5-2017

Survey of Parallel Processing on Big Data

Cheng Luo
leon.luocheng@gmail.com

Survey of Parallel Processing on Big Data

by

Cheng Luo

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of Requirements

for the Degree

Master of Science

in Computer Science

May, 2017

Starred Paper Committee:
Jie Hu Meichsner, Chairperson
Andrew A. Anda
Jim Q. Chen

Abstract

No doubt we are entering the big data epoch. The datasets have gone from small to super large scale, which not only brings us benefits but also some challenges. It becomes more and more difficult to handle them with traditional data processing methods. Many companies have started to invest in parallel processing frameworks and systems for their own products because the serial methods cannot feasibly handle big data problems. The parallel database systems, MapReduce, Hadoop, Pig, Hive, Spark, and Twister are some examples of these products. Many of these frameworks and systems can handle different kinds of big data problems, but none of them can cover all the big data issues. How to wisely use existing parallel frameworks and systems to deal with large-scale data becomes the biggest challenge. We investigate and analyze the performance of parallel processing for big data. We review and analyze various parallel processing architectures and frameworks, and their capabilities for large-scale data. We also present the potential challenges on multiple techniques according to the characteristics of big data. At last, we present possible solutions for those challenges.

Acknowledgement

I would like to thank my advisor Dr. Meichsner for offering a lot of valuable help and suggestions to my paper work. Without her help, I cannot finish this paper smoothly. I would also like to thank the committee members Dr. Anda and Dr. Chen for sharing their valuable time and advice on my paper research work.

Table of Contents

Section Page

List of Tables

List of Figures

1. Introduction

Today, more and more people use the Internet to achieve their needs for communication, shopping, transactions and so on. According to the research of IBM, there are 100 terabytes of data uploaded on Facebook, 294 billion emails sent, and 230 million tweets on Twitter every day, and these actions generated 5 exabytes data every two days in 2012 [1]. This rapidly growing flood of big data represents huge opportunities objectives such as big data-based offerings and business decisions making [2]. Determining how to quickly optimize the challenges such as analysis, searching, sharing, storage, and transfer of these data is an essential key to achieving success in the competitive digital world.

1.1 What is Big Data?

Big data is the collection of data sets that are complex and large, and they are hard to process with standard software [3]. To describe big data, there is no way we can avoid describing its major characteristics 5V: volume, velocity, variety, veracity and value as shown in Figure 1.

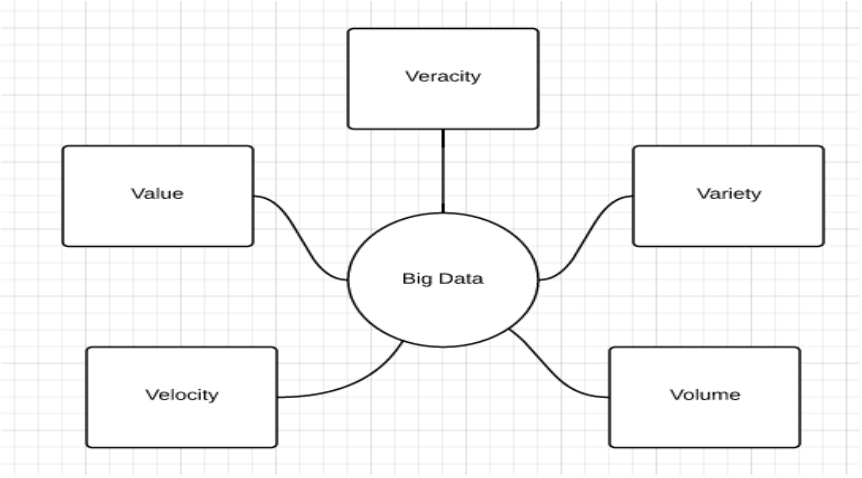

Figure 1: Big Data 5V [4]

1.1.1 Volume

Volume: This is the quantity of data that are generated not only from the Internet but also from the transaction data from internal of companies. With the data growth, the requirements of capacity for storing the data have increased. However, the cost of storage is decreasing. For instance, in 1956 IBM offered a 5MB hard drive storage system, which cost $50,000, but in 2013 the Western Digital MyBook WDBACW0020H with 2TB only cost $109.99 [5]. The reduction of cost is significant so that we can store more data with a cheaper device.

1.1.2 Velocity

Velocity: This is the speed of data creation. Compare to volume, the velocity of the data creation is even more important to many companies, because obtaining real time information allows companies to react more quickly in the digital world [6]. For example, on November 11, 2014, Alibaba received 285 million orders per minute and 93 billion dollar transactions in one day [7].

1.1.3 Variety

Variety: This is the category of big data. Big data originate from messages, social network, government data, and media outlets. There are three types of big data needed to be considered: structured data, unstructured data, and semi-structured data [4]. Structured data are considering to be the text or numeric, such as name or age. Unstructured data are in the form of PDF file, video, audio, images, etc. Semi-structured data are in the form of XML file, JSON file, log file [4].

1.1.4 Veracity

Veracity: This is accuracy, trustworthy and quality of the data. According to the survey of IBM, 1 out of 3 business leaders do not trust the data which was used to make decisions, and 27% of people responding to the IBM survey were not sure of how much of their data was accurate or not [8]. Big data quality, which depends on the veracity of source data, is very important for the analyzer to estimate their data accurately.

1.1.5 Value

Value: This is the new characteristic that was added to the previous 4Vs by many data scientists. We assert that the value is the most important characteristic in the 5V. Unless we can turn those large-scale datasets into value, those datasets are rubbish. It is important for companies to make data analysis cases by using these large-scale data, but it is very easy for companies to fall into the big data trap if they do not have a clear understanding about the costs and benefits of big data [9].

1.2 Where Does Big Data Come From?

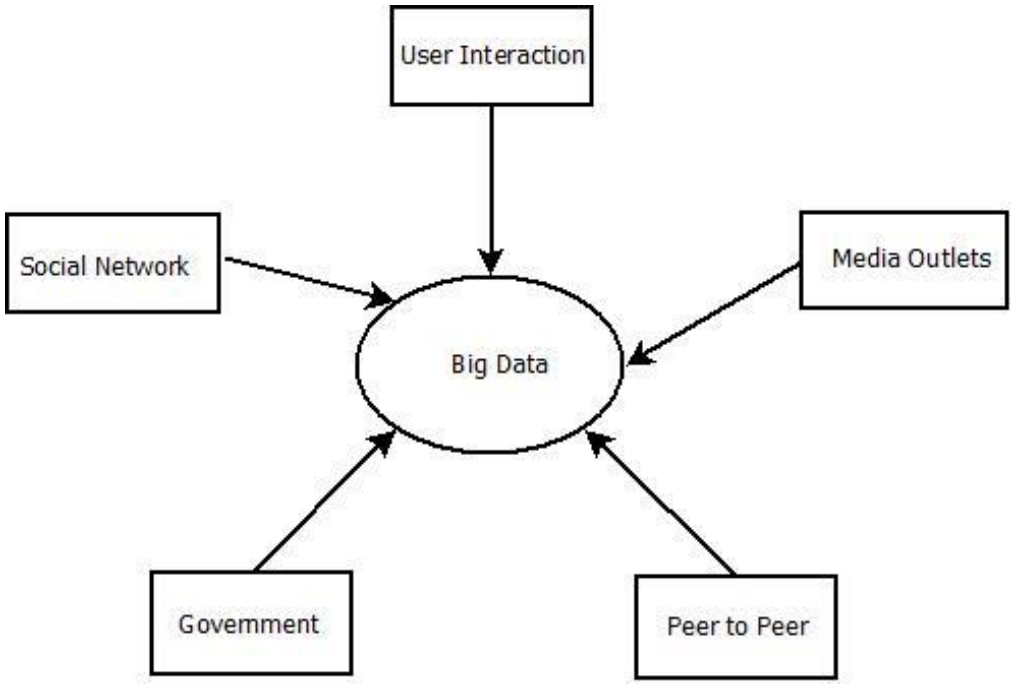

Figure 2: Big Data Source [10]

These large-scale datasets come from many various possibilities [10]:

1. Social networks: the growth of the social technologies, such as YouTube, Twitter and LinkedIn, generates billions of data for the business.
2. Government data: governments generate lots of data. Population, healthcare, legal information, weather and disaster forecasts; these kinds of data are inevitable for big data analysis and research.
3. Peer to peer communication: with the digital growth, more and more data come from peer to peer communications such as text messages, chat lines, and digital phone calls.

4. Media outlets: media websites are also a big part of the big data community. Digital books, magazines, web pages, images, and videos all belong to this.
5. User interaction: Currently, more companies are starting to build their own data for businesses by using websites. The most common data sources come from users' interactions. For example, an action, such as downloading, registration and completing an order from websites. Companies start to collect those data to classify different types of customers, and come out with strategies to further deal with different types of users. Those kinds of datasets could be sources of big data for companies to start analyses of their own businesses.

1.3 Parallel Processing in the Big Data Epoch

Parallel processing is considered "one of the cost-effective method for the fast solution of computationally large and data-intensive problems" [11]. Efficient parallel processing frameworks or applications are crucial for handling the performance and scalability requirements for big data researches. It has been over 30 years since people wrote SQL software and ran it in parallel, but this is not enough to deal with big data problems. Combining massively parallel processing and parallel computing environments, the parallel database systems provide speed, reliability, and capacity for data processing [12]. A parallel database system allows developers to partition the input data tables into different pieces according to the partitioning rules, which are range partitioning, round-robin, and hashing, and then put each piece for each processor through a single-machine program.

MapReduce, which is a parallel and distributed programming framework, has gained a lot of attention from web search companies. Because of its scalability and fault tolerance for

big data, MapReduce is very successful in processing big data. Therefore, many frameworks developed from MapReduce also gain massive success. In the early 2000s, some IT pioneering companies like Google and Yahoo had already built parallel infrastructures to handle searching problems, because the old infrastructures could not handle their workload well. Even though, there is a massive improvement in dealing with big data problems by using MapReduce with its high fault tolerance and scalability, it still contains some pitfalls, such as low efficiency and does not support high-level script language [13]. Some frameworks based on MapReduce have been proposed to solve these kinds of problems, such as the Apache Hadoop ecosystem, Hive, Pig, and Spark [14]. With the development of cloud services and parallel data warehouse systems, some companies have started to integrate their MapReduce and Hadoop applications with parallel data warehouse systems on the cloud. Even though using cloud service and parallel data warehouse system, some of the big data issues get relieved, such as increasing storage space and increasing data transmission speed, there are some challenges exist that still need to be overcame.

1.4 What is the Objective of This Paper?

In this paper, we present a survey of research on the parallel processing for big data through systems, architectures, frameworks, programming languages and programming models. We analyze the advantages and disadvantages of these parallel processing paradigms within the scope of the big data. We also list the potential challenges for multiple technique areas according to the characteristics of big data, and provide possible solutions for those challenges.

