RESEARCH ARTICLE OPEN ACCESS

Analysis of Various Tools in Big Data Scenario

Amarbir Singh[1], Palwinder Singh[2]
[1,2](Department of Computer Science, Guru Nanak Dev University, Amritsar)

----------**********----------

Abstract:

Big data is a term which refers to those data sets or combinations of data sets whose volume, complexity, and rate of growth make them difficult to be captured, managed, processed or analysed by traditional tools and technologies. Big data is relatively a new concept that includes huge quantities of data, social media analytics and real time data. Over the past era, there have been a lot of efforts and studies are carried out in growing proficient tools for performing various tasks in big data. Due to the large and complex collection of datasets it is difficult to process on traditional data processing applications. This concern turns to be further mandatory for producing various tools in big data. In this survey, a various collection of big data tools are illustrated and also analysed with the salient features.

Keywords **— Big data, Big data analytics, MapReduce, Data analysis, Hadoop.**

----------**********----------

I. INTRODUCTION

Big data is a vast quantity of data which extracts values by the process of capturing and analysis, this can be possible by innovative architectures and technologies. Nowadays from the platform of traffic management and tracking personal devices such as Mobile phones are useful for position specific data which emerges as novel bases of big data. Mainly the Big data have developed to increase the use of data demanding technologies for it. By using prevailing traditional techniques it is very challenging to achieve effective analysis of the huge size of data as Advances in information technology and its widespread growth in several areas of business, engineering, medical and scientific studies are resulting in information/data explosion. Meanwhile, on the market, big data have become the latest imminent technology, which can serve vast profits to the business organizations. This becomes essential because it contains several issues and challenges related in bringing and adapting, which need to be understood in this technology. The concept of big data deals with the datasets which continues to develop rapidly whereas that becomes tough to handle them by using the current concepts and tools in database management. Data capture, sharing, analytics, search, storage, visualization, etc., is the related difficulties in big data. Many challenges can be forwarded due to the several properties of big data like volume, variety, velocity, variability, value and complexity. Volume represent the size of the data how the data is large. Variety makes the data too big. Data comes from the various sources that can be of structured, unstructured and semi-structured type. Velocity defines the motion of the data streaming and speed of data processing. Variety represents the unreliability inherent in some sources of data [1]. Variability refers to the variation in the data flow rates. Big data are often characterized by relatively “low value density”, that is the data received in the original form usually has a low value relative to its volume. However, a high value can be obtained by analyzing large volumes of such data. Complexity refers to the fact that big data are generated through a myriad of sources. This imposes a critical challenge: the need to connect, match, cleanse and transform data received from different sources.

On the other hand scalability, real-time analytics, unstructured data, fault tolerance, etc., is the several challenges included in huge data management. Obviously the amount of data stored

in various sectors can vary in the data stored and created, i.e., images, audio, text information etc., from one industry to another. The main aim of big data analytics is to utilize the advanced analytic techniques besides very huge, different datasets which contain diverse sizes from terabytes to zettabytes and diverse types such as structured or unstructured and batch or streaming. Big data is useful for data sets where their size or type is away from the capability of traditional relational databases for capturing, managing and processing the data with low-latency. Thus the out coming challenges tend to the occurrence of powerful big data tools. From the practical perspective, the graphical interface used in the big data analytics tools leads to be more efficient, faster and better decisions which are massively preferred by analysts, business users and researchers [2].

II. IMPORTANCE OF BIG DATA

Big data enables organizations to accomplish several objectives [3] such as it helps to support real-time decisions and it includes various types of information that can be used in decision making. Apart from that big data helps to explore and analyze information and improve business outcomes and manage risk, now and in the future.

