

Graph DB--

Linkedin, Facebook uses Graph db- easy to handle heirarchy data.

Nodes and vertex, each person is node and check how related to other person.

Column Oriented--

Hbase, casandra

Hbase (column oriented)

dynamic schemas-- no relational model

auto sharing

replication

integrated cache

performacne, flexible,no joins, low cost

CAP Theorem:

1.Consistecy

2.Partitioning

3.Availability

------

Hbase is a hadoops DB which provides distibuted storage & retrival machanism using hdfs as

Java api for retrival and processing., column oriented,stores the data in H files.

-Hbase if the DML is performed it can happen in memory where us in RDMS DML has to happen in Disc.

-Hbase as in memeory

-Wirte many read meany

-Hbase is good for performaing DML in OLTP

-Hbase is schema.

-Hbase will not use Map reduce.

-Hbase will use Java APIs for getting and putting data.

-

------

Hbase Storage :

-table -- collection of row, its similar to database, you have single table have the customer, sales in the same record

-Column -- collection of key value pair.

-cell -- Each value of the column.

-timestamp -- version of the cell.

-Memstore -- used for update/deleet

-BlockCache-- Used for Read

-We ahve 8 region server, 8 region (similar to table), Each table as 4 column fam

ily, h File will be fore in the 200+ node.

ORC Snappy provides very good performance is Hive

----------------------------+++++

Advantage of Tiz over Hive:

Assume,

TIz is DAG , Hive is Non DAG,

TEZ Reusability of container - > you avoid the Container creation issue.

TEZ will write the intermediant data in Memory and final o/p is HDFS, MR writes to HDFs,

TEZ, MR is dynamic.

1. Single container

--------------------------------+++++++++++++++================================

Map reduce;

compo

combiner:

Combiner ,works on mapper and reducer phase.

Combiner reduces network traffic.

Combiner users reducer code, just need to extend it.

no of execution of combiner cann't be predicted it depends on the data. Mapper and reducer its not the case.

Combiner will work only on associative and commutative princile.

No of MApper is equal to no of input split.

No of mapper is less than or equal to no of block.

No of reducer is equal to number of partitions.

Map reduce reads the data from HDFS based on key value pairs using byte off set value through input split.

Byte offset value will be the key and one record will be value for that key. (if file format is textinputformat)

Mapper producess the o/p in key values pairs and spilled to local FS.

Mapper istance runs based on number of line.

Record reader reads the record based on the input format from the input format, whihc inturn reads from input split.

Joins are optizied in Hive and Pig.. In Mapper reduce we need to manage it.

Shuffler:

data from the local FS from various Mapper node to container memory of various reducer.

Input format types:

Textinputformat --> unstructed data

Keyvaluetextinputformat --> structured data

Nlineinputformat --> Mapper instances are reduced , and its not equal to number of records (merges line.)

Writable comparison --- Need more explation.

Map side join:

When you have 2 data set, one data set having 1million records (data in multiple blocks-- B,c,D) and other is having 1000 records (single data node-- A).

Then Data in Node A will be moved to all other node B,C,D, hence data i/o will be less. You also have Reducer side Join.

------------------------------++++++++++++++++++++++++++++++++++++++++

Map Reducer:

Mapper

Reducer

Driver

Note: if I can write the customized Reducer code extra there would be one more components.

Driver:

Main Method will be called.

Provide the cofigration info,

input format all configurations.

configuration conf = new configuration();

Configuration is Class

create a object called conf.

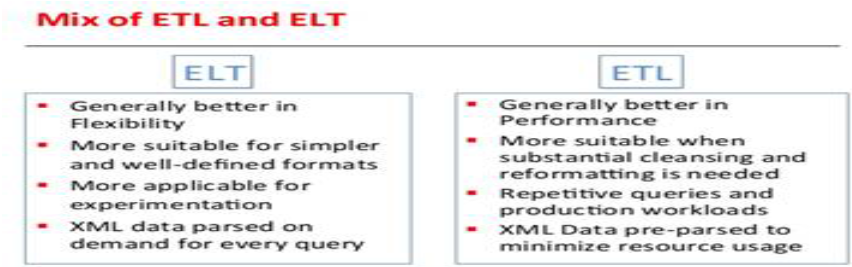
Job job = new Job(conf, "Trnding")

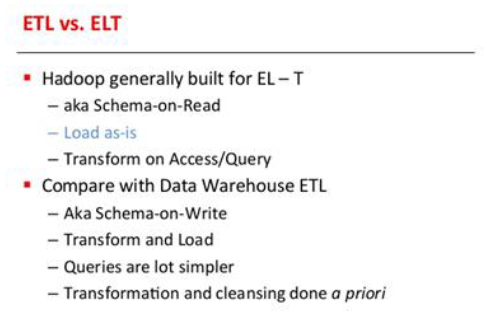
Job is a class

job is a object.

## Hadoop-Hive

* Structured/semi structured Data in HDFS logically into Tables.
* Uses the HiveQL Language.Very similar to SQL.
* Turns HiveQL into Map Reduce Jobs.
* Uses Map-Reduce for execution and HDFS for storage – but any system that implements Hadoop FS API.
* Structured data with rich data types (structs, lists and maps).
* Directly query data from different formats (text/binary) and file formats (Flat/Sequence).
* SQL as a familiar programming tool and for standard analytics.
* Allow embedded scripts for extensibility and for nonstandard applications Rich Meta Data to allow data discovery and for optimization.
* Hadoop is gengerally build for EL-T





Hive is good for

Works best for Batch processing and OLAP needs.

Provides SQL like environment and easily adaptable.

Capable of storing and processing huge volume of structure and semi structure data with

minimal coding effort.

Uses schema on read instead of schema on write, hence true raw data can be stored.

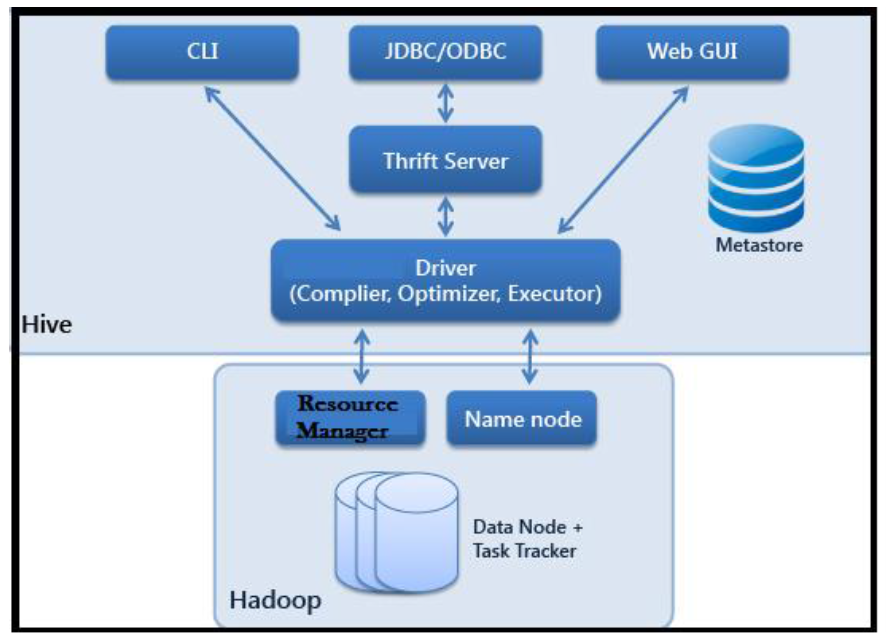
It is familiar, fast, scalable, and extensible.

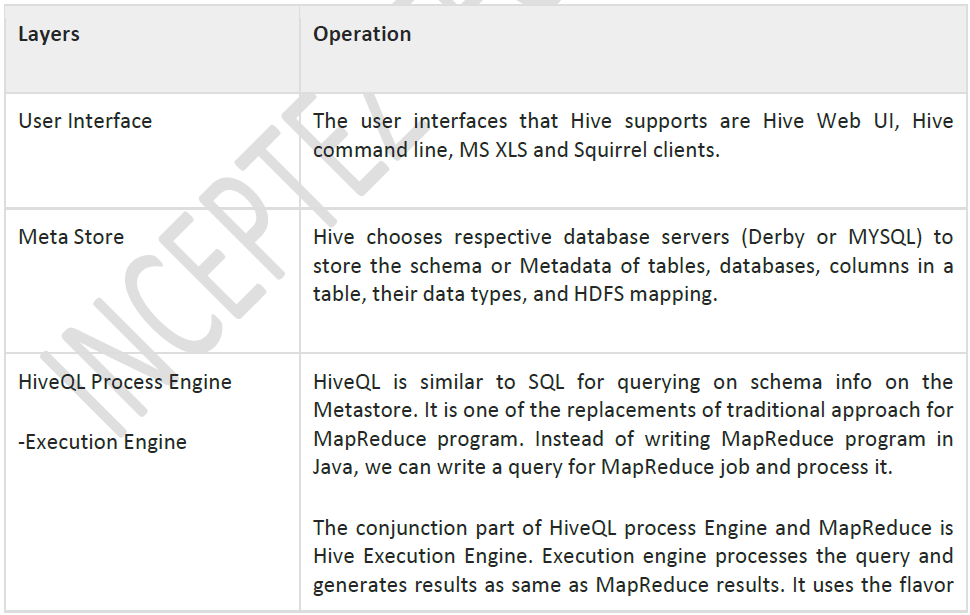
Hive is not good for

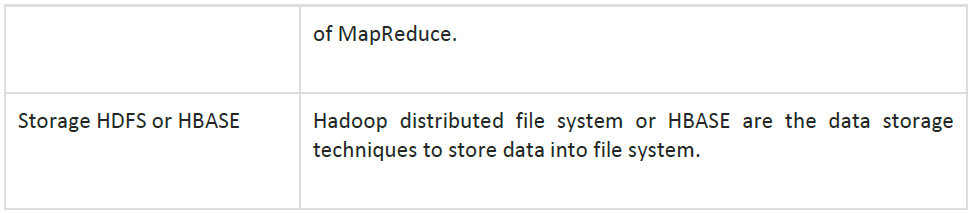
OLTP requirements where frequent changes happen in data.

