WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 2, Number 3, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v2n3.02
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Distributed Implementation of Computing Language Model in Cloud Computing Network

Yi-Ning Ou

Santa Clara University, Santa Clara, US

ABSTRACT

In this paper, the distributed implementation of Computing Language Model (CLM) in cloud computing network is studied and discussed. With the advent of the era of big data, the application of CLM in natural language processing (NLP), machine translation and other fields is increasingly extensive, and the demand for computing resources is also increasing. As an effective way to manage computing resources, distributed computing can make full use of resources in cloud computing environment and realize efficient execution of computing tasks. In this paper, the application of distributed computing methods such as data parallelism, model parallelism and hybrid parallelism in cloud computing environment is studied and analyzed, and the advantages and disadvantages of different methods in CLM implementation are discussed. The experimental results show that the hybrid parallel method can effectively combine the advantages of data parallelism and model parallelism, and improve the training efficiency and performance of CLM. This study provides important theoretical guidance and technical support for the distributed implementation of CLM in cloud computing networks, and is of great significance for further promoting the development of big data and AI technology.

KEYWORDS

Cloud Computing Network; Computing Language Model; Distributed Implementation

1. INTRODUCTION

In today's information age, the data scale is constantly expanding, and natural language processing (NLP) technology is more and more widely used in various fields. As an important part of NLP, Computing Language Mode (CLM) plays a key role in text generation, semantic understanding, machine translation and other tasks. The core goal of CLM is to model the probability distribution of language, so as to generate texts that conform to grammatical and semantic rules, or to understand and analyze input texts [1].

The traditional CLM is usually implemented on a single machine, but with the explosive growth of data and the increase of computing demand, the single machine model can no longer meet the needs of efficient processing of large-scale corpus [2-3]. In this context, distributed computing has become a necessary choice, which improves the computing efficiency and processing capacity by distributing computing tasks to multiple computers for parallel processing. As a flexible and efficient way to manage computing resources, cloud computing provides strong support for distributed computing [4]. In the cloud computing environment, users can dynamically apply for and release computing resources according to their own needs, which greatly reduces computing costs and improves resource utilization. Therefore, combining CLM with cloud computing to realize distributed computing has become a research direction with wide application prospects.

This paper will discuss the distributed method of implementing CLM in cloud computing network, aiming at improving the performance and scalability of CLM, so as to better meet the needs of large-scale data processing. By analyzing the principle of distributed computing and cloud computing environment, and discussing the implementation method of distributed CLM, this paper will provide theoretical and practical guidance for realizing efficient and reliable distributed CLM.

2. CLM OVERVIEW

NLP is an important branch in the field of AI, and its goal is to enable computers to understand, analyze and generate natural language texts. [5-6] CLM is one of the core components in NLP, which is mainly used to model the probability distribution of language, so as to realize tasks such as text generation, semantic understanding and machine translation.

Language modeling is one of the most basic tasks in CLM. The goal of language modeling is to estimate the probability of a sentence (or text sequence) appearing in the language. Usually, conditional probability is used to represent the probability of a sentence appearing in a language, that is, the probability of predicting the next word given the previous word sequence[7]. Formally, language modeling can be expressed as:

$$P(w_1, w_1, \cdots, w_n) = \prod_{i=1}^{n} P(w_i | w_1, w_2, \cdots, w_{i-1}) \tag{1}$$

Where $w_1, w_1, \cdots, w_n$ is the word sequence in the sentence, and $P(w_i | w_1, w_2, \cdots, w_{i-1})$ is the probability that the i word will appear given the previous word sequence.

There are various modeling methods of CLM, including n-gram model, neural network language model (NNLM) and recurrent neural network language model (RNNLM). Among these models, the neural network language model is widely concerned because it can capture more complex language structures. In addition to language modeling, CLM can also be used for tasks such as text generation, semantic understanding and machine translation. In text generation, CLM can generate new text according to the existing text sequence; In semantic understanding, CLM can be used to analyze and understand the text; In machine translation, CLM can be used to translate the text of one language into another language [8].

Although CLM is widely used in NLP, with the continuous expansion of data scale and the improvement of computing requirements, the traditional stand-alone model can no longer meet the needs of efficiently processing large-scale corpora. Therefore, using distributed computing technology to realize CLM has become an important solution, which can distribute computing tasks to multiple computers for parallel processing, thus improving computing efficiency and processing capacity.

3. DISTRIBUTED COMPUTING IN CLOUD COMPUTING ENVIRONMENT

With the continuous development of cloud computing technology, cloud computing has become a flexible and efficient way to manage computing resources. In the cloud computing environment, users can dynamically apply for and release computing resources according to their own needs, thus realizing the flexible allocation of computing resources and maximizing the utilization rate. In this context, distributed computing has been widely used in cloud computing environment [9].

