Chapter 2
Background

It is inevitable to use distributed computing for storing, processing and analyzing Big Data applications (Padole and Shah, 2018). Distributed computing is an environment which supports storage and processing of data using various systems distributed over a network. A distributed system forms a cluster for data storage, processing, and analytics. The cluster can be homogeneous or heterogeneous. In current era of fast growing technology, it is seldom to find merely a homogeneous cluster. Moreover, specifically for Big Data processing heterogeneous cluster is better choice compared to homogeneous one. The reason is, heterogeneous distributed cluster brings together distinct resources to facilitate better scalability. However, performance optimization, heterogeneity of resources, optimum scheduling, and load balancing are some of the key challenges to be dealt with, while implementing scalability model. Therefore, proper selection of distributed model and tools play a vital role while implementing distributed computing, for Big Data processing.

2.1 Big Data

When data storage and processing cannot be managed by a single computing system or with a conventional approach, it is termed as Big Data (Trujillo et al., 2015). The data is emerging at a tremendous pace, from huge range of sources and in variety of forms. Thus, it may comprise of structured and unstructured data which may not fit into traditional databases and storage systems.

According to the IBM study (Jacobson, 2018), social networking sites, sensor devices, IoT devices, public-private network cloud, and many more services generate nearly 2.5 quintillion bytes of data every day. The study also reveals that, out of the total data currently available all across the globe, 90% of it is generated in the past 2 years (Jacobson, 2018). Big data usually include data sets with sizes such as Petabytes or Zettabytes, which are beyond the capability of conventional tools and techniques for storage, processing and managing the data within a reasonable amount of time. Big data calls for cost-effective and innovative methods of information processing.

2.1.1 Big Data Characteristics

To understand, the enormity of Big Data, it is important to understand the characteristics of Big Data. These characteristics are described as volume, velocity, variety, and veracity, also commonly referred as 4 V's of Big Data, as shown in fig. 2.1.

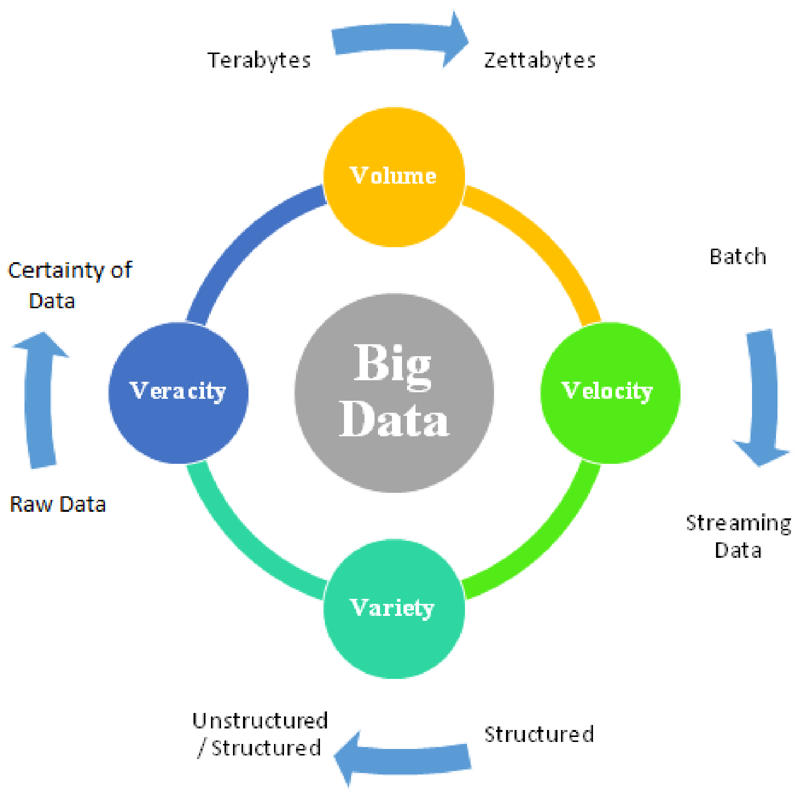

Figure 2.1 Big Data Characteristics

1. **Volume:** It refers to the colossal amount of data. The data is growing exponentially, and it is becoming more and more difficult to store and process this huge data. The volume of data is increasing much faster than the storage and processing speed of computational devices. Traditional Redundant Array of

Independent Disks (RAID) model no longer suffices the need of performance, scalability, and availability of Big Data processing.