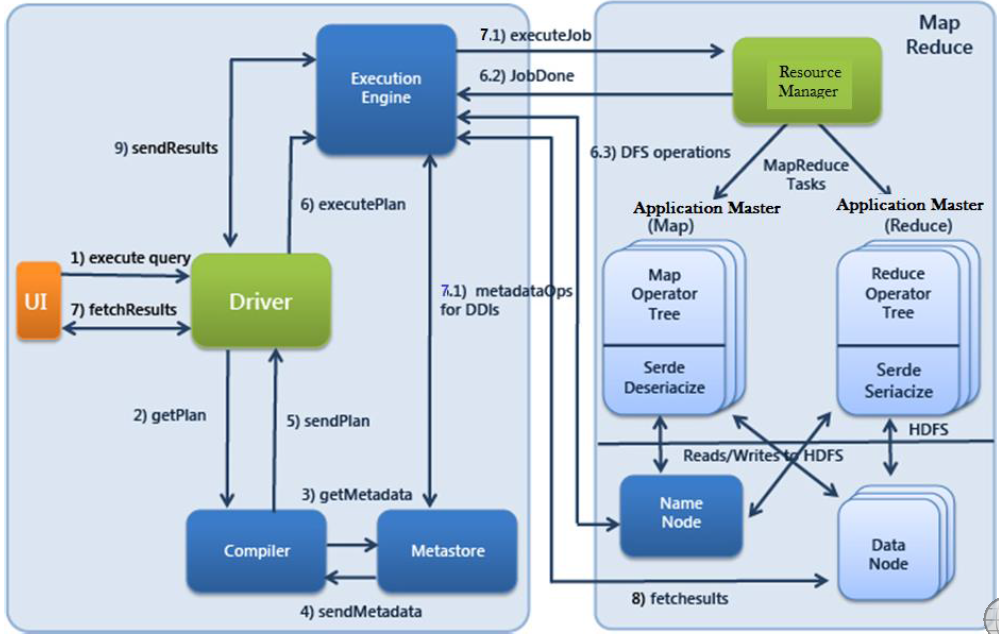
Not a relational database, it is a complementary approach to data warehouse on top of Hadoop.

Not good for real-time and interactive low latency queries.









## Hadoop-INTERVIEW QUESTIONS

1. What is your last project and how did you use Hadoop?

Have a well constructed answer for this question since most likely this will be your very first question in an interview.

* Mention about the company in 2 sentences.
* Explain the use case of the project.
* What are the input data points for Hadoop cluster – Database, flat files from other applications etc.
* Hadoop ecosystem tools you are using for specific tasks. For e.g. Sqoop for data ingestion, Pig for transformation, Hive for analysis etc.

Let’s assume you are working for a big online retail shopping company (Retail Inc.). The answer to the question could be something like below –

Retail Inc. is a fast growing online retail website specializing in Men’s and Women’s high fashion apparel. One of the key challenges we face at Retail Inc. is maintaining customer loyalty and to provide them with a customized online shopping experience.

Retail Inc. collects and analyzes large amounts of data from our customers 24×7 from several data points – websites, mobile apps, credit card, social media and coupons redemption. Data from these data points could be structured and unstructured in few cases.

All this data is collected, aggregated and analyzed in the Hadoop cluster to find customer’s shopping patterns and to provide them with a unique personalized shopping experience. The analysis that is done in our Hadoop cluster help our strategic teams make cross sell, up sell decisions and devise targeted marketing strategies for our customers.

2. What are your day to day responsibilities?

* Explain the key business areas you support.
* What you support and how – data ingestion, data transformation, administration, optimization etc.
* Tools you work with – Pig, Hive, Sqoop etc.

Sample Response

I am responsible for the data transformation tasks which are written using PIG scripts and the scripts run every night. I also work with the business to gather business requirements and write, optimize Hive queries for data analysis. I also work with the business to gather new requirements to write transformation tasks using Pig for which I design the data ingestion process using Sqoop.

3. Tell us about your Hadoop cluster?

Here are few items to keep in mind when you construct an answer.

* Distribution and version used – For e.g. CDH4 or HDP2
* No of nodes – For e.g. 60 node cluster
* Data volume of the cluster – For e.g. 60 TB cluster (180 TB with replication factor of 3)
* Mention whether you are running Hadoop 1 or Hadoop 2 – For e.g. YARN or JobTracker setup
* Special setup (if any) – HA, Federation etc.

Sample Response

We are running a CDH4 cluster with 120 Nodes. Our total data volume is 200 TB with a replication factor of 3. Our NameNode setup is configured in an Active-Standby High Availability setup. Our applications run on YARN.

4. How do you deploy MapReduce job or Pig or Hive scripts into your cluster?

Each company is different in how code or scripts are deployed in to production. A sample response can look something like below if you use Oozie to coordinate your jobs.

Sample Response

All our production jobs are coordinated using Oozie. Once the development process is complete we bundle the MapReduce job, Pig or Hive scripts into an Oozie job along with coordinator, workflow xml and properties files needed to run the job. The bundle is placed on one of our Edge nodes (look for the question on Edge node below) outside the cluster along with instructions to your admin or deployment team on how to start the Oozie job.

During deployment, the administrator will follow our instructions to pick up the files and deploy them to HDFS and start the job.

5. How did you decide on the number of nodes when you designed your cluster?

This question is aimed at someone who also involved in the initial design of the cluster. There is no straight forward answer to this question. Here are some of the practices or approaches that would go in when designing a cluster.

The primary reason for building a Hadoop cluster is to distribute the execution of your jobs to reduce the amount of execution time. The most practical way to do decide on the number of nodes is to start with a few node cluster in a development environment. Use the Benchmarking tools that come out of the box like TestDFSIO and MRBench to benchmark the current cluster. This will give you a very high level idea of whether your initial memory and CPU requirements in your initial test cluster will meet your needs.

Next step is to run an optimized MapReduce job or Pig/Hive script which you would except to run on a production environment. Make sure the job runs on the same amount of data as it would run in prod.

Now adjust the memory settings for your Map or Reduce and any other performance optimizations needed to make the job execute in the agreed SLA time. With this baseline at hand, think about the maximum number of scheduled jobs and adhoc requests your cluster will be expected to handle at peak usage time. Extrapolate your baseline in terms of memory, CPU and number of nodes to meet the SLA for your peak usage time. Always leave 20% room for unexpected spikes in usage.

Also remember the key benefit of Hadoop cluster is horizontal scalability so you add nodes to your cluster anytime.

By Kumar

DFSIO is part of the Hadoop distribution and can be found in "hadoop-mapreduce-client-jobclient-\*-tests.jar" for MR2. There are two types of DFSIO tools, but this article discusses TestDFSIO only. TestDFSIO is Distributed I/O Benchmark tool

6. How do you monitor jobs (MapReduce, Pig or Hive) in production?

Each company is different in how the jobs are monitored in production. A sample response can look like below. The answer is structured to match with our answer to the deployment question.

Sample Response

All our jobs are coordinated and executed using Oozie. Our support team uses the Oozie web interface to look up the status of the jobs. All jobs are configured to send out emails to the support team in case of failure and success. Upon receiving a failure email, the support team will try to investigate the problem and call the appropriate individuals as necessary.

Note: Administration tools like Cloudera Manager (for e.g.) also provide ways to monitor jobs in your cluster.

7. How do you debug a failed job in production?

A job could fail for several reasons in a distributed environment like Hadoop. The key to fix a failed job is to answer the below 3 questions in the same order as listed. 

* What failed?
* Where it failed?
* Why it failed?

There could you several reasons for failure. Some of the most common failure reasons are below.  Data related runtime exceptions  Resource related (for e.g. Out Of Memory)  Application or programming errors

Sample response

I will get the application id of the failed job and will go to the Application Master’s URL to look up the failed application. I will look for the overall progress made by the job. That is, I will look for how many mappers or reducers failed over all to get an idea about the extent of the failure. (This will answer what failed)

If the failure is with one or two mappers, I will drill down to the failed task note down the node it ran on and will inspect the logs from the Application Master UI itself (which will show only few lines from the error) to see the reason for the failure. (This will answer where it failed)

If you want to see the entire log you will ask the admin (if you don’t have necessary permissions) to get the entire log from the node or Cloudera Manager for the failed. Once the reason for failure is identified I will take appropriate action to fix the job. (This will answer why it failed)

8. How do you debug a performance issue or a long running job?

This is an open ended question and the interviewer is trying to see the level of hands-on experience you have in solving production issues. Use your day to day work experience to answer this question. Here are some of the scenarios and responses to help you construct your answer. On a very high level you will follow the below steps.

