Research on Parallel LSTM Algorithm Based on Spark

Yangyang Zhao, Wei Niu and Meinan Wang

May 8, 2021

Research on Parallel LSTM Algorithm Based on Spark

Zhao Yangyang[1], Niu Wei[1], Wang Meinan[1]

(1. Xi’an Aeronautics Computing Technique Research Institute, AVIC, Xi’an 710068, China)

Abstract: Aiming at the problems of large amount of data collected by airborne sensors, lack of data association, and low processing efficiency, this paper proposes a parallel LSTM algorithm model suitable for Spark platform. First, use the Spark platform to complete the traversal scan operation in the memory RDD of all nodes in the distributed cluster, and combine the directed acyclic graph to create a Pipeline pipeline to implement a parallel computing framework. An algorithm model to optimize the parameters of LSTM neural network is proposed, and load balancing processing method is introduced to realize that all nodes of the distributed system can share the computing tasks in a balanced manner. The experimental results show that compared to the stand-alone case, the parallelized LSTM algorithm improves the efficiency. The prediction efficiency of the LSTM algorithm model after load balancing processing is higher, which shows that the distribution of traversal tasks of each node is more balanced and the degree of parallelization is higher.

Key words: Spark; parallel computing; LSTM; load balancing processing

0 Introduction

The stable operation of aero engines is the key to ensuring the safety and reliability of the aircraft. Tracking the operating status of the engine and predicting possible future failures is a key and necessary part of the predictive maintenance of the aircraft.

There are three main methods for the health management system of aero-engines. One is the model-driven method, which is mainly based on the physical characteristics of the aero-engine to establish a physical failure model. For example, Luo[1] uses known mechanics and mechanical principles to analyze crack fatigue data. Carry out research and establish its physical failure model to predict the corresponding failure time. Such methods rely on physical failure mechanisms that are difficult to grasp, and are often targeted at a single model, with insufficient generalization capabilities; the second is knowledge-driven methods, which use expert systems and ontology reasoning knowledge to drive RUL prediction research, such as Hu[2], etc., considering maintenance In the actual working conditions of insufficient activity, the parameters are estimated based on the maximum likelihood estimation method and the Bayesian method, and the stochastic degradation model is established to derive the RUL probability density function; Sun[3] and others aim at the state characteristics of some potential failures of aeroengines, It is proposed to establish aeroengine RUL prediction model based on random filtering theory and K-means method. This kind of method is suitable for small sample data set processing, but it is not suitable for the processing and analysis of massive data. The third is data-driven method. The traditional method is to establish mathematical statistical model to analyze and process data. At present, it is mostly based on machine learning, deep learning and neural network methods to carry out theoretical research and application practice, comprehensively consider and weigh factors such as data scale, data structure characteristics, architecture computing power, algorithm robustness, model generalization ability, prediction accuracy, etc., on this basis Choose a reasonable method to achieve RUL prediction.

Long and short-term memory neural network (LSTM), as an improved time recurrent neural network (RNN), not only has neural network distributed storage, self-organization, self-adjustment and nonlinear fitting capabilities, but also can learn long and short-term time series information. It is suitable for processing and forecasting interval and delay events in time series.

The distributed computing architecture is suitable for the rapid processing and computing tasks of large-scale data. At this stage, Apache Hadoop, Spark and Flink are the main ones. Spark has the advantages of parallel computing on multiple nodes in the cluster that Hadoop MapReduce has. The support of iterative operations and low-latency interactive data mining tasks effectively solve the time problem of large-scale data parallel computing[4]. In this paper, a parallel LSTM algorithm model is designed in combination with the Spark architecture. The algorithm execution process is modeled based on a directed acyclic graph (DAG), and a load-balancing multi-node operation strategy is introduced to enable the algorithm model to iterate quickly and optimize LSTM network parameters.

1 Spark parallel computing framework

1.1 Parallel computing process of Spark architecture

The running architecture of Spark consists of Driver and Executor. Driver is responsible for DAG segmentation of user code and divide it into different stages, and then submit the task scheduling corresponding to each stage to Executor for calculation, so that Executor executes the same stage in parallel task, the specific calculation process is shown in Figure 1.

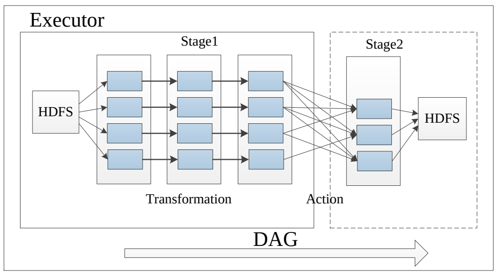


Fig.1 Spark parallel computing flowchart

In addition, the data is processed by the machine learning workflow (Pipeline) that creates the training model[5], and the pipeline processing flow is shown in Figure 2.