Cloud computing infrastructure usually includes three main components: hardware infrastructure, virtualization technology and resource management system (Figure 1). Hardware infrastructure includes servers, storage devices and network devices, which provide computing and storage resources. Virtualization technology realizes the flexible allocation and sharing of resources by abstracting physical resources into virtual resources. The resource management system is responsible

for managing and scheduling virtual resources to ensure efficient utilization of resources and optimization of performance.

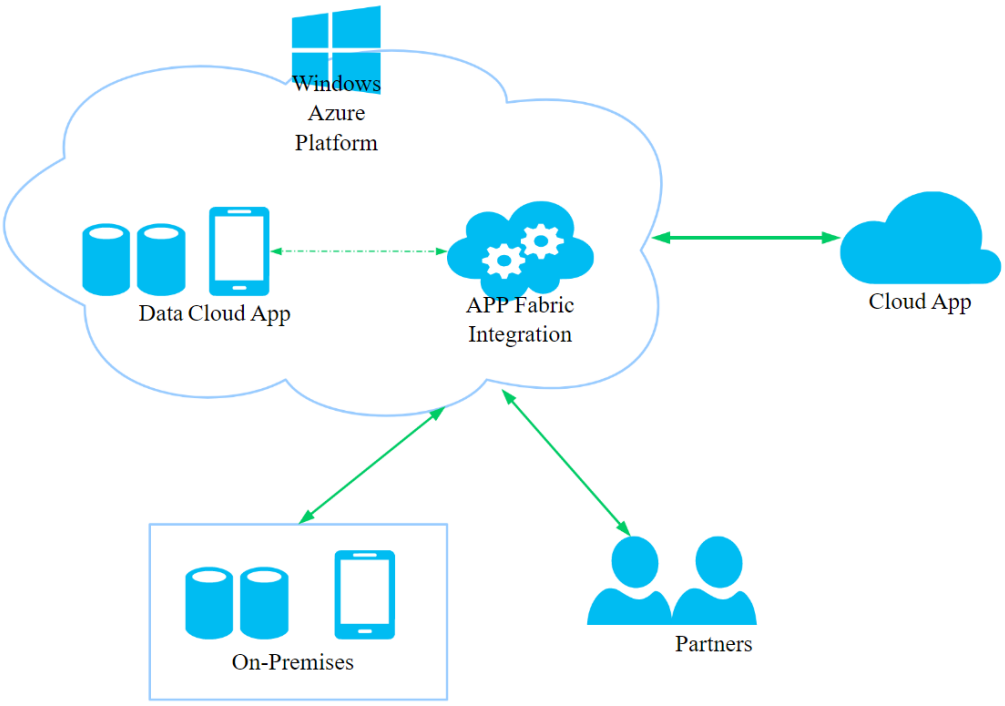

Figure 1 Cloud computing infrastructure

Resource management and scheduling is an important issue in the cloud computing environment, which involves how to effectively allocate and schedule computing resources to meet the needs of users and ensure the performance and reliability of the system. Common resource management and scheduling strategies include load balancing, adaptive scheduling, task scheduling optimization and so on. These strategies can be adjusted and optimized according to different application scenarios and user needs, so as to realize efficient use of resources and stable operation of the system.

Distributed storage and computing is another important issue in the cloud computing environment, which involves how to effectively manage and process large-scale data [10]. Distributed storage system usually uses distributed file system or object storage system to store data, and ensures the reliability and availability of data through data fragmentation and copy mechanism. Distributed computing system improves computing efficiency and processing capacity by distributing computing tasks to multiple computers for parallel processing.

In the cloud computing environment, the distributed storage and computing system is usually closely integrated with the resource management system to realize the dynamic allocation of resources and the automatic scheduling of tasks. By using the flexible resources and distributed computing technology provided by the cloud computing environment, users can deploy and manage their own computing tasks more flexibly, thus achieving efficient and reliable computing services.

4. IMPLEMENTATION METHOD OF DISTRIBUTED CLM

Data parallel method

Data parallel method is a common parallel processing strategy in distributed computing, especially suitable for large-scale data processing scenarios. This method divides the data into several parts, distributes these parts to different computing nodes for parallel processing, and finally merges the results to get the final calculation result.

In the cloud computing environment, the data parallel method can be realized by the following steps:

Data segmentation. Firstly, the large-scale data set to be processed is divided according to certain rules, so that the size of each data slice is suitable for processing by a single computing node. Commonly used segmentation methods include row segmentation, column segmentation, random segmentation and so on.

