Consultants

A Primer on Big Data Testing

For more information please contact
Maria Kelebeev
416-238-5333 ext 240
mkelebeev@qaconsultants.com

www.qaconsultants.com

PREFACE

This report is the output of a research project by QA Consultants - the North American leader in onshore software testing. This paper focuses on the primary challenges of testing Big Data systems and proposes methodology to overcome those challenges. Because of the complex nature of both Big Data and the highly distributed, asynchronous systems that process it, organizations have been struggling to define testing strategies and to set up optimal testing environments. The focus of this primer is on important aspects of methods, tools and processes for Big Data testing. It was completed with the support of the National Research Council of Canada.

TABLE OF CONTENTS

1 Big Data and Bad Data

> "70% of enterprises have either deployed or are planning to deploy big data projects and programs this year."
>
> *Analyst firm IDG*

> "75% of businesses are wasting 14% of revenue due to poor data quality."
>
> *Experian Data Quality Global Research report*

> "Big Data is growing at a rapid pace and with Big Data comes bad data. Many companies are using Business Intelligence to make strategic decisions in the hope of gaining a competitive advantage in a tough business landscape. But bad data will cause them to make decisions that will cost their firms millions of dollars.
>
> According to analyst firm Gartner, the average organization loses $8.2 million annually through poor Data Quality, with 22% estimating their annual losses resulting from bad data at $20 million and 4% putting that figure as high as an astounding $100 million. Yet according to the Experian Data Quality report, 99% of organizations have a data quality strategy in place. This is disturbing in that these Data Quality practices are not finding the bad data that exists in their Big Data."
>
> *Query Surge*

With respect to software development and verification processes, testing teams may not yet fully understand the implications of Big Data's impact on the design, configuration and operation of systems and databases. Testers need a clear plan to execute their tests but there are many new unknowns as Big Data systems are layered on top of enterprise systems struggling with data quality. Added to those struggles are the challenges of replicating and porting that information into the Big Data analytic and predictive software suite. How do you measure the quality of data, particularly when it is unstructured or generated through statistical processes? How do you confirm that highly concurrent systems do not have deadlock or race conditions? What tools should be used?

It is imperative that software testers understand that Big Data is about far more than simply data volume. For example, having a two-petabyte Oracle database doesn't necessarily constitute a true Big Data situation - just a high load one. Big Data management involves fundamentally different methods for storing and processing data, and the outputs may also be of a quite different nature. With the increased likelihood that Bad Data is imbedded in the mix, the challenges facing the quality assurance testing departments increase dramatically. This primer on Big Data testing provides guidelines and methodologies on how to approach these data quality problems.

2 Characteristics of Big Data

Big Data is often characterized as involving the so-called "three Vs": Volume, Velocity and Variety.

(Some authors also add a fourth characteristic: Veracity.)

2.1 Volume: The quantity of data

With the rise of the Web, then mobile computing, the volume of data generated daily around the world has exploded. For example, organizations such as Facebook generate terabytes of data daily that must be stored and managed.

As the number of communications and sensing devices being deployed annually accelerates to create the encompassing "Internet of Things," the volumes of data continue to rise exponentially. By recording the raw, detailed data streaming from these devices, organizations are beginning to develop high-resolution models based on all available data rather than just a sample. Important details that would otherwise have been washed out by the sampling process can now be identified and exploited. The bottleneck in modern systems is increasingly the limited speed at which data can be read from a disk drive. Sequential processing simply takes too long when such data volumes are involved. New database technologies that are resistant to failure and enable massive parallelism are a necessity.

2.2 Velocity: Streaming data

It is estimated that 2.3 trillion gigabytes of data are created each day. In our highly connected world, trends of interest may last only a few days, hours or even just minutes. Some important events, such as online fraud or hacking attempts, may last only seconds, and need to be responded to immediately. The need for near-real time sensing, processing and response is driving the development of new technologies for identifying patterns in data within very small and immediate time windows.

The Dawn of Big Data: the uncertainty of new information is growing alongside its complexity

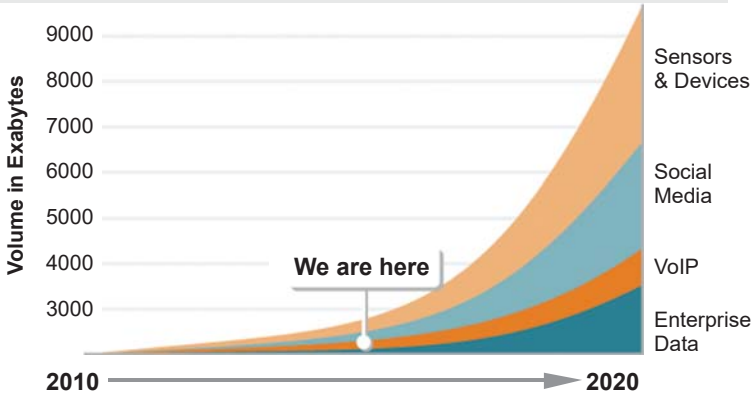


Chart courtesy of International Business Machines - Slide Share

2.3 Variety: Different types of data

A common theme in Big Data systems is that the source data is increasingly diverse, involving types of data and structures that are more complex and/or less structured than the traditional strings of text and numbers that are the mainstay of relational databases. Increasingly, organizations must be able to deal with text from social networks, image data, voice recordings, video, spreadsheet data, and raw feeds directly from sensor sources.

2

Even on the Web, where computer-to-computer communication ought to be straightforward, the reality is that data is messy. Different browsers send differently formatted data, users withhold information, and they may be using different software versions or vendors to communicate with you. Traditional relational database management systems are well-suited for storing transactional data but do not perform well with mixed data types.

In response to the “three Vs” challenge, Hadoop, an open source software framework, has been developed by a number of contributors. Hadoop is designed to capture raw data using a cluster of relatively inexpensive, general-purpose servers. Hadoop achieves reliability that equals or betters specialized storage equipment by managing redundant copies of each file, intentionally and carefully distributed across the cluster.

Hadoop uses its own distributed file system, HDFS, which extends the native file system of the host operating system, typically Linux or Windows.

A common Hadoop usage pattern involves three stages:

1. Loading data into HDFS
2. MapReduce operations
3. Retrieving results from HDFS

This process is by nature a batch operation, suited to analytical or non-interactive computing tasks. For this reason, Hadoop is not itself a full general-purpose database or data warehouse solution, but can act as an analytical adjunct to or the basis of one when supplemented with other tools.

