IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 3, Ver. III (May - June 2017), PP 33-37
www.iosrjournals.org

Anatomy of Hadoop Mapreduce Execution

Parvathy Gopakumar[1], Neethu Maria John[2]
[1]*(Computer Science, Mangalam College of Engineering,India)*
[2]*(Computer Science, Mangalam College of Engineering,India))*

Abstract : *Storing and monitoring Big data in widely distributed environments for 24/7 is a huge task for global service organizations. These datasets require high processing power which can't be offered by traditional databases as they are stored in an unstructured format. Apache Hadoop is open source software for reliable, scalable and distributed computing. This framework is inspired by Google's MapReduce structure in which application is broken down into numerous small parts and each part can be run in any node in the cluster. This paper contains detailed study of the execution of MapReduce programs over Hadoop cluster .It also discusses how the Hadoop platform offers an easy way of distributed Bigdata computing.*

Keywords: *Hadoop, Map Reduce, HDFS*

I. Introduction

MapReduce is a core component of the Apache Hadoop software framework. Hadoop enables resilient, distributed processing of massive unstructured data sets across commodity computer clusters, in which each node of the cluster includes its own storage. MapReduce serves two essential functions: It parcels out work to various nodes within the cluster or map, and it organizes and reduces the results from each node into a cohesive answer to a query. The volume and veracity of the datasets that are being stored and analyzed by the business are unforeseeable and the traditional technologies for data management such as relational databases cannot meet the current industry needs. Bigdata technologies like Hadoop play a vital role to address this issue. Primary aim of data analysis is to glean actionable logic that helps the business to tackle the competitive environment from this large unstructured datasets. This is possible in the current data centered environment by introducing Hadoop with its Mapreduce implementation . Bigdata analysis has a vital role in the industry to helps the organization to harness their transactional data and use it to identify new opportunities in a cost effective and efficient manner.

II. Problem Statement

Why Hadoop implements MapReduce paradigm?.Doug Cutting stated that with Hadoop, Bigdata is not treated like Bigdata, because a candidate is going to work with only some blocks of data. This can only be done if Hadoop implements the advantages of MapReduce programming like:
1. Data Locality: Pushing processing logic to where data resides.
2. Processing data in parallel, by launching mapper functions in separate nodes.

This structure helps Hadoop in effective utilization of cluster resources by consuming less network bandwidth. Hadoop contains two major components - a specific file system called Hadoop Distributed File System (HDFS) and a Map Reduce framework. Hadoop works on divide and conquer principle by implementing Mapper and Reducer in the framework. Mapper function splits the data into records and converts it into (key,value) pairs. Before feeding the output of the Mappers to Reducer an intermediate Sort and Shuffle phase is implemented in the MapReduce framework to reduce the work load at Reducer machine. The sorted (key,value)pair is given into Reducer phase. The Reducer function does the analysis of the given input and the result will be loaded to HDFS (eg.The maximum temperature recorded in a year, positive and negative ratings in a business etc.).The analyst has to develop Mapper and Reducer functions as per the demand of the business logic.

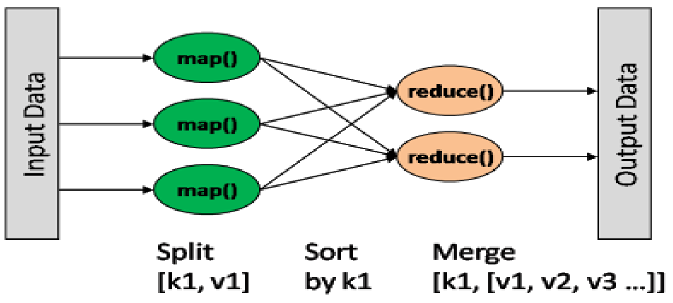

Figure 1 MapReduce

III. Mapreduce Components

1. Client: One who submits a MapReduce Job?
2. Resource Manager: Long lifed high quality hardware which deals with cluster level resource management.
3. Node Manager: Monitors resources on slave nodes.
4. Application Master: It coordinates and manages mapreduce jobs by negotiating with resource manager to schedule tasks. They are short lived and retain only one per application.
5. Container: Created by Namenode when requested by allocating certain amount of resources on the slave node.