Understand the symptom 🡪Analyze the situation 🡪Identify the problem areas 🡪 Propose solution

Scenario 1 - Job with 100 mappers and 1 reducer takes a long time for the reducer to start after all the mappers are complete. One of the reasons could be that reduce is spending a lot of time copying the map outputs. So in this case we can try couple of things.

* If possible add a combiner to reduce the amount of output from the mapper to be sent to the reducer.
* Enable map output compression - this will further reduce the size of the outputs to be transferred to the reducer.

Scenario 2 - A particular task is using a lot of memory which is causing the slowness or failure, I will look for ways to reduce the memory usage.

* Make sure the joins are made in an optimal way with memory usage in mind. For e.g. in Pig joins, the LEFT hand side tables are sent to the reducer first and held in memory and the RIGHT most table is streamed to the reducer. So make sure the RIGHT most table is largest of the datasets in the join.
* We can also increase the memory requirements needed by the map and reduce tasks by setting - mapreduce.map.memory.mb and mapreduce.reduce.memory.mb

Scenario 3 - Understanding the data helps a lot in optimizing the way we use the datasets in PIG and HIVE scripts.

* If you have smaller tables in join, they can be sent to distributed cache and loaded in memory on the Map side and the entire join can be done on the Map side thereby avoiding the shuffle and reduce phase altogether. This will tremendously improve performance. Look up USING REPLICATED in Pig and MAPJOIN or hive.auto.convert.join in Hive
* If the data is already sorted you can use USING MERGE which will do a Map Only join
* If the data is bucketted in hive, you may use hive.optimize.bucketmapjoin or hive.optimize.bucketmapjoin.sortedmerge depending on the characteristics of the data

Scenario 4 - The Shuffle process is the heart of a MapReduce program and it can be tweaked for performance improvement.

* If you see lots of records are being spilled to the disk (check for Spilled Records in the counters in your MapReduce output) you can increase the memory available for Map to perform the Shuffle by increasing the value in io.sort.mb. This will reduce the amount of Map Outputs written to the disk so the sorting of the keys can be performed in memory.
* On the reduce side the merge operation (merging the output from several mappers) can be done in disk by setting the mapred.inmem.merge.threshold to 0

9. How do you install, configure a Hadoop cluster?

There are several ways to install and configure a Hadoop cluster. Here are some of Hadoop management tools like

* Cloudera Manager,
* Apache Ambari,
* Tools like Puppet,
* Apache Whirr
* Manual setup with shell scripts (or other relevant scripting languages)

Sample response

In my client location, Cloudera Manager is used to configure, install, monitor, add or remove services, add or remove hosts to our Hadoop cluster

Note: In most companies developers will not have access to Cloudera Manager for the production environment. Only Administrators will have access to the tool.

10. How do you monitor your Hadoop cluster?

Hadoop related services can be monitored by tools like Cloudera Manager or Apache Ambari. These tools will provide a unified view of the cluster and will be primarily used by Hadoop Administrators. In many companies OS level monitoring will be handled by sophisticated tools like Ganglia, ITRS and will be handled by separate infrastructure teams.

Sample response

In our company we use Cloudera Manager to monitor Hadoop related services or get a unified status of the cluster. Our infrastructure team use Ganglia to monitor the nodes at the OS level.

11. How is security handled in your cluster?

We have seen Hadoop clusters where security is not configured at all but in places where security is configured below configuration is usually the norm.

Kerberos for authentication Apache Sentry for authorization

12. Where is your cluster hosted - in house or cloud?

A Hadoop cluster can be hosted in house, that is managed and maintained in one of the company’s datacenter and it can be hosted on the cloud like Amazon Web Services (AWS) or Google cloud etc. The answer to the question depends on your setup.

13. Have you used Amazon AWS?

This question is an important question to an interviewer especially if their cluster is hosted on AWS. Even if your cluster is hosted in house and even if you are not using AWS make sure you know the services offered by AWS and have a basic understanding of how to use them and how they work. Check out <http://aws.amazon.com/> for offers on new AWS accounts

14. What are the services you have used in AWS?

If you are not familiar with AWS make sure you get yourself familiarized at least with the following services. Sometimes no experience with AWS could be a deal breaker for some interviewers so make sure you at least have a basic understanding of different services offered by AWS.

* Elastic Cloud Compute (EC2)
* Elastic Map Reduce (EMR)
* Simple Storage Service (S3)
* Identity and Access Management (IAM)

fs.default.name

This is used to specify the default file system and defaults to your local file system that's why it needs be set to a HDFS address. This is important for client configuration as well so your local configuration file should include this element.

16. If the block size of the cluster is 64 MB and you have a file with 100 KB in size, how much space does the file really occupy when it is stored on disk?

File size on disk is determined by the cluster size (or block size) of the underlying local file system. If the cluster size of the local file system is 4 KB then the file of size 100 KB will need 25 clusters (25 \* 4 KB) to store the file on disk. So the answer is, the file size on disk would be 100 KB and not 64 MB

If the file size is 5 KB then 2 clusters are needed to store the file and the file size on disk would be 8 KB. HDFS does not use any more disk space than it is required by the local file system to store the file on disk.

17. Why Hadoop uses huge block size of 128 MB or 256 MB while storing files in HDFS?

Hadoop is designed to handle big files. If the block size is smaller, the NameNode will have to manage a lot of blocks and this would put an enormous stress on the NameNode and will create performance issues during HDFS reads and writes

The other important reason is to minimize disk seeks. A single HDFS block of 128MB will be written to disk sequentially. Therefore there is a fair chance that the data will be written into contiguous space on disk (consisting of multiple blocks next to each other). Since there is a good chance that the data will be stored in contiguous space the reads will be much faster since it avoids random seeks.

18. How corruption is handled by HDFS?

There are several reasons for data corruption - faults in a storage device, network faults, or buggy software.

 When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace.

 When a client retrieves file contents it verifies that the data it received from each DataNode matches the checksum stored in the associated checksum file.

 If the checksum did not match the client will retrieve that block from another DataNode that has a replica of that block.

 NameNode will be notified of the corrupted block and it will arrange for replicating the block to maintain the replication factor in the cluster.

Use ful for quiz

http://hadoopiq.blogspot.in/

http://hadooptutorial.info/forums/topic/250-hadoop-interview-questions-for-experienced-hadoop-developers/

http://www.bigdataanalyst.in/hadoop-mapreduce-interview-questions/

https://intellipaat.com/interview-question/big-data-hadoop-interview-questions/

Complete Tutorial/:

http://www.tutorialspoint.com/hadoop/hadoop\_mock\_test.htm

http://hadoopiq.blogspot.in/2014/08/100-top-hadoop-interview-questions-and.html

Java:

http://www.javatpoint.com/hadoop-interview-questions

## **Hadoop Admin Interview Questions**

7)

**Hadoop MapReduce Interview Questions**

23) What is a NameNode and what is a DataNode?

24) What is Shuffling in MapReduce?

25) Why would a Hadoop developer develop a Map Reduce by disabling the reduce step?

26) What is the functionality of Task Tracker and Job Tracker in Hadoop? How many instances of a Task Tracker and Job Tracker can be run on a single Hadoop Cluster?

27) How does NameNode tackle DataNode failures?

28) What is InputFormat in Hadoop?

29) What is the purpose of RecordReader in Hadoop?

30) What is InputSplit in MapReduce?

31)In Hadoop, if custom partitioner is not defined then, how is data partitioned before it is sent to the reducer?

32) What is replication factor in Hadoop and what is default replication factor level Hadoop comes with?

33) What is SequenceFile in Hadoop and Explain its importance?

34) If you are the user of a  [MapReduce framework](https://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83), then what are the configuration parameters you need to specify?

35) Explain about the different parameters of the mapper and reducer functions.

36) How can you set random number of mappers and reducers for a Hadoop job?

37) How many Daemon processes run on a Hadoop System?

38) What happens if the number of reducers is 0?

39) What is meant by Map-side and Reduce-side join in Hadoop?

40) How can the NameNode be restarted?

41) Hadoop attains parallelism by isolating the tasks across various nodes; it is possible for some of the slow nodes to rate-limit the rest of the program and slows down the program. What method Hadoop provides to combat this?

42) What is the significance of conf.setMapper class?

43) What are combiners and when are these used in a MapReduce job?

44) How does a DataNode know the location of the NameNode in Hadoop cluster?

45) How can you check whether the NameNode is working or not?

### **Pig Interview Questions**

46) When doing a join in Hadoop, you notice that one reducer is running for a very long time. How will address this problem in Pig?

47) Are there any problems which can only be solved by MapReduce and cannot be solved by PIG? In which kind of scenarios MR jobs will be more useful than PIG?

48) Give an example scenario on the usage of counters.

### **Hive Interview Questions**

49) Explain the difference between ORDER BY and SORT BY in Hive?

50)  Differentiate between HiveQL and SQL.

We would like to know about your experience in  [Hadoop](https://www.dezyre.com/Hadoop-Training-online/19)interviews. Please comment below to let us know if we missed any important question that is regularly asked in these interviews.

1. Why can't we use Java primitive data types in Map Reduce?  
2. Explain how do you decide between Managed & External tables in hive  
3. Can we change the default location of Managed tables  
4. What are the factors that we consider while creating a hive table  
5. What are the compression techniques and how do you decide which one to use  
6. Co group in Pig  
7. How to include partitioned column in data - Hive  
9. What hadoop -put command do exactly  
10. What is the limit on Distributed cache size?  
11. Handling skewed data  
12. What are the Different joins in hive?  
13. Explain about SMB join in Hive

**) What is Hadoop Map Reduce ?**

For processing large data sets in parallel across a hadoop cluster, Hadoop MapReduce framework is used.  Data analysis uses a two-step map and reduce process.