2. **Velocity:** It refers to the pace of data being generated by various organizations, applications, devices, and systems. With the increase in the volume of data, there is also a significant improvement in the speed of data generation. Data has turned from static to dynamic due to wide usage of social media, mobile applications, and IoT devices. Nowadays, batch-processing is no longer thinkable, as data generated by the applications is real-time, streaming data. Data access speed is very much important while processing the data. If data is generated too fast and data access latencies are too high, it degrades the performance.

3. **Variety:** It refers to the type of data being generated. Usually, Big Data is a raw form of data. It comprises structured, semi-structured and unstructured types of data. Due to the varieties of data type, it is essential to perform pre-processing on datasets prior to processing and analysis. Conventional processing and analysis approaches do not work efficiently while dealing with Big Data.

4. **Veracity:** It refers to the trust-worthiness and accuracy of data. It requires special attention to validate the authenticity of data. It is equally important to check whether the data that is being stored and mined is meaningful to the problem being analyzed.

2.1.2 Big Data Processing Platform

For large data storage, management, and processing, the conventional storage systems are not useful. The research community has proposed various platforms for Big Data processing. Grid, Cloud and Distributed Computing are few of them. For our thesis, our focus is on distributed computing. Distributed computing system framework fits the need of Big Data, e.g. distributed storage, efficient processing, scalability, elasticity, and management of large datasets. Distributed computing framework has achieved great success in processing clustered tasks, various Big Data application, and performing Big Data analytics. Following are the key characteristics requirements for the environment useful for big data processing

- **Scalable Computing Infrastructure:** Distributed computing provides powerful backstage scalable infrastructure support for Big Data processing. It also enables the distribution and management of Big Data across many nodes and disks.
- **Data Storage Framework:** The Big Data requires nodes which can store data in a distributed manner and when it comes to processing, provides support for execution of the data which shall be addressed by distributed computing.
- **Parallel / Distributed Programming Framework:** Distributed computing supports distributed programming for complex computations.
- **Analytics Framework:** Distributed computing provides an analytical platform for processing large volumes of persistent Big Data, in a highly distributed and efficient manner.

2.1.3 Challenges in Big Data

When data is in the large amount (Big Data) it also comes with major challenges like data storage, concurrent processing, and analysis of enormously growing data. Traditional data management tools are out of the scope for Big Data management as it cannot adhere to the unstructured and semi-structured form of data. To process a massive volume of data and analyze itself is a big challenge. Proper infrastructure needs to be developed. Few literature studies (Chaudari et al., 2011; Labrinidis and Jagadish, 2012; Agrawal et al., 2012) discuss the key problems of Big Data applications. The main challenges are described as follows (Chen et al., 2014):

- **Data representation:** It refers to how data is being stored in the computer system. Under heterogeneous system this task becomes further challenging. Effective data representation is helpful for effective analysis of data.
- **Redundancy reduction and data compression:** As Big Data mostly comprises of unstructured data, it may contain redundant data. It is important to clean and compress the data, by applying some pre-processing techniques, which can yield the cost-effective solution.

- **Analytical mechanism:** Inspection and analysis are the very important requirements of Big Data. To process the large volume of data and analyze the same in a limited amount of time is the biggest challenge of Big Data.
- **Data confidentiality:** Protecting the confidentiality of Big Data is also the concern of research. As data needs to be transmitted or shared for processing, a scheme that can provide security, privacy, and confidentiality of data, needs to be developed.
- **Expandability and scalability:** As data is growing exponentially, the system must be able to handle expandability and scalability of datasets. The seamless transition from small-scale data to large-scale data is required as data is increasing continuously. Moreover, the processing environment for Big Data should also be scalable, to meet the needs of data expansion or data growth.
- **Latency:** Latency is one of the key problems when you're dealing with huge amount of data. A latency not only affects the execution time sof job but it also affects various implicit parameters such as resource utilization, I/O overhead, processing performance and many more.
- **Optimization:** Optimization of algorithms and tools for Big Data processing is of prime concern. Various techniques are available, but it is not that commendably developed.