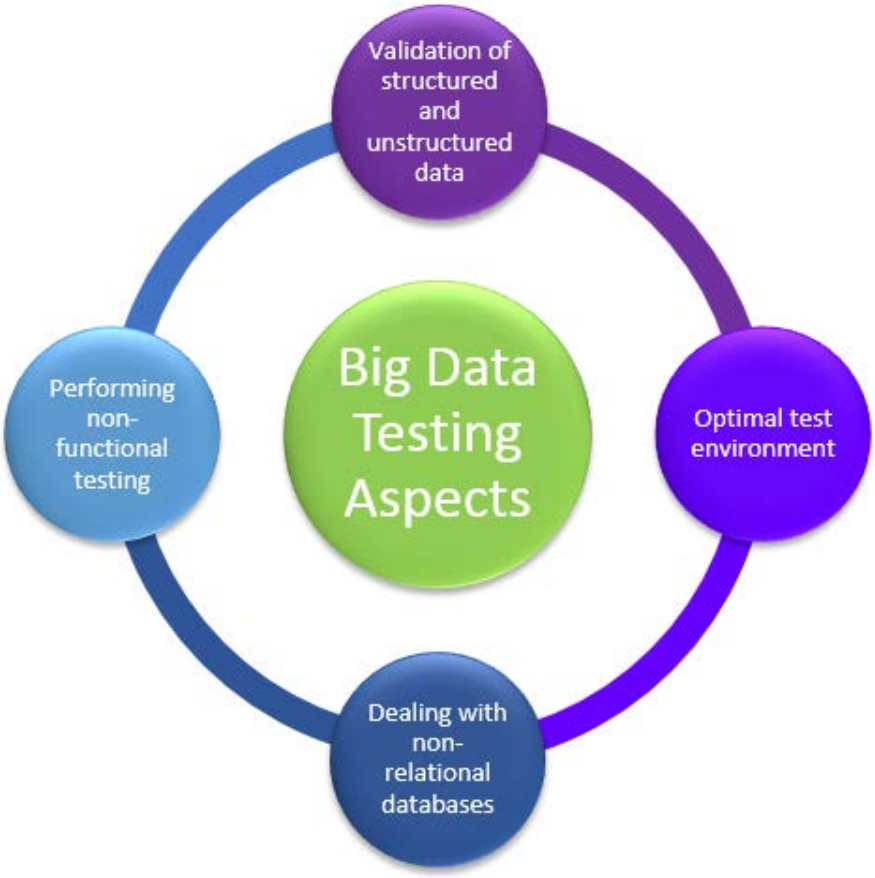
One of the best-known Hadoop users is Facebook, whose model follows this pattern. A MySQL database stores the core data, which is then reflected into Hadoop, where computations occur, such as creating recommendations for you based on your friends’ interests. Facebook then transfers the results back into MySQL for use in pages served to users.

Hadoop is not actually a single product but is instead a growing collection of components and related projects. Following are a few of the many components that would need to be tested for correct installation, configuration and functioning in a typical Hadoop environment.

1. Hadoop HDFS: the file system
2. Hadoop YARN: the Hadoop resource coordinator
3. Apache Pig: a query system
4. Apache Hive: a system that allows data stored in Hadoop to be structured as tables and queried using SQL
5. Apache HCatalog: the Hadoop metadata store
6. Apache Zookeeper: a coordination system
7. Apache Oozie: a process scheduling and sequencing system
8. Apache Sqoop: a tool for connecting Hadoop to relational databases
9. Hue: a browser-based user interface for several of the tools above

3 Big Data Testing Aspects

Some of the key aspects of Big Data testing are the following.



Poor implementation of the above key components of testing Big Data environments can lead to poor quality, delays in testing, and increased cost. Defining the test approach for process and data validation early in the implementation life cycle ensures that defects are identified as soon as possible, reducing the overall cost and time to implementation of the finished system. Performing functional testing can identify data quality issues that originate in errors with coding or node configuration; effective test data and test environment management ensures that data from a variety of sources is of sufficient quality for accurate analysis and can be processed without error.

Apart from functional testing, non-functional testing (specifically performance and failover testing) plays a key role in ensuring the scalability of the process. Functional testing is performed to identify functional coding issues and requirements issues, while non-functional testing identifies performance bottlenecks and validates the non-functional requirements. The size of Big Data applications often makes it difficult or too costly to replicate the entire system in a test environment; a smaller environment must be created instead, but this introduces the risk that applications that run well in the smaller test environment behave differently in production.

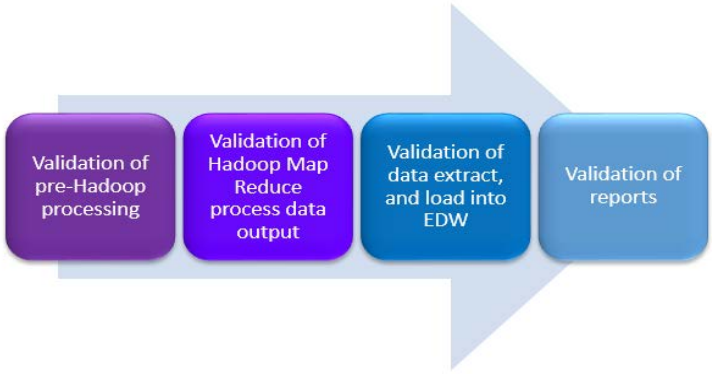
Therefore, it is necessary that the system engineers are careful when building the test environment, as many of these concerns can be mitigated by carefully designing the system architecture. A proper systems architecture can help eliminate performance issues (such as an imbalance in input splits or redundant shuffle and sort), but, of course, this approach alone doesn't guarantee a system that performs well.

Even with a smaller test environment, the data volumes may still be huge, requiring hours to run a single test. Careful planning is required to exercise all paths with smaller data sets in a manner that fully verifies the application but allows the test to run in a short enough period of time to allow repeated feasible testing.

The numerical stability of algorithms also becomes an issue when dealing with statistical or machine learning algorithms. Applications that run well with one dataset may abort unexpectedly or produce poor results when presented with a similar but poorly conditioned set of inputs. Verification of numerical stability is particularly important for customer-facing systems.

Two key areas of the testing problem are (1) establishing efficient test datasets and (2) availability of Hadoop-centric testing tools (such PigUnit, Junit for Pig, Hive UDF testing, and BeeTest for Hive).

Testing should include all four phases shown below. Data quality issues can manifest themselves at any of these stages.



4 Testing Methods, Tools and Reporting for Validation of Pre-Hadoop Processing

Big Data systems typically process a mix of both structured data (such as point-of-sale transactions, call detail records, general ledger transactions, and call center transactions), unstructured data (such as user comments, doctors' notes, insurance claims descriptions and web logs) and semi-structured social media data (from sites like Twitter, Facebook, LinkedIn and Pinterest). Often the data is extracted from its source location and saved in its raw or a processed form in Hadoop or another Big Data database management system. Data is typically extracted from a variety of source systems and in varying file formats, e.g. relational tables, fixed size records, flat files with delimiters (CSV), XML files, JSON and text files.

Most Big Data database management systems are designed to store data in its rawest form, creating what has come to be known as a "data lake," a largely undifferentiated collection of data as captured from the source. These DBMSs use an approach called "schema on read," i.e. the data is given a simple structure appropriate to the application as it is read, but very little structure is imposed during the loading phase.

The most important activity during data loading is to compare data to ensure extraction has happened correctly and to confirm that the data loaded into the HDFS (Hadoop Distributed File System) is a complete, accurate copy.

4

Data sources can include a local file system, HDFS, Hive tables, streaming sources, and relational or other databases. If the data is loaded into Hive, it can be validated and transformed using HiveQL, as one can do in SQL. If not, a MapReduce or equivalent process (e.g. Spark) will be needed.

Typical tests include:

1. **Data type validation.** Data type validation is customarily carried out on one or more simple data fields. The simplest kind of data type validation verifies that the individual characters provided through user input are consistent with the expected characters of one or more known primitive data types as defined in a programming language or data storage and retrieval mechanism.
2. **Range and constraint validation.** Simple range and constraint validation may examine user input for consistency with a minimum/maximum range, or consistency with a test for evaluating a sequence of characters, such as one or more tests against regular expressions.
3. **Code and cross-reference validation.** Code and cross-reference validation includes tests for data type validation, combined with one or more operations to verify that the user-supplied data is consistent with one or more external rules, requirements or validity constraints relevant to a particular organization, context or set of underlying assumptions. These additional validity constraints may involve cross-referencing supplied data with a known look-up table or directory information service such as LDAP.
4. **Structured validation.** Structured validation allows for the combination of any number of various basic data type validation steps, along with more complex processing. Such complex processing may include the testing of conditional constraints for an entire complex data object or set of process operations within a system.