**2) How Hadoop MapReduce works?**

In MapReduce, during the map phase it counts the words in each document, while in the reduce phase it aggregates the data as per the document spanning the entire collection. During the map phase the input data is divided into splits for analysis by map tasks running in parallel across Hadoop framework.

**3) Explain what is shuffling in MapReduce ?**

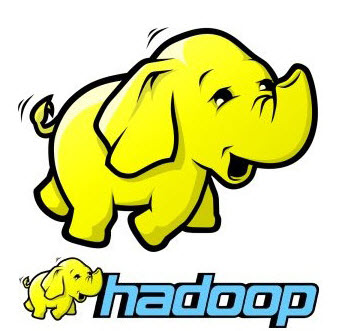
The process by which the system performs the sort and transfers the map outputs to the reducer as inputs is known as the shuffle

**4) Explain what is distributed Cache in MapReduce Framework ?**

Distributed Cache is an important feature provided by map reduce framework. When you want to share some files across all nodes in Hadoop Cluster, DistributedCache  is used.  The files could be an executable jar files or simple properties file.

**5) Explain what is NameNode in Hadoop?**

NameNode in Hadoop is the node, where Hadoop stores all the file location information in HDFS (Hadoop Distributed File System).  In other words, NameNode is the centrepiece of an HDFS file system.  It keeps the record of all the files in the file system, and tracks the file data across the cluster or multiple machines

[](http://career.guru99.com/wp-content/uploads/2014/08/7.jpg)

**6) Explain what is JobTracker in Hadoop? What are the actions followed by Hadoop?**

In Hadoop for submitting and tracking MapReduce jobs,  JobTracker is used. Job tracker run on its own JVM process

Hadoop performs following actions in Hadoop

* Client application submit jobs to the job tracker
* JobTracker communicates to the Namemode to determine data location
* Near the data or with available slots JobTracker locates TaskTracker nodes
* On chosen TaskTracker Nodes, it submits the work
* When a task fails, Job tracker notify and decides what to do then.
* The TaskTracker nodes are monitored by JobTracker

**7) Explain what is heartbeat in HDFS?**

Heartbeat is referred to a signal used between a data node and Name node, and between task tracker and job tracker, if the Name node or job tracker does not respond to the signal, then it is considered there is some issues with data node or task tracker

**8) Explain what combiners is and when you should use a combiner in a MapReduce Job?**

To increase the efficiency of MapReduce Program, Combiners are used.  The amount of data can be reduced with the help of combiner’s that need to be transferred across to the reducers. If the operation performed is commutative and associative you can use your reducer code as a combiner.  The execution of combiner is not guaranteed in Hadoop

**9) What happens when a datanode fails ?**

When a datanode fails

* Jobtracker and namenode detect the failure
* On the failed node all tasks are re-scheduled
* Namenode replicates the users data to another node

**10) Explain what is Speculative Execution?**

In Hadoop during Speculative Execution a certain number of duplicate tasks are launched.  On different slave node, multiple copies of same map or reduce task can be executed using Speculative Execution. In simple words, if a particular drive is taking long time to complete a task, Hadoop will create a duplicate task on another disk.  Disk that finish the task first are retained and disks that do not finish first are killed.

**11) Explain what are the basic parameters of a Mapper?**

The basic parameters of a Mapper are

* LongWritable and Text
* Text and IntWritable

**12) Explain what is the function of MapReducer partitioner?**

The function of MapReducer partitioner is to make sure that all the value of a single key goes to the same reducer, eventually which helps evenly distribution of the map output over the reducers

**13) Explain what is difference between an Input Split and HDFS Block?**

Logical division of data is known as Split while physical division of data is known as HDFS Block

**14) Explain what happens in textinformat ?**

In textinputformat, each line in the text file is a record.  Value is the content of the line while Key is the byte offset of the line. For instance, Key: longWritable, Value: text

**15) Mention what are the main configuration parameters that user need to specify to run Mapreduce Job ?**

The user of Mapreduce framework needs to specify

* Job’s input locations in the distributed file system
* Job’s output location in the distributed file system
* Input format
* Output format
* Class containing the map function
* Class containing the reduce function
* JAR file containing the mapper, reducer and driver classes

**16) Explain what is WebDAV in Hadoop?**

To support editing and updating files WebDAV is a set of extensions to HTTP.  On most [operating system](http://career.guru99.com/category/operating-system-2/) WebDAV shares can be mounted as filesystems , so it is possible to access HDFS as a standard filesystem by exposing HDFS over WebDAV.

**17)  Explain what is sqoop in Hadoop ?**

To transfer the data between Relational [database](http://career.guru99.com/category/database/) management (RDBMS) and Hadoop HDFS a tool is used known as Sqoop. Using Sqoop data can be transferred from RDMS like MySQL or Oracle into HDFS as well as exporting data from HDFS file to RDBMS

**18) Explain how JobTracker schedules a task ?**

The task tracker send out heartbeat messages to Jobtracker usually every few minutes to make sure that JobTracker is active and functioning.  The message also informs JobTracker about the number of available slots, so the JobTracker can stay upto date with where in the cluster work can be delegated

**19) Explain what is Sequencefileinputformat?**

Sequencefileinputformat is used for reading files in sequence. It is a specific compressed binary file format which is optimized for passing data between the output of one MapReduce job to the input of some other MapReduce job.

**20) Explain what does the conf.setMapper Class do ?**

Conf.setMapperclass  sets the mapper class and all the stuff related to map job such as reading data and generating a key-value pair out of the mapper

**21) Explain what is Hadoop?**

It is an open-source software framework for storing data and running applications on clusters of commodity hardware.  It provides enormous processing power and massive storage for any type of data.

**22) Mention what is the difference between an RDBMS and Hadoop?**

|  |  |
| --- | --- |
| **RDBMS** | **Hadoop** |
| RDBMS is relational database management system | Hadoop is node based flat structure |
| It used for OLTP processing whereas Hadoop | It is currently used for analytical and for BIG DATA processing |
| In RDBMS, the database cluster uses the same data files stored in shared storage | In Hadoop, the storage data can be stored independently in each processing node. |
| You need to preprocess data before storing it | you don’t need to preprocess data before storing it |

**23) Mention Hadoop core components?**

Hadoop core components include,

* HDFS
* MapReduce

**24) What is NameNode in Hadoop?**

NameNode in Hadoop is where Hadoop stores all the file location information in HDFS. It is the master node on which job tracker runs and consists of metadata.

**25) Mention what are the data components used by Hadoop?**

Data components used by Hadoop are

* Pig
* Hive

**26) Mention what is the data storage component used by Hadoop?**

The data storage component used by Hadoop is HBase.

**27) Mention what are the most common input formats defined in Hadoop?**

The most common input formats defined in Hadoop are;

* TextInputFormat
* KeyValueInputFormat
* SequenceFileInputFormat

**28) In Hadoop what is InputSplit?**

It splits input files into chunks and assign each split to a mapper for processing.

**29) For a Hadoop job, how will you write a custom partitioner?**

You write a custom partitioner for a Hadoop job, you follow the following path

* Create a new class that extends Partitioner Class
* Override method getPartition
* In the wrapper that runs the MapReduce
* Add the custom partitioner to the job by using method set Partitioner Class or – add the custom partitioner to the job as a config file

**30) For a job in Hadoop, is it possible to change the number of mappers to be created?**

No, it is not possible to change the number of mappers to be created. The number of mappers is determined by the number of input splits.

**31) Explain what is a sequence file in Hadoop?**

To store binary key/value pairs, sequence file is used. Unlike regular compressed file, sequence file support splitting even when the data inside the file is compressed.

**32) When Namenode is down what happens to job tracker?**

Namenode is the single point of failure in HDFS so when Namenode is down your cluster will set off.

**33) Explain how indexing in HDFS is done?**

Hadoop has a unique way of indexing. Once the data is stored as per the block size, the HDFS will keep on storing the last part of the data which say where the next part of the data will be.

**34) Explain is it possible to search for files using wildcards?**

Yes, it is possible to search for files using wildcards.

**35) List out Hadoop’s three configuration files?**

The three configuration files are

* core-site.xml
* mapred-site.xml
* hdfs-site.xml

**36) Explain how can you check whether Namenode is working beside using the jps command?**

Beside using the jps command, to check whether Namenode are working you can also use

/etc/init.d/hadoop-0.20-namenode status.

**37) Explain what is “map” and what is “reducer” in Hadoop?**

In Hadoop, a map is a phase in HDFS query solving.  A map reads data from an input location, and outputs a key value pair according to the input type.

In Hadoop, a reducer collects the output generated by the mapper, processes it, and creates a final output of its own.

**38) In Hadoop, which file controls reporting in Hadoop?**

In Hadoop, the hadoop-metrics.properties file controls reporting.

**39) For using Hadoop list the network requirements?**

For using Hadoop the list of network requirements are:

* Password-less SSH connection
* Secure Shell (SSH) for launching [server](http://career.guru99.com/category/server/" \o "server) processes

**40) Mention what is rack awareness?**

Rack awareness is the way in which the namenode determines on how to place blocks based on the rack definitions.

