Architectural Design Of Data Stream-Based Big Data Real-Time Analysis System

Tianfu College of SWUFE , Qiang Liu[1*], Junmin Lv[1], Xun Yuan[1], Renyi Luo[2], Dekui Lv[3]

1. Tianfu College of SWUFE ,Chengdu 610000
2. stuednt of Tianfu College of SWUFE ,Chengdu 610000
3. The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210000, China
** Corresponding author ,e-mail:55900673@qq.com*

Keywords: Data stream, Big data, Real-time computing (RT computing), Architectural design.

Abstract

Conducting the RT analysis of large-scale data stream severely challenges the IT hardware facilities and system Architecture. To cope with the needs of large-scale RT analysis, the article analyzes and compares the mainstream open-source technology and then designs the system architecture that consists of data collection, data distribution, real-time computing, persistent storage, resource allocation and scheduling.

1 Introduction

With rapid development of smartphone, Internet of things, cloud computing, wearable devices, social media and various of applications, sensors and access devices, the instantaneous data size produced by constant RT data stream such as "console, text files and logs" reaches the scale of millions even billions of records per second, which occurs generally TB or PB of data size. The features of traditional data processing system are low-response, high-latency which can not meet the needs of processing collection, aggregation, storing, analyzing and visualizing of these data. We are now facing both the challenges of massive instantaneous data shock and sub-second or even sub-millisecond response of RT analysis. for example: 3 million per second peak 5 million per second RT online message scale on "Double 11th" of Alibaba Group in 2015[1]. Hereby, requires fraud response, real-time warning and network attack recognition etc, which traditional data processing system can not realize.

The article compares and analyzes the computing framework of MapReduce and Spark, data collection and pretreatment method Flume and Kafka, the way of data persistence, therewith, using the Spark as the data computing framework to RT compute, Flume to data collect, Kafka to data preprocessing and distribution, Redis and Mysql as the data persistence storage, thereby design the data stream-based big data RT- analysis system.

2 Distributed storage and computing framework

Hadoop[2] shows good merits at coping with big data distributed storage, analyzing and processing big data files. However, stream data is generally time-limited with small value density and large data scale in one time window which unable to confirm the time and sequence of the data or to record[3], hence, conducting RT computing in terms of established time lag(sub-millisecond or sub-second) of data stream is revered as ideal way. MapReduce adopts batch processing mode which can not meet the needs toward the applications required high degree of response. Figure 1 shows data processing flow of MapReduce.

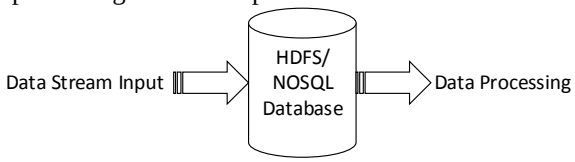

Figure 1: Traditional Data Processing

Spark is a fast distributed data processing(DDP) engine, adopts memory computing , which is 100 times faster than Hadoop MapReduce framework computing and 10 times faster than traditional disc computing[4]. Spark Streaming[5] is a distributed, high-fault-tolerant RT data processing system provided by Spark that supports multi-format of data recourse, which enables to store in data base such as HDFS, Hbase etc, after data processing, also, Spark Streaming is available for online RT interactive analysis or data analysis and processing by using machine learning and graphs processing algorithms. Figure 2 shows the data processing flow.

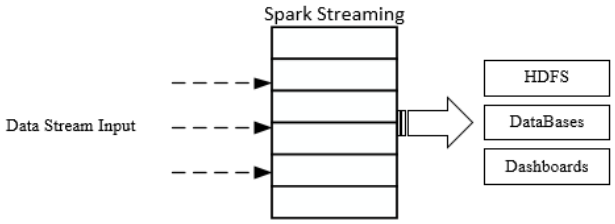

Figure 2: Spark Data Processing

MapReduce requires pre-store in disc or NoSQL in order to conduct batch processing, compare with it, the major advantage of Spark is to directly compute in memory and enables the processing data to be utilized in interactive analysis and data persistency in database, so Spark is faster speed than MapReduce.

Compared with Storm, Spark enables to realize RT data computing and batch computing, thus, Spark also produces widely adaptation in history data reusing, moreover, Spark SQL,MLlib and GraphX module realizes convenience in Ad hoc queries, machine learning and graph processing which provides support for further development .
Therefore, the solution architecture in this article uses Spark as distributed computing framework and adapt Spark streaming module in order to enhance the speed of response.