III. MAP-REDUCE

MapReduce is an algorithm design pattern that originated in the functional programming world. It consists of three steps. First, you write a mapper function or script that goes through your input data and outputs a series of keys and values to use in calculating the results. The keys are used to cluster together bits of data that will be needed to calculate a single output result. The unordered list of keys and values is then put through a sort step that ensures that all the fragments that have the same key are next to one another in the file. The reducer stage then goes through the sorted output and receives all of the values that have the same key in a contiguous block [4]. That may sound like a very roundabout way of building your algorithms, but its prime virtue is that it removes unplanned random accesses, with all scattering and gathering handled in the sorting phase. Even on single machines, this boosts performance, thanks to the increased locality of memory accesses, but it also allows the process to be split across a large number of machines easily, by dealing with the input in many independent chunks and partitioning the data based on the key. Hadoop is the best-known public system for running MapReduce algorithms, but many modern databases, such as MongoDB, also support it as an option. It's worthwhile even in a fairly traditional system, since if you can write your query in a MapReduce form, you'll be able to run it efficiently on as many machines as you have available.In the traditional relational database world, all processing happens after the information has been loaded into the store, using a specialized query language on highly structured and optimized data structures. The approach pioneered by Google, and adopted by many other web companies, is to instead create a pipeline that reads and writes to arbitrary file formats, with intermediate results being passed between stages as files, with the computation spread across many machines. Typically based around the Map-Reduce approach to distributing work, this approach requires a whole new set of tools, which are describe below.

A. Hadoop

Originally developed by Yahoo! as a clone of Google's MapReduce infrastructure, but subsequently open sourced, Hadoop takes care of running your code across a cluster of machines. Its responsibilities include chunking up the input data, sending it to each machine, running your code on each chunk, checking that the code ran, passing any results either on to further processing stages or to the final output location, performing the sort that occurs between the map and reduce stages and sending each chunk of that sorted data to the right machine, and writing debugging information on each job's progress, among other things. As you might guess from that list of requirements, it's quite a complex system, but thankfully it has been battle-tested by a lot of users [5]. There's a lot going on under the hood, but most of the time, as a developer, you only have to supply the code and data, and it just works. Its popularity also means that there's a large ecosystem of related tools, some that making writing individual processing steps easier, and others that orchestrate more complex jobs that

require many inputs and steps. As a novice user, the best place to get started is by learning to write a *streaming job* in your favorite scripting language, since that lets you ignore the gory details of what's going on behind the scenes. As a mature project, one of Hadoop's biggest strengths is the collection of debugging and reporting tools it has built in. Most of these are accessible through a web interface that holds details of all running and completed jobs and lets you drill down to the error and warning log files.

B. Hive

With Hive, you can program Hadoop jobs using SQL. It's a great interface for anyone coming from the relational database world, though the details of the underlying implementation aren't completely hidden. You do still have to worry about some differences in things like the most optimal way to specify joins for best performance and some missing language features. Hive does offer the ability to plug in custom code for situations that don't fit into SQL, as well as a lot of tools for handling input and output. To use it, you set up structured tables that describe your input and output, issue load commands to ingest your files, and then write your queries as you would in any other relational database [6]. Do be aware, though, that because of Hadoop's focus on largescale processing, the latency may mean that even simple jobs take minutes to complete, so it's not a substitute for a real-time transactional database.

C. Pig

The Apache Pig project is a procedural data processing language designed for Hadoop. In contrast to Hive's approach of writing logic-driven queries, with Pig you specify a series of steps to perform on the data. It's closer to an everyday scripting language, but with a specialized set of functions that help with common data processing problems. It's easy to break text up into component ngrams, for example, and then count up how often each occurs. Other frequently used operations, such as filters and joins, are also supported. Pig is typically used when your problem (or your inclination) fits with a procedural approach, but you need to do typical data processing operations, rather than general purpose calculations [7]. Pig has been described as "the duct tape of Big Data" for its usefulness there, and it is often combined with custom streaming code written in a scripting language for more general operations.

D. Cascading

Most real-world Hadoop applications are built of a series of processing steps, and Cascading lets you define that sort of complex workflow as a program. You lay out the logical flow of the data pipeline you need, rather than building it explicitly out of Map-Reduce steps feeding into one another. To use it, you call a Java API, connecting objects that represent the operations you want to perform into a graph. The system takes that definition, does some checking and planning, and executes it on your Hadoop cluster [8]. There are a lot of built-in objects for common operations like sorting, grouping, and joining, and you can write your own objects to run custom processing code.

E. Cascalog

Cascalog is a functional data processing interface written in Clojure. Influenced by the old Datalog language and built on top of the Cascading framework, it lets you write your processing code at a high level of abstraction while the system takes care of assembling it into a Hadoop job. It makes it easy to switch between local execution on small amounts of data to test your code and production jobs on your real Hadoop cluster [9]. Cascalog inherits the same approach of input and output taps and processing operations from Cascading, and the functional paradigm seems like a natural way of specifying data flows. It's a distant descendant of the original Clojure wrapper for Cascading, cascading-clojure.