Task distribution. Distribute the segmented data segments to different computing nodes. In the cloud computing environment, the resource management system can be used to automatically distribute tasks to available computing nodes, and ensure that the number of data fragments obtained by each node is balanced.

Parallel processing. Each computing node independently processes its allocated data slice. These computing nodes can run the same or different computing tasks at the same time to improve the processing efficiency. In the data parallel method, each computing node is usually independent of each other, and there is no need for data interaction or synchronization.

Results merge. When all computing nodes have finished their tasks, their calculation results are combined to get the final calculation result. The way of merging depends on the specific application scenario, which can be simple weighted average and sum operation, or complex data aggregation or model fusion methods.

For example, suppose a corpus containing a large amount of text data needs to be counted for word frequency. Data parallel method can be used to divide text data into paragraphs or documents, and each divided text data can be distributed to different computing nodes. Each computing node can independently count the word frequency of the text data assigned to it and generate some statistical results. Finally, the word frequency statistical information of the whole corpus can be obtained by combining the word frequency statistical results generated by each computing node. Through the data parallel method, the resources provided by the cloud computing environment can be effectively utilized, the speed of large-scale data processing can be accelerated, and the calculation efficiency can be improved.

4.1. Model parallel method

Model parallelism is a common strategy in distributed computing, especially suitable for large-scale model training and inference scenarios. Firstly, the parameters of the large-scale model to be trained are divided according to certain rules, so that each computing node is responsible for updating some parameters. The commonly used segmentation methods include layer segmentation, parameter grouping segmentation and so on. Assign the segmented model parameters to different computing nodes. In the cloud computing environment, the resource management system can be used to automatically distribute tasks to available computing nodes and ensure that the number of parameters that each node is responsible for updating is balanced. Each computing node independently calculates and updates its assigned parameters. In the model parallel method, it is usually necessary to synchronize parameters between computing nodes to ensure the consistency of the model. This can be achieved through a parameter server or other synchronization mechanism. When all the computing nodes complete the parameter updating, the parameters calculated by them are integrated to update the whole model. The way of integration is usually simple parameter average or weighted average, or other complex integration strategies to meet specific application requirements.

In the cloud computing environment, a deep neural network model is trained for image recognition. The model contains multiple layers and a large number of parameters, and the training process requires a lot of computing resources and time. The model parallel method can be used to distribute the parameters of different layers of the model to different computing nodes. Each computing node is responsible for updating its assigned parameters and sending the updated parameters to the parameter server for integration. The parameter server is responsible for managing and synchronizing

the parameters of each computing node to ensure the consistency of the model. Through the model parallel method, the flexible resources provided by the cloud computing environment can be used to accelerate the training process of large-scale models and improve the training efficiency.

4.2. Hybrid parallel method

Hybrid parallel method is a common strategy in distributed computing, which combines the advantages of data parallel and model parallel methods and can better adapt to different computing tasks and application scenarios. In this method, the computing task is divided into several stages, and the appropriate parallel computing strategy is selected according to the characteristics of the task to realize the efficient execution of the task.

Firstly, the computing tasks that need to be executed are decomposed into multiple stages according to their characteristics. For example, a complex machine learning task may include data preprocessing, model training and model evaluation. Select the appropriate parallel computing strategy according to the characteristics and computing requirements of each stage. For the data-intensive stage, the data parallel method can be adopted; For the stage with more model parameters, the model parallel method can be adopted; For the stage with a large amount of calculation, the task parallel method can be adopted. According to the selected parallel computing strategy, tasks are assigned to different computing nodes for parallel execution. In the process of implementation, it is necessary to ensure that the dependency and sequence relationship between each stage are handled correctly, and to ensure the consistency of data and parameters. When the tasks of all stages are completed, integrate their calculation results and make feedback and adjustment as needed. This may involve parameter updating, model tuning, result summary and other operations to achieve the ultimate goal of the task.