2. Parallel Database Systems

Back in the 1900s, Initially the data were mainly generated by companies was not that huge, so that the database management system could find the best approach to solve the data related problems [15]. With the Structured Query Language (SQL) becoming the standard query language, data scientists found out it is quite effective to deal with data problems by using SQL. However, with the development of technologies, data have been growing geometrically, and this method becomes infeasible because of the size of data. Two decades ago, a terabyte of data was considered an uncommonly large volume of data, but right now those sizes are common, even in small company's database or file system. For example, 20 petabytes are processed per day by Google; more than one million transactions per hour are processed by Walmart, and these transactions are more than 2.5 petabytes of data; AT&T has a 312 terabytes database which includes 1.9 trillion phone call records [16].

A theory called "Fourth paradigm", which are "experimental science", "theoretical science", "computational science" and "data-intensive science", is proposed by Microsoft research team to describe the development of science [17]. Most of interest regards, "data-intensive science" which is to develop a set of technologies for data analysis, data visualization and data management [17]. Currently, computer architectures are increasingly unbalanced; the growth of gap between hard disks and CPUs makes the fourth paradigm becoming massive challenges [18]. Lots of applications have generated and controlled terabytes and petabytes of distributed data, which requires high solid network environments and a gigantic I/O bandwidth for support. There is also a trend that the clusters of computers should be focused on managing and processing large-scale data rather than using the biggest

and fastest single computer [18]. A parallel database system is a high-performance database system established using massively parallel processing or parallel computing environments [19]. It allows multiple instances to share one physical database so that the shared device, software and data can be accessed by multiple client instances [12].

In this chapter, we present parallel database system and its architectures. We analyze the advantages and disadvantages of each architecture for handling big data problems. We also present symmetric multiple processing and massively parallel processing, which are widely utilized in parallel database systems, and analyze their advantages and disadvantages in big data area as well.

2.1 Partitioned Parallelisms

Relational queries are more ideally suited to parallel executions. Every relational query can be transferred into operations like scan or sort. Through the operators, source data can be produced as same as the client requests [19]. From the Figure 3, we can easily see that each data stream comes from the data source and then becomes an input of an operator1 which produces an output that is used as an input of operator2, and eventually the output is generated from merged result of operator2.

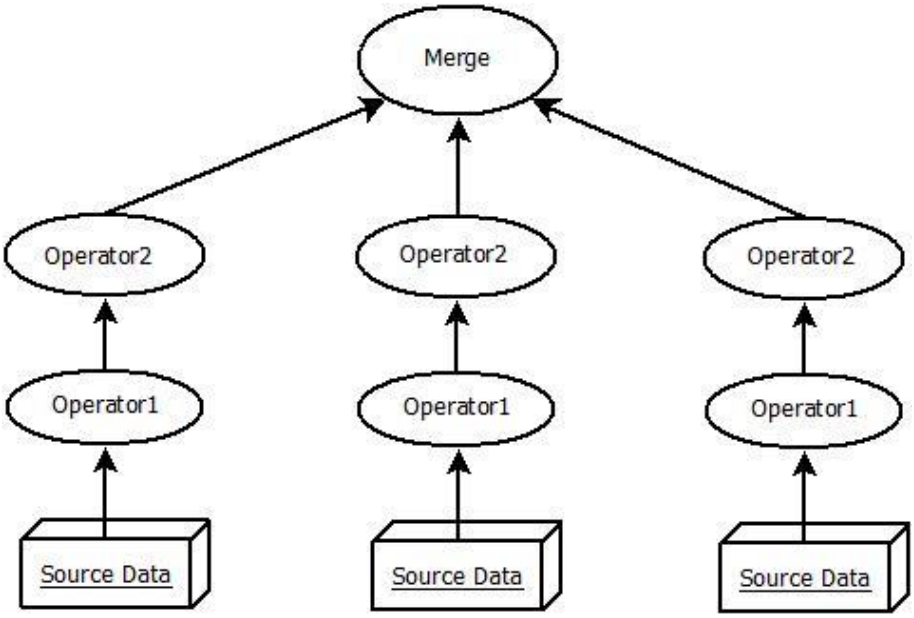

Figure 3 Partitioned Parallelisms [19]

This approach requires a merge based server that can handle the parallel execution of those operations. Without a high-speed network, this approach looks impossible. Right now, most of the parallel database systems use high-speed LAN as their workstations. Meanwhile, some of the companies use high-speed networks and distributed database technology to construct their parallel database systems which would cost more money.

2.2 Hardware Architectures

An ideal situation is that our database machine is a single super-fast CPU with unlimited disk, infinite RAM, unlimited bandwidth, and affordable cost [19]. With this kind of database machine, we do not need to worry about the speed up and scale up. However, the technology has not reached the level we would ideally want. It is not just simply adding multiple processors to speed up the database system. In some cases, adding one more

processor will slow down the system. Figuring to how efficiently build a multiple processors database system is important, and it is also a challenge. There are three types of architectures to use when building a parallel database system.

2.2.1 Shared-memory architecture

A shared-memory architecture [20] is shown in Figure 4. In this architecture, multiple processors must share the main memory space. There are some advantages and disadvantages if we choose a shared-memory architecture:

Advantage:

1. Global shared memory.
2. The data access is very easy to every processor by shared network bus.

Disadvantages:

1. The whole system is a lack of scalability for both memory and processors [19]. For future expansion, it is hard to increase the scalability without increased number of processors or memory.
2. Simply adding processors will increase the inner network traffic. It will also increase the difficulty on memory and processors management.
3. Programmers need to take responsibility for the synchronization management on memory access.

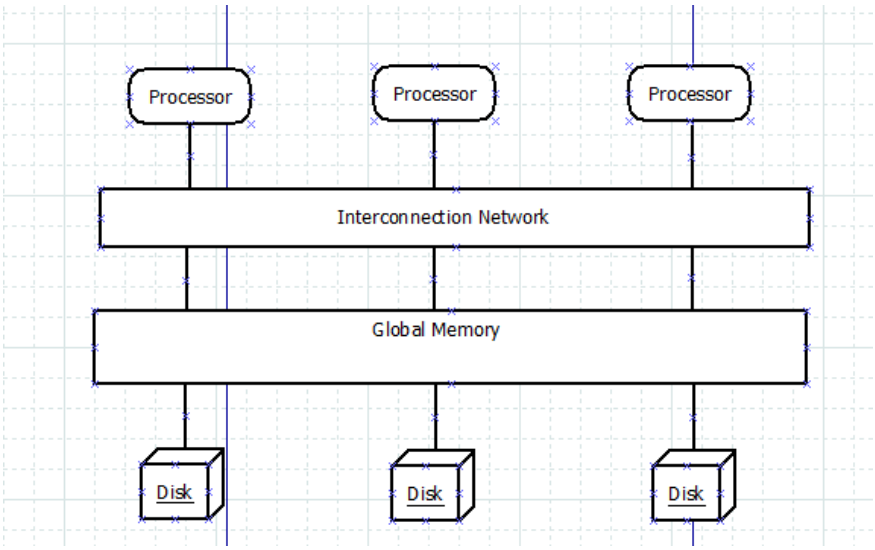

Figure 4. Shared-Memory Architecture [19]

2.2.2 Shared-disk architecture

Shared-disk architecture [20]: is shown in Figure 5, each processor in this architecture has its own private memory, but through an interconnection network, every processor can access all disks. There are some advantages and disadvantages if we choose shared-memory architecture.

Advantages:

1. With high availability, all data in the disks are accessible even if some of the processor nodes are dead.
2. Compared with the shared memory architecture, the shared disk database system is much easier to add more processors without slowdown the performance of the entire system.

Disadvantages:

1. Interconnection network bandwidth is the limitation for the scalability of the system. When processors or nodes reach out of the max limitation of system requests, they will cause high overload.
2. Difficult to set up the workload for each processor to push requests simultaneously. There may be high synchronization overhead.

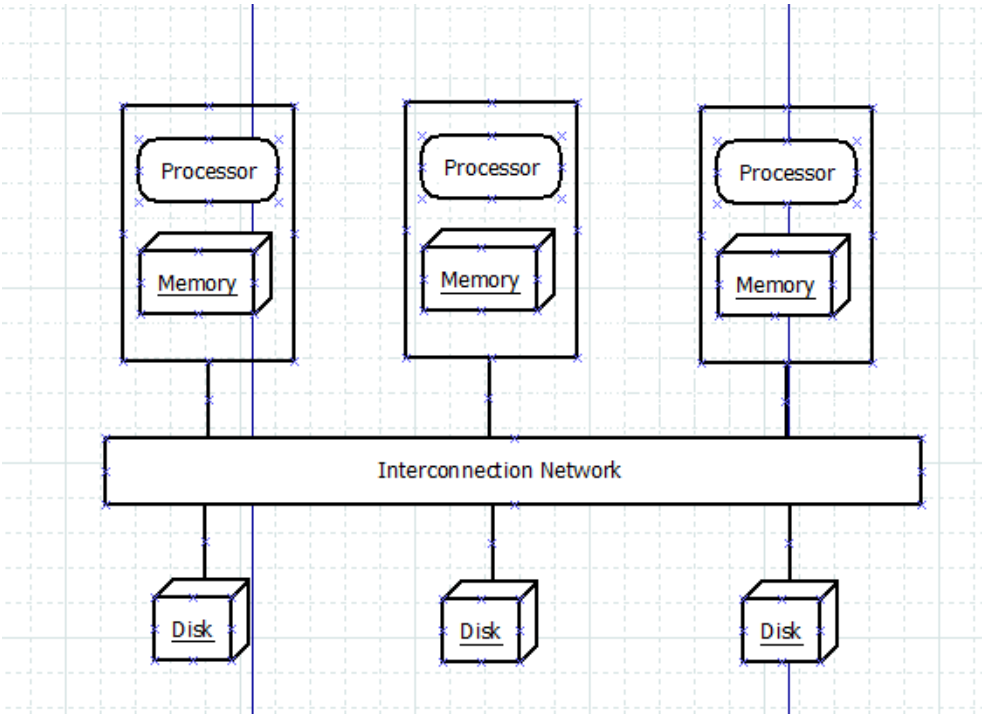

Figure 5. Shared-disk Architecture [20]

2.2.3 Shared-nothing architecture

Shared-nothing architecture [20]: is shown in Figure 6, each processor or node has its own memory and disk, which store processing data. We also can say each processor is a single server. And servers are connected to each other by an interconnection network. In such

architecture, massive data are distributed among every server. David Dewitt and Jim Gray wrote, “Parallel database architecture trended to shared nothing architecture.” [19]. Currently, the shared-nothing architecture is very common in many frameworks which are produced by processing large-scale data, such as the Hadoop and massively parallel database. For example, the Teradata Corporation built their DBC/1012 Database Machine by this model, and it is still used for their applications [21]. Shared-nothing architecture can be expanded to thousands of processors [19]. The whole system growth is practically unlimited, which means it provides scalability. In the meantime, it also has high fault tolerance, if one processor or node fails, another processor will continue the work. It looks like this type of architecture is much more complicated compared to shared-memory and shared-disk in the big data area, but it still has some pitfalls:

1. Because every processor is a single unit and has its own disk, it needs more coordination for those processors. Even though this architecture provides unlimited scalability, it does not mean adding more processors or node is easy. The system needs to be reconfigured, and this causes more work.
2. Every processor can only work on its own disk. If one processor dies, another processor cannot access the data on the dead processor unless there is a backup node.
3. From Figure 6, every processor is connected by an interconnection network. With slow network speed, it was hard to achieve the performance that we need, so the whole architecture requires a high-speed network.

