ISSN 2278-7763

A Discussion on Testing Hadoop Applications

Sevuga Perumal Chidambaram

ABSTRACT

The purpose of analysing the big data, using Hadoop framework, is to bring the useful insight from the huge size of the data. It provides great opportunities for the stakeholders to take the appropriate decision for improvement of the business.

The paper discusses the specific challages that we encounter during planning, preparing and executing the test for verifying the functionalities of Hadoop applications..

1 INTRODUCTION

Hadoop applications are used to process big data since the traditional systems are incapable of processing such volume of data.

The concept of both Big data and Data ware housing is that to have an architectural model for retrieving the data from different operational systems, transforming them in order to merge together and storing them in a centralized place with some measured data. However the Data Warehousing uses a single machine to store and process the data. Hadoop supports distributed data storage and parallel data processing for analyzing the Big data.

The distribution of data storage and data processing provides cheap and economical data storage and processing the data in parallel. However the solution architect and designer need to face a great amount of challenges to design the process and methods to achieve the desired results. The testing team needs to prepare a complex and detailed test approach that helps us handle the complexity involved in testing of big data.

The testing of Hadoop applications is far different from the testing of traditional systems. The testing of traditional systems by QA team mostly involves black box testing, however, the Hadoop application testing needs mostly grey box testing and some amount of white box and black box testing. The reason is that we need to verify the intermediate data and we should execute the test with different set of configuration factors.

The testing strategy for the Hadoop applications should contain the usual testing approaches that are carried out for Data ware housing and ETL process that includes

- Ensure that no data is lost while loading the data into HDFS from various sources
- Ensure that transformation of data according to the business rules specified
- Ensure that loading of the data within the specified time window.
- End to End functional testing
- UI testing
- Interface testing

However the purpose of this paper is discussing the specific challenges we, the testing team, are facing with testing Hadoop applications rather than discussing the traditional test approaches.

The challenges we are facing in testing Hadoop applications is categorized into three characteristics of the Big Data, such as Volume, Variety and velocity, for our convenience.

2 VOLUME

The Big data is handling a colossal volume of data. This demands to have an appropriate solution to handle different issues created by the huge volume of data. The source data from operational systems are stored in multiple systems rather than one system as in the case of Data Warehousing.

The volume of the data is too big to be stored in a single machine, hence Hadoop provides the solution of storing the data in a cluster of servers by slicing the data. It also provides cost effectiveness by supporting to have commodity servers in the cluster for data nodes.

We should have different test approach based on the testing environment where the test is being executed. The environment can be a single machine or pseudo – distributed mode or production like cluster (distributed nodes).

We need to select the size of the data, which is a sample of production data, based on the type of testing environment in which the testing is carried out. The available memory and storage area of the testing environment decides the amount of data that can be used for testing.

We should predict what kind of result we can expect from a particular sample of data as the results we obtain can be varied based on the sample of test data. In that case we should plan a range of samples and a method to analyse the results.

2.1 Data storage

The Hadoop Distributed File System has two types of nodes working as master-worker pattern. The NameNode is the master node and data nodes are worker nodes. The data is stored in the data nodes and Namenode manages the filesystem namespace. Secondary NameNode acts as either check point node or backup node for the NameNode.

In HDFS, the files are broken into data blocks and they will be stored in the data nodes. The data nodes are regularly sending heart beats to the NameNode with current details of data

ISSN 2278-7763

blocks.

We should evaluate that how the file system ensures the successful execution of our applications.
Our testing approach should contain the required methods to

- Ensure that Hadoop has stored the data across multiple data nodes with proper load balancing.
- Ensure that the design has enough hardware configurations for Namenode and Secondary Namenode to handle the number of blocks.
- Different block size can be used for different input files. Ensure that NameNode manages meta data of the data blocks with the available memory and communicate with data nodes efficiently.
- Ensure that the replication factor, that has been configured, is implemented by storing the data blocks with required redundancy. If various files are stored with different replication factor ensure that the configuration of such arrangement works fine to meet the specified requirement of the fault tolerance and data localization of the application.
- Ensure that the secondary NameNode is acting as either Check point node or back up node as per the configuration.
- Ensure that the secondary node stores the snap shot of FsImage and EditLog of NameNode at the predefined intervals.
- Ensure that the time required to get the system resume in case of failover of Name node is minimal.
- Ensure that when the client API write/read data the client, NameNode, and data nodes are communicating themselves efficiently.
- Ensure that the execution of application is not disturbed in the case of any data node failure.. The Hadoop frame work takes care of that kind of situation so that the impact to the execution of application is minimal. The Hadoop framework initiates the data processing in other data nodes that contain the replicated data blocks of failure data node.
- Ensure that after a new data node is added in the cluster to meet the requirement of scaling out, the administer can run the load balancer to distribute the load effectively among the data nodes.
- Ensure that when a data node is removed from the cluster the Namenode carries out the decommissioning process properly.
- Ensure that the architectural team provides guidance to the development/administration team to reconfigure the block size for the remaining files when Hadoop throws error even though HDFS has enough space to store the required files.
- When the checksum of any block is corrupted validate that the NameNode create a new block by copying the uncorrupted block in the appropriate data node to improve the fault tolerance and data localization