2.1.4 Big Data Optimization

For large business enterprises, it is indispensable to process large-scale data (Big Data) to get better insight into the business. There is no doubt that, processing of Big Data has become a challenging task, although many tools and techniques are available to process this flood of data. However, to process Big Data in an optimized way such that its overall performance doesn't degrade is a challenging task. Increasing rate of data will become critical to handle in future, hence, proper optimization techniques need to be applied for Big Data processing.

The major research problem is to provide proper optimization techniques for Big Data. Most of the optimized techniques which have been invented and researched are in context to traditional RDBMS. There are many challenges in applying existing optimization techniques for large data sets, which itself can be an area of research. Based on the objective, optimization techniques can be categorized as Performance Optimization, Ease-of-use, and Cost-Effectiveness. Performance optimization aims to reduce execution time to make data processing faster. Ease-of-use aims to make data processing tools easier to implement and use for the variety of datasets, while cost-effective optimization focuses on, to minimize the operating cost of the system. In the previous section i.e. Section 2.1.3, several challenges in Big Data processing have been discussed, out of which distributed computing provides the solution for scalability, storage, and distributed processing. Thus, the focus of our research work is to optimize storage and processing of Big Data through Distributed Computing.

2.2 Distributed Computing

Distributed Computing (DC) refers to performing computation using a system of loosely coupled computers striving to solve computationally intensive problems that are difficult to be computed using a single computer. Distributed computing is an effective alternative to expensive resource intensive computing, that is required to manage and process, complex computational problems (Lapkin, 2012). Distributed computing is used to solve complex computational problems that cannot be solved within a specified time frame on a single computer. The complex computational problems may involve either compute intensive or data-intensive processing.

2.2.1 Distributed Computing System (DCS)

"A Distributed Computing System, also referred as a Distributed System, is a collection of independent computers that appears to its end users, as a single computing system" (Tanenbaum and Van Steen, 2007). Distributed Computing System (DCS) refers to a system of multiple computers working on a single problem that is computationally intensive. A distributed computing system is a wide scale infrastructure that supports the sharing of resources, distribution transparency, scalability, single point failure handling and single system image concept in large-scale problem-solving. Distributed computing system provides, the aforementioned advantages, compared to the traditional centralized computing system. The general architecture of the distributed computing system is discussed in the next section i.e. Section 2.2.2.

2.2.2 Distributed Computing System Architecture

In computer architecture terminology, distributed computing system belongs to the class of loosely coupled Multiple Instruction, Multiple Data (MIMD) machines, with each node having an unshared memory (Ghosh, 2014). Fig.1 shows the typical architecture of DCS (Coulouris et al., 2005; Tanenbaum and Van Steen, 2007)

Distributed Computing systems are built using existing commodity hardware, operating systems (OS) and network. The hardware, OS, and network involved in forming DCS, may be of same type or different type i.e. they may be either homogeneous or heterogeneous in nature, respectively. A distributed system is a collection of individual computers connected via a network and middleware service which supports distributed storage and concurrent processing. The middleware enables distribution transparency, wherein the task submitted to Master is distributed amongst multiple Slaves. Thus, middleware serves as a service agent between distributed applications and machines. Middleware provides concurrent processing, and memory access services for faster processing. Figure 2.2 shows the architectural design which connects large number of computing devices with the fast network for transparent resource sharing among the distributed applications.

The distributed computing system comprises of the variety of hardware and software, to form a distributed platform. At a lower level, it is necessary to have multiple CPUs which are interconnected with each other by the network. At a higher level, those interconnected CPUs will communicate with each other through Middleware. The distributed computing system can be categorized as homogeneous or heterogeneous based on the uniformity or non-uniformity of hardware, OS, connection, architecture, and other components, respectively.