HDFS can support up to 200 PB of storage and a single cluster of 4,500 servers, with close to a billion files and blocks. When HDFS ingests data, it splits the file into smaller pieces and distributes them across different nodes in a cluster.

This splitting is key to Hadoop's ability to perform resilient parallel processing. For those transformations that can be parallelized, Hadoop will use the all the processors across the cluster to perform the computation as quickly as possible.

One node of the cluster is reserved to serve as the NameNode, which knows the identifiers (names) of each of the other nodes in the cluster. The client (the device initiating a data load or other computation) asks the NameNode for the list of DataNodes, the servers where the data resides. When a client writes data, it first asks the NameNode to choose DataNodes to host replicas of the first block of the file. The client organizes a direct pipeline from the source server and sends the data. When the first block is filled, the client requests a DataNode to be chosen for the next block, most likely a different one than received the first block. A new pipeline is organized, and the client sends the next block of data from the file.

1. Checkpoint Node

The Checkpoint Node periodically combines the existing checkpoint and journal to create a new checkpoint and an empty journal. The Checkpoint Node usually runs on a different host from the NameNode since it has the same memory requirements as the NameNode. It downloads the current checkpoint and journal files from the NameNode, merges them locally, and returns the new checkpoint back to the NameNode.

Creating periodic checkpoints is one way to protect the file system metadata. The system can start from the most recent checkpoint if all other persistent copies of the namespace image or journal are unavailable. Creating a checkpoint also lets the NameNode truncate the journal when the new checkpoint is uploaded to the NameNode. HDFS clusters run for prolonged periods of time without restarts during which the journal constantly grows. If the journal grows very large, the probability of loss or corruption of the journal file increases. Also, a very large journal extends the time required to restart the NameNode. For a large cluster, it takes an hour to process a week-long journal. Good practice is to create a daily checkpoint.

4

2. Backup Node

The Backup Node can be viewed as a read-only NameNode. It contains all file system metadata information except for block locations. It can perform all operations of the regular NameNode that do not involve modification of the namespace or knowledge of block locations. Use of a Backup Node provides the option of running the NameNode without persistent storage, delegating responsibility of persisting the namespace state to the Backup Node.

3. Upgrades and File System Snapshots

The snapshot (only one can exist) is created at the cluster administrator's discretion whenever the system is started. If a snapshot is requested, the NameNode first reads the checkpoint and journal files and merges them in memory.

Then it writes the new checkpoint and the empty journal to a new location, so that the old checkpoint and journal remain unchanged.

During handshake the NameNode tells DataNodes whether to create a local snapshot. The local snapshot on the DataNode cannot be created by replicating the directories containing the data files as this would require doubling the storage capacity of every DataNode on the cluster. Instead, each DataNode creates a copy of the storage directory and hard links existing block files into it. When the DataNode removes a block, it removes only the hard link, and block modifications during appends use the copy-on-write technique. Thus old block replicas remain untouched in their old directories.

4.1 Tools for validating pre-Hadoop processing

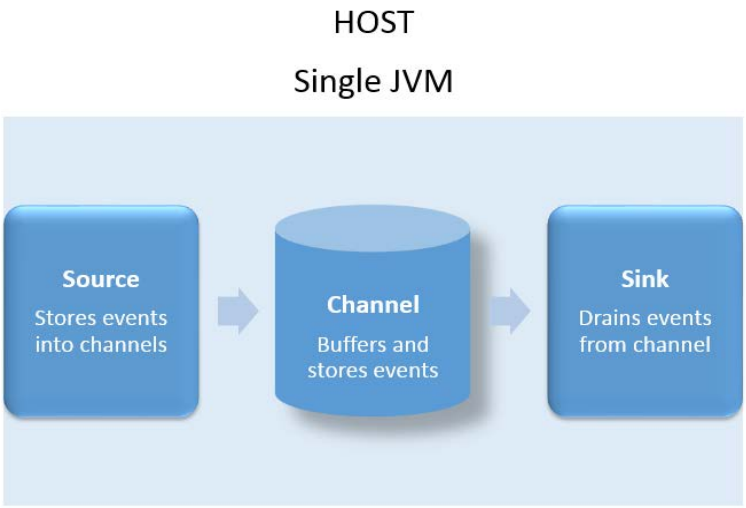
The following tools are components of the Hadoop ecosystem and can be used to assist with validating pre-Hadoop processing.

1. Apache Flume

Apache Flume is a reliable service for efficiently transferring large quantities of data into HDFS. Enterprises typically use Flume to ingest log files from application servers or other systems, often to archive them in compliance with prevailing regulations.

An important consideration when designing a Flume flow is the type of channel to use. There are two types: file channel and memory channel.

The file channel stores all events on disk so if the OS crashes or reboots, events that were not successfully transferred to the next node will not be lost. The memory channel buffers events in memory, so it is faster but less reliable should a failure occur.



4

2. Apache Sqoop

Apache Sqoop is a tool for transferring data between Hadoop and relational databases. For example, you can use Sqoop to import data from a MySQL or Oracle database into HDFS.

3. Hive

Hive is a tool that structures data in Hadoop into the form of relational-like tables and allows queries using a subset of SQL. Hive is a good tool for performing queries on large datasets, especially datasets that require full table scans. Hive can be used to support a tester who is interested in doing arbitrary queries to confirm values of calculated statistics or to run reasonableness tests across large swaths of data.

4. Pig

Apache Pig provides an alternative language to SQL, called Pig Latin, for querying data stored in HDFS. Pig does not require the data to be structured as tables, however, and can be more efficient than SQL if the queries involved require reuse of intermediate results. Pig comes with standard functions for common tasks like averaging data, working with dates, or finding differences between strings.

5. NoSQL

Not all Hadoop clusters use HBase or HDFS. Some integrate with NoSQL data stores that come with their own mechanisms for storing data across a cluster of nodes. This enables them to store and retrieve data with all the features of the NoSQL database, then use Hadoop to schedule data analysis jobs on the same cluster.

Most commonly this means Cassandra, Riak or MongoDB, and users are actively exploring the best way to integrate the two technologies. 10Gen, one of the main supporters of MongoDB, for instance, suggests that Hadoop can be used for offline analytics while MongoDB can gather statistics from the Web in real time.

6. Lucene/Solr

The most popular open source tool for indexing large blocks of unstructured text is Lucene. Written in Java, it integrates easily with Hadoop, enabling distributed, resilient text management. Lucene handles the indexing; Hadoop distributes queries across the cluster.

