Introduction to Big-Data

Ms.N.D.Sonwane [1], Mr.S.P.Taley [2]

[1] *Assistant Professor, Computer Science & Engineering, DBACER, Maharashtra, India*
[2] *Assistant Professor, Information Technology, DBACER, Maharashtra, India*

ABSTRACT

Big data is a blanket term for the non-traditional strategies and technologies needed to gather, organize, process, and gather insights from large datasets. While the problem of working with data that exceeds the computing power or storage of a single computer is not new, the pervasiveness, scale, and value of this type of computing has greatly expanded in recent years. In this article, we will talk about big data on a fundamental level and define common concepts you might come across while researching the subject. We will also take a high-level look at some of the processes and technologies currently being used in this space.

Keyword: - *Large Dataset, Resource Pooling, Easy scalability*

1. INTRODUCTION

An exact definition of "big data" is difficult to nail down because projects, vendors, practitioners, and business professionals use it quite differently. With that in mind, generally speaking, big data is:

- large datasets
- the category of computing strategies and technologies that are used to handle large datasets

In this context, "large dataset" means a dataset too large to reasonably process or store with traditional tooling or on a single computer. This means that the common scale of big datasets is constantly shifting and may vary significantly from organization to organization.

The basic requirements for working with big data are the same as the requirements for working with datasets of any size. However, the massive scale, the speed of ingesting and processing, and the characteristics of the data that must be dealt with at each stage of the process present significant new challenges when designing solutions. The goal of most big data systems is to surface insights and connections from large volumes of heterogeneous data that would not be possible using conventional methods.

In 2001, Gartner's Doug Laney first presented what became known as the "three Vs of big data" to describe some of the characteristics that make big data different from other data processing:

Volume

The sheer scale of the information processed helps define big data systems. These datasets can be orders of magnitude larger than traditional datasets, which demands more thought at each stage of the processing and storage life cycle.

Often, because the work requirements exceed the capabilities of a single computer, this becomes a challenge of pooling, allocating, and coordinating resources from groups of computers. Cluster management and algorithms capable of breaking tasks into smaller pieces become increasingly important.

Velocity

Another way in which big data differs significantly from other data systems is the speed that information moves through the system. Data is frequently flowing into the system from multiple sources and is often expected to be processed in real time to gain insights and update the current understanding of the system.

This focus on near instant feedback has driven many big data practitioners away from a batch-oriented approach and closer to a real-time streaming system. Data is constantly being added, massaged, processed, and analyzed in order to keep up with the influx of new information and to surface valuable information early when it is most relevant. These ideas require robust systems with highly available components to guard against failures along the data pipeline.

Variety

Big data problems are often unique because of the wide range of both the sources being processed and their relative quality.

Data can be ingested from internal systems like application and server logs, from social media feeds and other external APIs, from physical device sensors, and from other providers. Big data seeks to handle potentially useful data regardless of where it's coming from by consolidating all information into a single system.

1.1 **CHARACTERISTICS**

Various individuals and organizations have suggested expanding the original three Vs, though these proposals have tended to describe challenges rather than qualities of big data. Some common additions are:

- **Veracity**: The variety of sources and the complexity of the processing can lead to challenges in evaluating the quality of the data (and consequently, the quality of the resulting analysis)
- **Variability**: Variation in the data leads to wide variation in quality. Additional resources may be needed to identify, process, or filter low quality data to make it more useful.
- **Value**: The ultimate challenge of big data is delivering value. Sometimes, the systems and processes in place are complex enough that using the data and extracting actual value can become difficult.

1.2 **BIG DATA LIFE CYCLE**

So how is data actually processed when dealing with a big data system? While approaches to implementation differ, there are some commonalities in the strategies and software that we can talk about generally. While the steps presented below might not be true in all cases, they are widely used.

The general categories of activities involved with big data processing are:

- Ingesting data into the system
- Persisting the data in storage
- Computing and Analyzing data
- Visualizing the results

Before we look at these four workflow categories in detail, we will take a moment to talk about clustered computing, an important strategy employed by most big data solutions. Setting up a computing cluster is often the foundation for technology used in each of the life cycle stages.

Clustered Computing

Because of the qualities of big data, individual computers are often inadequate for handling the data at most stages. To better address the high storage and computational needs of big data, computer clusters are a better fit.