IV. Anatomy Of Hadoop Map-Reduce Execution

Once we give a MR job the system will enter into a series of life cycle phases:
1. Job Submission Phase
2. Job Initialization Phase
3. Task Assignment Phase
4. Task Execution Phase
5. Progress update Phase
6. Failure Recovery

In order to run the MR program the hadoop uses the command-'*yarn jar client.jar job-class HDFS-input HDFS-output directory'*,where yarn is an utility and jar is the command.Client.jar and job class name written by the developer. When we execute on terminal the Yarn will initiate a set of actions(see figure 2)
1. Loading configurations
2. Identifying command
3. Setting class path
4. Identifying the java class corresponding to the jar command

i.e..org.apache.hadoop.util.RunJar.Then it will set the user provided command to "*java. Org .apache. hadoop. util. RunJar job-class HDFS-input HDFS-output directory*".

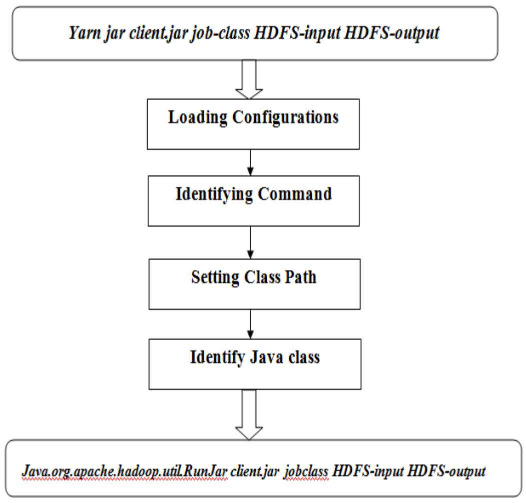
Yarn jar client.jar job-class HDFS-input HDFS-output
Loading Configurations
Identifying Command
Setting Class Path
Identify Java class
Java.org.apache.hadoop.util.RunJar client.jar jobclass HDFS-input HDFS-output

Figure 2 Yarn Actions

The execution system will create a RunJar class and call the main method (see org. apache.hadoop.util.RunJar GrepCode). Main method checks for the its arguments which are the client.jar and its job class name. Thus yarn will execute the job class (word count)with its arguments as the hdfs input and output. Job class then calls its own main method. The ToolRunner.run() method will call any of the trigger method i.e. submit, jobClient.runjob ,waitForComplition, and this will initiate the MR framework in edge node by checking the mapredsite.xml.

The 'map reduce.framework.name' field is visited and pulls the value which can be either YARN, LOCAL, CLASSIC, YARN-TEZ. Corresponding to the value specified a JobRunner instance is created. If the framework is using YARN then the instance created will be RemoteJobRunner, and LocalJobRunner if the value is LOCAL. The instance created then submit the application to the Resource Manager using RPC or Local protocols. Before submitting the application MR framework will create JobSubmitter(JS) instance and initialize the job submission phase of the MR job.

A.Job Submission Phase

The JobSubmitter invoke getNewApplicationId() on Resource Manager to get a new Application id for the job initiated. Then it validates input and output paths given by the user and checks whether the input file and output directory already exist or not. JobSubmitter contact the NameNode and get the metadata .It also compute the input splits ie. How many blocks of data are needed and also creates a shared directory inside Resource Manager with corresponding Application id. JobSubmitter then create a job.xml file which contains the following details of map reduce job, JobClassName,Input,Output,InputFormat,OutputFormat,Number of Splits,,Mapper, Reducer Class,Jar file name etc.. and copy the job.xml to shared directory .Ten copies of client.jar are created and keep the first copy in the shared directory and keeps the remaining nine copies in any configurable nodes in the network. Js submit this application-id to Resource Manager. The workflow is explained as(see figure-3):-