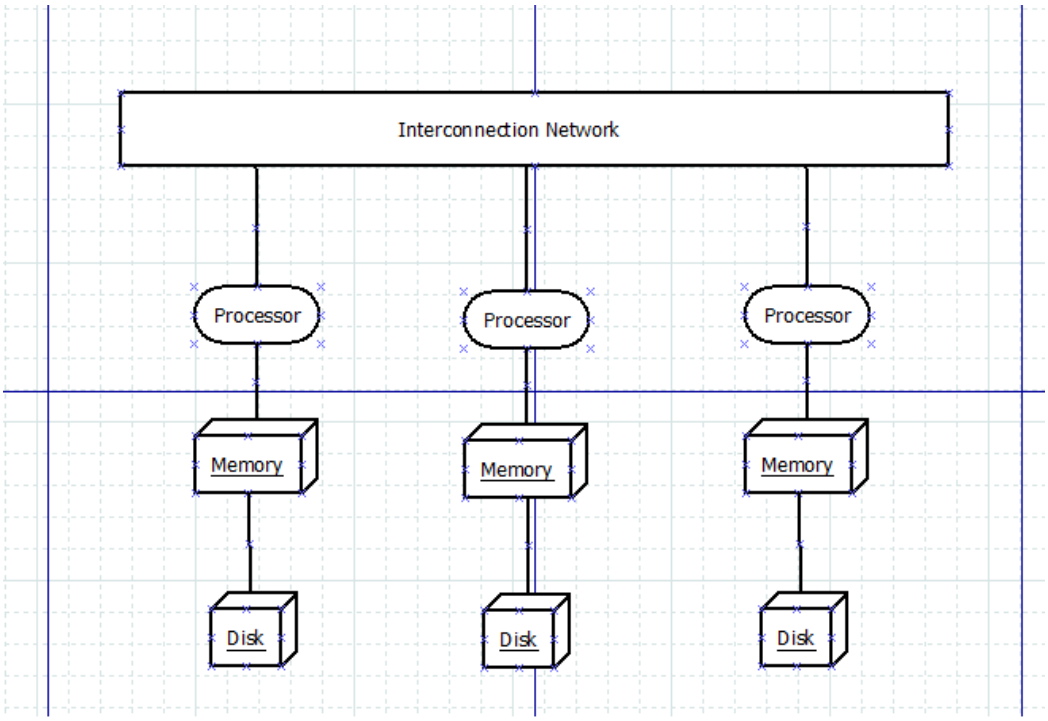

Figure 6. Shared-nothing Architecture [19]

2.2.4 Comparison of three parallel architectures

Table 1. Comparison of Three Parallel Architectures

	Advantage	Disadvantage
Shared-Memory Architecture	Easy to access the data	This system is lack of scalability.
	Memory are shared between processor	Adding processor will increase the inner network traffic.
		Programmer needs to take care the synchronization management on memory access.
Shared-Disk Architecture	High availability of data	Interconnection network bandwidth is the limitation of the scalability.
	Providing incremental growth	Difficult to set up the workload for management processor requests simultaneously.
Shared-Nothing Architecture	Providing unlimited growth	More coordination between processor is required.
	Fault tolerance	If one processor dies, that data is inaccessible unless there is a backup node.
		High-Speed network is required.

In Table 1, we compare advantages and disadvantages for each architecture. Neither shared-memory architecture nor shared-disk architecture performs scales very well on data-based applications. Also, they are very difficult to manage when number of processors and memories scale above a certain level. The network can also be another limitation that prevents shared-memory architecture and shared-disk architecture from performing well on big data-based applications. The network needs to have enough bandwidth to work effectively for thousands of processors. It is possible to build such a network but it will be expensive. So what is the most adaptable architecture that can easily fit into big data applications? Our answer is shared-nothing architecture. One of the most important reasons is that more and more high performance and low-cost commodity components are available right now, and they are improved a lot compared to twenty years ago. Another reason is that with high scalability and fault tolerance, shared-nothing architecture can fully be implemented into the database applications which can deal with the large-scale data problem. Even though shared-nothing architecture still has some pitfalls, some of pitfalls can be solved by the architecture that is developed later such like shared-disk combined with shared-nothing architecture, which is closer to most of the big data based applications' architecture. As shown from Figure 7, two shared-disk systems are linked to each other with the same hardware structure. The whole system layer was adopted from shared-nothing systems. This architecture provides shared-nothing architecture's high scalability and fault tolerance, and provides data sharing between multiple CPU processor. Research from Huaiming Song, XianHe Sun and Yong Chen [22] showed from a two node cluster up to a 16 node cluster, this hybrid combined

system is two times faster than the shared-nothing system. We can safely say that more and more companies are using such architecture to build their database or data analysis tool.

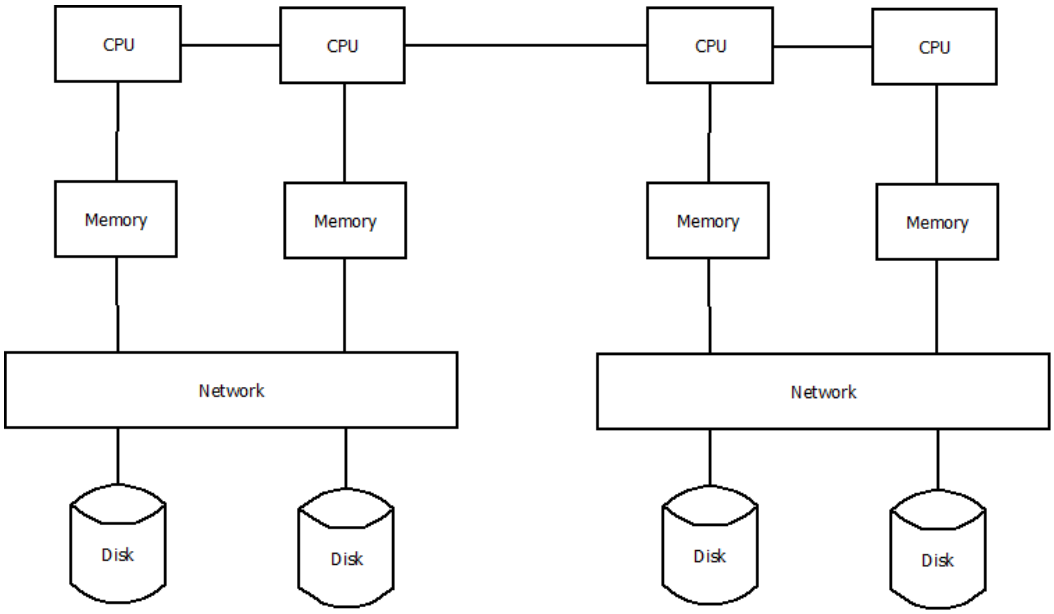

Figure 7. Shared Nothing & Shared Disk Combined System [20]

2.3 Symmetric Multiple Processing

Symmetric multiple processing [23] is a multiple processor system that has shared-memory architecture and is executed in a single operating system. In this system, each processor can work with any section of memory or disk shared.

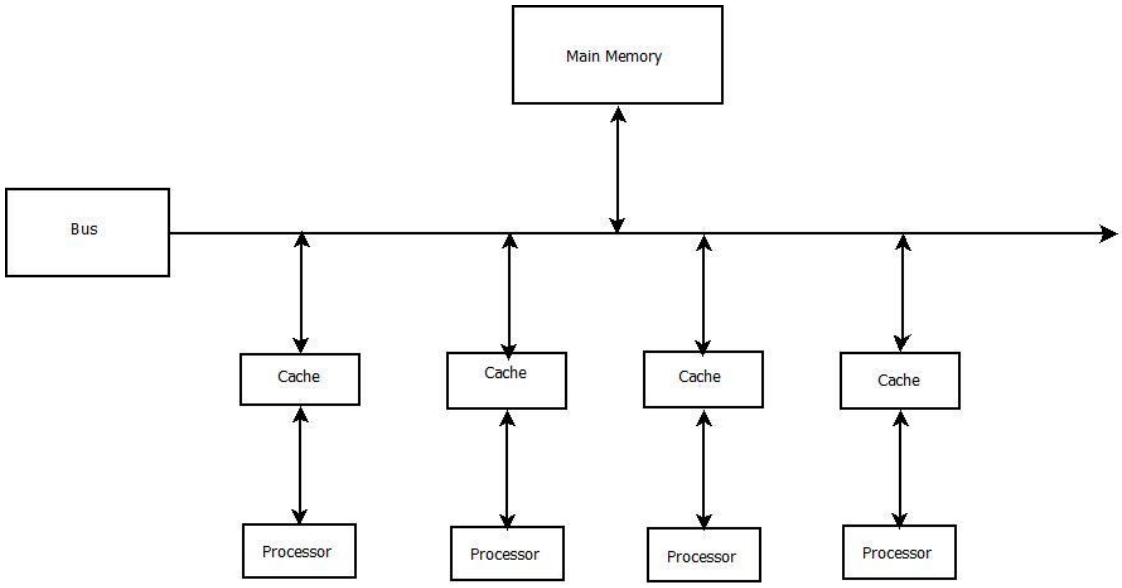

Figure 8. Symmetric Multiprocessor Systems [24]

Figure 8 shows how Symmetric Multiple Processing systems work. A shared bus and the shared memory are used by multiple processors. This means everything is processed on shared memory, disk, and I/O operations environments. This shared memory system has high-speed inner communication, memory and data sharing. Also, this system is very easy to manage physically. When big data emerged, symmetric multiple process system is not suitable for dealing with big data problem. When the bus is overloaded or when too many processors make data requests simultaneously, the system will generate data requests traffic. Because the system bus will become a bottleneck, it is hard to cover this limitation unless you spend most of your money to buy better hardware. This approach is to extend this limitation but not eliminate it.