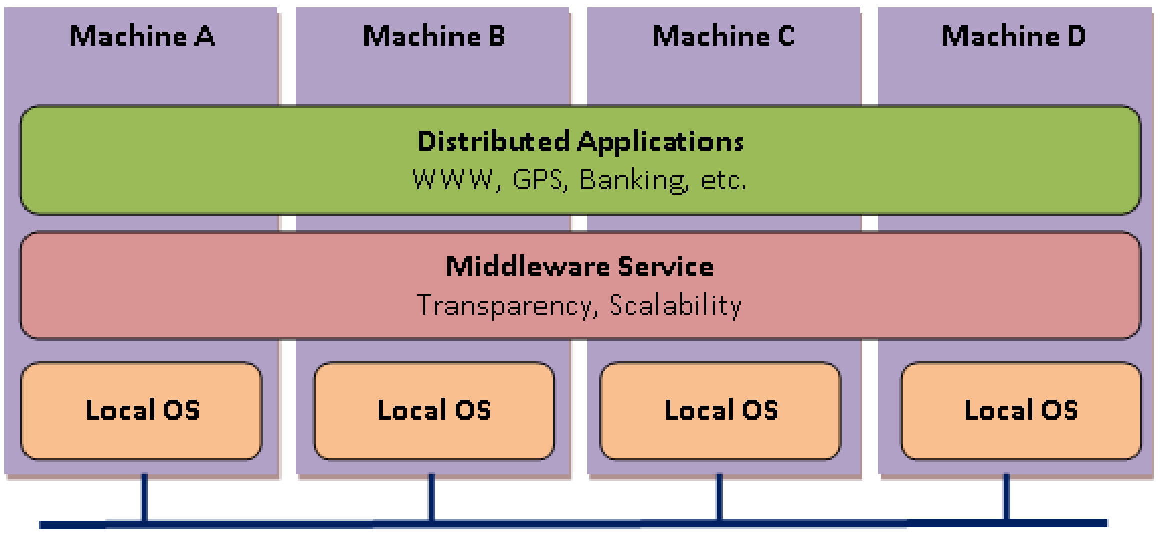

Figure 2.2 Distributed Computing System Architecture (Tanenbaum and Steen, 2007)

2.2.3 Homogeneous Distributed Computing System (HDCS)

A distributed computing system is said to be a Homogeneous Distributed Computing System: a) If hardware on each computing machine has the same architecture, processing capacity, and same storage representation. b) If the software (i.e. Operating system, Compiler etc.) on each computing machine has the same storage organization and similar operational speed. The requirements for a homogeneous distributed computing system are quite stringent and are frequently not met in the network of workstations, or PCs, even when each computer in the network is of the same make and model.

2.2.4 Heterogeneous Distributed Computing System (HeDCS)

Heterogeneous Distributed Computing System (HeDCS) is one which is not homogeneous. Heterogeneous distributed computing system supports different types of processors, hardware configurations, network, and operating systems. Distributed computing systems highly rely on the heterogeneity of processing nodes. A heterogeneous distributed system is mainly designed to achieve high performance by connecting distinct devices on a distributed platform. In this research study, the aim is to use Heterogeneous Distributed Computing System (HeDCS), hence, it is discussed in detail.

Computing performed using Homogeneous Distributed Computing System (HDCS) is referred as Homogeneous Distributed Computing (HDC) and likewise, the computing performed using Heterogeneous Distributed Computing System (HeDCS) is known as Heterogeneous Distributed Computing (HeDC). Henceforth, for the discussion, the term HeDC will be used to describe problem-solving using HeDCS.

2.3 Heterogeneous Distributed Computing

In a practical scenario, it is difficult to find a computing platform, with all computers involved in processing, to comprise of perfectly uniform configuration. Hence, the emphasis is using the heterogeneous set of computational resources, for solving computationally intensive problems.

A Heterogeneous Distributed Computing (HeDC) made up of a heterogeneous OS, CPUs, network, and protocols provide a way to connect distinct devices to perform distributed computation (Hwang and Briggs, 1985). Heterogeneity allows scaling of a cluster. HeDC is very well established computing paradigm to meet the data-intensive or process intensive requirements of Big Data applications. The examples of applications are weather forecasting, simulation and modeling, mapping of the human genome, Big Data processing, image processing, modeling of semiconductors, superconductors and banking systems.

