Dean&Francis
ISSN 2959-6157

Research Progress in the Field of MapReduce Framework Based on Serverless Computing

Jialing Xu[1, *]

[1]School of Computer Science and Technology, Dalian University of Technology, Dalian, China

*Corresponding author: 1829731518@mail.dlut.edu.cn

Abstract:
In recent years, big data technology has presented a rapid development trend with each passing day, and its application in all walks of life is becoming more and more extensive and in-depth. At the same time, traditional computing frameworks are gradually showing serious challenges in performance bottlenecks and scalability in the face of massive data processing requirements, making it difficult to meet the growing needs of data processing and highly concurrent access scenarios. Firstly, this paper introduces MapReduce framework and serverless platforms, and then introduces the MapReduce framework model based on serverless platforms from several parts of implementation, application scenarios and technical principles. This paper argues that the serverless-based MapReduce framework will provide a more efficient and flexible solution for big data processing and analyzing, which is of great significance in promoting the development of data-driven decision-making and intelligent applications. The main content of this paper will be introduced from the basic overview of MapReduce computing framework and serverless platform, the implementation of serverless-based MapReduce computing framework, technical principles and application scenarios.

Keywords: MapReduce framework; Serverless Computing; Big Data Technology.

1. Introduction

With the development of information science and computer science, the demand for data processing grows massively, and the traditional parallel computing processing is difficult to carry the intensity of its operation, and the emergence of distributed parallel computing mode solves this problem. In carrying out massive data processing and analysis, cloud computing makes full use of the advantages of high efficiency, low cost and easy maintenance to provide a good environment for data processing and analysis, and cloud computing has become the focus of parallel computing research. In cloud computing, Google, Amazon, IBM and Microsoft have taken the lead in cloud computing research and gained in-depth progress. At present, there are many applica-

ISSN 2959-6157

tion areas of cloud computing, involving cloud storage, information pushing, e-commerce, high-performance computing, search engine and other fields, for which large-scale data collection and analysis are carried out, and effective knowledge and benefits for each field are obtained by deeply utilizing the data processing results. Among them, MapReduce is one of the most commonly used programming models for analyzing large-scale data, as a distributed computing framework, it overcomes the shortcomings of the traditional distributed parallel program writing complexity, and provides a simplified way of writing. MapReduce was initially proposed by Google for a large amount of dispersed data and a huge amount of computation difficult to be completed by a single machine as a programming model, and on the basis of the partitioning MapReduce was originally proposed by Google as a programming model for large amounts of dispersed data and huge amounts of computation that are difficult to accomplish on a single machine, based on the idea of partitioning, to better accomplish parallel computing tasks, with the advantages of underlying transparency, high fault tolerance and load balancing. As the MapReduce model is more widely used, the algorithm optimization of this programming model is also more needed. Overseas scholars have made some optimizations for the deficiencies of the MapReduce model itself. The University of Illinois proposed a new parallel programming model-Barrierless MapReduce, which mainly eliminates the sorting and merging operation of the intermediate key-value pairs compared with the MapReduce model, although the Reduce function can directly process the intermediate key-value pairs, the model requires the user to make corresponding modifications to the Reduce function, which increases the programming burden of the user. Although the Reduce function can deal with intermediate key-value pairs directly, this model requires the user to make corresponding modifications to the Reduce function, which increases the burden of user programming. [1] To address the shortcomings of MapReduce model task scheduling, the domestic scholars Qianqian Dong took the local optimal solution of the ant colony algorithm as the initial solution of the simulated annealing algorithm to conduct global search, and judge whether to accept the local optimal solution of the current solution as the initial solution of the simulated annealing algorithm to conduct global search according to the Metropolis criterion, which can reduce the completion time of the task and ensure the balance of the resource load at the same time. In order to reduce the completion time of the task and ensure the resource load balancing. [2] To address the inefficient auto-generated MapReduce, WU proposes Cost-Aware SQL-to-MapReduce Translator (CAT), which provides a declarative SQL-like language and its auto-optimizing translator to enhance the auto-generation of MapReduce. [3] The paper will introduce the basic concepts of MapReduce computing framework and Serverless platform in detail, and discuss in depth the MapReduce computing framework realized based on Serverless architecture, including its technical principles, implementation methods, and practical applications in different application scenarios.

2. Overview of the MapReduce framework and Serverless Platform

2.1 MapReduce and Traditional System Architecture

The MapReduce model was proposed by Google in 2004, which is mainly applied to the computation of processing massively parallel data, and accomplishes the task of basic parallel computation through two functions, map and reduce. The core of the design of this program model consists of two main features, i.e., Map and Reduce operations. In the Map phase, the input data is partitioned into multiple chunks and distributed to different processing nodes. Each node runs a mapping function that processes the local data and generates intermediate key-value pairs. The next two intermediate results have the same result in the key values aggregated to the same node. The Reduce function takes the intermediate list and processes all the values for each key, finally generating a new list. The MapReduce programming model excels in efficiently utilizing distributed computing resources, automating data distribution, scheduling, and fault recovery.