2.4 Massively Parallel Processing

Massively parallel processing (MPP) is shown in Figure 9 is to use many processors to perform a single computation or a set of coordinated computations in parallel environments [25]. It is a strategy for dealing with large-scaling data. MPP allows each server or node that has its own memory and disk, and these nodes can share the workload. Compared to symmetric multi-processing (SMP), MPP also allows many databases to be searched in parallel and it is the infrastructure foundation for many of largest supercomputers right now. But in currently technology state, MPP systems still required high cost and complicated installation, which can only be afforded by largest companies and government organizations [26].

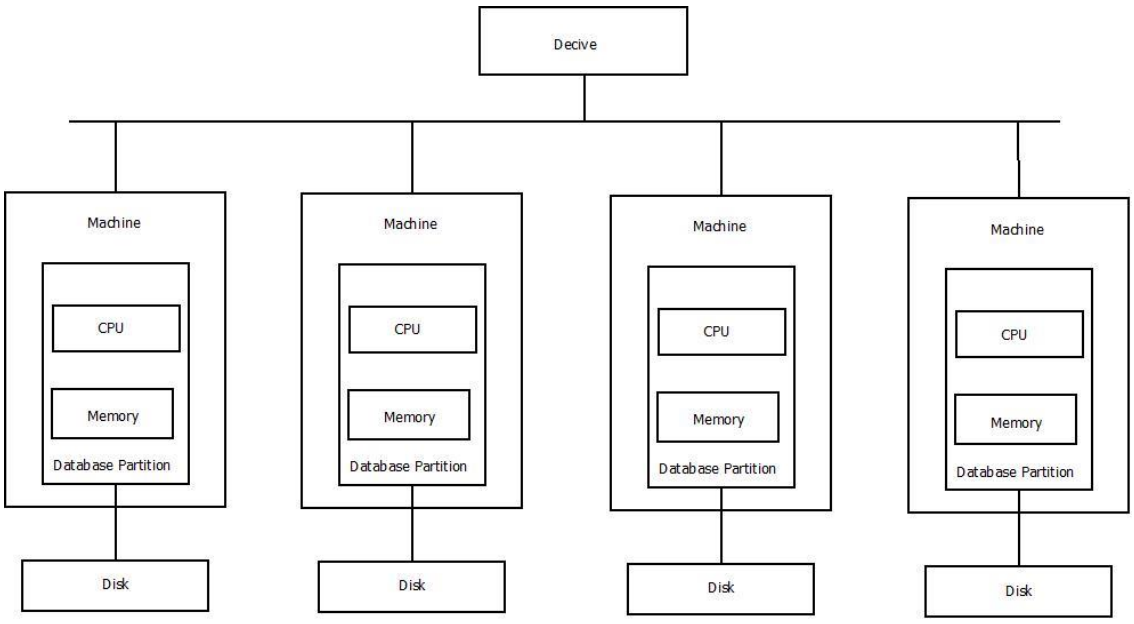

Figure 9. Massively Parallel Database Systems [25]

The most famous product for using MPP architecture is Microsoft SQL Server Parallel Data Warehouse (PDW). As shown from Figure 10, the parallel data warehouse has some different types of nodes to handle different actions:

- Control Node: This is an entry node for data read/write requests. Also, this node contains metadata about whole PDW system. From Figure 10, client drivers send all the requests to this node.
- Management Node: This is a node that handles all patching related actions, such as apply server service packs to Compute Node.
- Landing Zone: This is a node that allows PDW system to have such a high performance on data loading. The client can use a command line data loading tool called DWLoader [27] to bulk load data into PDW in parallel.
- Backup Node: This is a node that handles all backup actions.
- Compute Node: This is a special SQL Server instance that performs actual query processing. Every Compute Node is a SQL server instance.
- Storage Node: This is a node that stores all the data. Each Storage Node is connected to Compute Node directly and a Storage Node is connected to each other by the high-speed network.

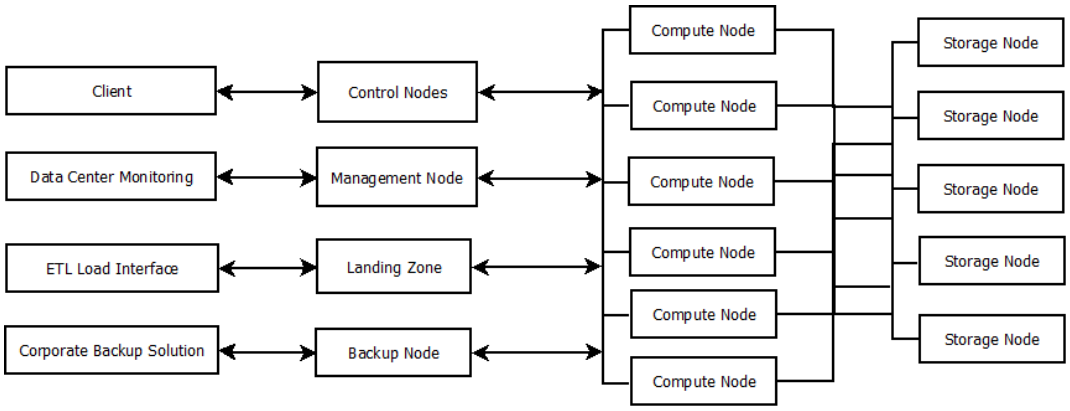

Figure 10. Microsoft Parallel Data Warehouse Architecture Diagram [28]

A massively parallel processing system allows PDW to have better performance when reading and writing data simultaneously. Also, it increases the performance when the system performs complex querying. Compared to the traditional Microsoft SQL server, which is a Symmetric Multiple Processing system, Microsoft SQL Server Parallel Data Warehouse contains a shared-nothing architecture, and it also contains MPP architecture. PDW has multiple physical nodes that can run its own instance of SQL server with its own CPU processor, memory, and disk. Microsoft SQL Server PDW can also be deployed on commodity hardware. Thus, it provides low-cost services per Terabytes. In spite of that, Microsoft SQL Server PDW is still expensive for some small companies. The total costs of hardware and installation are around 2 million dollars without software license [29]. According to the release of Microsoft SQL Server 2012 PDW release documentation, Microsoft Parallel Data Warehouse is 50 times faster than traditional data warehouse built on symmetric multi-processing database system in performance [30]. Except for Microsoft PDW, MPP has been widely utilized by other companies' products such as EMC Greenplum, Oracle Exadata, HP Vertica and IMB Netezza.

3. Parallel Processing Techniques and Frameworks

Recently 20 years, more and more parallel processing techniques and frameworks are coming out, and they are implemented and used in many areas such as government, healthcare, bank, weather, transportation, social media, and education. Data grows faster than what we can picture. The biggest challenge is how to handle large scale data. In this chapter, we present multiple parallel processing frameworks widely used in big data area. We compare and analyze the advantages and disadvantages of these parallel processing frameworks within the scope of the big data.

3.1 MapReduce and Hadoop

3.1.1 MapReduce

MapReduce [31] is a parallel programming model which was introduced for processing and generating large datasets. There are two functions in MapReduce program: map function and reduce function. Both map and reduce function are written by user. Map is used to accept input data and produce a set of intermediate (key, value) results, and then send the results to reduce function. Reduce function will accept the results and merge them together to output file. Every reduce function normally generated zero or one output. Even though MapReduce only provides two simple functions, the amount of large scale data analytical problem such like data mining, SQL query, data graphic processing, and machine language learning can be handled by it [32].

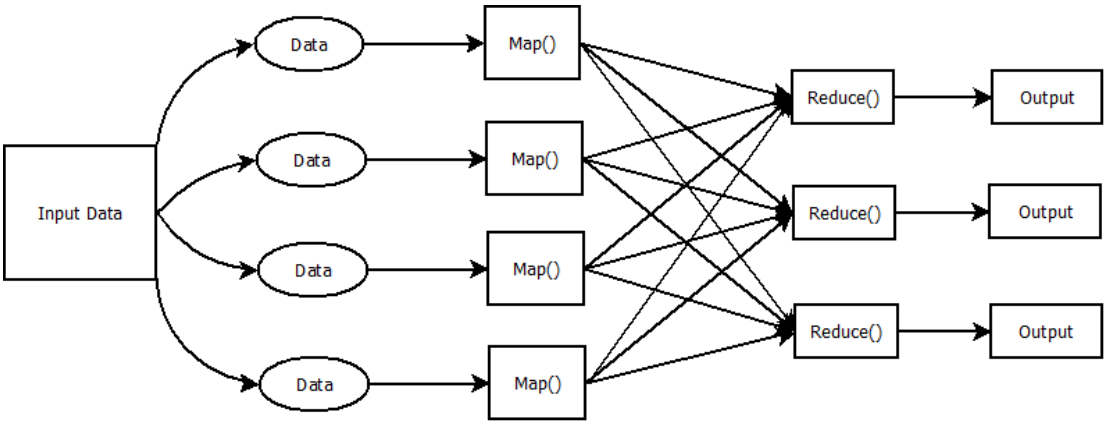

Figure 11: The MapReduce Programming Mode [31]

Figure 11 shows the MapReduce programming working flow. Input data are portioned by the system into multiple small trunks which take by map function as an input. And then, results of map function will be redistributed and shuffled. Those shuffled results are sent to reduce function through processing and stored into the result file. The authors of MapReduce introduce an example that is to count occurrences of word from large datasets [31]. The map function take input pair, in this case, key is document name and value is document contents, and perform a search for every word in the document content. The result of map function is every word and its associated occurrences in this content, map function will send the result to Reduce function after finishing processing. The reduce function sums up all of occurrences and generates a total number of occurrences for this word as an output.

3.1.2 Hadoop

Hadoop [14], [33] is a very popular open-source that can support applications, which process amount of data in a timely manner. Hadoop has been used in many companies like Amazon, Facebook, and Google. For processing data, Hadoop splits the amount of data into

small trunks and sends them to each of nodes which can process those required datasets in parallel under MapReduce program. There are many frameworks added into Hadoop right now, but the MapReduce and HDFS are core design of Hadoop.

Hadoop Distributed File System (HDFS) [33] is a file system that can run on low-cost machines with a highly fault tolerance. “Hadoop Distributed File System provides high throughput access to application data, and it is suitable for applications that have large data sets” [34].

