The Fault Tolerance of Big Data Systems

Xing Wu[1,2(✉)], Zhikang Du[1], Shuji Dai[1], and Yazhou Liu[2]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China
{xingwu,duzhikang,daishuji}@shu.edu.cn
[2] Key Laboratory of Image and Video Understanding for Social Safety,
Nanjing University of Science and Technology, Nanjing, China
yazhouliu@njust.edu.cn

Abstract. When the size of the data itself becomes part of the problem, big data era is approaching. Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis. Fault tolerance is of great importance for big data systems, which have potential software and hardware faults after their development. This paper introduces some popular applications and case studies of big data mining. The architecture of big data's individual components has parallel and distributed features, including distributed data processing, distributed storage and distributed memory, this paper briefly introduces Hadoop architecture of big data systems. Then presents some fault tolerance work recently in the big data systems such as batch computing, stream computing, Spark and Software defined networks, which shows great efforts to the capability of massive big data systems, and makes some comparison with each other.

Keywords: Fault tolerance · Big data · Hadoop · Spark · SDN

1 Introduction

Recently, big data becomes a highlighted buzzword in industry. Moreover, big data mining has almost immediately followed up as an emerging, interrelated research area. Databases, data warehouses, data marts and other information management technologies were about to solve the problem of large scale data.

However, "Big data" has become hot as an exclusive noun due to the rapid development of Internet, cloud computing, mobile and Internet of Things in recent years. Ubiquitous mobile devices, RFID, wireless sensors generate data all the time; hundreds of millions of users of Internet services always generate a huge amount of interactive data. The amount of data to be processed is too large, thus the business needs and competitive pressures require a real-time and effective data processing.

The traditional techniques cannot deal with such a large scale of data to satisfy the real business needs. Thus a number of new technologies have been developed and adopted, which includes distributed cache, distributed database, distributed file system, distributed storage scheme, no-SQL databases and so on. Among all these new technologies, fault tolerance is an inevitable part for big data systems.

J. Cao and J. Liu (Eds.): MiPAC 2016, CCIS 686, pp. 65–74, 2017.
DOI: 10.1007/978-981-10-3996-6_5

All big data systems need tolerate software and hardware faults remaining in the system after its development, which will benefit the systems in different ways including failure recovery, lower cost, improved performance and etc. A fault tolerance is a setup or configuration that prevents a computer or network device from failing in the event of an unexpected problem or error [1], as illustrated in Fig. 1. To make a computer or network fault tolerant requires users or companies to think how a computer or network device may fail and take steps that helps prevent that type of failure [2].

Fig. 1. Fault, error and failure

The paper is group by the following sections. Section 2 describes some popular applications of big data, including the applications in engineering, science, transportation and other fields. Section 3 discusses the technical architectures of big data systems, and uses Hadoop as an intuitive example. Section 4 summarizes the fault tolerant mechanisms and methods of some popular architectures of big data systems. Section 5 draws the conclusion of this paper.

2 Applications of Big Data

2.1 General Applications

Development of large data industry will promote the development of the world economy which has far-reaching significance from extensive to intensive changes to enhance the competitiveness of enterprises and the impact of the government's management capacity. Gathering the mass original data together, by the intelligent analysis and data mining technology, we analysis the potential of the data to forecast the development trend of the future things, helping people to make correct decisions to improve the operation efficiency of various areas and get greater benefits. Usually, the application of big data is the largest commercial areas, especially in electronic commerce. Its large market and its huge amount data from the market are particularly applicable to large data analysis techniques to predict and analysis to reduce costs, improve efficiency. In addition, the transportation, energy, electronics and other fields have conducted extensive research and application.

2.2 Case Studies of Big Data

There are some famous case studies listed in this paper.

Science Research. The data flow of the Large Hadron Collider (LHC) in experiments consists of 25 petabytes before replication and reaches up to 200 petabytes after replication.

Public Administration. The Obama administration project is a big initiative where a government is trying to find the uses of the big data which eases their tasks somehow and thus reducing the problems faced. It includes 84 different Big data programs which are a part of 6 different departments.

Business. Amazon uses data storage and processing 500,000 third-party electricity supplier sales data, optimizing the sales process and reduces costs.