2.2 Managing Tasks

Hadoop contains computational daemons to manage the tasks. It contains master/worker pattern for computation like the storage architecture.
The JobTracker is the master which determines the execution plan for the jobs submitted. The TaskTrackers are the workers that are running in every data node to manage the work in that particular data node. TaskTracker initiates task instances to do the actual work.
The JobTracker, TaskTracker and task instances are communicating regularly to carry out the execution successfully by handling the adverse events, that may happen while execution, effectively.

The task tracker marks the task instance as fail when it does not receive any progress from the task instance for a certain time, configured through the property "mapred.task.timeout". In the testing environment, this property needs to be set with a proper value so that we should make some instances as failed and evaluate how the JobTracker and TaskTracker handle this kind of situation.We should also monitor the JobTracker which creates a new task instance for the same data block in some other DataNode.
Our testing approach should contain a way to verify how the Jobtracker handle failure and blacklisting of TaskTracker through the properties "mapred.tasktraker.expiry.interval", "mared.max.tracker.failure"and"mapred.max.tracker.blacklists'.

3 VARIETY

The sources of data to Data Warehousing is relational data, it is mostly structured data. However the Hadoop handles not only the structured data but also semi structured and unstructured data.
The sources of data received by Hadoop can be produced by

ISSN 2278-7763

customers or operating personnel or the machines.
As far as the structured data is considered, we can use the traditional methods of ETL process with little changes however when the semi-structured and/or unstructured data is also involved in the design, this initiates us to prepare a complex testing approach to verify the intermediate level data and validate the final result.
The unstructured data can contain uneven length of records and the positions of the characters, we are interested, can be varied record by record. In that case we need to have a sample of data which contains representative of different length of records to verify the programs. Ensure that the program is flexible enough to handle all kind of production records and produce the designed result.

3.1 Skipping bad records

Hadoop can handle variety of data and it receives data from various sources. The possibility of handling different kind of records by the program is high.
When the application encounters unexpected bad records the application crashes after making number of task attempts as per the configuration.
If the system is complex and it is expected that it would receive bad records then the system is designed with a way to skip the bad records. The architecture team chooses this design when the missing of some records will not make much differece in the result and analysis can be done without difficulty.
In our testing environment, we should have representation of the bad records in our input and observe how the system handles that kind of records and we should also evaluate the impact of skipped records in the result.

3.2 Joining Data

The joining of data from different sources also brings difficulties to the testing team. The structured, semi-structured and unstructured data can be joined after cleaning of the data is done. The data needs to be properly transformed so that it can be joined with other data.
We should have proper sampling of data that contains representation of all kind of production data from all sources to ensure that the joining of data is being done in a proper way and we get the results as expected.

4 VELOCITY

Since Big data handles huge volume and great variety of data the designing of velocity of data demands to have an excellent architecture that takes care of the performance of the system well.
The Hadoop frame work achieves parallel processing by distributed data processing. However it also presents the challenge of heavy network load. Hence a careful design of the architecture, which reduces the amount of data that transfers across the data nodes, becomes an utmost important requirement.
The performance testing of the Hadoop application needs evaluation of the complete application architecture to produce detailed analysis.

4.1 Compression

The sequence files are used to obtain the performance enhancement by serialization of the data. The data will be used in binary form. To increase the performance further, the intermediate data can be compressed. The compression is not only used for performance improvement but also for effectively utilizing the storage space.
The compressed intermediate data needs to be decompressed in the memory before it is processed by the reducer.
If the compression is used for the sole purpose of performance improvement, we should execute a range of samples with compression and with no compression. We then analyze how efficiently the compression is used to improve the performance.
The compression is not only used for the intermediate data, it could also be used for the output of reducer when that is sent to the next MapReduce function as input. In our performance testing, this also needs to be evaluated by using the data with compression and without compression.