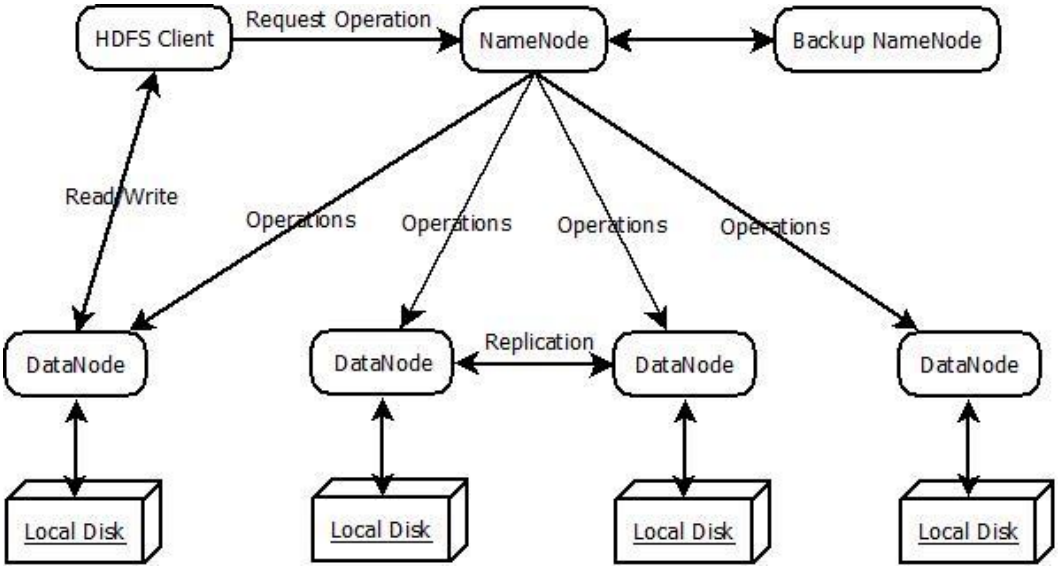

Figure 12. Hadoop Distributed File System Architecture [34]

As shown in Figure 12, all the operations in Hadoop Distributed File System are controlled by metadata in NameNode. Namespace operations, such as open or close files, and rename files or directories, are managed by the NameNode [35]. Some of HDFS will have the Backup NameNode, which is a backup for the whole file system once NameNode is shut down or disconnected. A user/client can set up block creation, deletion and replication request

for each DataNode through NameNode, and those block operations can be changed whenever the user or client want. Normally, an input file will be divided into one or multiple trunks and each trunk will be stored in every block in DataNodes. The DataNode also responds to the read and write requests from user/client. Each DataNode has its own local disk, so the DataNode also perform data replication between each other. Because Hadoop is an open-source framework, lots of implementation modules are added to the Hadoop echo system, such as Pig, Hive, and Spark. These frameworks will be introduced in this chapter later.

3.1.3 Performance of MapReduce and Hadoop

Table 2. Advantage & Disadvantage of MapReduce and Hadoop [14], [33]

Advantage	Disadvantage
Data locality	Network communication overload
Scalability	Real-Time processing
Own storage infrastructure(HDFS)	
Fault tolerance	

The advantages and disadvantages of MapReduce and Hadoop are summarized in Table 2. To gain high performance, MapReduce or Hadoop assigns those massive workloads to each DataNode or server which has required data stored. Data locality is an advantage, while at the same time a disadvantage for the performance. Hadoop does not recommend that we deploy data locally in Hadoop environment because the extra network communication overload will cause a performance bottleneck. The bigger the system becomes; the bigger performance issue will become. The second advantage for MapReduce and Hadoop is

scalability. MapReduce can widely spread the massive workload into multiple small trunks to hundreds and thousands of cheap machines and execute it in parallel. This advantage can be used to run applications in thousands of server nodes to deal with terabytes of data. The third advantage is the MapReduce and Hadoop has own storage infrastructure. Whenever a client wants to do some action with the data processing, Hadoop can quickly locate the data. The fourth advantage is fault tolerance, which is also important for Hadoop to deal with the big data problem. When MapReduce starts to spill amount of data into small trunks and distribute into multiple individual servers, the client can set up the replication function for those data and copy the data to another server in the system. So, if one node is dead, there is another copy that will be available for use.

Of course, there are some situations which are not suitable for using MapReduce and Hadoop. For example, real-time processing is not suitable for MapReduce and Hadoop. MapReduce and Hadoop is considered as a batch processing framework that requires the data were loaded into the distributed file system and then process data via MapReduce, and during the processing, the data were batched to each node for the job. Network overloading is also a limitation for using Hadoop, and whenever there is data processing request via the user, the amount of data will be shuffled over cross the network through server nodes. This will become a bottleneck on performance for MapReduce and Hadoop system.

In 2010, Dawei Jiang, Beng Ooi, Lei Shi and Sai Wu did a couple of tests on MapReduce performance via using Hadoop on Amazon EC2 cloud environments [32]. They tested the performance of on MapReduce from different perspectives which include Grep, Selection, Aggregation and Join on Max 100 nodes with 850GB storage and 7.5 GB memory

in Hadoop system, and compared with some of the results to the parallel database system (DBMS-X and Vertica). During the Grep and Aggregation task, Hadoop and MapReduce can obtain the same performance as DBMS-X but less effective than Vertica. Hadoop is also less effective compared to both DBMS-X and Vertica on Join task.

3. 2 Pig and Hive

For the database programming language, SQL is easier to write, modify and understand. But MapReduce programming needs developers to write a program by using a low-level programming language. To overcome this problem, the MapReduce community has started to work on some projects that allow MapReduce programming to perform some common data sets tasks such like Join operation by using high-level languages. Pig [36] and Hive [37] are two excellent projects, which are implemented via high-level languages combined with MapReduce model.

3.2.1 Pig

Pig is a programming tool that is designed for handling and processing large scale data on the top of MapReduce and Hadoop [33]. Pig can handle multivalued and nested data structures which are hard to be directly processed by using MapReduce. Pig is combined by two pieces: 1. Pig Latin. 2. Hadoop MapReduce compiling environment.

Pig Latin is a new programming language which is produced by Yahoo research team [36]. Pig Latin combines high-level querying language and low-level MapReduce programming language.

The data model of Pig Latin contains scalar type and complex type. The scalar type, which is atomic value, is the simple types that appear in most programming languages, such

as an integer, a string, and a number etc. The three complex types are Tuple, Bag, and Map as shown in Table 3. The Tuple is a fixed-length, ordered collection of pig data elements which can be any type of data. The Bag is an unordered collection of Tuple. The Map is a data item collection. Every item in Map is a (key, value) pair that has an item and its associated key. The data item in the map can be a scalar type or complex type. To maintain the efficiency of search, the best choice for the key is atomic type of data [36]. Map constants are formed using brackets to delimit the map. It also supports User-Defined Function which allows executed tasks written via Java or Python as well [33].

Table 3: Examples of Tuple, Bag, and Map in Pig Latin

Tuple	(‘apple, ‘pen’)
Bag	{(‘apple’, ‘pen’) (‘phone’, (‘apple’, ’pen’))}
Map	[‘age’ -> 20 ‘Fan’ -> {(‘apple’, ‘phone’)}]

Pig does not only include the advantage that Hadoop and MapReduce have, but also has an ability that allows the user to control execution steps via a set of operators which are provided by Pig [33]. It also decreases the development time especially considering the MapReduce job’s complexity, time spent and maintenance of the programs.

3.2.2 Hive

Hive [37], [38] is a data warehouse system that build on Hadoop. It provides multiple functionalities to help user to deal with Hadoop system, such as data query, data management,

and data analysis. Initially Hive was developed by Facebook, but now, it becomes a part of Apache Hadoop project, and has been widely used by many companies as a big data processing platform [33]. In Hive, there is a query language that pretty much like SQL, called HiveQL. It allows the user to plugin MapReduce scripts into queries as a custom option. During Hive processing, the HiveQL is transferred to MapReduce jobs and Hadoop distributed file system operations.

The data in the Hive was stored in tables [37], [38]. The tables are very similar to the table in traditional database system and each table has a corresponding Hadoop distributed file directory. Each table consists of many rows, and each row consists of many columns. Each column required an associated type which is a complex type or primitive type [37].

- Primitive Types: This type includes integers, floating point number and string.
- Complex types: This type includes associative arrays, lists and structs.
    - Associative arrays: Map<key, value>
    - Lists: List<element>
    - Structs: struct<name1: type1, name2: type2>

A complex associated type structure also can be combined by these three complex types, for example, List<struct<A: string, B: string>>> represents a list of structs that contains two string field named A and B.

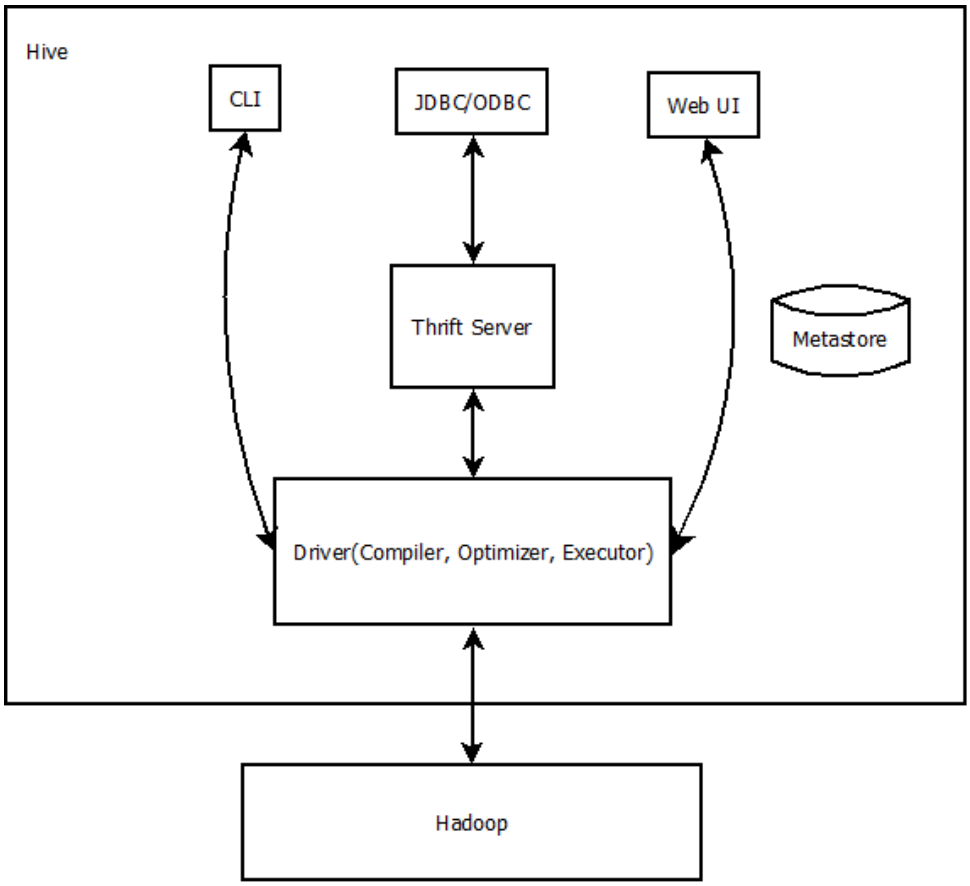

Figure 13: Hive Architecture [38]

As shown in Figure 13, Hive is a framework that is integrated with Hadoop system. How to make it easy to use is the number one priority. Metastore is the system catalog, and it is controlled by thrift server. It stores metadata about the tables, such as table schemas, locations, and associated types. Hive provides three interfaces which include Command line, web UI, and API(JDBC/ODBC). These interfaces allow an external user to interact with Hive by sending queries, instructions, and monitoring status requests. Other than these, Hive has its own thrift server that can transfer another programming language into HiveQL. Currently, Hive supports programming languages include C++, Java, PHP, Python, Perl and Ruby [37].