5

Testing Methods, Tools and Reporting for Hadoop MapReduce Processes

Hadoop MapReduce is a software framework for easily writing applications that process vast amounts of data (multi-terabyte datasets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

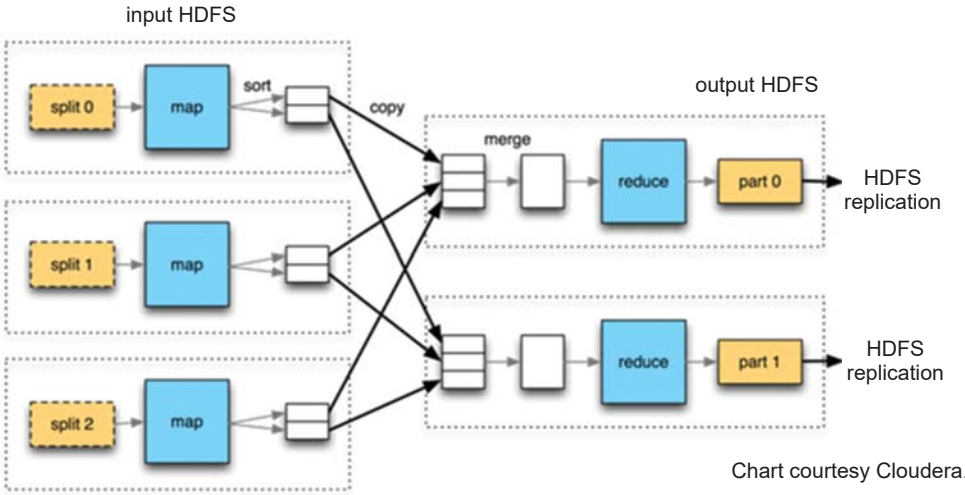


Chart courtesy Cloudera.com

5

A MapReduce job usually splits the input data-set into independent chunks that are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file system. The framework takes care of scheduling tasks and monitoring them, and re-executes the failed tasks.

The MapReduce framework runs on the same nodes where the data is stored, usually in HDFS. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

In Hadoop 1, MapReduce was essentially the only computational framework. In Hadoop 2, YARN ("Yet Another Resource Negotiator") was introduced to support a variety of frameworks, including MapReduce. The computation framework now consists of a single master Resource Manager, per- node Node Managers, and a framework-specific ApplicationMaster responsible for negotiating with the Resource Manager for containers with the required resources, and working with the Node Managers to execute and monitor those resource containers. An application, via the ApplicationMaster, asks for containers with a specific number of CPUs, memory, etc., and then the Node Managers of each node that will be participating in the computation provision them.

5.1 Methods and Tools

1. MRUnit - Unit testing for MR jobs

MRUnit is a tool that was developed by Cloudera and released back to the Apache Hadoop project. It can be used to unit-test map and reduce functions. MRUnit lets users define key-value pairs to be given to map and reduce functions, and it tests that the correct key-value pairs are emitted from each of these functions. MRUnit tests are similar to traditional unit tests in that they are simple, isolated, and don't require Hadoop to be running.

2. Local job runner testing - Running MR jobs on a single machine in a single JVM

Traditional unit tests and MRUnit tests should do a sufficient job of detecting bugs early, but neither will test MR jobs with Hadoop. The local job runner lets you run Hadoop on a local machine, in one JVM, making MR jobs a little easier to debug in the case of a job failing.

To enable the local job runner, set "mapred.job.tracker" to "local" and "fs.default.name" to "file:///some/local/path" - which are the default values.

Running bin/hadoop will start a JVM and will run your job for you. Creating a new hadoop-local.xml file is recommended. You can then use the -config parameter to tell bin/hadoop which configuration directory to use. If you'd rather avoid fiddling with configuration files, you can create a class that implements Tool and uses Tool Runner, and then run this class with bin/hadoop jar foo.jar com.example.Bar -D mapred.job.tracker=local -D fs.default.name=file:/// (args) where Bar is the Tool implementation.

To start using the local job runner to test your MR jobs in Hadoop, create a new configuration directory that is local job runner enabled and invoke your job as you normally would, remembering to include the -config parameter, which points to a directory containing your local configuration files. The user will have to ensure that input files are set up correctly and output directories don't exist before running the job.

The -conf parameter also works in 0.18.3 and lets you specify your hadoop-local.xml file instead of specifying a directory with -config.

The difficulty with this form of testing, however, is verifying that the job ran correctly. Simply basing success on exit codes isn't quite good enough. At the very least, you'll want to verify that the output of your job is correct. You may also want to scan the output of bin/hadoop for exceptions. You should create a script or unit test that sets up preconditions, runs the job, diffs actual output and expected output, and scans for raised exceptions. This script or unit test can then exit with the appropriate status and output specific messages explaining how the job failed.

5

3. Pseudo-distributed testing - Running MR jobs on a single machine using Hadoop

The local job runner lets you run your job in a single thread. Running an MR job in a single thread is useful for debugging, but it doesn't properly simulate a real cluster with several Hadoop daemons running (e.g. NameNode, DataNode, TaskTracker, JobTracker, SecondaryNameNode). A pseudo-distributed cluster is composed of a single machine running all Hadoop daemons. This cluster is still relatively easy to manage (though harder than local job runner) and tests integration with Hadoop better than the local job runner does.

To start using a pseudo-distributed cluster to test your MR jobs in Hadoop, follow the aforementioned instructions for using the local job runner, but in your precondition setup include the configuration and startup of all Hadoop daemons. Then, to start your job, just use bin/hadoop as you would normally.

4. Full integration testing - Running MR jobs on a QA cluster

Probably the most thorough yet most cumbersome mechanism for testing your MR jobs is to run them on a QA cluster composed of at least a few machines. By running your MR jobs on a QA cluster, you will be testing all aspects of both your job and its integration with Hadoop.

Running your jobs on a QA cluster has many of the same issues as the local job runner. For example, you will have to check the output of your job for correctness. You may also want to scan the stdin and stdout produced by each task attempt, which will require collecting these logs in a central place and grepping them. Scribe is a useful tool for collecting logs, though it may be superfluous depending on your QA cluster.

6

Testing Methods, Tools and Reporting for Data Extract and EDW Loading

As the Big Data technology and services market is showing a huge annual growth rate worth billions of dollars, data warehouses have a vital role to play in Big Data. Companies rely on data warehouses as they have to collect information on their business operations, markets and client behavior to identify patterns, and collect the results to identify more business opportunities and operational improvements.

6.1 Methods

The major part of the data warehouse system is data extraction, transformation and loading (ETL). The goal is to extract the data, often from a variety of different systems, and transform it so that it is uniform in terms of format and content, and, finally, to load the data into a warehouse where it can serve as the basis for business intelligence needs.

The integrity of the data must be maintained at every step. It must be stored clearly and concisely without loss, and should be accessible to all authorized professionals. So, for the data warehouses to deliver value, they require careful ETL testing to ensure that processes work as required.

The different methods for ETL testing depend on the challenges faced in performing this testing. The following are some of the main challenges to overcome.

- No user interface - In data warehouse testing, no user interface is present but only data and its relationships are there. In order to test this type of data, the ability to look at data, validate data processing rules, and analyze final data output are required. Consequently, knowledge of database query languages like SQL is essential for testers to do this accurately, where traditional manual testing skills are not enough.
- Huge volume of data - Millions of transactions can be happening every day. It is a challenge to verify the extraction, transformation and loading of that data in the real-time environment as the code is updated.

- Variety of sources - Typically, a wide variety of systems feed daily transactional data to a data warehouse. Some of the data may even come from systems used in cloud computing or hosted by a third party. Similarly, the format and content of the data will vary. It is often a huge challenge to merge the data while making sure that everything gets processed consistently and in relation to each other.
- Bad or missing data - The information collected from the various source systems may not be complete, may have many special cases requiring exception processing, or may be of poor quality generally.
- Non-static rules - The source systems will likely change over time as a result of release upgrades with attendant changes in data content and structure. There should be ways to cope with these changes without having to change the design of the data warehouse.