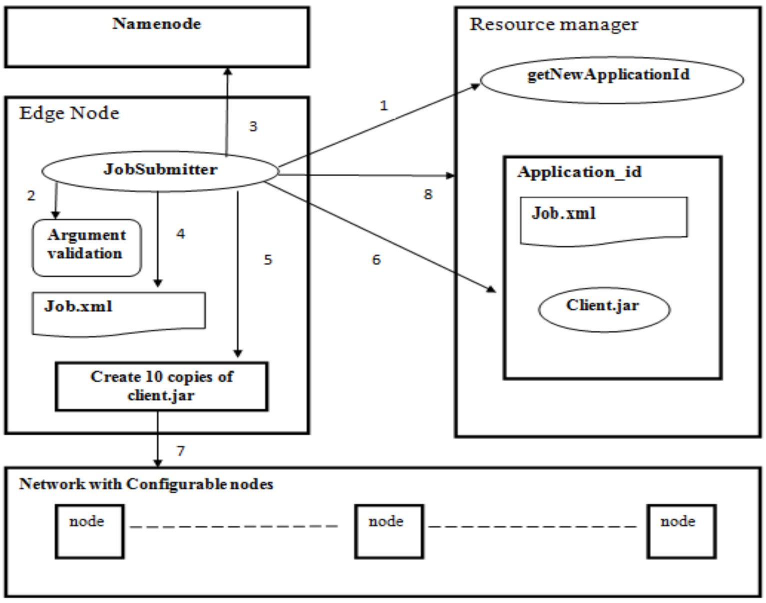

Figure -3 Job Submissions

1. Generate new application id.
2. Validate the user input.
3. Gather metadata from NameNode.
4. Create job.xml file
5. Create ten copies of client.jar.
6. Create a shared directory(job.xml,client.jar).
7. Keep nine copies of client.jar in Datanodes.

B.Job Initialization

Once the job is submitted by the JS the control will move to the cluster side from edge node. Resource Manager contains a candidate called Application Manager which maintains a queue of application ids submitted by the clients. The default name of the queue is ‘default’. The applications submitted in the queue are scheduled by ‘YARN Scheduler’ .To execute the job an Application Master is required, which cannot be created by YARN Scheduler. So a request is given to the Resource Manager to create Application Master. Resource manager will contact all Node Managers(NM) in the cluster to check the container-0 specification (minimum 2 GB and maximum 20GB).If any Node Manager possesses container-0 specification will respond to RM with a positive signal. In the new generations of Hadoop, NM continuously sends heart beats to the RM with its specifications so it’s possible for the RM to select NM without network overhead.

Resource Manager will identify what type of application we are executing (YARN is not a MapReduce specific platform, it’s a diversified programming approach).Since we are executing MapReduce application type,it wil request thenode manager to create JVM in container-0 and deploy MRAppMaster which is a MapReduce specific AppMaster. This AppMaster acts as the leader which manages the execution of the specified job by initiating it with the application id. AppMaster then go and contact the shared directory in the RM to get the job.xml file. It creates Mappers based on number of input splits specified in the file.ie Number of Mappers =Number of input splits. Reducer count is decided by the developer based on the output load and can be specified in job.xml.The Job Initialization phase will come to an end when the mapper ids for the required splits are created and add in the queue. The workflow is explained as (see figure-4):-

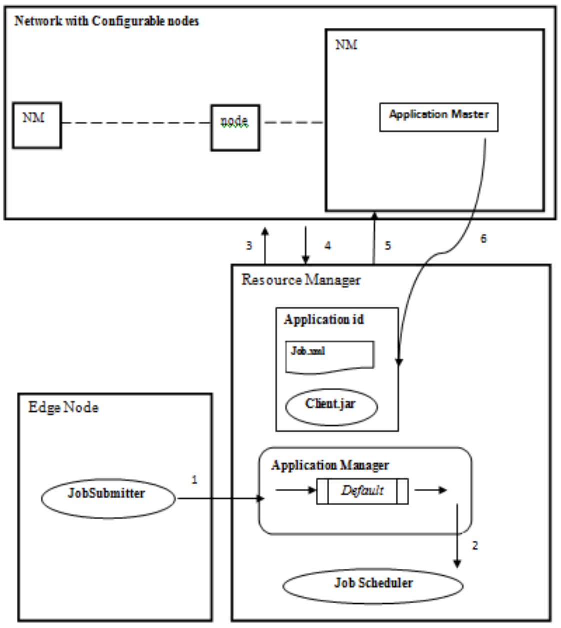

Figure-4 Job Initialization

1. Add application id into default queue.
2. JobScheduler starts scheduling.
3. Check with NM for container-0 specification.
4. NM replays with its own specification.
5. Select and register the NM, create AppMaster for execution.
6. Pull shared information and creates required mapper ids.

C.TASK ASSIGNMENT PHASE

One of the advantages of Hadoop framework is that the data splits are distributed across the network in multiple data nodes .Thus phase three deals with deploying the mapper task in data node which contains corresponding input splits(see figure 5). Thus the burden of processing in one node is distributed among multiple nodes. Application Master uses the data locality concept. It contacts RM for negotiating resources and submit the data node information which satisfy the data locality criteria and keeps the informations.AM initiate YARN child creation on each specified data node with the same application id as their task id and submit the mapper id to be executed and it also create a temporary local directory in the specified data node. Thus the phase three for task assignment comes to an end.