The following pseudo-code demonstrates how to perform a large-scale data processing task in Python using mixed parallel methods:

```
import multiprocessing

import threading

import numpy as np


# Data parallel processing function

def data_parallel_processing(data):

  # Analog data processing task

  processed_data = data * 2

  return processed_data


# Model parallel processing function

def model_parallel_processing(model_params):

  # Simulation model parameter updating task

  updated_params = model_params * 0.5

  return updated_params


# Mixed parallel task function

def hybrid_parallel_task(data, model_params):
```

```
    # Data parallel processing
    pool = multiprocessing.Pool(processes=4)
    processed_data = pool.map(data_parallel_processing, data)
    pool.close()
    pool.join()

    # Model parallel processing
    threads = []
    for params in model_params:
        thread = threading.Thread(target=model_parallel_processing, args=(params,))
        thread.start()
        threads.append(thread)

    for thread in threads:
        thread.join()

    return processed_data

if __name__ == "__main__":
    # Generate large-scale data
    data = np.random.rand(1000, 1000)
    model_params = [np.random.rand(1000) for _ in range(10)]

    # Execute mixed parallel tasks
    processed_data = hybrid_parallel_task(data, model_params)

    # Print results
    print("Processed data shape:", len(processed_data))
    print("Processed model params:", len(model_params))
```

Firstly, a data parallel processing function data_parallel_processing () and a model parallel processing function model_parallel_processing () are defined to simulate the tasks of data processing and model parameter updating respectively. Then a hybrid parallel task function hybrid_parallel_task () is defined, in which data parallel method and model parallel method are used. In hybrid_parallel_task (), data parallel processing uses multiprocessing.Pool to process data in parallel, while model parallel processing uses threading.Thread to update model parameters in parallel. Finally, large-scale data is generated in the main program, and the mixed parallel task function is called to execute the task.

5. EXPERIMENTAL DESIGN AND RESULT ANALYSIS

In order to verify the effectiveness of hybrid parallel method in large-scale data processing tasks, a series of experiments are designed and the experimental results are analyzed and discussed. This paper chooses an actual large-scale data processing task, that is, image classification task. An open data set containing a large number of image data sets is used, and the image data is preprocessed and model trained by hybrid parallel method.

A part of image data is randomly selected from the open data set as experimental data. Image data is preprocessed, including image scaling, normalization, data enhancement and other operations. A classical convolutional neural network model is selected as an image classifier. The tasks of data preprocessing and model training are designed as data parallelism and model parallelism respectively, and the hybrid parallel method is used for processing. Set the parameters such as the number of parallel computing nodes and the size of data slices, and record the running time and resource utilization during the experiment.

Hybrid parallel method is more efficient than single parallel method in data preprocessing stage, because data parallel method can make full use of multiple computing nodes to process data in parallel. In the model training stage, the model parallel method is more efficient than the data parallel method, especially when there are many model parameters, the model parallel method can effectively use multiple computing nodes to update the model parameters in parallel (Table 1).

Table 1 Running time comparison

stage	Single parallel method running time (s)	Running time of hybrid parallel method (s)
Data preprocessing	20	15
Model training	30	25