**41) Explain what is a Task Tracker in Hadoop?**

A Task Tracker in Hadoop is a slave node daemon in the cluster that accepts tasks from a JobTracker. It also sends out the heartbeat messages to the JobTracker, every few minutes, to confirm that the JobTracker is still alive.

**42) Mention what daemons run on a master node and slave nodes?**

* Daemons run on Master node is “NameNode”
* Daemons run on each Slave nodes are “Task Tracker” and “Data”

**43) Explain how can you debug Hadoop code?**

The popular methods for debugging Hadoop code are:

* By using web interface provided by Hadoop framework
* By using Counters

**44) Explain what is storage and compute nodes?**

* The storage node is the machine or computer where your file system resides to store the processing data
* The compute node is the computer or machine where your actual business logic will be executed.

**45) Mention what is the use of Context Object?**

The Context Object enables the mapper to interact with the rest of the Hadoop

system. It includes configuration data for the job, as well as interfaces which allow it to emit output.

**46) Mention what is the next step after Mapper or MapTask?**

The next step after Mapper or MapTask is that the output of the Mapper are sorted, and partitions will be created for the output.

**47) Mention what is the number of default partitioner in Hadoop?**

In Hadoop, the default partitioner is a “Hash” Partitioner.

**48) Explain what is the purpose of RecordReader in Hadoop?**

In Hadoop, the RecordReader loads the data from its source and converts it into (key, value) pairs suitable for reading by the Mapper.

**49) Explain how is data partitioned before it is sent to the reducer if no custom partitioner is defined in Hadoop?**

If no custom partitioner is defined in Hadoop, then a default partitioner computes a hash value for the key and assigns the partition based on the result.

**50) Explain what happens when Hadoop spawned 50 tasks for a job and one of the task failed?**

It will restart the task again on some other TaskTracker if the task fails more than the defined limit.

**51) Mention what is the best way to copy files between HDFS clusters?**

The best way to copy files between HDFS clusters is by using multiple nodes and the distcp command, so the workload is shared.

**52) Mention what is the difference between HDFS and NAS?**

HDFS data blocks are distributed across local drives of all machines in a cluster while NAS data is stored on dedicated hardware.

**53) Mention how Hadoop is different from other data processing tools?**

In Hadoop, you can increase or decrease the number of mappers without worrying about the volume of data to be processed.

**54) Mention what job does the conf class do?**

Job conf class separate different jobs running on the same cluster.  It does the job level settings such as declaring a job in a real environment.

**55) Mention what is the Hadoop MapReduce APIs contract for a key and value class?**

For a key and value class, there are two Hadoop MapReduce APIs contract

* The value must be defining the org.apache.hadoop.io.Writable interface
* The key must be defining the org.apache.hadoop.io.WritableComparable interface

**56) Mention what are the three modes in which Hadoop can be run?**

The three modes in which Hadoop can be run are

* Pseudo distributed mode
* Standalone (local) mode
* Fully distributed mode

**57) Mention what does the text input format do?**

The text input format will create a line object that is an hexadecimal number.  The value is considered as a whole line text while the key is considered as a line object. The mapper will receive the value as ‘text’ parameter while key as ‘longwriteable’ parameter.

**58) Mention how many InputSplits is made by a Hadoop Framework?**

Hadoop will make 5 splits

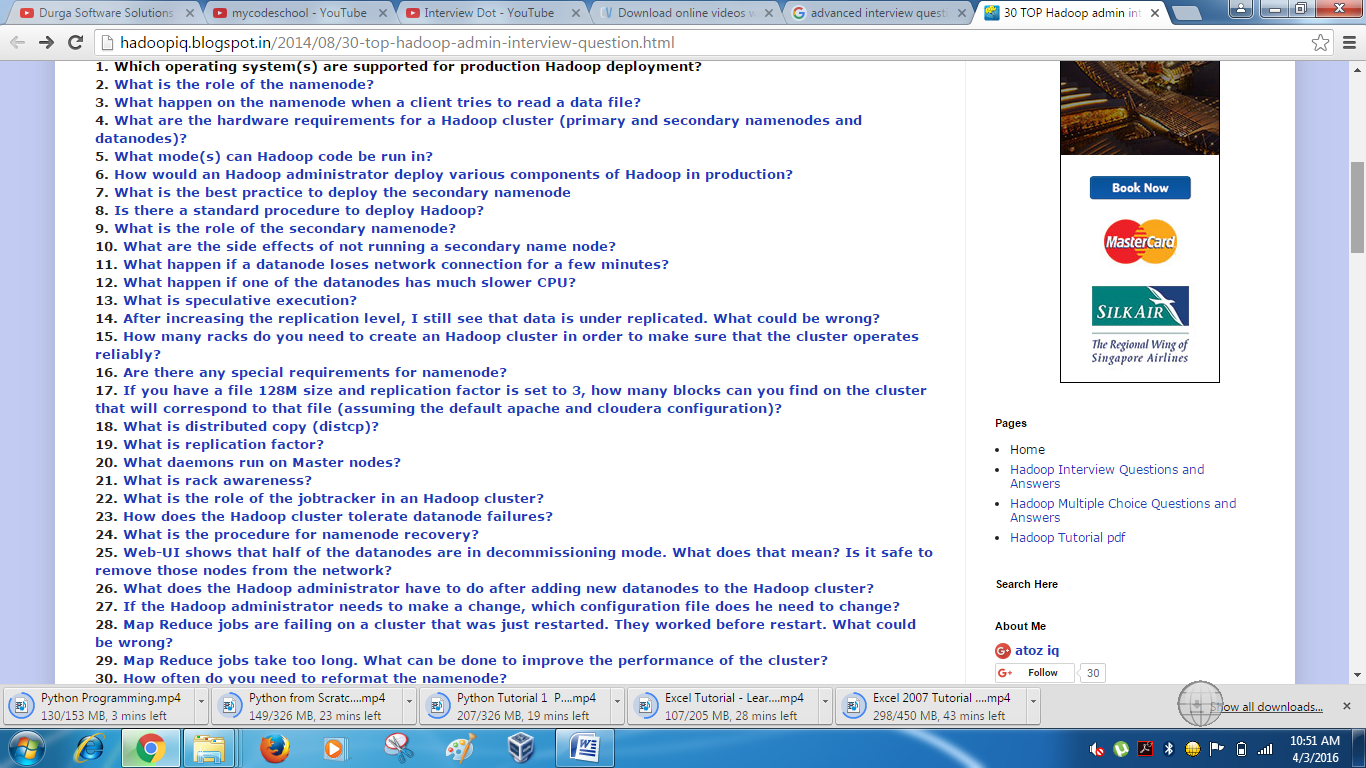
* 1 split for 64K files
* 2 split for 65mb files
* 2 splits for 127mb files

**59) Mention what is distributed cache in Hadoop?**

Distributed cache in Hadoop is a facility provided by MapReduce framework.  At the time of execution of the job, it is used to cache file.  The Framework copies the necessary files to the slave node before the execution of any task at that node.

**60) Explain how does Hadoop Classpath plays a vital role in stopping or starting in Hadoop daemons?**

Classpath will consist of a list of directories containing jar files to stop or start daemons