Energy. With the analysis technology of big data, electrified wire net connects GPS and GIS and detects the fault of power transmission lines made by lightning and electrical traveling waves to make sure the reliability of electrical energy supply.

Traffic. Using of large data storage and analysis of the state of technology in aviation engine sensor data acquired thousands, such as Oil temperature, vibration frequency data, monitoring and forecasting the engine health condition, to ensure its reliable operation of the route running. Similar technology is also used in the automotive and motorcycle.

Health. The use of technology to monitor CT scanners and other large medical equipment status data, data processing a large number of sensors to ensure that services are available [2].

3 The Architecture of Big Data

Big data systems, have a basic part of the storage, processing, memory, network, etc. But based on the "4V" features of big Data technology, architecture of big data individual components has parallel and distributed features, including distributed data processing, distributed storage and distributed memory.

The big data platform Hadoop [3] utilizes MapReduce architecture to achieve a distributed data processing, in conjunction with HDFS distributed file system to achieve efficient, fault-tolerant and stable large data solution, as shown in Fig. 2. HDFS is a fault-tolerant and self-healing, distributed file system, the purpose of the standard server clustering into a large-scale expansion of the data pool. HDFS is the working load of large-scale data processing, engage in scalability, flexibility and throughput and specialized development. It accepts any data format for high bandwidth flow is optimized, can be extended to the environment in the 100 PB data over the deployment.

In HDFS, data replication in multi nodes will protect and maintain the computing performance. MapReduce is a highly scalable, parallel processing architecture, which is connected with HDFS and works together. The MapReduce and Hadoop, the calculation is executed in a data storage place, instead of moving the data to calculate the execution. The same physical data storage and computing nodes in the cluster on coexist. MapReduce can handle very large amounts of data, through the advantages of the nearest data, which is not the bottleneck bandwidth traditional limitation. The MapReduce will work load, divided into several parts and can be executed in parallel.

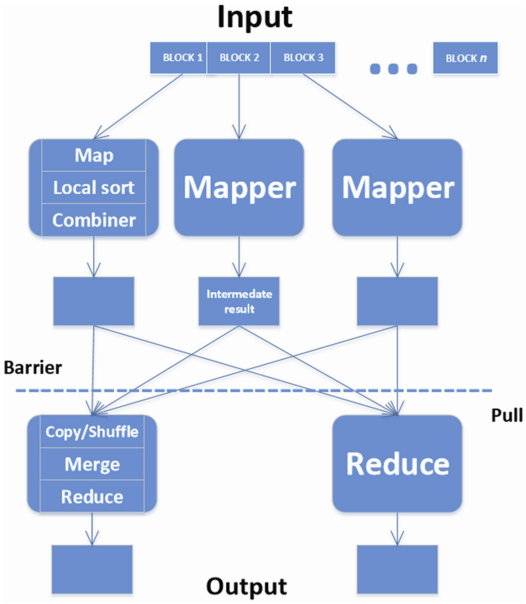

Fig. 2. The architecture of Hadoop

The traditional big data processing system use the disk to store the data. In recent years, with a substantial decline in the price of the hardware memory and real-time data processing based on WEB. In the face of big data processing needs more large-scale software vendors have introduced the database system based on their memory, such as HANA from and TimesTen from Oracle.

The same Spark big data platforms based on distributed memory also emerge as the times require. Big data platform Hadoop Spark and traditional distributed computing based on MapReduce is realized, but not for Hadoop Spark is to establish a set of distributed memory system, the task can be intermediate results stored in memory instead of distributed file system, and the need for iterative operation for distributed computing can achieve more efficient calculation. The core of Spark is to establish a flexible distributed data set RDD (Resilient Distributed Dataset). RDD is for distributed memory abstraction, RDD represents has been partitioned and can be operated in parallel data collection, RDD can be stored in memory, eliminating the need for large amounts of disk MapReduce operation. While its level of abstraction to avoid the direct operation of the memory, which can achieve the underlying hardware failures overshadowed by automatic reconfiguration for fault tolerance mechanisms.