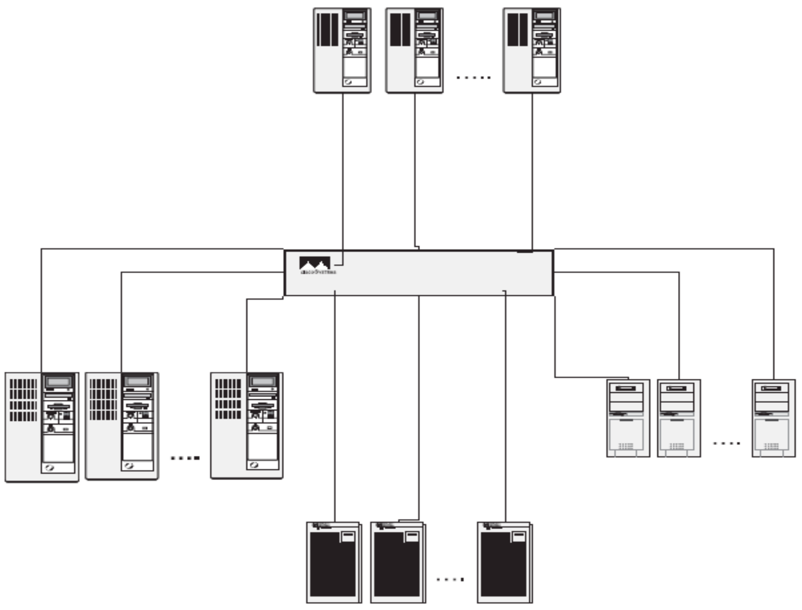

Figure 2.3 Heterogeneity of Nodes

Figure 2.3 gives an idea about the heterogeneity of nodes. In today's era where technology is changing every day, it is infeasible to have cluster nodes of similar types. Therefore, it is important to have distributed functionality though systems, networks, and configurations are heterogeneous in nature.

2.3.1 HeDC Challenges

Heterogeneous Distributed Computing comes with novel challenges due to non-uniformity, the variety of programming models, and overall varied system capability. The following factors are to be considered while applying HeDC for large datasets.

- Different instruction set and memory set architectures
- Library and OS services are not uniformly available on distributed nodes
- CPUs have different performance level and power consumption
- Compute elements have different cache structure, network architecture

Above factors may result in performance degradation while working on large data sets, also referred to as Big Data. Big Data processing would be difficult to be performed on a single computer; it is also not feasible to be performed on homogeneous distributed computing, due to scalability issues. Hence, there is a need for HeDC for processing large data sets i.e. Big Data.

2.4 Distributed File System

Distributed File System (DFS) is a specialized storage facility which allows splitting the data file into chunks and store them across multiple machines. The stored data can be accessed and processed in parallel for data processing. Unlike the local file system, data is not stored at a single place but it is spread out across multiple storage resources, connected via a high-speed network. Looking at the storage size and processing requirement of Big Data, it is inevitable to use a distributed file system concept. Even though data files are stored on multiple machines, DFS provides a simplified interface for managing, accessing and storing data on it. There are many different types of distributed file systems available, especially to deal with Big Data which is discussed in great detail in Chapter 4. For any DFS, it is important to have features such as fault-tolerance support, scalable architecture and transparency.

- **Fault-tolerance:** It is important that data be always available, when data is distributed using DFS. DFS should be strong enough to handle faults or failures. Issues may be related to data non-availability, hardware failure, network congestion or network failure. DFS should be fault-tolerant against these types of problems.
- **Scalability:** In the case of Big Data, it is growing continuously so the DFS implemented in DCS, should be scalable elastically according to the size and requirement of data. Scalability is an important aspect, necessary to be achieve, to deal with accessibility, latency, resource availability, and parallel processing issues.
- **Transparency:** Regardless of where the data is stored and processed, a user should be able to see it as a single system. DFS should be transparent to the clients and they should be able to experience the same performance as they deal with the localfile system.

2.4.1 Fault-tolerance Support

In the real world, hardware, software, network failures are bound to happen. It is inevitable to design a system in which these problems don't arise. Therefore, the only solution is to design a system which always has an alternative solution to the problems of data unavailability, replica management, and load balancing. Here we present the key features of DFS for maintaining the high availability and fault-tolerant support.