3 Data collection

In order to realize advanced specific functional areas of system architecture and to realize the designing ideal of hierarchicalization and modularity for ensuring the system expansibility for coping with various data source, high concurrent and high concurrency of data sources, thus, the architecture adapts Flume as data collection module.

Flume is a distributed, highly reliable mass data collection, aggregation and transmission system that contains the characteristics of customizability, high-fault-tolerance and recovery mechanism. The data pattern enables to augment. Figure 3 shows the Flume data collection process.

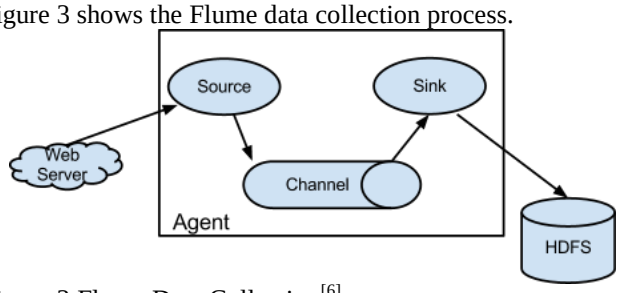

Figure 3:Flume Data Collecting[6]

Flume realizes mass data collection with "three-tier architectures": first, source tier, deploys process(JVM) on each information collection required server, which is used in collecting log data of the server. Second, Channel tier, receives and stores the data from source tier, directly transmit to the next tier or written to local file system until the completion of Sink tier in order to ensure the major data will be correctly processed in the condition of unexpected shutdown or program error, third, Sink tier, transmits or stores the data from Channel, for example, transmitting the disc files from HDFS to Kafka. Sink deletes the data from Channel only when the data processing is completed. Flume architecture provides data stability and reliability of "end to end" with Source and Sink tiers sealed in transaction. Flume ensures the expandability of data collection system with linear extension model. In Source tier, every machine deploys the process that provides the unlimitedly horizontal scaling when server data increase. Generally, one process enables to complete the correspondent data collection. Process adapts load balancing mechanism that enables channel tier to correctly receive the mass data when transmitted from Source to Channel. Every Channel provides indiscriminating service and enables to linear expansion, moreover, Sink tier also enables to realize linear expansion on account of distributed-storage of HDFS and Kafka.

4 Data pre-processing and distribute

Flume collects the data, thereafter, requires to conduct format specification and aggregation for data stream, then to transmit the next step for RT analysis or storing in HDFS, NoSQL, RDBMS for persistent storage. The system adapts Kafka as data pre-processing and distributing platform.

Kafka is a highly-reliable data stream processing platform that operates above cluster which allows to establish RT data stream analysis system or applications. Data stores in terms of categories, which is named "Topic" of each category, that enables to store in different servers. "Topic" that above each server is named "Partition", which messages accordingly store in corresponding Partition as the way of "append". The consumer accesses the data according to the "Topic" which does not need to know which server the Partition stores in. Once producer publishes the messages, consumer subscribes and processes the "Topic" messages through the producer. One "Topic" is able to store in numerous servers, the data distributing reliability is ensured even N-1 numbers of servers failure occur. Figure 4 shows the operating architecture of Kafka.

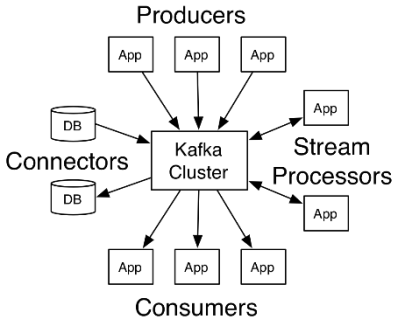

Figure 4: Kafka streaming platform[7]

5 Data persistence

Spark is able to realize persistent data storage according to different application demand. There are Hbase, MongoDB, Cassandra and Redis etc in major NoSQL database storage scheme.

Hbase[8] is a distributed, extensible open-source database. With compactly integrating with Hadoop, Hbase is able to operate billions of rows and millions of columns that provides Write-ahead Log to ensure data security, which is widely used at present.

Redis is a memory-based non-relational database, with the characteristic of fast speed, provides approximately 100,000 ops/s read/write performance with 10-70 microsecond delay[9]. Redis supports integrated mapping between "Key" and five different "values" which enables to store key-value data from memory to the disc, also, Redis supports data types as string, list, set, hash table, ordered set etc.