The traditional system architecture of MapReduce usually consists of multiple components to support large-scale data processing. This architecture is mainly led by the Hadoop ecosystem, an open source distributed computing framework developed by the Apache organization. The system consists of two components: the Hadoop File System (HDFS) and MapReduce, which is a framework for processing data from the Hadoop file system. HDFS is an open-source derivative of the Google File System (GFS); its goal is to achieve high redundancy and availability of data and to reduce the cost of hard disk drives. The idea of HDFS is to distribute redundant data to many machines (called data nodes) in the network. The idea of HDFS is to distribute redundant data across many machines (called data nodes) in a network. This is done by splitting the data into chunks of a certain size and then storing these chunks randomly many times, with a copy of each chunk stored on a different node. [4] In addition to storing the data, HDFS is also used for storing the data on the network. [4] In addition to storing data, Hadoop runs MapReduce with each machine in the cluster acting as both a data node and a task tracker. This means that both data storage and analytics run on the same physical machine, which reduces network traffic. [4]

2.2 Introduction to Serverless Platform and Mainstream Platforms

Serverless computing is a cloud computing model that allows developers to build and run applications without the need for managed servers. In this model, the cloud provider automatically handles the allocation, scaling, and management of resources. In recent years, there has been a rise in the serverless computing model, in which users do not need to explicitly manage servers; instead, the cloud provider dynamically allocates resources to execution units, i.e., function calls. Users write functions using a service-supported programming language without having to make assumptions about which underlying computing infrastructure the function calls will run on. By creating stateless functions, it is possible to use a highly parallel execution model where multiple concurrent invocations of the same function can be triggered based on events occurring in the infrastructure. [5]

Currently the mainstream serverless platforms are AWS Lambda, Azure Functions, Google Cloud Functions and Alibaba Cloud Function Compute, etc. AWS Lambda is a Serverless computing service provided by Amazon that supports multiple programming languages (Node.js, Java, C#, Python, and Go) with a high degree of parallelism. AWS Lambda can be triggered by a variety of event sources, such as HTTP requests (via Amazon API Gateway), file uploads (via Amazon S3), database changes (via Amazon DynamoDB) etc. This event-driven architecture gives applications the flexibility to respond to a variety of inputs. Users pay only for the computing resources they use, billed by execution time and number of requests. This on-demand model can significantly reduce costs and is especially suited for short execution times. Azure Functions is a serverless computing service provided by Microsoft Azure that supports an event-driven programming model, scales automatically, and responds quickly to events. It is particularly well suited for building small services, automated tasks, and microservice architectures. Compared to AWS Lambda, Azure Functions minimizes the impact of cold starts through methods such as resident instances and warming up, allowing function applications to remain running without requests.

3. Technical Principles of Serverless Computing

3.1 Separation Of Computing and Storage

In serverless platforms, storage-computing separation usually uses a distributed architecture where storage and computation resources are deployed on different nodes. The storage nodes are responsible for maintaining the data and the compute nodes perform the processing tasks. In store-computer separation, data can be preprocessed at the storage layer and then the processed data is passed to the compute resources. This reduces the load on the compute nodes and increases the speed of computation. At the same time, a cache is set up on the compute node to store frequently used data and reduce frequent accesses to storage resources. Computing resources can be elastically scaled up or down according to the number of requests, while storage resources can grow independently of computing resources. This asynchronous processing model ensures that when some nodes fail, they can still be accessed through other nodes, reducing the idle time of computing resources and improving the throughput of the system.

3.2 Scalability and Elasticity for Serverless Computing

Serverless computing can automatically scale based on real-time workloads. When requests increase, the platform can automatically launch more compute instances to handle them; when the load decreases, it releases unneeded resources. This resource allocation method effectively improves the utilization of computing resources. Functions in serverless are stateless, and proceeding independently can make alleviate the complex problem of synchronizing states in front of the system, thus improving the flexibility of computation. Serverless is equipped with automatic failover and self-healing capabilities. When a computing instance fails, the platform will automatically start a new instance to continue processing tasks, ensuring high service availability.