6.2 Different methods for ETL testing

There are two high-level approaches to ETL data validation, as follows.

- **Method I:** The data in the data sources is validated directly in the data warehouse. This approach validates that all the data in the source appears in the data warehouse according to the business rules. This approach does not validate the intermediate staging area and transformation processes between the source and data warehouse.
- **Method II:** The data is validated from the data sources through each step of the extract, including the final load in the data warehouse. Also, the data is validated at each transformation. For example, in the first stage, data is extracted from the file and checked for completeness. In a second stage, any unwanted/junk information is removed that is not needed to be processed. In a third stage, data translation is performed to clean it and make it uniform.

As in the second method, each step in the process needs to be verified. It is more time-consuming but makes it much easier to track down any problems that occur. On the other hand, the first method is less time-consuming, but any errors are more difficult to track down.

6.3 Areas of ETL testing

There are a lot of areas to be tested in the whole process of ETL.

- Is expected data coming from the source file? After the data has been loaded, a full inventory must be completed to ensure that the correct records made it into the right tables.
- Are all the records in the correct place? It is also important to confirm that the correct numbers of records are being processed and that each data field from each source loads in the right order.
- Check for duplicate data. It is important that a check for duplicates is performed as data moves across landing, staging, and finally into the data warehouse.
- Are all data transformation rules correct? It must be verified that the transformation rules include all possible cases and treat them correctly.
- Does system generate error logs? In case of a system crash or a power outage, it is important to verify that the system does not lose data or create duplicates if restarted. Key facts about each run should be logged for future analysis.
- Performance monitoring. It is important to track the performance of ETL operations over time to predict growth rates and increase system resources if necessary in advance of the development of bottlenecks.
- Is data being filtered correctly? Sometimes the data may be problematic and needs to be skipped for the time being while the processing carries on for the balance of the data. The system must log the rejected data and possibly set it aside for further investigation before attempting again to process it.

6.4 Tools

There are many commercial ETL tools available on the market. Some of them are described here.

1. SQL Server Integration Services (SSIS): SSIS tools connect and transform a variety of data sources.
2. Informatica PowerCenter: Informatica is a unified enterprise data integration platform for accessing, discovering and integrating data from virtually any business system, in any format.

6

3. OpenText Integration Center: This tool offers a data and content integration platform that unifies information that crosses application boundaries, consolidating and transforming data and content throughout the entire information ecosystem.
4. Cognos Data Manager: This tool is provided by IBM. It provides dimensional ETL capabilities for high-performance business intelligence.
5. SAP Data Services: This tool allows organizations to easily explore, extract, transform and deliver data anywhere, at any frequency. It ensures the integrity of the data, maximizes developer productivity, and accelerates data integration performance for all operational and analytic initiatives.

   SAP Data Services provide pervasive, open and extensible on-premise and cloud source and target support - structured, text, Big Data, social, spatial, SAP and non-SAP.
6. Oracle Warehouse Builder (OWB): This tool takes raw data, typically in different formats and different systems, and transforms it into high-quality information that is optimized for business reporting and analytics.

7

Testing Methods, Tools and Reporting on Analytics

Analytics is a multi-dimensional discipline. It involves extensive use of mathematics and statistics, as well as descriptive techniques and predictive models to gain valuable knowledge from data. Big Data analytics refers to the process of collecting, organizing and analyzing large sets of data to discover patterns and reporting useful information.

Firms may commonly apply analytics to business data, to describe, predict and improve business performance. Specifically, areas within analytics include predictive analytics, enterprise decision management, retail analytics, store assortment and stock-keeping unit optimization, marketing optimization and marketing mix modeling, web analytics, sales force sizing and optimization, price and promotion modeling, predictive science, credit risk analysis, and fraud analytics. Since analytics can require extensive computation, the algorithms and software used for analytics often employ the most current methods in computer science, statistics, and mathematics.

The value and insights that Big Data bring can only be realized by effective presentation and visualization of information for your business. Visual representations help you see patterns in the data that make it easier to make meaningful decisions. The data is usually displayed in one of the following forms.

- **Dashboards:** High level representations of key business performance indicators used to measure financial performance, internal operations, product innovation, and/or customer satisfaction
- **Data Mining:** Providing tools and a process for statistically analyzing large quantities of data to uncover patterns and predict trends
- **Reporting:** Including ad-hoc, operational, embedded, and production reports
- **Interactive analysis:** For multi-dimensional views of data through data aggregation and drill-down

7.1 Four Big Data reporting strategies

1. Performance Management

Performance management involves understanding the meaning of Big Data in company databases using predetermined queries and multidimensional analysis. The data used for this analysis is typically transactional, e.g. years of customer purchasing activity, or inventory levels and turnover. Ideally managers can ask questions, such as which are the most profitable customer segments, and get answers in real time that can be used to help make short-term business decisions and longer-term plans.

7

Most business intelligence tools today provide a dashboard capability. The user, often the manager or analyst, can choose which queries to run, and can filter and rank the report output by certain dimensions (e.g. region), as well as drill down/up on the data. Multiple types of reports and graphs make it easy for managers to look at trends.

2. Data Exploration

Data exploration makes heavy use of statistics to experiment and get answers to questions that managers might not have thought of previously. This approach leverages predictive modeling techniques to predict user behavior based on their previous business transactions and preferences. Cluster analysis can be used to segment customers into groups based on similarity in ways that may not be obvious a priori. Once these groups are discovered, managers can perform targeted actions such as customizing marketing messages, differentiating services, and cross/upselling to each unique group.

3. Social Analytics

Social analytics measure the vast amount of non-transactional data that exists today. Much of this data exists on social media platforms, such as conversations and reviews on Facebook, Twitter and Yelp.

Social analytics measure three broad categories: awareness, engagement and word-of-mouth or reach. Awareness looks at the exposure or "mentions" of social content and often involves metrics, such as the number of video views and the number of followers or community members. Engagement measures the level of activity and interaction among platform members, such as the frequency of user-generated content. Social analyzers need a clear understanding of what they are measuring. For example, a viral video that has been viewed 10 million times is a good indicator of high awareness, but it is not necessarily a good measure of engagement and interaction.

4. Decision Science

Decision science involves experiments and analysis of non-transactional data, such as consumer¬generated product ideas and product reviews, to improve the decision-making process.

Decision scientists, in conjunction with community feedback, determine the value, validity, feasibility and fit of these ideas and eventually report on if or how they plan to put these ideas into action.

Organizations need effective testing in all these domains.

7.2 Methodology for report testing

Big Data analytics solutions are built to report and analyze data from data warehouses. They can vary in size and complexity depending on the needs of the business, underlying data stores, number of reports and number of users. The key focus is on validating layout format as per the design mock-up, style sheets, prompts and filter attributes and metrics on the report. Verification of drilling, sorting and export functions of the reports in the Web environment is also done. Data generated on the reports should be correct as per business logic. The test team needs to target the lowest granularity that is present in the data warehouse, understanding each report and the linkages of every field displayed in the report with the schema (star and snow). Tracing its origin back to the source system is a big challenge and a time-consuming process.

Building such a reporting solution isn't trivial, and testing the solution for release sign-off isn't, either. One of the challenges a test manager faces is what to test, or, more precisely, what not to test. While in reality every report can be tested, doing so increases the project budget and timeframes.