4 Fault Tolerance in Big Data

The calculating pattern of big data is divided into batch computing and stream computing [4]. We first calculate the mass data storage, and then the static data of stored centralized computing. Hadoop is a typical batch of big data computing architecture, static data is

responsible for the HDFS distributed file system storage, and is assigned by MapReduce computational logic to each data node for data calculation and value discovery. In stream computing, we cannot determine the arrival time and the arrival of the order of the data, all the data cannot be stored. Therefore, no streaming data storage, but when the flow of data in real-time arrival data is calculated directly in memory. Therefore, streaming data is no longer saved, but immediate real-time data is calculated in the memory when data of the flow arrival directly.

Such as Storm of Twitter [5], S4 of Yahoo, they are typical of the streaming data computing architecture, the data was calculated in the topology in task, and outputs the valuable information. Flow calculation and batch calculation are respectively applicable to different big data application scenarios: For some application scenarios, such as storage before computation, real-time demand not high, at the same time, the accuracy and completeness of the data being more important, batch computing mode is more suitable. For some application scenarios, such as without first storage, directly data calculation, real time being very strict, but the accuracy of the data requiring a little loose. Due to the application of the memory database, memory database fault tolerance has become a hot issue. Spark etc. are based on the calculation of distributed memory, using abstract RDD technique to realize fault tolerance mechanism to deal with memory fault of distributed memory in the system. In addition to the above two models of fault tolerance, a new method of fault tolerance appears recently, namely SDN (Software Defined Networking) fault-tolerant technology [6]. A level of abstraction will be established in the network hardware, such as routers, switches, gateway and other infrastructure, a virtual grid structure given software, namely SDN. All big data platforms run on SDN, while SDN is to deal with failure on the network. Thus the big data can be free from network fault tolerance, and we will focus on software failure.

4.1 Fault Tolerance in Batch Computing

In Hadoop batch calculation model, the applications of two main fault tolerant mechanisms to deal with failure are the data replication and the rollback mechanism [7].

The Data Replication. In data replication mechanism, a copy of the data will be in several different data nodes. When it needs data replication, any data node, that its communication is not the busy can copy data. The main advantage of this technology is that it can instantly recover from failure. But in order to achieve this kind of fault tolerance, storing data in different nodes will consume large amounts of resources, such as a waste of large amounts of memory and resources. When copying across different nodes, there is the possibility of data inconsistency. But the technology provides instantaneous fault recovery fast. Compared with the rollback method, this method is used more frequently.

Rollback Mechanism. The copy report will be saved in a fixed time interval. If failure occurs, the system is just to back up a save point, then starts the operation again from that point. The method adopts the rollback concept, that is, the system will return to the previous work. But this method increased the execution time of the whole system,

because the rollback need to back up and to check on a save a consistent state, thus increasing the time. Compared with the first method, defects of the method is too time consuming, but needs less resources.

4.2 Fault Tolerance in Stream Computing

In stream computing system, there are four kinds of strategy to realize fault tolerance mechanism. Those are passive standby, active standby, upstream backup and a recent study of operator state management [8].

Passive Standby. System will be regular to back up the latest state on the master node to a copy of the replica node. When fault occurs, the system state will be restored from the backup data. Passive replication strategy support the case with data load higher, throughput larger, But the recovery time is longer, backup data can be saved in the distributed storage to reduce recovery time. This way is more suitable for precise data recovery, uncertainty computing applications. In the current, it is widely used in the calculation of streaming data.

Active Standby. When system transmits data for the master node, it also transmits a copy of the data for a copy of the node at the same time. When the master node failed, a copy of the node completely takes over the work, and the deputy nodes need to assign the same system resources. Fault recovery time is shortest in this way, but smaller data throughput.it also wastes more system resources.

Upstream Backup. Every master node is recording its own state and the output data to a log file. When a master node failed, the master node of upstream will replay a copy of the data in a log file to the corresponding node, for recalculating the data. They need longer time to reconstruct the state of the recovery, so that fault recovery time tends to be long. As system resources are scarce, upstream backup strategy is a better option in a state of the circumstances of fewer operators.

Operator State Management. This method will display the state operator to stream processing system through state management main types, And according to the time interval, it will back up its state to the upstream nodes. In any malfunction, the system will spread laterally to replace the fault node, and recover from the upstream node state. This method does not need longer reconstruction time, but also take up less computing resource.