F. Mrjob

Mrjob is a framework that lets you write the code for your data processing, and then transparently run it either locally, on Elastic MapReduce, or on your own Hadoop cluster. Written in Python, it doesn't offer the same level of abstraction or built-in operations as the Java-based Cascading. The job specifications are defined as a series of map and reduce steps, each implemented

as a Python function [10]. It is great as a framework for executing jobs, even allowing you to attach a debugger to local runs to really understand what's happening in your code.

G. Caffeine

Even though no significant technical information has been published on it, I'm including Google's Caffeine project, as there's a lot of speculation that it's a replacement for the MapReduce paradigm. From reports and company comments, it appears that Google is using a new version of the Google File System that supports smaller files and distributed masters. It also sounds like the company has moved away from the batch processing approach to building its search index, instead using a dynamic database approach to make updating faster. There's no indication that Google's come up with a new algorithmic approach that's as widely applicable as MapReduce, though I am looking forward to hearing more about the new architecture.

H. S4

Yahoo! initially created the S4 system to make decisions about choosing and positioning ads, but the company open sourced it after finding it useful for processing arbitrary streams of events. S4 lets you write code to handle unbounded streams of events, and runs it distributed across a cluster of machines, using the ZooKeeper framework to handle the housekeeping details. You write data sources and handlers in Java, and S4 handles broadcasting the data as events across the system, load-balancing the work across the available machines. It's focused on returning results fast, with low latency, for applications like building near real-time search engines on rapidly changing content [11]. This sets it apart from Hadoop and the general MapReduce approach, which involves synchronization steps within the pipeline, and so some degree of latency. One thing to be aware of is that S4 uses UDP and generally offers no delivery guarantees for the data that is passing through the pipeline. It usually seems possible to adjust queue sizes to avoid data loss, but it does put the burden of tuning to reach the required level of reliability on the application developer.

I. MapR

MapR is a commercial distribution of Hadoop aimed at enterprises. It includes its own file systems that are a replacement for HDFS, along with other tweaks to the framework, like distributed name nodes for improved reliability. The new file system aims to offer increased performance, as well as easier backups and compatibility with NFS to make it simpler to transfer data in and out. The programming model is still the standard Hadoop one; the focus is on improving the infrastructure surrounding the core framework to make it more appealing to corporate customers[12].

J. Acunu

Like MapR, Acunu is a new low-level data storage layer that replaces the traditional file system, though its initial target is Cassandra rather than Hadoop. By writing a kernel-level key/value store called Castle, which has been open-sourced, the creators are able to offer impressive speed boosts in many cases. The data structures behind the performance gains are also impressive. Acunu also offers some of the traditional benefits of a commercially supported distribution, such as automatic configuration and other administration tools.

K.Flume

One very common use of Hadoop is taking web server or other logs from a large number of machines, and periodically processing them to pull out analytics information. The Flume project is designed to make the data gathering process easy and scalable, by running agents on the source machines that pass the data updates to collectors, which then aggregate them into large chunks that can be efficiently written as HDFS files. It's usually set up using a command-line tool that supports common operations, like tailing a file or listening on a network socket, and has tunable reliability guarantees that let you trade off performance and the potential for data loss.

L.Kafka

Kafka is a comparatively new project for sending large numbers of events from producers to consumers. Originally built to connect LinkedIn's

website with its backend systems, it's somewhere between S4 and Flume in its functionality. Unlike S4, it's persistent and offers more safeguards for delivery than Yahoo!'s UDP-based system, but it tries to retain its distributed nature and low latency [13]. It can be used in a very similar way to Flume, keeping its high throughput, but with a more flexible system for creating multiple clients and an underlying architecture that's more focused on parallelization. Kafka relies on ZooKeeper to keep track of its distributed processing.

M. Azkaban

The trickiest part of building a working system using these new data tools is the integration.
The individual services need to be tied together into sequences of operations that are triggered by your business logic, and building that plumbing is surprisingly time consuming. Azkaban is an open source project from LinkedIn that lets you define what you want to happen as a job flow, possibly made up of many dependent steps, and then handles a lot of the messy housekeeping details. It keeps track of the log outputs, spots errors and emails about errors as they happen, and provides a friendly web interface so you can see how your jobs are getting on. Jobs are created as text files, using a very minimal set of commands, with any complexity expected to reside in the
Unix commands or Java programs that the step calls.