When it comes to the actual validation of the data in a data warehouse, the testing approach is well- defined and time-tested. The tester has the option of using either a sampling strategy or performing exhaustive verification. Given the large data volumes involved, the decision as to which strategy to take for any given dataset is best made using a "risk-based testing" technique to assess and prioritize the test scope to consider the associated business risk that could arise from errors in the data. The primary purpose of this technique is to perform effective testing within the limited timeframe and limited resources available for testing by working through the prioritized list in the order of high to low risk scope items.

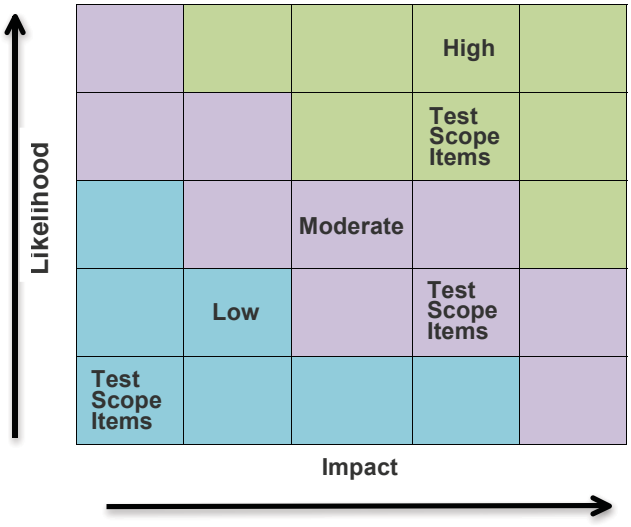
7

Identifying the test scope and carrying out the risk assessment are two significant activities that should collectively be performed by the project team members. It is important to include a cross¬section of the project team to get a good balance of experience and knowledge of the systems and business. Each item should be assessed for complexity and business risk factors (e.g. data sensitivity and impact on reputation). This information will be used when estimating the required testing effort.

These should be quantified, considering the likelihood of the risk occurring and the impact of the risk in terms of cost, time and quality, and rating them on a scale of 1 to 5. The rating scale of 1 for the likelihood represents least likely and 5 represents most likely. The rating scale of 1 for impact represents least significant impact and 5 represents the most significant. Risk exposure is calculated by multiplying the likelihood and the impact.

Once the risk exposure assessment is completed, the team categorizes the test scope items into high, moderate or low-risk groups, and determines an appropriate combination of testing techniques, such as basic unit testing, partial system testing or full user acceptance testing. Each testing type (i.e. unit testing, system testing and user acceptance testing) can be broken down into basic, partial and full testing.

An example of the system risk assessment is shown below.



1. Validating the Dashboard Report Model

The design of these reports is critical understanding for the tester. Insight such as what content uses which information maps, what ranges are leveraged in which indicators, and where interactions exist between indicators is required to build a full suite of test cases. If any measures are defined in the report itself, these should be verified as accurate - but all other data elements that are pulled straight from the table/information map should already have been validated from one of the above two sections.

It is essential to address:

- Understanding each report and the mapping of every field displayed in the report with the schema, tracing its origin back to the source system
- Verification of the GUI: layout format, style sheets, prompts and filters attributes and metrics on the report as per mock ups
- Verification of drilling (drill down, drill through), sorting and export functions of the reports, including testing of different data sets (different regions, periods of time) and usability testing
- Verification of the data at the lowest granularity level that is present in the data warehouse against the report using data validation
- Verification of report format and content by appropriate end users
- Verification of accuracy and completeness of the scheduled reports
- Analytics are working
- Previewing and/or exporting of reports to different formats, such as spreadsheet, pdf, html and email to ensure they display accurate and consistent data
- Print facility, where applicable, produces expected output
- Where graphs and data exist in tabular format, both should reflect consistent data
- Table analysis
  - Business rule analysis
  - Functional dependency
  - Column set analysis

7

2. Checking the Source Record Count and Target Record Count

- Verifying the source record count and target record count match
- Data consistency validation

3. Authentication Testing

Authentication testing is the process used to verify that only authorized users can access the system.

4. Data Level Security

Authorization is the process used to verify that a user has been granted sufficient privileges to perform the requested action.

Data level security validation means users will be able to see only particular data for the given permission. For example, both the Eastern and Western region sales managers will be seeing the same reports but the data visible to them in the reports will be Eastern and Western region sales data respectively. Object level security means to validate whether the particular user is able to access the particular dashboard or folder, etc.

5. Bursting the Reports

Bursting the reports means distributing the reports based on the regions, e.g. if there are four regional reports, ensure that the reports are properly distributed to (and only to) the appropriate parties.

6. Buzz Matrix Validation

Alerting also needs to be validated, i.e. that the alerts are produced and delivered when any variables being tracked cross their alerting thresholds.

7. User Acceptance Criteria

Users typically have an existing legacy mechanism to verify if what is displayed in the new solution makes sense. One should dig into this and understand how the end users built the project acceptance criteria. Testers should challenge the assumptions made by the business community in deriving the acceptance criteria. This activity helps get an end-user perspective built into the testing efforts from early on.

8. Time Series Functions Validation

Time series functions provide the ability to compare business performance with previous time periods, allowing the analysis of data that spans multiple time periods. These will also need to be verified for correct functioning.

9. End-to-End Testing

Although individual components of the Big Data system may be behaving as expected, there may be issues in how those components interact dynamically. Thus, execution and validation of end-to-end runs are recommended. Along with data reconciliation discrepancies, issues such as resource contention or deadlocks might surface. The end-to-end runs will further help in ensuring the data quality and performance acceptance criteria are met.

7.3 Apache Falcon

Falcon simplifies the development and management of data processing pipelines with a higher layer of abstraction, taking the complex coding out of data processing applications by providing out-of-the-box data management services. This simplifies the configuration and orchestration of data motion, disaster recovery and data retention workflows.

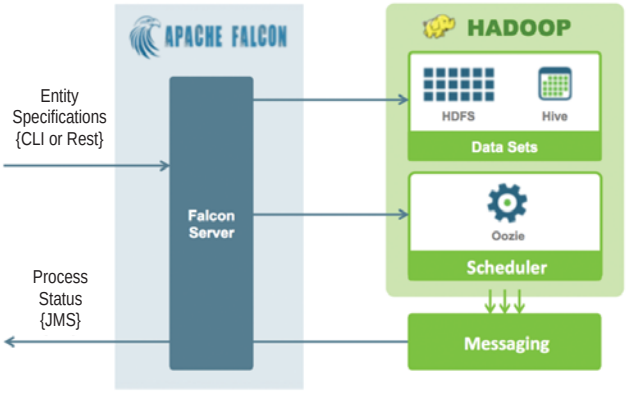


Chart courtesy of Daystrom.com

Falcon runs as a standalone server as part of your Hadoop cluster. Falcon is a data feed processing and management system aimed at bringing more rigorous data governance to Hadoop.

7

Apache Falcon meets enterprise data governance needs in three areas.

1. Centralized data life cycle management

- Centralized definition and management of pipelines for data ingest, process and export
- Disaster readiness and business continuity
- Out-of-the-box policies for data replication and retention
- End-to-end monitoring of data pipelines

2. Compliance and audit

- Visualize data pipeline lineage
- Track data pipeline audit logs
- Tag data with business metadata

3. Database replication and archival

- Replication across on-premises and cloud-based storage targets: Microsoft Azure and Amazon S3
- Data lineage with supporting documentation and examples
- Heterogeneous storage tiering in HDFS
- Definition of hot/cold storage tiers within a cluster

Falcon provides a single interface to orchestrate data life cycle across clusters. It provides the key services data processing applications needed so sophisticated data life cycle management can easily be added to Hadoop applications. Complex data-processing logic is handled by Falcon instead of hard¬coded in apps. There is faster development and higher quality for ETL, reporting and other data- processing apps on Hadoop.