4.3 Fault Tolerance in Spark

Compared with the parallel file system, the big data system using memory access speed, based on distributed memory, has more efficient data processing ability. Its face fault-tolerant mechanism will be different due to different ideas of this. For example, the RDD memory abstraction technique was used to realize fault tolerance mechanism for Spark framework [9].

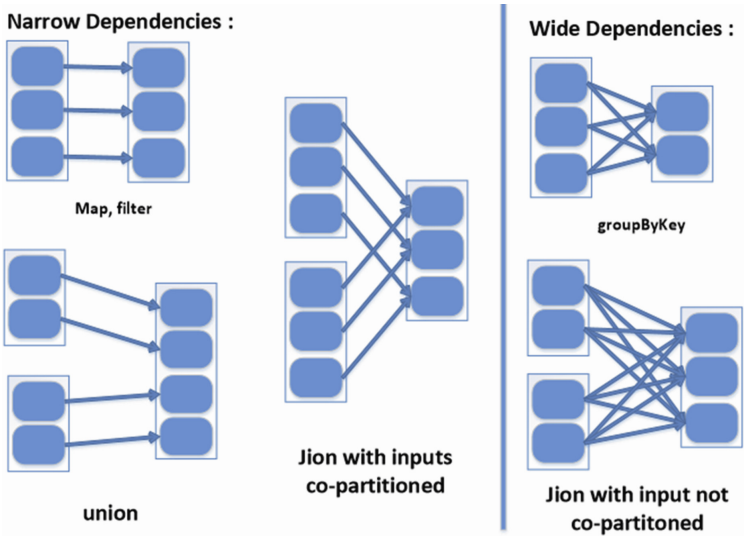

Fig. 3. Two types of dependencies in Spark

RDD (Resilient Distributed Dataset) stored all calculation data in a Distributed memory. RDD stored data in the cluster of memory by partition way. There are two kinds of operation: The Transformation and The Action. Transform is the operation that a kind of RDD is converted to another RDD, similar to Hadoop Map operation, whose operator definition is rich, including map, join, filter, groupByKey operations and so on. Action is similar to Reduce in Hadoop. The output is an aggregation function values such as the count, or a collection [10].

The Spark uses two methods to realize fault tolerance. One method is traditional checkpoint, which is to restore the backup RDD data set. This method will store RDD in file system on a regular basis, similar to Hadoop. Another method is to use lineage to realize fault tolerance mechanism. This method is used to establish the Transformation operation of a data set, used for data recovery. It is similar to the upstream backup strategy in the loss calculation system, its application range depending on the dependent type. As shown in Fig. 3, the Spark of the two types of dependence, Narrow dependence and upstream and downstream RDD is saved in the same node. So the node downtime cannot continue to back up. In wide dependence, upstream and downstream RDD is saved on different nodes. When the downstream is down, it can be recovery through the upstream. Different operations (union, map, join, etc.) will produce different depend on the type. There are two kinds of fault tolerance mechanism in the Spark. It depends on the case. The both kinds of recovery methods used in distributed memory system need to consume a lot of time Due to the attention of the computational efficiency, there is the corresponding loss on the efficiency of the fault recovery.

4.4 Fault Tolerance in SDN

Software Defined network (Software Defined Networking, SDN) is a revolutionary change in the field of network architecture in recent years. The core idea of SDN is that the data layer and the control layer of traditional network equipment should be separated, and the function of the control layer focuses on controllers. We can manage and configure a variety of network devices via Standardized interfaces in a centralized controller. That will provide more possibilities for the design, management and use of network resources, which are more likely to promote the innovation and development of the network [11] (Fig. 4).