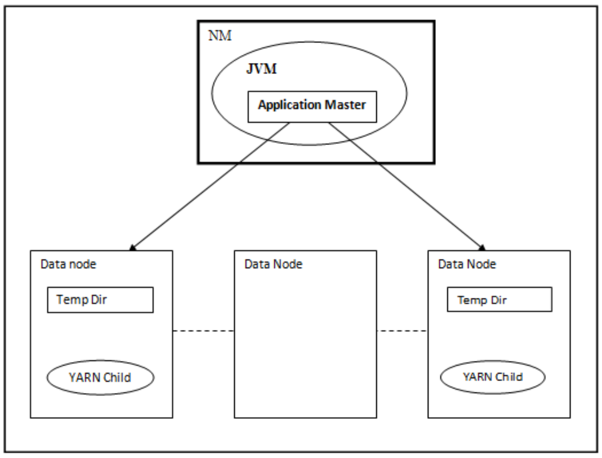

Figure-5 Task Assignment

D.Task Execution Phase

At this point the control is transferred to the particular data nodes which possess the YARN child. It then contact the shared directory on RM to get the related information and copy the files from the shared directory of RM to the locally created temporary directory ie the client.jar and job.xml etc..This is the main theme of hadoop which is pushing the process to data, thus by making data locality possible.

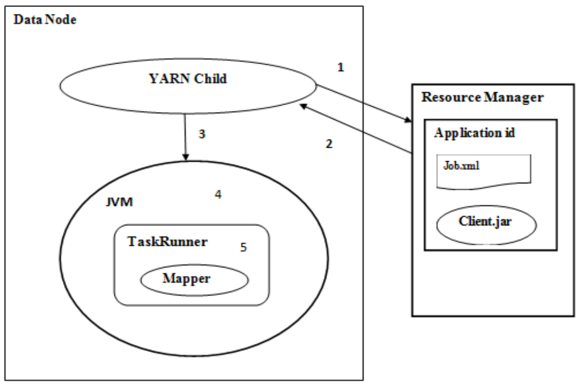

Figure- 6 Task Execution

It then sets the class path to the temporary directory because client.jar should be visible to the JVM which is going to be created by the YARN child. For executing the mapper function the JVM is created and execute a MR related framework called TaskRunner which is a java class object executing on the top of locally created directory .TaskRunner will identify which class should run over JVM ie. The Mapper class in client.jar.It then starts executing the Mapper program. The workflow is explained as(see figure-6):-

1. Request the shared informations.
2. RM transfers the requested files.
3. YARN child creates JVM for execution.
4. Create TaskRunner to run mapper locally.
5. Execute the Mapper program.

E. Progress Update

Once the execution started in distributed data nodes the progress have to be send to the respective initiative modules. Mapper and Reducer JVM execution environment sends progress report to the corresponding AM periodically (every 1second).AM accumulate progress from all MR tasks and communicate to client only if a change in progress happen for every 3 seconds. In reverse client also sends the request of completion in every 5 seconds. Once all the tasks are completed AM cleanup the temporary directory and send the response to the client with the results. Mappers and reducers intermediate output is deleted only when all the tasks got the completion response, otherwise if a reducer fails it still needs the output from the mapper programs.

F.Failue Recovery

Hadoop provides a facility to store the trace of the user and system operation by using FSImage and Edit.logs in the Namenode.The Secondary Namenode checkpoint backup mechanism provides a hard backup technique for hadoop in case if the Namenode goes down.All the informations regarding the executions are saved and if at any point of time the server goes down system calls the log files and restart the execution.Thus the failure handiling is done by the assistance of the framework itself .

V. Conclussion

Hadoop provides flexible architecture which enables industrialist and even starters to learn and analyse social changes.This is achieved by the highly efficient,fault tolerent framework called MapReduce.This paper is a detailed study of how MR jobs are executed in the hadoop cluster and how the distributed parallel processing is achieved by pushing process to data in the framework.

Reference

[1] Apache Hadoop. [Online]. Available: http://hadoop.apache.org
[2] Apache Hadoop-Streaming. [Online]. : http://hadoop-streaming.apache.org
[3] Cassandra wiki, operations. [Online]. Available: http://wiki.apache.org/cassandra/Operations
[4] NOSQL data storage [online]: http://nosql-database.org
[5] National energy research scientific computing center. [Online]. Available: http://www.nersc.gov
[6] Apache Hadoop [Online]: http://spark.apache.org
[7] Data Mining with BigData by Xindong Wu, Xingquan Zhu, Gong-Qing Wu, Wei Ding , 1041-4347/13© 2013 IEEE
[8] Tom White –Hadoop The Definitive Guide 3rd EditionJan 2012.pdf
[9] Online tutorial www.edureka.co/big-data-and-hadoop