4.2 JVM replication factor

The JVM replication factor through "mapred.submit.replication" property creates the number of copies of JAR in the cluster. This factor is designed for the production environment, however, if the testing environment is not that big of production cluster we need to choose an appropriate number for this property to analyze the performance of the system based on this factor.
The number can be decided with the consultant of architect and development teams.
If we use the same number as in the production environment for this property in testing environment also, we can get better performance result in the testing environment. since it has limited nodes. However it leads to wrong conclusion as we cannot expect this kind of performance in production environment. So choosing an appropriate factor for testing environment is very important.

4.3 Scheduler

The testing team should identify that which scheduler is going to be used in the production environment. We need to create a situation accordingly in the testing environment to evaluate the scheduler how it handles the jobs that are submitted from various clients.
We should submit the job with different priorities and monitor how the scheduler handles jobs with higher priorities.

4.4 Shuffle, Sort, Partitioner and Combiner

The outputs of map function are transferred to the reducer

ISSN 2278-7763

function. This process is called Shuffle. The map functions do not write its outputs directly to the disk instead, the outputs of the mappers are sorted in the memory by a thread and then it is written to the disk. When the sorted data in the buffer reached to a certain percentage, configurable, the data is spilled to the disk.
In the testing environment the "io.sort.mb", which sets the memory size of the buffer, and the "io.sort.spill.percent , which sets the threshold of the buffer, are set with different sets of values to monitor the performance of the mapper.

In the reducer side, we can monitor the performance of the program by using different merge factor.
By default, the reducers output the records with partial sort. If we are testing an application, that contains the required customization to produce the records with total sorting, we should make sure that the system produces the records with total sort within a specified time window.
We can use the same approach for the application which has the required customization to produce the records with secondary sort.
If more than one reducer is used then the partitioning of data, based on the keys, is also carried out in the memory before spilling it into disk. While using either the default partitioner or the customized partitioner we should observe if one or few reducers receive more data than the other reducers. The distribution of skew of data to reducers will highly impact the performance of the whole application.
We should monitor the performance of the reducers. If we identify one or few reducers take more time to process the data we need to report to the development and architecture teams.
The next function we should watch for performance is the combiner function if it is used. We should evaluate how the use of combiner improves the performance of the application. The combiners are used to reduce the size of the data transferred across network from mapper to reducer. The combiner can be executed zero, one, or more than one times in the mapper side.
In our testing environment we can use various sample of data with various configuration to monitor how the combiner improves the overall performance of the application.
The test strategy should contain an approach to evaluate the effectiveness of JVM reuse, defined in the system, and how it improves the performance of the application.

4.5 Speculative executions

The speculative execution is set on by default to optimize the process. We need to observe the behaviour of the system when we identify any slow execution of task instance and evaluate how the speculative execution improves the performance of the application.

4.6 Block and input splits

The block is the physical division of data and input split is the logical division of the data. An individual mapper is assigned to every input split.
We should observe how the configured block and input splits are working together to achieve the localization of the data.
An application can have different block size for various files as well as different input format. We should evaluate how effectively these are configured.
The custom input format " CombineInputFormat" is used to handle a large number of small size files. This format combines more than one file to create a single input split. Here also we should monitor how the localization of data is achieved when the files are combined together to form as one.
If the design needs not to split the file then the isSplittabe method is designed to return false. We should evaluate how it impacts the performance of the application.

4.7 Joining Data

In Hadoop the joining of data can happen either in the mapper side or in the reducer side.
There is no concern of memory requirement for all nodes of cluster in the reducer side join. However the performance of the process is of more important. We should monitor the performance of the system with a range of sample data to analyze the capability of the system.
In the mapper side joining, the smaller source of the two sources, that are to be joined, is stored in the memory of all the mappers. This is achieved by the distributed cache mechanism. We should get the maximum size of the smaller source that would be expected in the production environment for using in our testing environment. We can execute the joining process with this smaller source in our testing environment to ensure that no error relating to memory requirement happens.

4.8 Counters

The counters used in the MapReduce program are of more helpful to the testing team for analyzing the metrics.
Apart from build-in counters the development team can use user defined counters also for any important parameters used in the program. The testing team can use the metrics created by both build-in counters and user defined counters to produce a more informative testing report.

5 TOOLS

Hadoop applications has other eco tools along with hadoop core system. Every tool used in the Hadoop applications has specific purpose to support core system. For example Oozie is used for work flow management, Hive and Pig programming are used to facilitate the ease of writing the program. Sqoop is used to transfer relational data from external sources to Hadoop or vice versa, Flume is used to transfer bulk log data from web server, Zookeeper is used to coordinate the distributed servers, HBase is used for real time access, etc..
There can be a main test strategy which also includes the test approach for the tools used in the applications or we can have separate test strategy for these interfaces.
We can discuss the purpose of some tools and how we can prepare the approach to test the functionalities of those tools

ISSN 2278-7763

in this section.