8 Testing Methods, Tools and Reporting on Performance and Failover Testing

8.1 Performance testing

Through performance testing in Big Data applications, we can achieve the following objectives.

1. Obtain and understand the actual performance under load of Big Data applications, such as response time, maximum online user data capacity size, and maximum processing capacity
2. Identify performance limits and conditions that can cause performance problems
3. Gain insight that could be used to optimize parameters that influence performance (e.g. hardware configuration, software configuration and application code)

Performance testing should be conducted by setting up large volumes of data with an environment similar to production. In addition to the usual performance metrics such as job completion time, data throughput and memory utilization, the following should also be noted and/or tracked.

- Data storage
- Commit logs
- Concurrency
- Caching
- JVM parameters
- MapReduce configuration
- Message queue configuration

8.2 Failover testing

Hadoop installations typically have dozens or hundreds, if not thousands of nodes, each with four or more disk drives. Some disk drives or even entire nodes will fail every week. HDFS architecture is designed to detect these failures and automatically recover to proceed with processing despite these failures. Failover testing validates the recovery process and ensures data processing continues correctly when switched to other data nodes.

As Hadoop involves radically different architectures from traditional information-processing systems, verification of the design and configuration is critical. The NameNode, in particular, remains a single point of failure and should be maintained in a high availability configuration and submitted to appropriate failover testing.

8

8.3 Methods and tools

While it is important to have some testing of newly installed systems, many of the important performance tests that need to be carried out on Big Data systems will be more realistic with a populated installation. Virtual machines and the ability to "snapshot" and roll back to a known hard disk state are very useful tools for this kind of operation in smaller scale testing, but are of limited use for Big Data tests given the scale of storage involved. In order to overcome this problem, we can make use of various techniques to pre-populate data into the system before executing new tests.

Some possible techniques are:

- Static Installation - Configuring the software against a static, "read-only" installation can be useful for testing query performance against a known dataset for performance benchmarking.
- Backup/Restore - Using the backup/restore and disaster recovery features of the system to restore an existing installation in a known state. As well as being an excellent way of restoring an installation, this also helps to test the backup and recovery mechanisms themselves through real use.
- Data Replication - If the software supports quick import or replication methods, we can leverage these to populate an installation with bulk data far more quickly than through the standard importing interfaces. For example, we utilize a product feature to support geographic replication of data across servers to bulk insert prebuilt data into archives far more rapidly than the standard import process. Once we have reached a suitable capacity, we can then switch to standard importing to test performance.
- Rolling Installation - Having an installation that is in a "rolling state," whereby tests import new data and archive out old data at a continuous rate. This allows for testing at a known capacity level in a realistic data life cycle without the lead time of building up an archive for each iteration, with the additional benefit of boosting our version of compatibility testing with an installation that has been running over a long period of time with many software versions.

The following industry-standard tools can be used to carry out performance and failover testing.

- Performance test tools
  - YCSB (Yahoo Cloud Serving Benchmark), SandStorm, JMeter
- Monitoring tools
  - Nagios, Zabbix, Ganglia, JMX utilities
- Diagnostic tools
  - visualVM, AppDynamics, Compuware

8.3.1 Jepsen

Modern software systems are composed of dozens of components that communicate over an asynchronous, inherently unreliable network. Understanding the reliability of a dynamic distributed system requires careful analysis of the reliability of the network itself. The root issue is essentially one of shared state: a set of nodes that must exchange information through network connections, e.g. "Did I like that post?"; "Was my write successful?"; "Will you thumbnail my image?"; "How much is in my account?"

It's not just the network. Garbage collection pauses in the JVM or heavy activity by your neighbor in a server running on the cloud can also manifest in slowdowns that are virtually indistinguishable from a network partition. Even in the best-managed data centers, things go wrong; disks fail, switches malfunction, power supplies get shorted out, RAM modules die, and a distributed system that runs on a large scale should strive to work around such issues as much as possible.

Here are some examples of the kind of questions that need to be addressed.

- How are the states of the system causally connected?
- Will the system survive the total failure of one or more nodes?
- Will the data survive a complete power failure?
- Will the data survive the destruction of an entire data center?
- What if communications lines between components are severed?
- What happens if data communications degrade badly so some messages are delivered but not all?

8

Jepsen is a tool that simulates network partitions and tests how distributed data stores behave under them. Jensen will cut off one or more nodes from talking to other nodes in a cluster while continuing to insert, update or look up data during the partition, as well as after the partition heals, to find if they lose data, read inconsistent data or become unavailable.

Jepsen works by setting up the data store under test on five different hosts (typically Linux Containers on a single host for simplicity). It creates a client, for the data store under test, pointing to each of the five nodes to send requests.

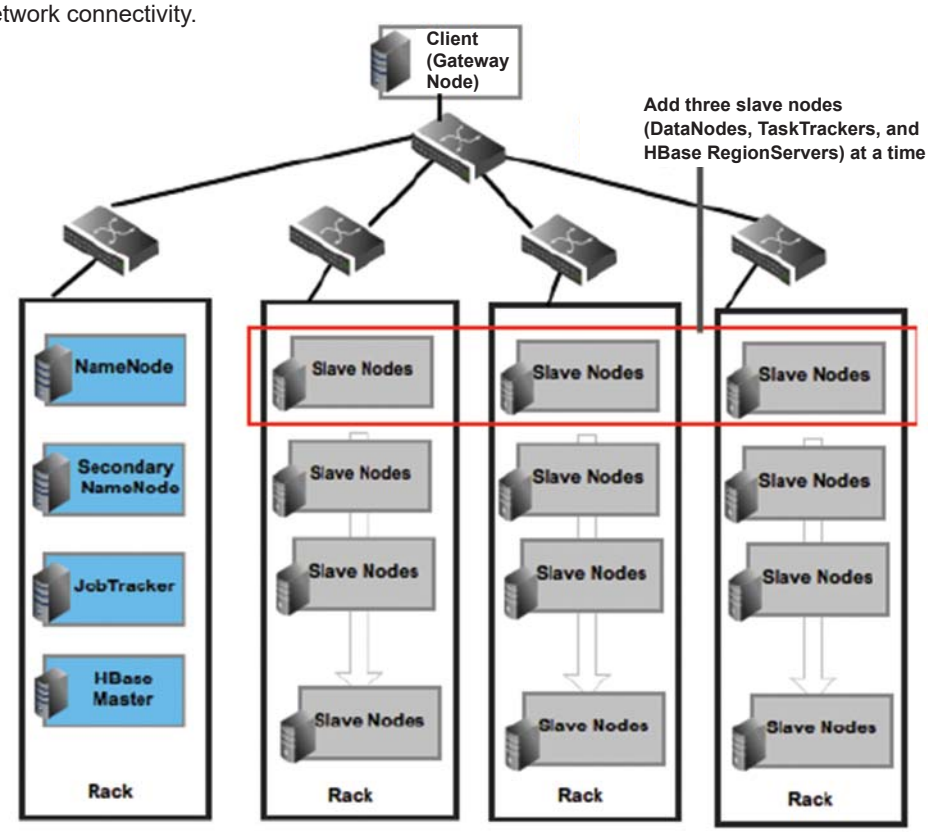
It also creates one or more copies of a special client called a "Nemesis," which instead of talking to a data store, wreaks havoc in the cluster by, for example, cutting links between nodes using IP tables. Then it proceeds to make requests concurrently against different nodes while alternately partitioning and healing the network. At the end of the test run, it heals the cluster, waits for the cluster to recover, and then verifies whether the intermediate and final state of the system is as expected.