Big data clustering software combines the resources of many smaller machines, seeking to provide a number of benefits:

- Resource Pooling: Combining the available storage space to hold data is a clear benefit, but CPU and memory pooling is also extremely important. Processing large datasets requires large amounts of all three of these resources.
- High Availability: Clusters can provide varying levels of fault tolerance and availability guarantees to prevent hardware or software failures from affecting access to data and processing. This becomes increasingly important as we continue to emphasize the importance of real-time analytics.
- Easy Scalability: Clusters make it easy to scale horizontally by adding additional machines to the group. This means the system can react to changes in resource requirements without expanding the physical resources on a machine.

Using clusters requires a solution for managing cluster membership, coordinating resource sharing, and scheduling actual work on individual nodes. Cluster membership and resource allocation can be handled by software like Hadoop's YARN (which stands for Yet Another Resource Negotiator) or Apache Mesos.

The assembled computing cluster often acts as a foundation which other software interfaces with to process the data. The machines involved in the computing cluster are also typically involved with the management of a distributed storage system, which we will talk about when we discuss data persistence.

Ingesting Data into the System

Data ingestion is the process of taking raw data and adding it to the system. The complexity of this operation depends heavily on the format and quality of the data sources and how far the data is from the desired state prior to processing.

One way that data can be added to a big data system are dedicated ingestion tools. Technologies like Apache Sqoop can take existing data from relational databases and add it to a big data system. Similarly, Apache Flume and Apache Chukwa are projects designed to aggregate and import application and server logs. Queuing systems like Apache Kafka can also be used as an interface between various data generators and a big data system. Ingestion frameworks like Gobblin can help to aggregate and normalize the output of these tools at the end of the ingestion pipeline.

During the ingestion process, some level of analysis, sorting, and labelling usually takes place. This process is sometimes called ETL, which stands for extract, transform, and load. While this term conventionally refers to legacy data warehousing processes, some of the same concepts apply to data entering the big data system. Typical operations might include modifying the incoming data to format it, categorizing and labelling data, filtering out unneeded or bad data, or potentially validating that it adheres to certain requirements.

With those capabilities in mind, ideally, the captured data should be kept as raw as possible for greater flexibility further on down the pipeline.

Persisting the Data in Storage

The ingestion processes typically hand the data off to the components that manage storage, so that it can be reliably persisted to disk. While this seems like it would be a simple operation, the volume of incoming data, the requirements for availability, and the distributed computing layer make more complex storage systems necessary.

This usually means leveraging a distributed file system for raw data storage. Solutions like Apache Hadoop's HDFS filesystem allow large quantities of data to be written across multiple nodes in the cluster. This ensures that the data can be accessed by compute resources, can be loaded into the cluster's RAM for in-memory operations, and can gracefully handle component failures. Other distributed filesystems can be used in place of HDFS including Ceph and GlusterFS.

Data can also be imported into other distributed systems for more structured access. Distributed databases, especially NoSQL databases, are well-suited for this role because they are often designed with the same fault tolerant considerations and can handle heterogeneous data. There are many different types of distributed databases to choose from depending on how you want to organize and present the data.

Computing and Analyzing Data

Once the data is available, the system can begin processing the data to surface actual information. The computation layer is perhaps the most diverse part of the system as the requirements and best approach can vary significantly depending on what type of insights desired. Data is often processed repeatedly, either iteratively by a single tool or by using a number of tools to surface different types of insights.

Batch processing is one method of computing over a large dataset. The process involves breaking work up into smaller pieces, scheduling each piece on an individual machine, reshuffling the data based on the intermediate results, and then calculating and assembling the final result. These steps are often referred to individually as splitting, mapping, shuffling, reducing, and assembling, or collectively as a distributed map reduce algorithm. This is the strategy used by Apache Hadoop's MapReduce. Batch processing is most useful when dealing with very large datasets that require quite a bit of computation.

While batch processing is a good fit for certain types of data and computation, other workloads require more real-time processing. Real-time processing demands that information be processed and made ready immediately and requires the system to react as new information becomes available. One way of achieving this is stream processing, which operates on a continuous stream of data composed of individual items. Another common characteristic of real-time processors is in-memory computing, which works with representations of the data in the cluster's memory to avoid having to write back to disk.

Apache Storm, Apache Flink, and Apache Spark provide different ways of achieving real-time or near real-time processing. There are trade-offs with each of these technologies, which can affect which approach is best for any individual problem. In general, real-time processing is best suited for analyzing smaller chunks of data that are changing or being added to the system rapidly.