1. Define Sequence file in Hadoop?
2. What is meant by Replication factor?
3. List out the key components of HBase and tell when you should use it?
4. How the file system check is done in HDFS?
5. Define speculative execution?
6. Sqoop in Hadoop, Define?
7. Define webDav in Hadoop?
8. Define the function of Mapper?
9. Define Distributed Caxhe in Hadoop?
10. Differentiate HDFA and NAS?
11. **Q1. Name the most common Input Formats defined in Hadoop? Which one is default?**
12. The two most common Input Formats defined in Hadoop are:
13. – TextInputFormat
14. - KeyValueInputF6ormat
15. - SequenceFileInputFormat
16. TextInputFormat is the Hadoop default.
17. **Q2. What is the difference between TextInputFormat and KeyValueInputFormat class?**
18. **TextInputFormat:** It reads lines of text files and provides the offset of the line as key to the Mapper and actual line as Value to the mapper.
19. **KeyValueInputFormat:** Reads text file and parses lines into key, Val pairs. Everything up to the first tab character is sent as key to the Mapper and the remainder of the line is sent as value to the mapper.
20. **Q3. What is InputSplit in Hadoop?**
21. When a Hadoop job is run, it splits input files into chunks and assign each split to a mapper to process. This is called InputSplit.
22. **Q4. How is the splitting of file invoked in Hadoop framework?**
23. It is invoked by the Hadoop framework by running getInputSplit()method of the Input format class (like FileInputFormat) defined by the user.
24. **Q5. Consider case scenario: In M/R system,** **- HDFS block size is 64 MB**
25. **- Input format is FileInputFormat**
26. **– We have 3 files of size 64K, 65Mb and 127Mb**
27. **How many input splits will be made by Hadoop framework?**
28. Hadoop will make 5 splits as follows:
29. - 1 split for 64K files
30. - 2 splits for 65MB files
31. - 2 splits for 127MB files
32. **Q6. What is the purpose of RecordReader in Hadoop?**
33. The InputSplit has defined a slice of work, but does not describe how to access it. The RecordReader class actually loads the data from its source and converts it into (key, value) pairs suitable for reading by the Mapper. The RecordReader instance is defined by the Input Format.
34. **Q7. After the Map phase finishes, the Hadoop framework does “Partitioning, Shuffle and sort”. Explain what happens in this phase?**
35. **Partitioning:** It is the process of determining which reducer instance will receive which intermediate keys and values. Each mapper must determine for all of its output (key, value) pairs which reducer will receive them. It is necessary that for any key, regardless of which mapper instance generated it, the destination partition is the same.
36. **Shuffle:** After the first map tasks have completed, the nodes may still be performing several more map tasks each. But they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers. This process of moving map outputs to the reducers is known as shuffling.
37. **Sort:** Each reduce task is responsible for reducing the values associated with several intermediate keys. The set of intermediate keys on a single node is automatically sorted by Hadoop before they are presented to the Reducer.
38. **Q8. If no custom partitioner is defined in Hadoop then how is data partitioned before it is sent to the reducer?**
39. The default partitioner computes a hash value for the key and assigns the partition based on this result.
40. **Q9. What is a Combiner?**
41. The Combiner is a ‘mini-reduce’ process which operates only on data generated by a mapper. The Combiner will receive as input all data emitted by the Mapper instances o4n a given node. The output from the Combiner is then sent to the Reducers, instead of the output from the Mappers.
42. **Q10. What is JobTracker?**
43. JobTracker is the service within Hadoop that runs MapReduce jobs on the cluster.
44. **Q11. What are some typical functions of Job Tracker?**
45. The following are some typical tasks of JobTracker:-
46. - Accepts jobs from clients
47. - It talks to the NameNode to determine the location of the data.
48. - It locates TaskTracker nodes with available slots at or near the data.
49. - It submits the work to the chosen TaskTracker nodes and monitors progress of each task by receiving heartbeat signals from Task tracker.
50. **Q12. What is TaskTracker?**
51. TaskTracker is a node in the cluster that accepts tasks like MapReduce and Shuffle operations – from a JobTracker.
52. **Q13. What is the relationship between Jobs and Tasks in Hadoop?**
53. One job is broken down into one or many tasks in Hadoop.
54. **Q14. Suppose Hadoop spawned 100 tasks for a job and one of the task failed. What will Hadoop do?**
55. It will restart the task again on some other TaskTracker and only if the task fails more than four (default setting and can be changed) times will it kill the job.
56. **Q15. Hadoop achieves parallelism by dividing the tasks across many nodes, it is possible for a few slow nodes to rate-limit the rest of the program and slow down the program. What mechanism Hadoop provides to combat this?**
57. Speculative Execution.
58. **Q16. How does speculative execution work in Hadoop?**
59. JobTracker makes different TaskTrackers pr2ocess same input. When tasks complete, they announce this fact to the JobTracker. Whichever copy of a task finishes first becomes the definitive copy. If other copies were executing speculatively, Hadoop tells the TaskTrackers to abandon the tasks and discard their outputs. The Reducers then receive their inputs from whichever Mapper completed successfully, first.
60. **Q17. Using command line in Linux, how will you**
61. - **See all jobs running in the Hadoop cluster**
62. **- Kill a job?**
63. Hadoop job – list
64. Hadoop job – kill jobID
65. **Q18. What is Hadoop Streaming?**
66. Streaming is a generic API that allows programs written in virtually any language to be used as Hadoop Mapper and Reducer implementations.
67. **Q19. What is the characteristic of streaming API that makes it flexible run MapReduce jobs in languages like Perl, Ruby, Awk etc.?**
68. Hadoop Streaming allows to use arbitrary programs for the Mapper and Reducer phases of a MapReduce job by having both Mappers and Reducers receive their input on stdin and emit output (key, value) pairs on stdout.
69. **Q20. What is Distributed Cache in Hadoop?**
70. Distributed Cache is a facility provided by the MapReduce framework to cache files (text, archives, jars and so on) needed by applications during execution of the job. The framework will copy the necessary files to the slave node before any tasks for the job are executed on that node.
71. Q21. What is the benefit of Distributed cache? Why can we just have the file in HDFS and have1 **the application read it?**
72. This is because distributed cache is much faster. It copies the file to all trackers at the start of the job. Now if the task tracker runs 10 or 100 Mappers or Reducer, it will use the same copy of distributed cache. On the other hand, if you put code in file to read it from HDFS in the MR Job then every Mapper will try to access it from HDFS hence if a TaskTracker run 100 map jobs then it will try to read this file 100 times from HDFS. Also HDFS is not very efficient when used like this.
73. **Q.22 What mechanism does Hadoop framework provide to synchronise changes made in Distribution Cache during runtime of the application?**
74. This is a tricky question. There is no such mechanism. Distributed Cache by design is read only during the time of Job execution.
75. **Q23. Have you ever used Counters in Hadoop. Give us an example scenario?**
76. Anybody who claims to have worked on a Hadoop project is expected to use counters.
77. **Q24. Is it possible to provide multiple input to Hadoop? If yes then how can you give multiple directories as input to the Hadoop job?**
78. Yes, the input format class provides methods to add multiple directories a1s input to a Hadoop job.
79. **Q25. Is it possible to have Hadoop job output in multiple directories? If yes, how?**
80. Yes, by using Multiple Outputs class.
81. **Q26. What will a Hadoop job do if you try to run it with an output directory that is already present? Will it**
82. **- Overwrite it**
83. **- Warn you and continue**
84. **- Throw an exception and exit**
85. The Hadoop job will throw an exception and exit.
86. **Q27. How can you set an arbitrary number of mappers to be created for a job in Hadoop?**
87. You cannot set it.
88. **Q28. How can you set an arbitrary number of Reducers to be created for a job in Hadoop?**
89. You can either do it programmatically by using method setNumReduceTasks in the Jobconf Class or set it up as a configuration setting.
90. **Q29. How will you write a custom partitioner for a Hadoop job?**
91. To have Hadoop use a custom partitioner you will have to do minimum the following three:
92. - Create a new class that extends Partitioner Class
93. - Override method getPartition
94. - In the wrapper that runs the Mapreduce, either
95. - Add the custom partitioner to the job programmatically using method set Partitioner Class or – add the custom partitioner to the job as a config file (if your wrapper reads from config file or oozie)
96. **Q30. How did you debug your Hadoop code?**
97. There can be several ways of doing this but most common ways are:-
98. - By using counters.
99. - The web interface provided by Hadoop framework.
100. **Q31. Did you ever built a production process in Hadoop? If yes, what was the process when your Hadoop job fails due to any reason?**
101. It is an open-ended question but most candidates if they have written a production job, should talk about some type of alert mechanism like email is sent or there monitoring system sends an alert. Since [Hadoop works on unstructured data,](http://wiziqlmp.wpengine.com/decoding-big-data-analytics-hadoop/) it is very important to have a good alerting system for errors since unexpected data can very easily break the job.
102. [. jobtracker](http://www.wiziq.com/blog/tag/jobtracker/)

#### Hadoop is a complex framework. Some interview questions can be really simple like “How do you debug a performance issue or a long running job?”  but difficult to answer on the spot if you are not prepared. Below are some sample and common questions you would hear in an interview. The Hadoop Developer Interview guide has more complex and difficult to answer scenario based questions than what is shown below.

### How do you debug a performance issue or a long running job?

This is an open ended question and the interviewer is trying to see the level of hands-on experience you have in solving production issues. Use your day to day work experience to answer this question. Here are some of the scenarios and responses to help you construct your answer. On a very high level you will follow the below steps.

Understand the symptom¬  
 Analyze the situation¬  
 Identify the problem areas¬  
 Propose solution¬

**Scenario 1** – Job with 100 mappers and 1 reducer takes a long time for the reducer to start after all the mappers are complete. One of the reasons could be that reduce is spending a lot of time copying the map outputs. So in this case we can try couple of things.

1. If possible add a combiner to reduce the amount of output from the mapper to be sent to the reducer  
2. Enable map output compression – this will further reduce the size of the outputs to be transferred to the reducer.

**Scenario 2** – A particular task is using a lot of memory which is causing the slowness or failure, I will look for ways to reduce the memory usage.

1. Make sure the joins are made in an optimal way with memory usage in mind. For e.g. in Pig joins, the LEFT hand side tables are sent to the reducer first and held in memory and the RIGHT most table in streamed to the reducer. So make sure the RIGHT most table is largest of the datasets in the join.  
2. We can also increase the memory requirements needed by the map and reduce tasks by setting – mapreduce.map.memory.mb and mapreduce.reduce.memory.mb

**Scenario 3** – Understanding the data helps a lot in optimizing the way we use the datasets in PIG and HIVE scripts.

1. If you have smaller tables in join, they can be sent to distributed cache and loaded in memory on the Map side and the entire join can be done on the Map side thereby avoiding the shuffle and reduce phase altogether. This will tremendously improve performance. Look up USING REPLICATED in Pig and MAPJOIN or hive.auto.convert.join in Hive  
2. If the data is already sorted you can use USING MERGE which will do a Map Only join  
3. If the data is bucketted in hive, you may use hive.optimize.bucketmapjoin or  
hive.optimize.bucketmapjoin.sortedmerge depending on the characteristics of the data

**Scenario 4** – The Shuffle process is the heart of a MapReduce program and it can be tweaked for performance improvement.

1. If you see lots of records are being spilled to the disk (check for Spilled Records in the counters in your MapReduce output) you can increase the memory available for Map to perform the Shuffle by increasing the value in io.sort.mb. This will reduce the amount of Map Outputs written to the disk so the sorting of the keys can be performed in memory.  
2. On the reduce side the merge operation (merging the output from several mappers) can be done in disk by setting the mapred.inmem.merge.threshold to 0

### Assume you have Research, Marketing and Finance teams funding 60%, 30% and 10% respectively of your Hadoop Cluster. How will you assign only 60% of cluster resources to Research, 30% to Marketing and 10% to Finance during peak load?