4. Implementation of MapReduce Framework for Serverless Computing

4.1 Motivation for Combining Serverless Computing with MapReduce Models

The high degree of parallelism supported by AWS Lambda is one of the main features of the service. Reports from the workshops and the State of Serverless Computing Symposium at the First International Workshop on Serverless Computing (WoSC) in 2017 indicate that a hot research topic in serverless computing will be its use in parallel programming, and that how to extend FaaS for MapReduce is A challenge. [5] While Hadoop is the most commonly used Big Data processing framework, it has a steep learning curve given the number of components and their inherent complexity. By minimizing the number of components in the framework and abstracting infrastructure management, the combination of Serverless platforms

ISSN 2959-6157

and MapReduce can simplify the process of data processing. In addition to the advantage of reduced complexity, the combination of the two saves development and O&M costs over ad hoc MapReduce workload solutions. Serverless platforms only charge for the execution of functions, and on-demand allocation makes it easy to accurately budget data processing needs based on queries.

Combined with a serverless platform, MapReduce jobs can be deployed and launched faster. The serverless architecture allows for auto-scaling, which enables the system to dynamically allocate resources when processing large amounts of data, improving task completion efficiency. The serverless platform's automatic fault recovery capability ensures that tasks can be automatically retried or recovered in the event of an error, enhancing system reliability.

4.2 Implementing the MapReduce Framework Model on the Serverless Platform

To accomplish the task of running MapReduce on serverless platforms, the University of Valencia team developed MARLA (MApReduce on LAmbda), an open source lightweight framework using the Python language.

4.2.1 Architectural Design

In the Molto team's research, the MapReduce framework consists of three sets of coordinator, mapper, and reducer functions and two S3 buckets (one for input data and one for output data). When the dataset is uploaded to the input S3 bucket, the coordinator Lambda function is activated to compute the optimal size of the data partition, and then the mapper and reducer functions are executed sequentially. The framework supports automatic partitioning of input data, and the user only needs to define the mapper and reducer functions in Python.

If an exception occurs during the run, then the Lambda function call is terminated, but not retried indefinitely. Faults are handled differently on the coordinator, mapper or reducer, and some of the processed data may remain in the S3 output bucket to be handled by the user. Currently only column-based plain text data files are supported, the input data does not need to be pre-partitioned, the user's "map" and "reduce" functions must be implemented in Python, and AWS service access credentials and IAM roles need to be provided. [5]

4.2.2 AWS Lambda Performance Evaluation

AWS Lambda is compared to the SRAM benchmark by testing a subset of queries using the same benchmarks as SRAM, including scan queries and aggregated queries. The results show that AWS Lambda exhibits uneven performance behaviour and does not provide a homogeneous compute environment even if the functions have the same memory. The test results show that AWS Lambda has a heterogeneous compute infrastructure, that there is a significant correlation between the execution time of Lambda calls and the underlying allocated VMs, and that the CPU model is a good indicator of VM performance, as well as the performance of Lambda function calls. When also accessing the network performance of S3, the tests show that the network behaviour has unpredictable fluctuations in a small fraction of the calls, and that the increase in upload time does not seem to be related to the performance of the VMs, but still has a significant impact on the global execution time. In terms of isolated network usage performance, tests showed that latency was not generated by the number of calls using the network, but by the number of concurrent calls. [5]

4.3 Application Scenarios of MapReduce Framework Model for Serverless Based Computing

4.3.1 Big Data Processing and Applied Analytics

In the scenarios of log analysis, click stream analysis, user data behaviour analysis, etc., the MapReduce programming model based on serverless computing can efficiently complete the tasks of data partitioning and distributed computing. Nanjing University has developed a large-scale network community discovery algorithm based on MapReduce, which makes use of the "circle of friends coefficient" technique, the "two-stage k-centre clustering" technique, and the "community fusion" technique to accurately and efficiently optimize the modularity of the community. The "circle of friends coefficient" technique, "two-stage k-centre clustering" technique, "community fusion with modularity optimisation" technique accurately and efficiently discovers the community structure in large-scale social networks, and shows its superiority in comparison with other community discovery algorithms. [6] In distributed systems, the MapReduce programming model can be used to analyse large numbers of log files to identify potential problems, performance bottlenecks or security vulnerabilities. For example, error log analysis for e-commerce websites, online services, server performance monitoring, etc. Search engines can analyse user query logs to optimise the ranking of search results and understand user needs and behavioural patterns.

Md. Mobin Akhtar's team used MapReduce framework for student health data collected and normalised by IoT devices. In the Map phase, the framework gives the normalised data to an autoencoder and a one-dimensional convolutional neural network (1DCNN), respectively, for extracting the deep features of the student health information, and in the Reduce phase, the Adaptive Bird-Rat-Swarm Optimisation (ABRSO) is used for the optimal feature selection to augment the student health monitoring

(HM) system. [7] The combination of MapReduce and Convolutional Neural Networks can take advantage of distributed computing in order to improve the efficiency of large-scale data processing and the speed of model training.