The above examples represent computational frameworks. However, there are many other ways of computing over or analyzing data within a big data system. These tools frequently plug into the above frameworks and provide additional interfaces for interacting with the underlying layers. For instance, Apache Hive provides a data warehouse interface for Hadoop, Apache Pig provides a high level querying interface, while SQL-like interactions with data can be achieved with projects like Apache Drill, Apache Impala, Apache Spark SQL, and Presto. For machine learning, projects like Apache SystemML, Apache Mahout, and Apache Spark's MLlib can be useful. For straight analytics programming that has wide support in the big data ecosystem, both R and Python are popular choices.

2. BIG DATA GLOSSARY

While we've attempted to define concepts as we've used them throughout the guide, sometimes it's helpful to have specialized terminology available in a single place:

- **Big data**: Big data is an umbrella term for datasets that cannot reasonably be handled by traditional computers or tools due to their volume, velocity, and variety. This term is also typically applied to technologies and strategies to work with this type of data.
- **Batch processing**: Batch processing is a computing strategy that involves processing data in large sets. This is typically ideal for non-time sensitive work that operates on very large sets of data. The process is started and at a later time, the results are returned by the system.
- **Cluster computing**: Clustered computing is the practice of pooling the resources of multiple machines and managing their collective capabilities to complete tasks. Computer clusters require a cluster management layer which handles communication between the individual nodes and coordinates work assignment.
- **Data lake**: Data lake is a term for a large repository of collected data in a relatively raw state. This is frequently used to refer to the data collected in a big data system which might be unstructured and frequently changing. This differs in spirit to data warehouses (defined below).
- **Data mining**: Data mining is a broad term for the practice of trying to find patterns in large sets of data. It is the process of trying to refine a mass of data into a more understandable and cohesive set of information.
- **Data warehouse**: Data warehouses are large, ordered repositories of data that can be used for analysis and reporting. In contrast to a *data lake*, a data warehouse is composed of data that has been cleaned, integrated with other sources, and is generally well-ordered. Data warehouses are often spoken about in relation to big data, but typically are components of more conventional systems.
- **ETL**: ETL stands for extract, transform, and load. It refers to the process of taking raw data and preparing it for the system's use. This is traditionally a process associated with data warehouses, but characteristics of this process are also found in the ingestion pipelines of big data systems.
- **Hadoop**: Hadoop is an Apache project that was the early open-source success in big data. It consists of a distributed filesystem called HDFS, with a cluster management and resource scheduler on top called YARN (Yet Another Resource Negotiator). Batch processing capabilities are provided by the MapReduce computation engine. Other computational and analysis systems can be run alongside MapReduce in modern Hadoop deployments.
- **In-memory computing**: In-memory computing is a strategy that involves moving the working datasets entirely within a cluster's collective memory. Intermediate calculations are not written to disk and are instead held in memory. This gives in-memory computing systems like Apache Spark a huge advantage in speed over I/O bound systems like Hadoop's MapReduce.
- **Machine learning**: Machine learning is the study and practice of designing systems that can learn, adjust, and improve based on the data fed to them. This typically involves implementation of predictive and statistical algorithms that can continually zero in on "correct" behavior and insights as more data flows through the system.
- **Map reduce (big data algorithm)**: Map reduce (the big data algorithm, not Hadoop's MapReduce computation engine) is an algorithm for scheduling work on a computing cluster. The process involves splitting the problem set up (mapping it to different nodes) and computing over them to produce

intermediate results, shuffling the results to align like sets, and then reducing the results by outputting a single value for each set.

- **NoSQL**: NoSQL is a broad term referring to databases designed outside of the traditional relational model. NoSQL databases have different trade-offs compared to relational databases, but are often well-suited for big data systems due to their flexibility and frequent distributed-first architecture.
- **Stream processing**: Stream processing is the practice of computing over individual data items as they move through a system. This allows for real-time analysis of the data being fed to the system and is useful for time-sensitive operations using high velocity metrics.

3. CONCLUSIONS

Big data is a broad, rapidly evolving topic. While it is not well-suited for all types of computing, many organizations are turning to big data for certain types of workloads and using it to supplement their existing analysis and business tools. Big data systems are uniquely suited for surfacing difficult-to-detect patterns and providing insight into behaviours that are impossible to find through conventional means. By correctly implement systems that deal with big data, organizations can gain incredible value from data that is already available.

4. REFERENCES

[1] Justine Ellingwood "Digital Ocean",September 2016.

[2] Dr. Arvind Sathi "Big Data Analytics Disruptive Technologies for Changing the Game" First Edition First Printing — October 2012