Capacity scheduler in Hadoop is designed to support this use case. Capacity scheduler supports hierarchical queues and capacity can be defined for each queue.

For this use case, you would have to define 3 queues under the root queue and give appropriate capacity in % for each queue.

Illustration

Below properties will be defined in capacity-scheduler.xml

<property>  
<name>yarn.scheduler.capacity.root.queues</name>  
<value>research,marketing,finance</value>  
</property>

<property>  
<name>yarn.scheduler.capacity.research.capacity</name>  
<value>60</value>  
</property>

<property>  
<name>yarn.scheduler.capacity.research.capacity</name>  
<value>30</value>  
</property>

<property>  
<name>yarn.scheduler.capacity.research.capacity</name>  
<value>10</value>  
</property>

### How do you benchmark your Hadoop cluster with tools that come with Hadoop?

**TestDFSIO¬**

TestDFSIO gives you an understanding of the I/O performance of your cluster. It is a read and write test for HDFS and helpful in identifying performance bottlenecks in your network, hardware and set up of your NameNode and DataNodes.

**NNBench¬**

NNBench simulate requests for creating, reading, renaming and deleting files on HDFS and is useful for load testing NameNode hardware configuration

**MRBench¬**

MRBench is a test for the MapReduce layer. It loops a small MapReduce job for a specific number of times and checks the responsiveness and efficiency of the cluster.

Illustration

TestDFSIO write test with 100 files and file size of 100 MB each.

$ hadoop jar /dirlocation/hadoop-test.jar TestDFSIO -write -nrFiles 100 -fileSize 100

TestDFSIO read test with 100 files and file size of 100 MB each.

$ hadoop jar /dirlocation/hadoop-test.jar TestDFSIO -read -nrFiles 100 -fileSize 100

MRBench test to run a lob of 50 small test jobs

$ hadoop jar /dirlocation/hadoop-test.jar mrbench -numRuns 50

NNBench test that creates 1000 files using 12 maps and 6 reducers.

$ hadoop jar /dirlocation/hadoop-test.jar nnbench -operation create\_write \  
-maps 12 -reduces 6 -blockSize 1 -bytesToWrite 0 -numberOfFiles 1000 \  
-replicationFactorPerFile 3

### Assume you are doing a join and you notice that all but one reducer is running for a long time how do you address the problem in Pig?

Pig collects all of the records for a given key together on a single reducer. In many data sets, there are a few keys that have three or more orders of magnitude more records than other keys. This results in one or two reducers that will take much longer than the rest. To deal with this, Pig provides skew join.

In the first MapReduce job pig scans the second input and identifies keys that have so many records.¬  
 In the second MapReduce job, it does the actual join.¬  
 For all except the records with the key(s) identified from the first job, pig would do a standard join.¬  
 For the records with keys identified by the second job, bases on how many records were seen for a given key, those records will be split across appropriate number of reducers.¬  
 The other input to the join that is not split, only the keys in question are then then split and then replicated to each reducer that contains that key¬

Illustration

jnd = join cinfo by city, users by city using ‘skewed’;

### What is the difference between SORT BY and ORDER BY in Hive?

ORDER BY performs a total ordering of the query result set. This means that all the data is passed through a single reducer, which may take an unacceptably long time to execute for larger data sets.

SORT BY orders the data only within each reducer, thereby performing a local ordering, where each reducer’s output will be sorted. You will not achieve a total ordering on the dataset. Better performance is traded for total ordering.

Assume you have a sales table in a company and it has sales entries from salesman around the globe. How do you rank each salesperson by country based on their sales volume in Hive?

Hive support several analytic functions and one of the functions is RANK() and it is designed to do this operation.

Lookup details on other window and analytic functions – https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics

Illustration

Hive>SELECT  
rep\_name, rep\_country, sales\_volume,  
rank() over (PARTITION BY rep\_country ORDER BY sales\_volume DESC) as rank  
FROM  
salesrep;

### What is Speculative execution?

A job running on a Hadoop cluster could be divided in to many tasks. In a big cluster some of these tasks could be running slow for various reasons, hardware degradation or software miconfiguration etc. Hadoop initiates a replica of a task when it sees a tasks which is running for sometime and failed to make any progress, on average, as the other tasks from the job. This replica or duplicate exeuction of task is referred to as Speculative Execution.

When a task completes successfully all the duplicate tasks that are running will be killed. So if the original task completes before the speculative task, then the speculative task is killed; on the other hand, if the speculative task finishes first, then the original is killed.

### What is the benefit of using counters in Hadoop?

Counters are a useful for gathering statistics about the job. Assume you have a 100 node cluster and a job with 100 mappers is running in the cluster on 100 different nodes. Lets say you would like to know each time you see a invalid record in your Map phase. You could add a log message in your Mapper so that each time you see an invalid line you can make an entry in the log. But consolidating all the log messages from 100 different nodes will be time consuming. You can use a counter instead and increment the value of the counter every time you see an invalid record. The nice thing about using counters is that is gives you a consolidate value for the whole job rather than showing 100 separate outputs.

### What is the difference between an InputSplit and a Block?

Block is a physical division of data and does not take in to account the logical boundary of records. Meaning you could have a record that started in one block and ends in another block. Where as InputSplit considers the logical boundaries of records as well.

### Can you change the number of mappers to be created for a job in Hadoop?

No. The number of mappers is determined by the no of input splits.

### How do you do a file system check in HDFS?

FSCK command is used to do a file system check in HDFS. It is a very useful command to check the health of the file, block names and block locations.

Illustration

hdfs fsck /dir/hadoop-test -files -blocks -locations

### What are the parameters of mappers and reducers function?

Map and Reduce method signature tells you a lot about the type of input and ouput your Job will deal with. Assuming you are using TextInputFormat, Map function’s parameters could look like –

LongWritable (Input Key)  
Text (Input Value)  
Text (Intermediate Key Output)  
IntWritable (Intermediate Output)

The four parameters for reduce function could be –

Text (Intermediate Key Output)  
IntWritable (Intermediate Value Output)  
Text (Final Key Output)  
IntWritable (Final Value Output)

### How do you overwrite replication factor?

There are few ways to do this. Look at the below illustration.

Illustration

hadoop fs -setrep -w 5 -R hadoop-test

hadoop fs -Ddfs.replication=5 -cp hadoop-test/test.csv hadoop-test/test\_with\_rep5.csv

### What are the functions of InputFormat?

Validate input data is present and check input configuration  
Create InputSplits from blocks  
Create RecordReader implementation to create key/value pairs from the raw InputSplit. These pairs will be sent one by one to their mapper.

### What is a Record Reader?

A RecordReader uses the data within the boundaries created by the input split to generate key/value pairs. Each of the generated Key/value pair will be sent one by one to their mapper.

### What is a sequence file in Hadoop?

Sequence file is used to store binary key/value pairs. Sequence files support splitting even when the data inside the file is compressed which is not possible with a regular compressed file. You can either choose to perform a record level compression in which the value in the key/value pair will be compressed. Or you can also choose to choose at the block level where multiple records will be compressed together.

**1) Hadoop Basic Interview Questions**

**1.** What is Big Data?

Any data that cannot be stored into traditional RDBMS is termed as Big Data. As we know most of the data that we use today has been generated in the past 20 years. And this data is mostly unstructured or semi structured in nature. More than the volume of the data – it is the nature of the data that defines whether it is considered as Big Data or not.

**2.** What do the four V’s of Big Data denote?

IBM has a nice, simple explanation for the four critical features of big data:   
a) Volume –Scale of data   
b) Velocity –Different forms of data   
c) Variety –Analysis of streaming data   
d) Veracity –Uncertainty of data

IBM has a nice, simple explanation for the four critical features of big data:   
a) Volume –Scale of data   
b) Velocity –Different forms of data   
c) Variety –Analysis of streaming data   
d) Veracity –Uncertainty of data

For more the Basic questions and answers [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#basicAnswers)

**2) Hadoop HDFS Interview Questions**

**1.** What is a block and block scanner in HDFS?

Block - The minimum amount of data that can be read or written is generally referred to as a “block” in HDFS. The default size of a block in HDFS is 64MB.

Block Scanner - Block Scanner tracks the list of blocks present on a DataNode and verifies them to find any kind of checksum errors. Block Scanners use a throttling mechanism to reserve disk bandwidth on the datanode.

**2.** Explain the difference between NameNode, Backup Node and Checkpoint NameNode.

**NameNode**: NameNode is at the heart of the HDFS file system which manages the metadata i.e. the data of the files is not stored on the NameNode but rather it has the directory tree of all the files present in the HDFS file system on a hadoop cluster. NameNode uses two files for the namespace-

fsimage file- It keeps track of the latest checkpoint of the namespace.

edits file-It is a log of changes that have been made to the namespace since checkpoint.

**Checkpoint Node-**

Checkpoint Node keeps track of the latest checkpoint in a directory that has same structure as that of NameNode’s directory. Checkpoint node creates checkpoints for the namespace at regular intervals by downloading the edits and fsimage file from the NameNode and merging it locally. The new image is then again updated back to the active NameNode.

**BackupNode:**

Backup Node also provides check pointing functionality like that of the checkpoint node but it also maintains its up-to-date in-memory copy of the file system namespace that is in sync with the active NameNode.