4.3.2 Text Processing and Indexing

The combination of MapReduce and serverless computing can be applied to large-scale text processing, such as building search engine indexes, document classification, and natural language processing tasks. The elastic scalability of serverless architectures enables them to efficiently respond to big data processing tasks. Zhiang Wu et al. proposed an effective cost-aware SQL-mo-reduce converter (CAT) to enhance the efficiency of automatically generated MapReduce programs.CAT has two distinctive features. Firstly, it defines two intra-sql correlations: generalized job flow correlation (GJFC) and input correlation (IC), based on which a set of looser merging rules is introduced, and a cost estimation model for mapping reduced job flows is adopted to provide a more efficient choice of mapping reduce job flows. [3]

Reverse indexing maps each word in a document to a list of documents that contain that word, thus helping the user to find documents that contain a specific word, a method that is common in the field of information retrieval. The MapReduce model based on serverless platform can overcome its shortcomings of large data volume and resource consumption, and process large-scale data quickly with its advantages of distributed computing.

4.3.3 Image Classification and Image Annotation

MR-SMO is a MapReduce based distributed SMO algorithm for automatic image annotation which maintains high accuracy while reducing training time for large scale datasets. [8] Min Chen developed mapreduce based CNN (MCNN) algorithm to solve the task of image classification, furthermore, the algorithm solves the overfitting problem by adding the idea of a dropout layer in the network. Due to the limitations of CNN itself, it was added to a cloud computing cluster to implement a MapReduce (MCNN) based parallel CNN algorithm. The developed algorithm exploits the computational structure inherent in CNNs to enable parallelization for higher processing speed. [9] With MapReduce on a serverless platform, image classification tasks can be processed efficiently and scalable to meet the demands of modern big data and deep learning tasks.

5. Challenges and Prospects

5.1 The Challenges of Serverless Computing

In Serverless environment, the performance of function execution is affected by many factors such as function cold start, resource allocation and network latency. When a large amount of data transfer is involved, there is uncertainty in MapReduce's task execution time as different resources may be allocated for each run. Many Serverless platforms (e.g., AWS Lambda) have strict limitations on function execution time and resources, and MapReduce tasks may need to be split into smaller subtasks, which increases the complexity of development and the difficulty of task management.

5.2 Future Perspectives on the Combination of Serverless Computing and the MapReduce Framework

Future Serverless platforms may integrate smarter data slicing and scheduling algorithms to handle MapReduce tasks more efficiently. Bandwidth bottlenecks and latency issues are reduced by automatically identifying and optimizing data transfer paths and sharding strategies. The Serverless platform is extending function execution time and resource limits, which may in the future allow a single function to handle larger tasks, reducing the complexity of splitting MapReduce tasks into smaller ones.

6. Conclusions

This paper discusses the research background and significance of serverless and MapReduce, summarises the concept and basic framework of MapReduce, and enumerates the improvements in algorithm design, framework construction, and other aspects of the current serverless-based MapReduce programming model as well as the applications in data retrieval and image recognition. Based on the results of this paper, research can be centered on the optimization of MapReduce programming model in terms of data transfer, automatic scheduling and so on.

References

[1] Verma A, Zea N, Cho B, Gupta I & Campbell R H. Breaking the MapReduce stage barrier. Cluster Computing, 2010, 16(1), 191-206.

[2] Lehrack S, Duckeck G & Ebke J. Evaluation of Apache Hadoop for parallel data analysis with ROOT. Conference on High Energy Physics, 2014.

[3] Wu Z, Song A, Cao J, Luo J & Zhang L. Efficiently translating complex SQL query to MapReduce jobflow on cloud. IEEE (2), 2020.

[4] Yang H, Dasdan A, Hsiao R & Douglas Stott J P. Map-reduce-merge: simplified relational data processing on large clusters. Acm Sigmod International Conference on Management of Data, 2007.

[5] Gimenez-Alventosa V, Molto G & Caballer M. A framework

ISSN 2959-6157

and a performance assessment for serverless mapreduce on aws lambda. Future Generation Computer Systems, 2019, 97(AUG.), 259-274.
[6] Wang H C, Dai H P, Chen C P, Chen S S & Chen G H. A mapreduce-based algorithm for large-scale web community discovery. Computer Science, 2024, 51(4), 11-18.
[7] Akhtar M M, Shatat A S A, Al Hashimi M, Zamani A S, Rizwanullah M, Mohamed S S I. MapReduce with Deep Learning Framework for Student Health Monitoring System using IoT Technology for Big Data.Journal of Grid Computing (4), 2023.
[8] Alham N K, Li M, Hammoud S, Liu Y & Ponraj M. A distributed SVM for image annotation. IEEE, 2010.
[9] Chen M. Classification with Convolutional Neural Networks in MapReduce. Journal of Computer and Communications (08), 2024, 174-190.