- **Replication:** Replica placement and management are very important to maintain data availability. If multiple copies of data (called replica) are stored on distinct racks and geographic locations, then, even if one or more system or server crashes, data always remains available.
- **Load balancing:** The different file system uses a different load balancing approach to deal with fault-tolerance and performance issues. In the case of

system failure or non-availability of data, the system must be able to migrate the data automatically to other nodes to avoid the trade-off of the performance degradation and data non-availability.

- **Synchronization:** In DFS, synchronization between replicas, nodes, and process must be considered for making system fault-tolerant. If data is updated it is also important to update all copies of data in order to avoid the possibility of accessing the older data.

2.4.2 Scalability Support

To make scalable architecture, it is important to address the two important issues: high latency and network congestion. If scaling is not done properly it creates a lot of issues such as high latency, communication delay, and network congestion. Therefore, the performance of the system may get degraded. In DFS, scaling can be achieved vertically (scale up) and horizontally (scale out) as shown in Fig. 2.4.

Vertical scaling (aka scale up) means to scale the existing system with upgrading hardware and software support. It means it upgrades the storage, memory and processing capacity of the existing system only. But even that has a limitation as you cannot upgrade the system beyond certain limit. Solution to the problem is, horizontal scaling (aka scale out) which allows a system to scale out horizontally by adding more computing resources rather than upgrading the existing one. Using the concept of horizontal scaling we can store more amount of data as scaling can be achieved extensively.

2.4.3 Transparency Support

Transparency hides the complexity of system implementation and configuration from the end-users. DFS server mainly provides location transparency and replication transparency. In DFS, location transparency means hides the storage location where data is stored and replication transparency means hides the location of replica where it is stored. These data can be accessed by naming services which maps the logical and physical data location.