The life cycle of HiveQL was managed by Hive driver during compilation, optimization, and execution, and it also establishes the connection between Hive and Hadoop NameNode, and it can push operation commands into Hadoop system [38]. Hive is a tool used for scalability, extensibility, and job handling. It does not focus on how to speed up performance. The real-time query is not optimized, and even for the smallest job, it takes few minutes.

3.2.3 Performance of Pig and Hive

Hive and Pig are not designed for speed up performance. They are used as a High-level programming language or query language that can be added into MapReduce and Hadoop. But, we still need to know whether Pig or Hive for doing our tasks. From the research by Stewart, Thrinder, and Loidl [39], the performance of Hive (Version 0.4.0) and Pig (Version 0.6) under Hadoop with 32 nodes running on Linux machine is compared. They validated benchmarks from Apache that Hive gains the quickest runtime performance for every benchmark which includes on scaling input size, scaling processing units, and computation size per reduce task. But there is a conflict that based on the research from Benijamin and Dr. McBrien [40]. From their research, they ran performance testing on 6 nodes which have 2 dual-core intel Xeon CPU with 4GB memory. Pig's performance consistently beats Hive with the following results: Pig is 46% faster than Hive on arithmetic operation; Pig is 36% faster than Hive on Joining datasets. Who has the correct answer? No one is wrong. They just tested the performance of Pig and Hive from a difference perspective. When we decide whether to use Pig or Hive, we need to consider what kind of situation and environment that we are going to deploy and use. But for sure, there are some points that we need to notice:

1. A number of nodes will impact the performance between Pig and Hive. Both experiments [39], [40] are running on a smaller cluster which is not really conducting the performance result to show which one is better. Therefore, considering selecting either Pig or Hive as part of data analysis tool is really depends on how you set up the cluster environments and how many processors you are going to choose.
2. Depends on what kind of query you are going to perform and what kind of datasets you are going to deal with. Complexity of datasets and queries will be a big factor of performance.

3.3 Spark

Apache Spark [41], [42] is a cluster computing framework that initially developed by a research group from UC Berkeley. Spark is built out to deal with the problems that cannot be handled by MapReduce and Hadoop. MapReduce and Hadoop are very successful frameworks, but they contain a lot of pitfalls. When dealing with an algorithm or an application apply to iterative jobs, every MapReduce job have to reload the data from disk which cause massive delay [41]. The issue will cause those algorithms or applications cannot run efficiently by using MapReduce and Hadoop. Spark introduces two abstractions: Resilient Distributed Datasets and parallel operations used to handle Resilient Distributed Datasets.

3.3.1 Resilient Distributed Datasets and parallel operations

Resilient Distributed Dataset (RDD) [41], [43] is a read-only collection of datasets. RDD are divided into multiple trunks which can be recovered, and they stored in a set of machine nodes. Users can store RDD in shared memory across machine nodes to reuse it in

multiple parallel operations. RDD also can recover the lost partition of data. A dataset has enough information about how the lost data are derived from other datasets and then rebuilt those data. RDD balances the scalability, reliability, and reusability in a nice spot. This ability supports Spark to suit for a variety of applications.

Resilient Distributed Dataset could be created via four different approaches [41]:

1. The dataset which comes from a file system.
2. The data trunk which comes from parallelizing Scala [44] collection.
3. The dataset which is transferred from an existing RDD.
4. The dataset which comes from changing the persistence of an existing RDD. Normally, RDD in Spark are materialized due to the demand that being used in a parallel operation and it will be discarded from memory after use. But the Spark user can change this behavior via save the RDD in the cache, which is required enough memory to support or distributed it into file system.

When a programmer starts to use Resilient Distributed Dataset, they can perform several parallel operations as well, which means Spark supports those parallel operations [41]. A REDUCE operation can be used by driver program to combine dataset elements to produce a result. A COLLECT operation is very similar to the Map function that send all datasets elements to the REDUCE operation. The FOREACH operation is very similar to a loop so that each element in the dataset will be passed through a user provided function. Spark also supports two restricted shared variables [41] which are broadcast and accumulator. Spark allows programmers to create broadcast variables that collect a large read-only piece of data which are used in multiple parallel operations, and distribute these broadcast variables to each

node. Spark also allows programmers to create accumulator variables to act as a counter in MapReduce, so that it can provide a more convenient syntax for parallel sum.

3.3.2 Spark architecture

There are three type of components which are included in Spark as shown in Figure 14. They are driver program, management framework and nodes. User can create application by using Scale, Java and Python, and built it on Spark. Driver program is created by user or application, and required a SparkContext object to be included for connecting to the management framework, such as Spark standalone management system or Hadoop management tool YARN. Management Framework is used to manage resources in Spark cluster. Each node in Spark cluster will require an Executor which is deployed by user application. The Executor contains the cache memory and executes tasks. The SparkContext object can directly send user application requests to Executor to perform tasks. Spark can be deployed as a stand-alone server or on the Hadoop system.

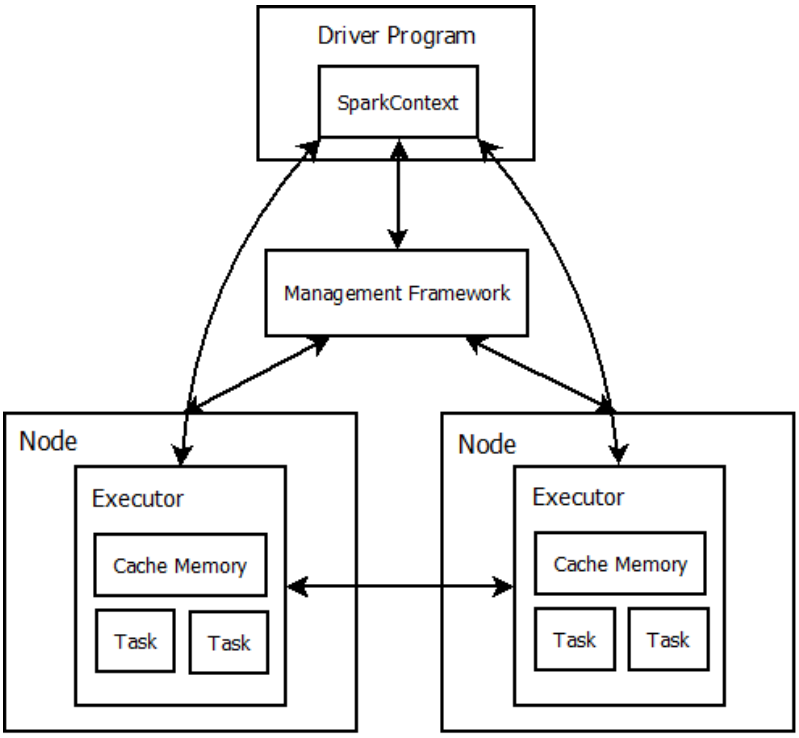

Figure 14. Spark Architecture [45]

3.3.3 Performance of Spark

Compared to Hadoop, it is much faster when performing iterative jobs. According to experiments presented by author of Spark [41], Spark is 10 times faster than Hadoop on iterative jobs, logistic regression. Authors of Spark use Spark to perform interactive query on 39 GB dump data of Wikipedia among 15 nodes. Because Spark store those data across in the memory, each query only takes 0.5 to 1 seconds compare to Hadoop. They also compared the preformance between Hadoop and Spark on logistic regression. It take 127s for hadoop to perform each iteration task, but only take 6s for Spark. The reason is the same, because of Spark use memeory to load the datasets instead of load data from file system.

3.4 Twister

Twister is a MapReduce extension framework which has many enhancements to support MapReduce works [46]. Twister adds some programming models which improve MapReduce's architecture and allows MapReduce to have more capabilities to execute more classes of applications. It provides a lot of features to support MapReduce computations as follows [47]:

- Distinction between variable and static data.
- Use publish-subscribe messaging method for communication.
- Configurable cacheable map or reduce Tasks.
- Support iterative MapReduce jobs.
- Use local disk to store data
- A lightweight program.
- User-friendly, provides a management tool to manage data and supports regular MapReduce computations as well.

Also, Twister group recently developed Twister 0.9 to support more generic usages, but we are not going to present that here since it only focuses on friendly usage and adds more extensions. Twister 0.9 is still considered as a young framework since it is only produced and supported by a small research group.

3.4.1 Programming model and architecture

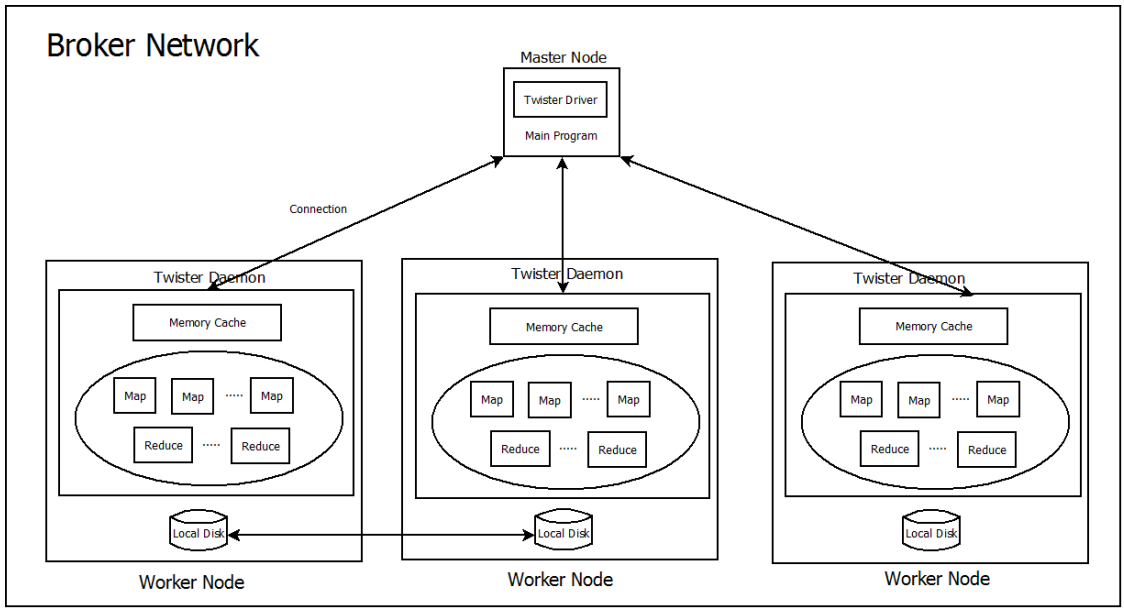

Figure 15. Twister Architecture and Broker Network [46]

To make Iterative MapReduce computation more efficiently, Twister builds a network which is called Broker Network as shown in Figure 15. In Broker Network, Twister runs their own Java virtual machine called Daemon, which runs on every worker node. Each Twister Daemon has its own memory cache and control multiple map and reduce jobs. Daemon also has responsibilities to notify worker node status and events controlling. The daemon could read data from the local disks of a node or receive data directly from another job via broker network. Normally, Twister assumes data files are separated by users or customers, then those small data files are directly sent to map or reduce job. Therefore, a large input data required to be divided by user into smaller multiple trunks for processing efficiency. Twister also allows user to send datasets directly to map job, but this approach is inefficient to pass large datasets due to network bandwidth limitation. These Daemons are connected to each other via Broker

Network for receiving commands and data. Twister provides a client driver which is included in Master Node that contains the programming API. This driver can send input data messages and commands to Daemon via Broker Network. Twister uses a publish-subscribe messaging method to deal with different types of communication requirements, which includes handling control events message, sending and receiving data between client and Twister daemons and transferring intermediate data between Map and Reduce tasks [46].