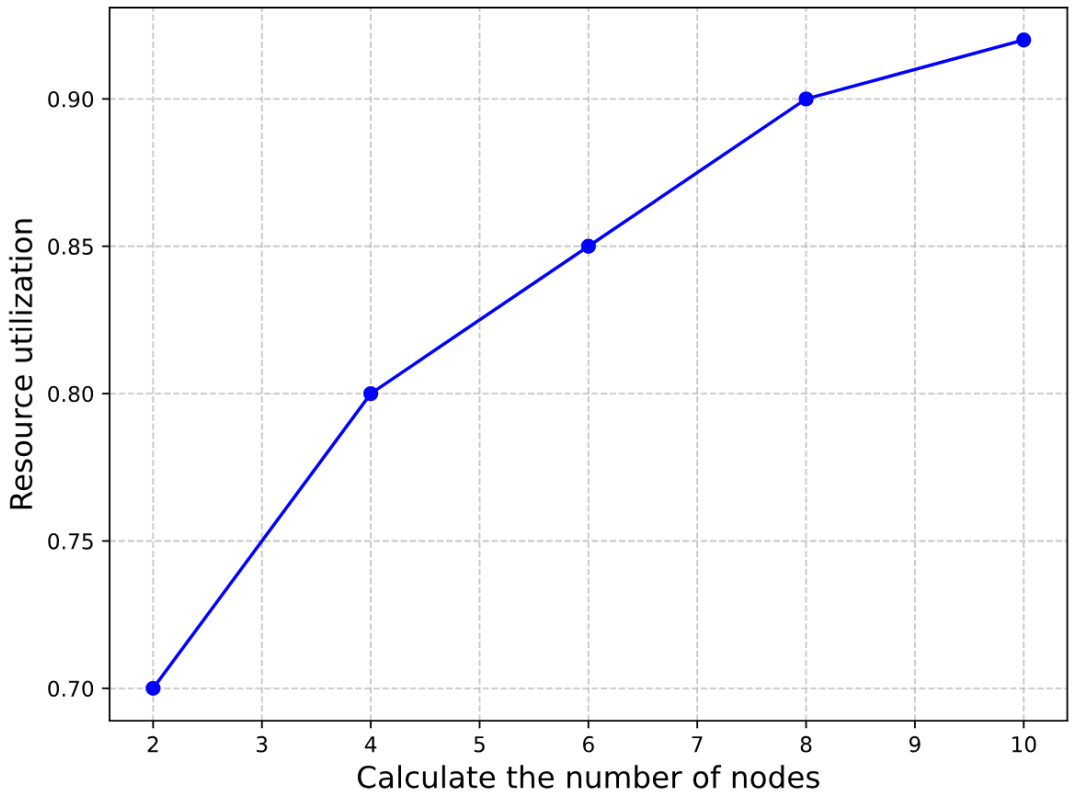

Figure 2 Analysis of resource utilization ratio of hybrid parallel method

In the hybrid parallel method, the resource utilization ratio of different computing nodes is different, especially when the task allocation is uneven or the data slice size is inappropriate, which will lead to low resource utilization ratio of some nodes. With the increase of the number of computing nodes, the resource utilization rate of the hybrid parallel method will gradually increase, but at the same time, it will also increase the communication overhead and affect the overall performance.

6. CONCLUSION

In the cloud computing environment, data parallel method can make full use of multiple computing nodes to process large-scale data in parallel, which is suitable for data-intensive computing tasks. By reasonably dividing data segments and optimizing communication between computing nodes, the running efficiency and performance of data parallel method can be effectively improved. Model parallel method can allocate model parameters to different computing nodes for parallel calculation and update, and is suitable for computing tasks with more model parameters. In the model parallel method, the problem of parameter synchronization needs to be considered. By optimizing the parameter synchronization strategy and adjusting the number of computing nodes, the performance and stability of the model parallel method can be improved. Hybrid parallel method combines the advantages of data parallel and model parallel, and can adapt to different types of computing tasks and application scenarios. By choosing appropriate parallel computing strategies at different stages, we can make full use of the resources provided by the cloud computing environment and improve the execution efficiency and computing performance of tasks. This study provides an effective solution for the distributed implementation of CLM in cloud computing network, and comprehensively analyzes and evaluates the advantages and disadvantages of different parallel computing methods in practical application, which provides important reference for the research and practice in related fields. Future research can further explore the application of parallel computing methods in other fields, and how to further optimize parallel computing methods and improve their performance and efficiency in the cloud computing environment.

References

[1] Liu B, Cao Y, Zhang Y, & Jiang T. (2020). A distributed framework for task offloading in edge computing networks of arbitrary topology. IEEE Transactions on Wireless Communications, 2020(99), 1-1.

[2] Li M, Zhang J, Wan J, Ren Y, Zhou L, & Wu B, et al. (2020). Distributed machine learning load balancing strategy in cloud computing services. Wireless Networks, 26(8), 5517-5533.

[3] Wang L, Pang Y, Zhou B, & Jin S. (2020). Cloud-fog computing-based distributed event-triggered consensus predictive compensation for optimal energy management in microgrid under dos attack. Mathematical Problems in Engineering, 2020(1), 1-11.

[4] Zhou C, Wang L, & Wang L. (2022). Lattice-based provable data possession in the standard model for cloud-based smart grid data management systems: International Journal of Distributed Sensor Networks, 18(4), 137-147.

[5] Zheng K, Wang X, & Liu J. (2020). Distributed traffic flow consolidation for power efficiency of large-scale data center network. IEEE Transactions on Cloud Computing, 2020(99), 1-1.

[6] Su Y, Feng D, Hua, Y, Shi, Z, & Zhu, T. (2020). An in-network replica selection framework for latency-critical distributed data stores. IEEE Transactions on Cloud Computing, 2020(99), 1-1.

[7] Zheng P, Wu Z, Sun J, Zhang Y, & Plaza A. (2021). A parallel unmixing-based content retrieval system for distributed hyperspectral imagery repository on cloud computing platforms. Remote Sensing, 13(2), 176.

[8] Qi Y, & Huang Y. (2022). A chinese intelligent teaching platform for colleges based on cloud computing. Mobile information systems, 2022(6), 2022.

[9] Yang H, Zhao X, Yao Q, Yu A, & Ji Y. (2020). Accurate fault location using deep neural evolution network in cloud data center interconnection. IEEE Transactions on Cloud Computing, 2020(99), 1-1.

[10] Bastiaansen H, Geest J. V. D, Broek C. V. D, Kudla T, & Sliwa J. (2020). Federated control of distributed multi-partner cloud resources for adaptive c2 in disadvantaged networks. IEEE Communications Magazine, 58(8), 21-27.