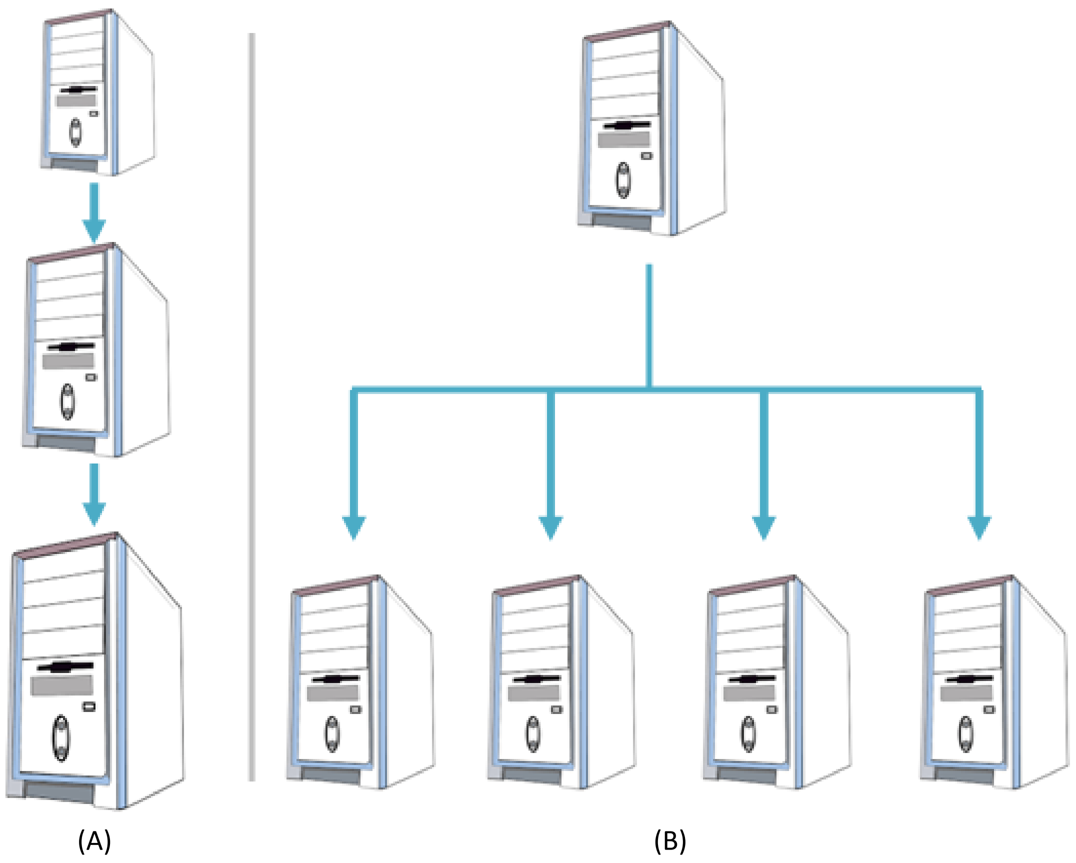

Figure 2.4 (A) Vertical Scaling (B) Horizontal Scaling

The main advantage of transparent DFS is that it allows simple architecture for storage as, a user need not worry about the physical location of data storage. Furthermore, it automatically decides the storage location based on data placement strategy and also manages the replication of each block of data to avoid the system failure.

2.5 Scheduling in Heterogeneous Distributed Computing

Scheduling plays a prominent role in distributed computing performance. Moreover, if scheduling is not done properly it becomes the bottleneck in terms of throughput of the system. Proper choice of scheduling helps to improve resource utilization and job completion time. Considering the fact, scheduling algorithms are NP-complete (Ullman, 1975), Fig 2.5 shows the hierarchical classification of the scheduling algorithms.