N. Oozie

Oozie is a job control system that's similar to Azkaban, but exclusively focused on Hadoop. This isn't as big a difference as you might think, since most Azkaban uses I've run across have also been for Hadoop, but it does mean that Oozie's integration is a bit tighter, especially if you're using the Yahoo! distribution of both. Oozie also supports a more complex language for describing job flows, allowing you to make runtime decisions about exactly which steps to perform, all described in XML files [14]. There's also an API that you can use to build your own extensions to the system's functionality. Compared to Azkaban, Oozie's interface is more powerful but also more complex, so which you choose should depend on how much you need Oozie's advanced features.

O. Greenplum

Though not strictly a NoSQL database, the Greenplum system offers an interesting way of combining a flexible query language with distributed performance. Built on top of the Postgres open source database, it adds in a distributed architecture to run on a cluster of multiple machines, while retaining the standard SQL interface. It automatically shards rows across machines, by default based on a hash of a table's primary key, and works to avoid data loss both by using RAID drive setups on individual servers and by replicating data across machines. It's normally deployed on clusters of machines with comparatively fast processors and large amounts of RAM, in contrast to the pattern of using commodity hardware that's more common in the web world.

IV. CONCLUSION

Early interest in big data analytics focused primarily on business and social data sources, such as e-mail, videos, tweets, Facebook posts, reviews, and Web behavior. The scope of interest in big data analytics is growing to include data from intelligent systems, such as in-vehicle infotainment, kiosks, smart meters, and many others, and device sensors at the edge of networks—some of the largest-volume, fastest-streaming, and most complex big data. So in order to handle the growing complexity this paper provides an in-depth analysis of various tools which are used in current big data scenario.

ACKNOWLEDGMENT

Authors wish to thank Guru Nanak Dev University, Amritsar for the guidance and Support.

REFERENCES

[1] http://www01.ibm.com/software/in/data/bigdata/Mark Troester, "Big Data Meets Big Data Analytics", retrieved 10/02/14, 2013.
[2] http://www.slideshare.net/HarshMishra3/harsh-big-data-seminar-report Garlasu, D.; Sandulescu, V. ; Halcu, I. ; Neculoiu, G.,"A Big Data implementation based on Grid Computing", Grid Computing, 2013.
[3] http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data.
[4] Sagiroglu, S.; Sinanc, D.,"Big Data: A Review" , 20-24 May 2013.
[5] Mukherjee, A.; Datta, J.; Jorapur, R.; Singhvi, R.; Haloi, S.; Akram, W., "Shared disk big data analytics with Apache Hadoop", 18-22 Dec.,2012.
[6] http://dashburst.com/infographic/big-data-volume.variety-velocity/
[7] http://www01.ibm.com/software/in/data/bigdata/Mark Troester (2013), "Big Data Meets Big Data Analytics", www.sas.com/resources/.../WR46345.pdf, retrieved 10/02/14.

[8] Chris Deptula, “With all of the Big Data Tools, what is the right one for me”,www.openbi.com/blogs/chris%20Deptula,retrieved 08/02/14.
[9] Brian Runciman(2013), “IT NOW Big Data Focus, AUTUMN 2013”, www.bcs.org, retrieved 03/02/14.
[10] Neil Raden (2012), “ Big Data Analytics Architecture”,www.teradata.com/ Big-Data Analytics”, retrieved 15/03/14.
[11] K. Bakshi, "Considerations for Big Data: Architecture and Approach", Aerospace Conference IEEE, Big Sky Montana, March 2012
[12] Sagiroglu, S.; Sinanc, D. ,(20-24 May 2013),”Big Data: A Review”
[13] Grosso, P.; de Laat, C.; Membrey, P., ” Addressing big data issues in Scientific Data Infrastructure” , 20-24 May 2013.
[14] Divyakant Agrawal, UC Santa Barba, Philip Bernstein, Microsoft Elisa Bertino,…(2012), “ Challenges and Opportunities with Big Data”, “www.cra.org/ccc/../BigdataWhitepaper.pd..., retrieved 15/03/14.