5.1 Pig Programming

Pig programming is a high level data processing. When the MapReduce program is being developed one has to think in the way of mapper and reducer and data should be assumed to be keys and values. However in the traditional way, for the ETL process, the programmer can think through a data processing task that contains data flow operations such as loops and filters. Pig aids the programmer to think through data processing, rather than key and values pairs. Pig produces a chain of MapReduce programs internally to accomplish these tasks.

Pig provides programmers with a simple way to write a program in single or double digit number of statements to get the required report. The same report can be produced in MapReduce program that contains hundreds of lines.

The Pig programming is mainly used for ETL process, research on raw data and iterative processing.

One of the advantages, we have when verifying the functionality of the Pig programs is that we can break the Pig program at any point where we need to verify the intermediate data .

The DEFINE and ILLUSTRATE expressions in Pig program can be of more useful for us to analyze the data transformations happened by the expressions.

Pig program can be used along with MapReduce programs. The MapReduce programs and Pig programs are used in the data pipe line. The output of the MapReduce program can be fed as input for the Pig program and vice versa. We can verify the output of the program for its correctness, no loss of data and consistency before it is fed as input to the next program.

5.2 Hive

Hive is another one high level language. It uses a SQL like language, called as HiveQL, to process the data. It is so handy for the people who are so familiar with SQL language. It also produces MapReduce programs to process the data.

Hive uses partitioning and bucketing to improve the performance of the process. In our test environment we should evaluate how portioning and bucketing improve the performance of process.

Hive produces metadata for the tables being created. We can dig more details about the tables from this metadata database.

If Hive programs are interconnected with other programs, we should have an approach to test at the interface.

5.3 Oozie

The Apache Oozee, which is the workflow system used in Hadoop, is a great aid for the testing team. We can change the workflow using this tool based on the testing need.

The Oozie contains action nodes and control-flow nodes. The control-flow node makes the decision based on the current condition and instructs the action node to perform the work flow task accordingly.

There would be a number of paths in the work flow to connect the start and end point of the workflow. We should evaluate each and every path of the workflow for which we should generate different conditions in the testing environment.

The testing team can also adjust the workflow if they need any intermediate data for verification.

5.4 Sqoop

Sqoop is an open source tool that allows users to extract data from a structured data store into Hadoop for further processing. Sqoop has an extensive framework that provides bulk data transfer capabilities for importing data from/ exporting data to external systems from Hadoop. Sqoop uses MapReduce programs that connects the source database and reads the table in parallel.

We should make sure that no data is lost while transferring, no inconsistency of data and no redundancy of data.

Sqoop can directly load the data to Hive tables from source tables. In that case, the metadata for the tables imported into Hive is created by Sqoop and Hive uses that information to process the tables in Hive programs.

We should make sure that the metadata created by Sqoop makes no inconsistency so that Hive works fine with that metadata.

5.5 Flume

Flume is the tool which is used to collect data from web servers and transfer them to Hadoop. Agent, one of the components of Flume, works in the web server and it sends the data to collector, another one component of Flume, which in turn push the data to Hadoop.

Flume is used to push large volume of log data from various web servers to HDFS for further processing in Hadoop.

We should verify the configuration of all the components used in the Flume to evaluate how Flume, as whole, works to transfer the data without any error with an efficient manner.

We should make sure that no data is lost while transferring, no inconsistency of data and no redundancy of data.

5.6 Zookeeper

Hadoop applications are working in a cluster of servers. These servers need to be coordinated to work fine and if failure happens in any server, that would be taken care of and the applications that are running at that time should not be much impacted by this failure.

Zookeeper is an open source tool that manages the distributed processes in large system to synchronize with each other. Zookeeper helps a cluster of servers avoid single point of failure.

In a distributed application, when a message is sent across the network between two nodes and the receiver fails, the sender does not know whether the receiver got the message or not. This is called partial failure.

If the current master server in a cluster fails Zookeeper should select another node as new master and delegate the responsibilities of the master tasks to the new master.

In our testing, we should monitor how Zookeeper manages

ISSN 2278-7763

the partial failure and the failure of the current master.

6 Conclusions

It is evident that the Big data testing with Hadoop applications is a different world rather than the traditional testing. It is a new game where we need to implement different kinds of strategies to achieve the objectives of our testing.
It may present a completely new challenge while we are in the process of testing, for which we need to try with various approaches and discover the best methods.