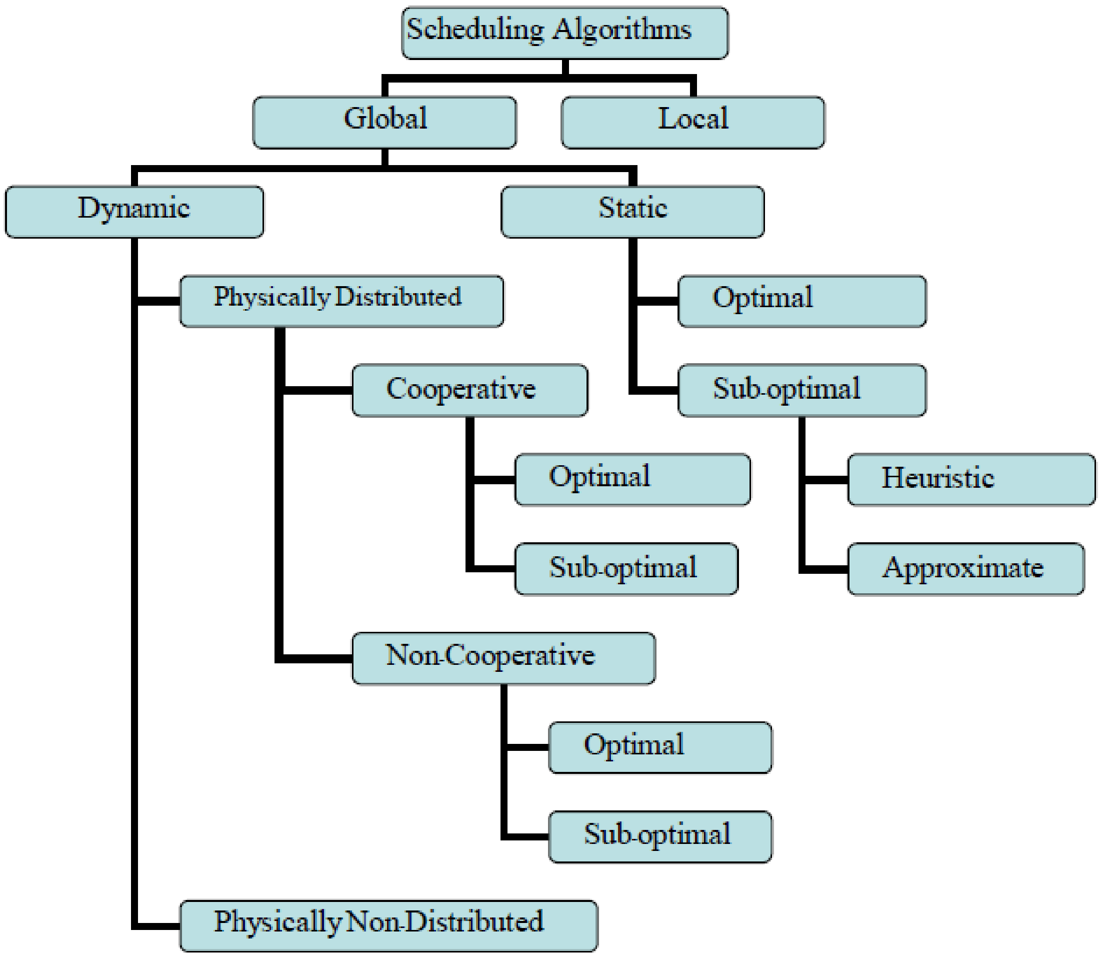

Figure 2.5 Hierarchical classification of scheduling algorithms (Padole and Shah, 2018)

The classification shown in Fig. 2.5, helps to understand the behaviour of scheduling algorithms. The important characteristics are considered for better understanding of classification of scheduling algorithms, although, there are many characteristics which can be explored further. . First, all scheduling algorithms are categorized as local and global subsequently subcategorized in two groups such as static and dynamic schedulers. Hierarchy of scheduling algorithm classification is discussed as follows:

- Local & Global – Local scheduling performed on a single system which has single CPU to schedule the task while Global scheduling is performed in multisystem and multiprocessor environment. Distributed job scheduling comes into the category of global scheduling, which is taken into consideration for further tree expansion.

- Static & Dynamic – Global schedulers are static or dynamic in nature. Static schedulers run on predefined conditions of scheduling and do not consider any priority or deadline of jobs/tasks for scheduling at runtime. Whereas, dynamic schedulers are more appropriate for scheduling in distributed computing, as it requires scheduling of job/task at runtime based upon various considerations like resource requirement to execute the task and availability of resources across distributed system. The static scheduler would need to know the resource requirement of the task in advance, before the processing of the task begins whereas, dynamic schedulers work dynamically depending upon the resource requirement of the task and the availability of resources, at runtime.

- Optimal & Suboptimal – All static and dynamic schedulers are subcategorized as the optimal and suboptimal. Optimal scheduling can be performed when all resource requirements for the application is known to us before scheduling. Optimal scheduling tries to match the criterion like optimal makespan, resource sharing, and concurrency control. But as stated earlier, scheduling problems are NP-complete (Ullman, 1975), it is difficult to identify resource requirement apriori. Therefore, suboptimal scheduling is a better option for this kind of situation. Suboptimal scheduling do not assure the best or optimal solution, but provide sub solutions with guaranteed results.

- Approximate & Heuristic – As the name implies approximate solution is guaranteed by the approximate scheduling approach. It gives approximate performance based upon available resources and scheduling parameters. Heuristic scheduling gives an accurate result rather than approximate. The heuristic approach is widely adopted as it supports both static and dynamic scheduling.

- Distributed & Non-distributed – In global, dynamic category scheduling algorithms are divided into two broad subcategories, the schedulers which support distributed environment and the schedulers which are not. Non-distributed schedulers can run on a centralized system environment only, which has lack of scalability, non fault-tolerant, and performance issues. On the other hand, scheduling algorithms can be distributed on multiple machines and can perform the scheduling jobs on multiple systems.

- Cooperative & Non-cooperative – Distributed schedulers are cooperative or non-cooperative in nature. For cooperative scheduling, processors work together to make any decision for application scheduling. In contrast, non-cooperative scheduling processors work independently without concern of other processor and its effect, while making any decisions.

2.6 Framework for Big Data

There is a need for computing service which can answer key problems such as latency, scalability, resource sharing, and parallel processing. Distributed computing meets the need for Big Data. It is not necessary that all complex computing problems required specialized service like distributed computing. But the Big Data processing is struggling to get performance due to high latency and compatibility of hardware support. Distributed computing leverages the commodity hardware by effective resource sharing, concurrent processing and reducing latency.

Basically, inspiration is to use distributed storage and processing to deal with the 4 V's of Big Data. In order to meet the requirement of Big Data processing specialized framework is required which can provide services for distributed storage and processing. Hadoop is a widely adopted framework for the said purpose. The next section introduces "Apache Hadoop" – The Distributed Framework.