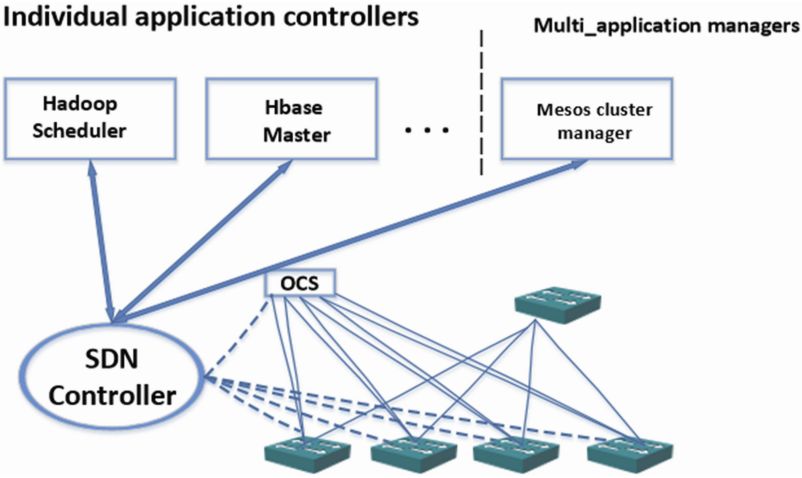

Fig. 4. The big data applications based on SDN

From a virtual data center level, SDN centralized master data center network topology information of control surface, and that the virtual machine's MAC/IP/location attribution can realize flexible programmable ability, and can keep fault in physical layer away from the application layer. That will provide a more flexible and more stable and reliable virtual network for network virtualization, automation, all sorts of network services and big data.

Big Data applications are no longer connected through the traditional network hardware and provide services, but by SDN control operation in the virtual network. When network hardware has the failure, big data applications should keep away from hardware fault by SDN controller. SDN is responsible to maintain the availability of virtual network by calling the backup hardware. So that big data applications don't have to focus on network level fault, and only focus on the software failure.

5 Conclusion

Throughout the past 20 years, development of IT technology and IT industry, based not only on the progress and development of the technology itself, contains more business model and application of the success of the marketing mode. Big data has been deeply rooted in every area of life and technology, and it has made a great contribution for business, engineering, health care, public administration. With commonly used Hadoop platform as an example, this paper discussed the basic architecture of big data technology, and provides big data applications in various industries. While fault tolerance is an important issue in big data applications, fault-tolerant mechanism ensures that large data software and applications can run stably and reliably, and continue to provide service for various industries. This paper summarizes the technology of data in a variety of fault tolerant mechanism, and provides the comparison. It also provides reference of high availability and high reliability for big data applications.

Acknowledgements. This paper is supported by the project 61303094 supported by National Natural Science Foundation of China, by the Science and Technology Commission of Shanghai Municipality (16511102400), by Innovation Program of Shanghai Municipal Education Commission (14YZ024).

References

1. Jhawar, R., Piuri, V., Santambrogio, M.: A comprehensive conceptual system-level approach to fault tolerance in cloud computing. In: 2012 IEEE International Systems Conference (SysCon), pp. 1–5. IEEE (2012)
2. Dyavanur, M., Kori, K.: Fault tolerance techniques in big data tools: a survey. Int. J. Innovative Res. Comput. Commun. Eng. **2**(2), 95–101 (2014)
3. Parker, P.A.: Discussion of "reliability meets big data: opportunities and challenges". Qual. Eng. **26**(1), 117–120 (2014)
4. Shvachko, K., Kuang, H., Radia, S., et al.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
5. Neumeyer, L., Robbins, B., Nair, A., et al.: S4: distributed stream computing platform. In: 2010 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 170–177. IEEE (2010)
6. Jones, M.T.: Process real-time big data with Twitter Storm. IBM Tech. Libr. **14**(2), 1–5 (2013)
7. Reitblatt, M., Canini, M., Guha, A., et al.: Fattire: declarative fault tolerance for software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 109–114. ACM (2013)
8. Antoniu, G., Costan, A., Bigot, J., et al.: Scalable data management for map-reduce-based data-intensive applications: a view for cloud and hybrid infrastructures. Int. J. Cloud Comput. **2**(2), 150–170 (2013)
9. Hwang, J.H., Balazinska, M., Rasin, A., et al.: High-availability algorithms for distributed stream processing. In: Proceedings of 21st International Conference on Data Engineering 2005, ICDE 2005, pp. 779–790. IEEE (2005)
10. Zaharia, M., Chowdhury, M., Das, T., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, p. 2. USENIX Association (2012)

11. Zaharia, M., Chowdhury, M., Franklin, M.J., et al.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, p. 10 (2010)
12. Kim, H., Santos, J.R., Turner, Y., et al.: Coronet: fault tolerance for software defined networks. In: 2012 20th IEEE International Conference on Network Protocols (ICNP), pp. 1–2. IEEE (2012)