3.4.2 Performance of Twister

In Hadoop, each node only can execute either Map or Reduce job, however in Twister, the hybrid mechanism allows Map and Reduce job to be sent to one Daemon, which is a single Java virtual machine that controls multiple Map and Reduce jobs. Map jobs will directly push the result to Reduce tasks via the publish/subscribe messaging mechanisms. Those results are already buffered for Reduce tasks execution. Therefore, Twister directly can handle those intermediate data after Map processing and store them into cache memory for this purpose. If the result is relatively bigger, Twister will store the result in local disk instead of memory.

From the experiment results obtained by Zacharia Fadika [48], for iterative application tests which require multiple iteration execution on the same datasets, Twister showed two to five times faster compared to Hadoop because of the publish-subscribe messaging mechanism. In Hadoop, the data needs to be loaded between file system and memory during the iteration processing. But there is a limitation, according to the experiments by Zacharia Fadika [48], the size of transfer data between Map and Reduce is less than 93MB to ensure Twister gaining better performance over Hadoop. Increasing the data size over 93MB will cause

Twister to store the data into local disks rather than the memory. Zacharia Fadika' group also compared other areas between Twister and Hadoop as showed in Table 4.

Table 4. Comparison between Twister and Hadoop [48]

	Twister	Hadoop
Multiple Iterations	2 to 5 times faster	
CPU-Intensive	93% CPU utilization	89% CPU utilization
Speedup improvement (Nodes scaled up from 8 to 64)	factor of 7.5	factor of 4
Memory footprint	500MB data in memory	100MB data in memory
Fault tolerance	No support for fault tolerance	Support fault tolerance

4. Challenges for Companies and Businesses on Big Data

It is easy to see that more and more people are talking about “Big Data”, and more companies start to hire data analyst and try to convert massive unorganized datasets into business values. Many cloud services are produced such as Microsoft Azure, Google cloud, and Amazon web services. But still, the more exposure people get from “Big Data”, the more expectations people put on “Big Data”. Sometimes it is out of your expectation when you feel you can solve these challenges. In this chapter, we present multiple techniques challenges on big data area. These challenges include visualization, volume, variety, veracity and transmission speed. We also present possible solution for those challenges.

4.1 Visualization

The first challenge is visualization. When big data come out, people know it is there, but hard to see it. Even right now, the data that most of the companies get are some boring statistics. It is very important to use visualization to help people to get a complete view on big data. For example, Microsoft built a Digital Crimes Unit on Microsoft Cloud [49] to visualize information to detect cyber crime in real time and help people and organizations get safer.

Because the big data is unorganized, large and rapid growth, dynamics and scalability are two major challenges in big data visualization [50]. Of course, to fully take advantage on visualization of big data, not only those challenges mentioned above but also other challenges [51] listed as follows:

- Meeting the need for speed
    - Improve the hardware, such as increasing the memory, switching hard drive from HDD to SSD and adding powerful parallel processing machine. This is the easiest way to do but the cost is expensive. Another approach is to store data in RAM via grid computing.
- Understanding the data
    - Have the proper domain expertise in place and make sure people who are dealing with the dataset has a deep understanding.
- Data quality
    - Keep the datasets clean and stay in structure through the data process
- Displaying meaningful results
    - Integrate the meaningful datasets into a group, so that the group of datasets can be displayed effectively.
- Dealing with outliers
    - Remove them or separate them.

4.2 Volume, Variety and Veracity

4.2.1 Volume

Another challenge is the volume of big data. The data grows with an incredible speed from the government, social media, education, technology. How do we store these data? What is the cost for storage? Currently, solutions for storage on Big data is Parallel/distributed database system, distributed file system, and warehouse system.

Distributed file system is the most widely used, because of Hadoop's dominance on big data technology area, and it is cheaper when compared with the data warehouse systems. As mentioned in Chapter 2, Microsoft parallel data warehouse system is a shared nothing architecture and massively parallel database management system. It is 50 times faster than a regular parallel database, but it also more expensive than a regular SQL database. Compared to a regular SQL database server you can download for free on the express version or hundreds for a developer version. Microsoft warehouse system's cost is over $1 million. From a white paper research that was published by Value Prism Consulting [52], Microsoft provided the most valuable warehouse system by comparing with other four enterprise data warehouse systems from Pivotal, IBM, Oracle and Teradata according to the five-year net present value includes one-time appliance hardware costs, software license purchases, installation costs, annual maintenance & support, and management labor costs.

Table 5. Total Cost of Ownership for Enterprise Data Warehouse (US dollar) [52]

Cost	**Microsoft**	**Oracle**	**Pivotal**	**IBM**	**Teradata**
Appliance	1,330,100	18,476,000	6,385,300	2,235,000	2,798,200
Installation	10,200	10,500	11,900	Omit	8,000
One-time Cost	1,340,000	18,487,000	6,397,000	2,235,000	2,806,000
Maintenance + Support	373,200	4,042,700	1,340,800	335,000	657,600
Labor	125,500	266,900	110,500	496,800	583,600
Facilities	19,200	22,100	17,600	14,200	17,300
Annual License	578,000	4,332,000	1,469,000	846,000	1,259,000

From Table 5, we can clearly to see that Microsoft is the cheapest large data warehouse system compared with rest of four companies. But still, it is considered very expensive to academy research and small businesses. The benefit of using warehouse system is “unlimited storage space”. Basically, you can safely to say that your important data will never be deleted or eliminated due to the storage space issue. Also, technologies are being developed around in the big data area for almost 20 years since MapReduce came out, so the storage is not a real problem because of these two main reasons:

- More companies are willing to spend money on increasing hard drive, cloud and warehouse to storage their essential customer data.
- Hard drive is cheaper year after year.

4.2.2 Variety

For big data problem, the variety of data is as important as the volume of data. The amount of data could give people a huge of opportunities to start to build the data infrastructure. But the question is how to increase the variety of the data infrastructure, especially in machine learning, artificial intelligence, and geolocation. For those big information companies like Google and Microsoft, they have large customers base. They can gather the amount of feedback information or product runtime data from their products, therefore improving their products. But for small businesses, they do not have enough money to build a large warehouse system or have a large of customers. How do they start to build their own data infrastructure and variety? Currently, the most common way is using cloud technology which is based on parallel/distributed infrastructure. With reasonable price and

infinite virtual cloud machine, the small companies also can start to build its own business big data.

Improving the quality of data is considered another challenge. The most noticeable problem with big data is that data are large and unorganized, which is against lots of technique models and methods. These models and methods are based on structured data. However, most of the sources of big data come from multiple varieties inbound sources. Data are not well-prepared for usage, like massive web server traffic log, location information, source devices and end device information, etc. To get data to be ready for usage, a method must be applied like sorting. Also, to clean unstructured data up it will take more time than that analyzing for [53].

Of course, we need to have new technique modes or methods to deal with these issues. MapReduce and Hadoop have provided us a basic idea about how to deal those unstructured data, therefore lots of big data applications use them as infrastructure and build above them. Even MapReduce and Hadoop can deal with unstructured data. It is better to improve the quality that would fit into currently technique models and methods.

4.2.3 Veracity

The veracity of big data also should be considered one of the challenge. Not all of data came from trustworthy, reliable source. The data measured or collected from any source should be detected to ensure it is trustworthy before any possible corruption or manipulations [54]. In most of the case, the raw data collected from original sources need to be filtered or pre-processed to avoid obvious irrelevant information. Due to its large volume, the raw data are very difficult to be filtered or pre-processed adequately. The larger data volume is, the

more difficult filtering becomes. The velocity of processing will be impacted due to the size of data too.

Currently, the best technology software to overcome this challenge is Hadoop and cloud service. With capabilities of Hadoop and cloud, filtering processes of the raw data are guaranteed in a relative speed. Even though it is not fast enough, but Hadoop and cloud can relieve the velocity problem which is a massive factor to filter those unstructured data.

4.3 Transmission Speed

The transmission speed of data is considered the bottleneck of the performance during the data processing. As we know, distribute file system and network sharing is good enough for storing the data but not really designed for transmission speed. Lots of parallel frameworks rely on them to process the big data related issues. How is the performance about these frameworks? A very interesting research [55] has compared Hadoop and parallel SQL database management system based on a 100-node cluster system. The result showed SQL database management system is significantly faster than Hadoop, but spent more time to load the execution data. Not like parallel SQL database management system, MapReduce program needs to scan all the tables and then perform querying.

Of course, the transfer speed can be improved from hard drive from HDD to SSD if money is not counted into the consideration. There is a very interesting research [56] that mainly discussed the impact on MapReduce& Hadoop when using different storage hardware like HDDs and SSDs. Within the same Sequential R/W Bandwidth (1300 MBPS) HDD-11 and SSD, this research talked about the difference from two perspectives: establish a brand-new cluster, and improve existing cluster. SSDs is significantly improved compare to HDDs

on MapReduce shuffle and HDFS read and write, but it did not really impact too much if it is CPU related jobs. One more important factor that SSD was not widely used because of its costs. With the same bandwidth and smaller capacity, SSD is 3 times expensive than HDD-11 [56].

4.4 Solution for Challenges

When technologies move to the Cloud computing, everything is changed and looks much easier to people. The easiest way to explain the purpose of the cloud is to use other available machines for your own tasks. Cloud service is dynamic and scalable. It can dynamically scale to meet user's requirements, instead of having user or companies to deploy their own management sources. Using cloud service saves more physical spaces and labor sources. Many major companies have started to use cloud services as part of their own businesses, because storing data in the cloud space is more convenient and cost less. Under the massive cloud infrastructure, large IT companies integrated cloud computing into their own products, deployed their services and applications into the cloud, therefore providing a convenient approach for their customers to use, such as Microsoft Azure, Amazon EC2, and Google Cloud. Many parallel processing frameworks have been integrated with cloud services, such as Hadoop and Spark. It is very convenient for customer to get a brand new Hadoop server with customized configurations on Cloud services.