The most current update can be found at
https://aphyr.com/
Source code can be obtained at
https://github.com/aphyr/jepsen

9 Infrastructure Setup, Design, and Implementation

Hadoop workloads tend to vary a lot and it takes experience to correctly anticipate the amounts of storage, processing power and inter-node communication that will be required for different kinds of jobs.

For a Hadoop (or HBase) cluster, it is critical to accurately predict the size, type, frequency and latency of analysis jobs to be run. Begin small and gain experience by measuring actual workloads during a pilot project. This way you should be able to scale the pilot environment without making significant changes to the existing servers, software, deployment strategies or network connectivity.



Hortonworks.com NOTE: DataNodes, TaskTrackers, and RegionServers are typically co-deployed

9

Hadoop and HBase clusters have two types of machines.

1. Masters (the HDFS NameNode, the MapReduce JobTracker, and the HBase Master)
2. Slaves (the HDFS DataNodes, the MapReduce TaskTrackers, and the HBase RegionServers)

The DataNodes, TaskTrackers and HBase RegionServers are co-located or co-deployed for optimal data locality. In addition, HBase requires the use of a separate component (ZooKeeper) to manage the HBase cluster.

9.1 Hardware selection for master nodes (NameNode, JobTracker, HBase Master)

The master nodes, being unique, have significantly different storage and memory requirements from the slave nodes.

Storage options

Using dual NameNode servers is recommended - one primary and one secondary. Both NameNode servers should have highly reliable storage for their namespace storage and edit-log journaling. Typically, hardware RAID and/or reliable network storage are justifiable options. The master servers should have at least four redundant storage volumes, some local and some networked, but each can be relatively small (typically 1TB).

Memory sizing

The amount of memory required for the master nodes depends on the number of file system objects (files and block replicas) to be created and tracked by the NameNode. 64 GB of RAM supports approximately 100 million files. Some sites are now experimenting with 128 GB of RAM, for even larger namespaces.

Processors

The NameNodes and their clients are very "chatty". We therefore recommend providing 16 or even 24 CPU cores to handle messaging traffic for the master nodes.

9.2 Hardware selection for slave nodes (DataNodes, TaskTrackers and RegionServers)

Typically, dual-socket servers are optimal for Hadoop deployments. For medium to large clusters, using these servers is a best choice over the entry-level servers because of the load-balancing and parallelization capabilities. In terms of density, it is advisable to select server hardware that fits into a small number of rack units. Typically, 1U or 2U servers are used in 19" racks or cabinets.

Storage options

For general-purpose Hadoop applications, we recommend using a relatively large number of hard drives (typically eight to twelve SATA LFF drives) per server.

It is important to note that Hadoop is storage intensive and seek efficient, but does not require fast and expensive hard drives. If your workload pattern is not I/O intensive, it is safe to add only four or six disks per node. Note that power costs are proportional to the number of disks and not to terabytes. We therefore recommend that you add disks for storage and not for seeks.

Memory sizing

In a Hadoop cluster, it is critical to provide sufficient memory to keep the processors busy without swapping and without incurring excessive costs for non-standard motherboards. Depending on the number of cores, your slave nodes typically require 24 GB to 48 GB of RAM for Hadoop applications. For large clusters, this amount of memory sufficiently provides extra RAM (approximately 4 GB) for the Hadoop framework and for your query and analysis processes (HBase and/or Map/Reduce).

9

Power consideration

Power is a major concern when designing Hadoop clusters. Instead of purchasing the biggest and fastest nodes, it is important to analyze the power utilization for the existing hardware. We observed huge savings in pricing and power by avoiding the fastest CPUs, redundant power supplies, etc. For slave nodes, a single power supply unit (PSU) is sufficient.

Processors

Although it is important to understand your workload pattern, for most systems we recommend using medium clock speed processors with less than two sockets. For most workloads, the extra performance per node is not cost-effective. For large clusters, use at least two quad core CPU for the slave machines.

9.3 Infrastructure setup key points

1. It is not a general practice to deploy Hadoop nodes as virtual machines for many different reasons, primarily for I/O performance and other shared properties.
2. For performance and scale benefits, you should run the NameNode, Secondary NameNode and/or Checkpoint Node, Job Tracker and the HBase (or any DB) Master as dedicated standalone nodes where these processes are not shared with other Hadoop processes on the same server.
3. It is critical to distribute the network workload evenly across the cluster. It is important to understand the use case and communication patterns across the Hadoop components, such as the replication factors, NameNode and Job Tracker placements, and cluster size dependencies on the mappers to reducers ratios. For example, unless you are in a small test environment, it is asking for trouble to deploy Hadoop in production with a single reducer process.
4. In general, storage for Hadoop deployments is always local disks and/or JBOD attached. SAN attached Hadoop clusters would not work. HDFS is capable of handling huge dataset sizes distributed over large clusters of compute, so tuning HDFS is another factor that drives Hadoop performance. For example, increasing the HDFS block size to 128 MB (from default 64 MB) is a recommended best practice. Also, backing up critical Hadoop configuration files like the journal, checkpoint file, etc., should be built into the operations rules.
5. Hadoop deployment requirements, use cases, tools and application interfaces are going to vary from one environment to other, so no one design is going to applicable in all situations.
6. The Hadoop app/dev, testing teams and infrastructure team (facilities/server/network) must have joint planning meetings to strategize on the goals.

Conclusion

In conclusion, if an organization applies the right test strategies and follows best practices, it will improve Big Data testing quality, which will help to identify defects in early stages and reduce overall cost.

To be successful, Big Data testers have to learn the components of the Big Data ecosystem from scratch. Since the market has created fully automated testing tools for Big Data validation, the tester has no other option but to acquire the same skill set as the Big Data developer in the context of leveraging Big Data technologies like Hadoop. This requires a tremendous mindset shift for both testers and testing units within organizations. In order to be competitive, companies should invest in Big Data-specific training needs and developing the automation solutions for Big Data validation.

About QA Consultants

QA Consultants is an award-winning provider of software testing and quality assurance solutions. We are the trusted testing company for businesses, government departments and institutions. Over the last 20 years we have successfully delivered 5,000+ mission-critical projects in the private, public and not-for-profit sectors. Within those sectors, QA Consultants has extensive testing experience and depth in the following industries: automotive, banking, consumer goods, insurance, media and advertising, public affairs, retail, technology, and travel and tourism.

In the ongoing effort to maintain our status as leader in research and innovation, and with the continuing support of the National Research Council of Canada, QAC founded, developed and built a large facility in Toronto devoted solely to testing. The Test Factory™ is a continuous quality test lab incorporating a precise combination of intelligence applied to advanced levels of automation.

The Test Factory pairs the skilled labour and expertise of our staff (mostly computer science graduates) with our proprietary testing methodologies. This unique blend provides our clients with unparalleled and superior quality service. QAC's 30,000-square-foot testing facility delivers onshore quality and performance at low offshore prices. The Test Factory™ alone or in partnership with our Managed Consulting Services and On Demand Testing™ delivers highly effective testing and QA solutions to our wide range of clients.