MongoDB is an open-source database applied to various scales of enterprises, industries and applications that supports

single server deployment or mass, multi-data center architecture. With the advantage of memory computing, MongoDB provides high-performance data read/write operations which is now utilized as major database[10]. Generally, data stream-based RT analysis system contains rare document information, hence, MongoDB is not considered in persistent storage layer in this paper.

Apache Cassandra is a set of open-source, distributed NoSQL database system, which originally developed by Facebook that applied to store single format data with complete distributed architecture of Google BigTable data module and Amazon Dynamo. In 2008, Cassandra was released as open source, with good extensibility and performance, Cassandra was applied by well-known websites as Apple, Comcast, Instagram, Spotify, eBay, Rackspace, Netflix etc, which becomes a popular distributed structured data storage scheme[11]. Cassandra performs better in read/write performance than Hbase[12].

MySQL server is a fast, multithreaded open-source multi-users RDBMS. MySQL server applies to heavy load production system and software embedding for large-scale deployment[13]. With the characteristics of high performance, low cost and high reliability, MySQL becomes the popular open-source database that is used for small and medium websites on internet extensively. As MySQL continues to mature, it applies to more large websites and applications as Wikipedia, Google and Facebook[14]. The system architecture design chooses MySQL as preferred as relational persistent database layer.

6 Resource scheduling and management

The initial Hadoop MapReduce architecture was designed an independent distributed computing framework that can not operate the others, in which can hardly update and maintain. Jobtracker focuses on operating the distribution and monitoring of Job on the TaskTracker that causes high pressure and the possibility of single point of failure. If failure occurs on Jobtracker, it disables the entire system. Over-concentrated task mode causes resource waste when operating more tasks.

To solve the problem of former framework, the new Hadoop MapReduce architecture conducts refactoring and call MapReduceV2 or YARN. Figure 5 shows the framework operating process of YARN.

New architecture splits the former JobTracker,Resource Manager (RM) conducts resource allocation and abandons the tasks of monitoring and reboot etc, New architecture does not conduct monitor or track the running state of applications. ApplicationMaster(AM) requests resources from RM, collaborates with Node Manager(NM), to complete the monitoring of the task operation and state. NM completes the management for the actual resource (memory, CPU, internet and disc etc,) of applications and reports the status to RM.

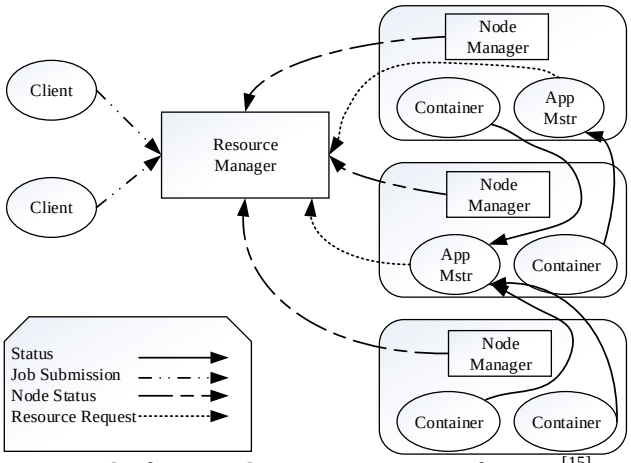

Figure 5: the framework operating process of YARN[15]

Compare with former Hadoop MapReduce, new architecture splits the former “JobTracker” and “TaskTracker” into three parts of “RM”,”NM” and “AM”, in order to reduce the pressure and resource consumption for safer operation, better scalability and maintenance.

Different frameworks are able to operate in YARN architecture as follow: MapReduce, Storm, Spark and Tez etc, in which provides support for application integration. YARN is utilized as resource scheduling and management platform in the system architecture.

7 Architecture of Data stream-based Big data Real-Time Analysis System

In summary, Architectural design of Data stream-based Big data Real-Time Analysis System shows in figure 6.

System core consists of 5 layers: data collection layer, which adapts Flume to collect data and log from applications and devices, next, distribute to Spark memory computing framework through Kafka preprocessing. After the completion of RT analysis from Spark Streaming, system persistently stores the data that are required long-term stored to NoSQL or MySQL, therewith, to combine the query result views of batch processing and RT computing, thus to utilize MLib machine learning algorithm of Spark framework to mine the larger value of the data.