For more the Hadoop HDFS Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#hdfs)

**3) MapReduce Interview Questions**

**1.** Explain the usage of Context Object.

Context Object is used to help the mapper interact with other Hadoop systems. Context Object can be used for updating counters, to report the progress and to provide any application level status updates. ContextObject has the configuration details for the job and also interfaces, that helps it to generating the output.

**2.** What are the core methods of a Reducer?

The 3 core methods of a reducer are –

1)setup () – This method of the reducer is used for configuring various parameters like the input data size, distributed cache, heap size, etc.

Function Definition- public void setup (context)

2)reduce () it is heart of the reducer which is called once per key with the associated reduce task.

Function Definition -public void reduce (Key,Value,context)

3)cleanup () - This method is called only once at the end of reduce task for clearing all the temporary files.

Function Definition -public void cleanup (context)

For more the MapReduce Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#mapReduce)

**4) Hadoop HBase Interview Questions**

**1.** When should you use HBase and what are the key components of HBase?

HBase should be used when the big data application has –

1)A variable schema

2)When data is stored in the form of collections

3)If the application demands key based access to data while retrieving.

Key components of HBase are –

Region- This component contains memory data store and Hfile.

Region Server-This monitors the Region.

HBase Master-It is responsible for monitoring the region server.

Zookeeper- It takes care of the coordination between the HBase Master component and the client.

Catalog Tables-The two important catalog tables are ROOT and META.ROOT table tracks where the META table is and META table stores all the regions in the system.

For more the Hadoop HBase Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#hbase)

**5) Hadoop Sqoop Interview Questions**

**1.** Explain about some important Sqoop commands other than import and export.

**Create Job (--create)**

Here we are creating a job with the name my job, which can import the table data from RDBMS table to HDFS. The following command is used to create a job that is importing data from the employee table in the db database to the HDFS file.

$ Sqoop job --create myjob \

--import \

--connect jdbc:mysql://localhost/db \

--username root \

--table employee --m 1

**Verify Job (--list)**

‘--list’ argument is used to verify the saved jobs. The following command is used to verify the list of saved Sqoop jobs.

$ Sqoop job --list

**Inspect Job (--show)**

‘--show’ argument is used to inspect or verify particular jobs and their details. The following command and sample output is used to verify a job called myjob.

$ Sqoop job --show myjob

**Execute Job (--exec)**

‘--exec’ option is used to execute a saved job. The following command is used to execute a saved job called myjob.

$ Sqoop job --exec myjob

For moreHadoop Sqoop Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#sqoop)

**6) Hadoop Flume Interview Questions**

**1.** Explain about the core components of Flume.

The core components of Flume are –

Event- The single log entry or unit of data that is transported.

Source- This is the component through which data enters Flume workflows.

Sink-It is responsible for transporting data to the desired destination.

Channel- it is the duct between the Sink and Source.

Agent- Any JVM that runs Flume.

Client- The component that transmits event to the source that operates with the agent.

For more Hadoop Flume Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#flume)

**7) Hadoop Zookeeper Interview Questions**

**1.** Can Apache Kafka be used without Zookeeper?

It is not possible to use Apache Kafka without Zookeeper because if the Zookeeper is down Kafka cannot serve client request.

**2.** Name a few companies that use Zookeeper.

Yahoo, Solr, Helprace, Neo4j, Rackspace

For more Hadoop Zookeeper Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#zookeeper)

**8) Pig Interview Questions**

**1.** What do you mean by a bag in Pig?

Collection of tuples is referred as a bag in Apache Pig

**2.** Does Pig support multi-line commands?

Yes

For more Pig Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#pig)

**9) Hive Interview Questions**

**1.** What is a Hive Metastore?

Hive Metastore is a central repository that stores metadata in external database.

**2.** Are multiline comments supported in Hive?

No

For more Hive Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#hive)

**10)** **Hadoop YARN Interview Questions**

**1.** What are the stable versions of Hadoop?

Release 2.7.1 (stable)

Release 2.4.1

Release 1.2.1 (stable)

**2.** What is Apache Hadoop YARN?

YARN is a powerful and efficient feature rolled out as a part of Hadoop 2.0.YARN is a large scale distributed system for running big data applications.

For more Hadoop YARN Interview Questions [click here](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#yarn)

Read the complete article at ​[**Top 100 Hadoop Interview Questions and Answers**](http://www.dezyre.com/article/-top-100-hadoop-interview-questions-and-answers-2015/159#basicAnswers)

You might also be interested in

[Difference between pig and hive.](http://www.dezyre.com/article/difference-between-pig-and-hive-the-two-key-components-of-hadoop-ecosystem/79)

[Hadoop MapReduce vs. Apache Spark.](http://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83)

### 1) What is Hadoop?

Hadoop is a distributed computing platform. It is written in Java. It consist of the features like Google File System and MapReduce.

### 2) What platform and Java version is required to run Hadoop?

Java 1.6.x or higher version are good for Hadoop, preferably from Sun. Linux and Windows are the supported operating system for Hadoop, but BSD, Mac OS/X and Solaris are more famous to work.

### 3) What kind of Hardware is best for Hadoop?

Hadoop can run on a dual processor/ dual core machines with 4-8 GB RAM using ECC memory. It depends on the workflow needs.

### 4) What are the most common input formats defined in Hadoop?

These are the most common input formats defined in Hadoop:

1. TextInputFormat
2. KeyValueInputFormat
3. SequenceFileInputFormat

TextInputFormat is a by default input format.

### 5) What is InputSplit in Hadoop? Explain.

When a hadoop job runs, it splits input files into chunks and assign each split to a mapper for processing. It is called InputSplit.

### 6) How many InputSplits is made by a Hadoop Framework?

Hadoop will make 5 splits as following:

* One split for 64K files
* Two splits for 65MB files, and
* Two splits for 127MB files

### 7) What is the use of RecordReader in Hadoop?

InputSplit is assigned with a work but doesn't know how to access it. The record holder class is totally responsible for loading the data from its source and convert it into keys pair suitable for reading by the Mapper. The RecordReader's instance can be defined by the Input Format.

### 8) What is JobTracer in Hadoop?

is a service within Hadoop which runs MapReduce jobs on the cluster.

### 9) What are the functionalities of JobTracer?

These are the main tasks of JobTracer:

* To accept jobs from client.
* To communicate with the NameNode to determine the location of the data.
* To locate TaskTracker Nodes with available slots.
* To submit the work to the chosen TaskTracker node and monitors progress of each tasks.

### 10) Define TaskTracker.

TaskTracker is a node in the cluster that accepts tasks like MapReduce and Shuffle operations from a JobTracker.

### 11) What is Map/Reduce job in Hadoop?

Map/Reduce is programming paradigm which is used to allow massive scalability across the thousands of server.

Actually MapReduce refers two different and distinct tasks that Hadoop performs. In the first step maps jobs which takes the set of data and converts it into another set of data and in the second step, Reduce job. It takes the output from the map as input and compress those data tuples into smaller set of tuples.

### 12) What is Hadoop Streaming?

Hadoop streaming is a utility which allows you to create and run map/reduce job. It is a generic API that allows programs written in any languages to be used as Hadoop mapper.

### 13) What is a combiner in Hadoop?

A Combiner is a mini-reduce process which operates only on data generated by a Mapper. When Mapper emits the data, combiner receives it as input and sends the output to reducer.

### 14) Is it necessary to know java to learn Hadoop?

If you have a background in any programming language like C, C++, PHP, Python, Java etc. It may be really helpful, but if you are nil in java, it is necessary to learn Java and also get the basic knowledge of SQL.

### 15) How to debug Hadoop code?

There are many ways to debug Hadoop codes but the most popular methods are:

* By using Counters.
* By web interface provided by Hadoop framework.

### 16) Is it possible to provide multiple inputs to Hadoop? If yes, explain.

Yes, It is possible. The input format class provides methods to insert multiple directories as input to a Hadoop job.

### 17) What is the relation between job and task in Hadoop?

In Hadoop, A job is divided into multiple small parts known as task.

### 18) What is distributed cache in Hadoop?

Distributed cache is a facility provided by MapReduce Framework. It is provided to cache files (text, archives etc.) at the time of execution of the job. The Framework copies the necessary files to the slave node before the execution of any task at that node.

### 19) What commands are used to see all jobs running in the Hadoop cluster and kill a job in LINUX?

Hadoop job - list

Hadoop job - kill jobID

### 20) What is the functionality of JobTracker in Hadoop? How many instances of a JobTracker run on Hadoop cluster?

JobTracker is a giant service which is used to submit and track MapReduce jobs in Hadoop. Only one JobTracker process runs on any Hadoop cluster. JobTracker runs it within its own JVM process.

Functionalities of JobTracker in Hadoop:

* When client application submits jobs to the JobTracker, the JobTracker talks to the NameNode to find the location of the data.
* It locates TaskTracker nodes with available slots for data.
* It assigns the work to the chosen TaskTracker nodes.
* The TaskTracker nodes are responsible to notify the JobTracker when a task fails and then JobTracker decides what to do then. It may resubmit the task on another node or it may mark that task to avoid.

### 21) How JobTracker assign tasks to the TaskTracker?

The TaskTracker periodically sends heartbeat messages to the JobTracker to assure that it is alive. This messages also inform the JobTracker about the number of available slots. This return message updates JobTracker to know about where to schedule task.

### 22) Is it necessary to write jobs for Hadoop in Java language?

No, There are many ways to deal with non-java codes. HadoopStreaming allows any shell command to be used as a map or reduce function.