The biggest reason to choose cloud service is not only its storage spaces but also the speed of the processing requested operations [53]. With thousands of machine nodes, unlimited CPU, RAM and other resources, cloud service can provide more than people's imagination. The main limitation of the speed probably is the network bandwidth, because the

restrictions of physical difficulty to move hundreds of terabytes of pebibytes of data across the network. Companies can save money on hardware by using cloud service but they must spend money to improve network bandwidth. The cost will not be high for small applications but can be significantly high for data-intensive applications.

5. Conclusion

Without a doubt, parallel processing has been so successful in dealing with big data. As a parallel processing programming model, MapReduce is the main power to drive the big data technologies, but it still contains lots of pitfalls, which result in lots of extension technologies or frameworks, like Spark, Twister, Pig, and Hive. MapReduce also cannot store the big data so that all the MapReduce applications which deal with big data problems require a large-scale file system or database systems that allow input and output data retrieved or stored efficiently [15]. To choose these frameworks wisely to handle different situations will be important. If you want a framework to help you handle interactive related jobs, Spark will be your best choice over Twister or MapReduce, because Spark is more mature than Twister and much faster than MapReduce. Twister is a young project which is still under development and mainly does not have fault tolerance. When considering integrating high-level scripting languages on Hadoop to help performing data queries, Pig and Hive will be the best choice. Cloud service and database warehouse systems improve big data processing in a completely new level compared to MapReduce and Hadoop. Today, most of the big data technology companies deploy their own applications on cloud servers, which are integrated with Hadoop and Spark, two of most effective big data frameworks currently. More space, cheaper and easier to maintenance are three main reasons for companies to build their own data systems, especially for small and middle business.

In this paper, we discussed parallel systems which are currently used for big data processing. We first introduced parallel database systems and their own architectures and then compared the different architectures of parallel database systems and showing their

advantages and disadvantages. Second, we talked about parallel frameworks and systems used in the big data area. MapReduce and Hadoop, and its extension frameworks are disscussed in detail. Finally, we discussed the potential challenges on multiple technique areas according to the characteristics of big data and possible solution for those challenges. There are lots of parallel data processing systems which are not included in this survey, but future research opportunities can be added as well.

References

[1] G. Noseworthy, "Infographic: Managing the big flood of big data in digital marketing," 24 April 2012. [Online]. Available: http://analyzingmedia.com/2012/ infographic-big-flood-of-big-data-in-digital-marketing/.

[2] T. H. Davenport and J. Dyche, *Big Data in Big Companies*. SAS Institute, Inc., 2013.

[3] C. Snijders, U. Matzat, and U.-D. Reips, "'Big data': Big gaps of knowledge in the field of internet science," *International Journal of Internet Science*, vol. 7, pp. 1-5, 2012.

[4] "5v's of Big data," 10 July 2015. [Online]. Available: https://onlinebigdatahadoop. wordpress.com/2015/07/10/5vs-of-big-data/.

[5] "Cost of hard drive storage space." [Online]. Available: http://ns1758.ca/winch/ winchest.html.

[6] A. McAfee and E. Brynjolfsson, "Big data: The management revolution," pp. 59-68, October 2012.

[7] M. Lee, "Alibaba's 24-hour online sale rakes in over $9 billion GMV," 12 Nov 2014. [Online]. Available: http://www.alizila.com/alibabas-24-hour-online-sale-rakes-over-9-billion-gmv.

[8] "Extracting business value from the 4 V's of big data," IBM. [Online]. Available: http://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data.

[9] B. Marr, "Big data: The 5 Vs everyone must know," 6 3 2014. [Online]. Available: https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know. [Accessed 9 9 2015].

[10] R. Peplau and R. Barbedo, "Where did all this Big data come from," 29 October 2013. [Online]. Available: http://www.nonlinearcreations.com/Digital/how-we-think/ articles/2013/10/Where-did-all-this-Big-Data-come-from.aspx.

[11] A. Grama, A. Gupta, . V. Kumar, and . A. Gupta, *Introduction to Parallel Computing* (2nd Edition). Boston: Pearson, 2003.

[12] "Parallel processing & parallel databases," [Online]. Available: https://docs.oracle.com/cd/A58617_01/server.804/a58238/ch1_unde.htm#2934.

[13] K. H. Lee, Y. J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: A survey," *ACM SIGMOD Record*, vol. 40, no. 4, pp. 11-20, 2011.

[14] "Hadoop." [Online]. Available: https://hadoop.apache.org/.

[15] C. Dobre and F. Xhafa, "Parallel programming paradigms and frameworks in Big data era," *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 710-738, 2014.

[16] B. Davis, "How much data we create daily," 24 11 2013. [Online]. Available: http://blogs.teradata.com/international/how-much-data-we-create-daily/. [Accessed 1 11 2015].

[17] T. Hey, S. Tansley, and K. Tolle, *The Fourth Paradigm: Data-intensive Scientific Discovery*. Microsoft Research, 2009.

[18] G. Bell, J. Gray, and A. Szalay, "Petascale computational systems," *Computer* , vol. 39, no. 1, pp. 110-112, Jan 2006.

[19] D. Dewitt and J. Gray, "Parallel database system: The future of high performance database systems," *Communications of the ACM*, vol. 35, no. 6, pp. 85-98, June 1992.

[20] "Parallel hardware architecture." [Online]. Available: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SPS73/chap3.htm#shdisk.

[21] R. D. Sloan, "A practical implementation of the data base machine - teradata DBC/1012," *0073-1129-1/92 IEEE,* pp. 320-327, 1992.

[22] H. Song, X.-H. Sun, and Y. Chen, "A hybrid shared-nothing/shared-data storage scheme for large-scale data processing," IEEE, 2001.

[23] P. H. Van der Veen, "Symmetric multi-processor system," US Patent US7103631 B1, 5 9 2006.

[24] "Symmetric multiprocessing," 26 April 2017. [Online]. Available: https://en.wikipedia.org/wiki/Symmetric_multiprocessing.

[25] A. Grishchenko, "Distributed systems architecture," 13 July 2015. [Online]. Available: https://0x0fff.com/hadoop-vs-mpp/.

[26] G. Edmondson, "Massively parallel processing and the parallel data warehouse," 15 April 2012. [Online]. Available: https://garrettedmondson.wordpress.com/2012/04/15/massively-parallel-processing-and-the-parallel-data-warehouse/.

[27] B. Kess, "dwloader command-line loader," 14 March 2017. [Online]. Available: https://msdn.microsoft.com/en-us/sql/analytics-platform-system/dwloader.

[28] "Introduction to Microsoft SQL server Parallel Data Warehouse (PDW)," 8 May 2014. [Online]. Available: http://binaryworld.net/blogs/pdw-presentation-microsoft-parallel-data-warehouse-aps-analytics-platform-system/.

[29] J. Serra, "Microsoft SQL server Parallel Data Warehouse (PDW) Explained," 26 8 2011. [Online]. Available: http://www.jamesserra.com/archive/2011/08/microsoft-sql-server-parallel-data-warehouse-pdw-explained/.

[30] "Microsoft SQL server 2012 Parallel Data Warehouse," Microsoft Corporation., 2013.

[31] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107-113, January 2008.

[32] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The Performance of MapReduce: An in-depth study," in *The 36th International Conference on Very Large Data Bases*, Singapore, 2010.

[33] T. White, *Hadoop--The Definitive Guide*. M. Loukides, Ed., Sebastopol, CA: O'Reilly Media, Inc, 2009.

[34] D. Borthakur, "HDFS architecture guide," 04 08 2013. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

[35] J. Hanson , "An introduction to the Hadoop Distributed File System," 01 February 2011. [Online]. Available: https://www.ibm.com/developerworks/library/wa-introhdfs/.

[36] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: A not-so-foreign language for data processing," in *SIGMOD'08*, 2008.

[37] A. Thusoo, J. . S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, . P. Wyckoff, and R. Murthy, "Hive–A petabyte scale data warehouse using," in *ICDE Conference 2010*, 2010.

[38] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, . P. Wyckoff, and R. Murthy, "Hive--A warehousing solution over a Map-Reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626-1629, August 2009.

[39] R. J. Stewart, P. W. Trinder, and H.-W. Loidl, "Comparing high level MapReduce query languages," in *APPT'11 Proceedings of the 9th International Conference on Advanced Parallel Processing Technologies*, 2001.

[40] B. Jakobus and P. McBrien, "Pig vs Hive: Benchmarking high level query," IBM, 2014.

[41] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, Boston, 2010.

[42] "Spark," [Online]. Available: http://spark.apache.org/.

[43] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for In-memory cluster computing".

[44] "Scala programming language," [Online]. Available: http://www.scala-lang.org/.

[45] "Spark 2.1.0 cluster mode overview." [Online]. Available: http://spark.apache.org/docs/latest/cluster-overview.html.

[46] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative MapReduce," in *HPDC '10 Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, Chicago, 2010.

[47] J. Ekanayake, J. Qiu, H. Li, B. Zhang, and G. Fox, "Twister--Iterative MapReduce," [Online]. Available: http://www.iterativemapreduce.org/.

[48] Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan, "Benchmarking MapReduce implementations for application usage scenarios," in *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*, 2011.

[49] "Microsoft Azure," Microsoft, [Online]. Available: https://azure.microsoft.com/.

[50] L. Wang, G. Wang, and C. A. Alexander, "Big data and visualization: Methods, challenges and technology progress," *Digital Technologies,* vol. 1, no. 1, pp. 33-38, 2015.

[51] "Five big data challenges and how to overcome them with visual analytics," SAS Institute, Inc, 2013.

[52] "Microsoft analytics platform system delivers best TCO-to-Performance," Value Prism Consulting, 2014.

[53] A. Adrian TOLE, "Big data challenges," *Database Systems Journal* , vol. IV, pp. 31-40, 3/2013.

[54] S. Yin and O. Kaynak, "Big data for modern industry: Challenges and trends," *Proceedings of the IEEE*, vol. 103, no. 2, pp. 143-146, Feb 2015.

[55] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," ACM, Providence, 2009.

[56] K. Kambatla and Y. Chen, "The truth about mapreduce performance on SSDs," 12 March 2014. [Online]. Available: http://blog.cloudera.com/blog/2014/03/the-truth-about-mapreduce-performance-on-ssds/.

[57] S. Sakr, A. liu, D. M. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, vol. 13, no. No.3, pp. 311-336, 2011.

[58] S. Pallickara, G. Fox, and J. Ekanayake, "MapReduce for data intensive scientific analyses," in *ESCIENCE '08 Proceedings of the 2008 Fourth IEEE International Conference on eScience*, 2008.