System advantages:

7.1 High reliability

Adapts cluster deployment, which Provides guarantee of reliability in each layer for data collection, distribution and preprocessing, RT analyzing and layer persistence.

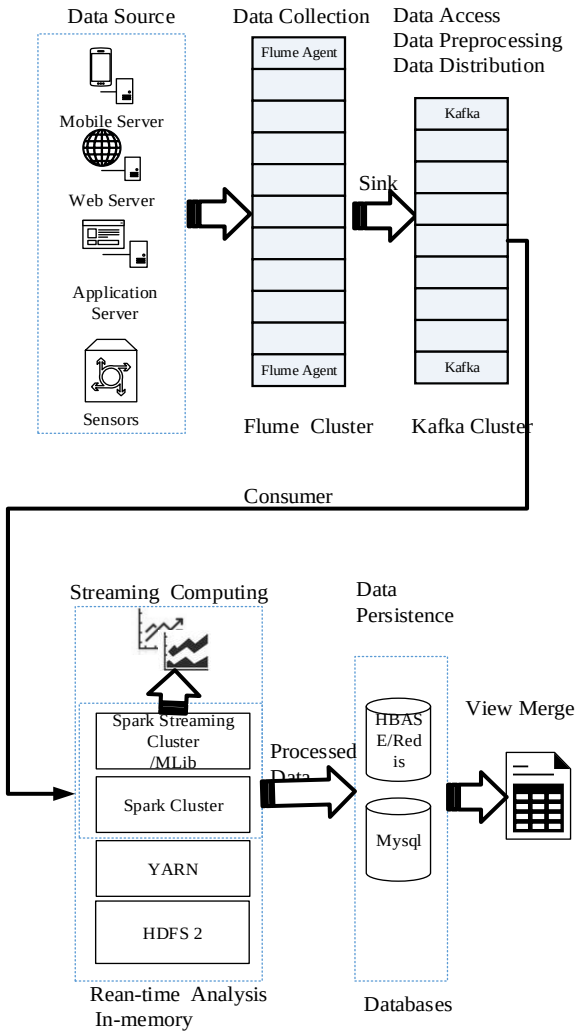

Figure 6: Architecture of Data Flow Real-time Analysis

7.2 Strong expendability

Can easy to cope with instantaneous peak value and linear scalability in each layer.

7.3 Cost controllable

System design adapts major open-source framework and software to ensure the convenience of system deployment and updating in order to realize cost saving.

7.4 Maintainability

System designed with the ideal of loosely coupled which adapts layered architecture with clear functional distinction to ensure the system maintainability.

7.5 High security

System ensures the security of data either the process of transmission or data persistent storage.

7.6 Load-balance

Using load-balancing mechanism in storage platform and computing framework to ensure reasonable dispatching and distributing of storage and computing resource.

References

[1] Real - time calculation of massive data Tec <https://yq.aliyun.com/articles/2971?spm=5176.100239.blogcont4199.10.DnTNKt>
[2] http://hadoop.apache.org/
[3] SUN Da-Wei, ZHANG Guang-Yan, ZHENG Wei-Min. Big Data Stream Computing:Technologies and Instances[J]. Ruan Jian Xue Bao/ Journal of Software, 2014, (4): 839-862.
[4] Apache Spark™ is a fast and general engine for large-scale data processing.< http://spark.apache.org/>
[5] Spark Streaming makes it easy to build scalable fault-tolerant streaming applications. <http://spark.apache.org/streaming>
[6] Data flow mode. <https://flume.apache.org/FlumeUserGuide.html>
[7] Introduction. <https://kafka.apache.org/intro>
[8] < http://hbase.apache.org/>
[9] Introduction to Redis. <https://redis.io/topics/introduction>
[10] Introduction of MongoDB. <https://www.mongodb.com/cn>
[11] Cassandra.< https://zh.wikipedia.org/wiki/Cassandra>
[12] Apache Cassandra NoSQL Performance Benchmarks.<https://academy.datastax.com/planet-cassandra/nosql-performance-benchmarks>
[13] <http://dev.mysql.com/doc/refman/8.0/en/introduction.html>
[14] Mysql. <https://zh.wikipedia.org/wiki/MySQ>
[15] Detailed of new MapReduce framework Yarn. <http://www.ibm.com/developerworks/cn/opensource/os-cn-hadoop-yarn>