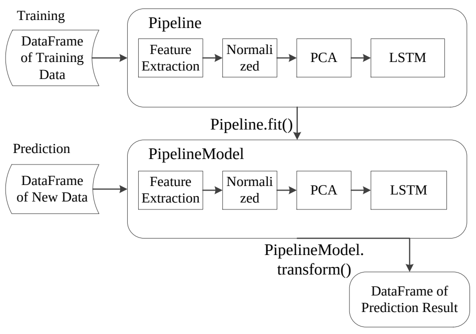


Fig. 2 Training model flow on the Pipeline

Using machine learning algorithm training data in Pipeline, a PipelineModel is generated after execution. Pipeline and PipelineModel help to ensure that the feature processing flow of training data and test data is consistent. A Pipeline structure contains one or more Stage modules for data processing. Each stage will complete a data processing task, including feature extraction, normalization processing, and feature dimensionality reduction. With these stages that deal with specific problems, Organize and create a Pipeline in a specific logical sequence, making pipelined machine learning more efficient and faster than single-step independent modeling.

1.2 Load balancing strategy of each node on the Spark platform

In the Spark platform, the multi-node technology in the distributed system is used to parallelize the algorithm model to improve the efficiency of the algorithm. In this paper, the load balancing strategy is adopted to ensure that each node can share the traversal task in the algorithm in a balanced way, and give full play to the potential of each node to participate in the operation. The specific implementation method is as follows[6].

Set the number of loads as *n*, and set up the load monitoring model in t period. The changes of the load are irregular on the time axis. Analyzing the load connected to the Spark platform can predict the number of traversed loads in the time series. The actual number of neural network layers to be traversed in the previous time period and the predicted number are used to establish the prediction model for the current time period, and the weights are constantly revised to approximate the actual number. Suppose *M(t)* is the predicted value of load quantity in *t* period, *m(t)* is the actual load quantity in *t* period, and ω is the weight factor. The predicted value of the load quantity in the next time period can be calculated according to formula (1):

$$M_{t+1} = M_t + \omega(m_t - M_t) \qquad (1)$$

Suppose ε is the predicted change value of the load quantity in two consecutive times, then:

$$\varepsilon = M_k - M_{k-1} \qquad (2)$$

In order to solve the error caused by a single calculation, this paper adopts the method of averaging multiple predicted values to increase the prediction accuracy. The calculation method is shown in formula (3):

$$\overline{M} = \frac{\sum_{k=1}^{n} M_k}{n} \qquad (3)$$

Judge the load change according to the difference between $|\varepsilon|$ and $\overline{M}$. When $|\varepsilon| > \overline{M}$, it means that the load of the node has changed significantly in the current time period, continuing the load time slice of each node in the previous time period Set the strategy; when $|\varepsilon| < \overline{M}$, it means that the load change of the node is small, and measures need to be taken to adjust the load time slice strategy of the node in time and increase its load appropriately.

2 Parallel LSTM algorithm design

The LSTM network standard module includes the standard cyclic layer in the RNN network, and introduces a threshold mechanism—"memory" control gate to control the accumulation speed of information. The specific structure is shown in Figure 3.

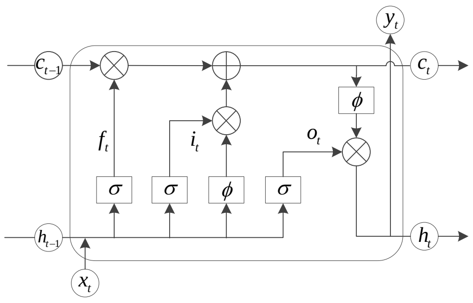


Fig. 3 The LSTM Schematic diagram of LSTM network standard module

The LSTM network standard module is divided into two parts, namely the long-term state c_t and the short-term state h_t. At the same time, three control gates are added along the state path[9]: forget gate f_t, input gate i_t, and output gate o_t to adjust information.

(1) Forget gate: It is to control at time t, the information component of the previously long-term state c_{t-1} after passing f_t are kept in the current c_t, expressed as $f_t \otimes c_{t-1}$. The realization formula of f_t is:

$$f_t = \sigma(\omega_f^{\mathrm{T}} \cdot [h_{t-1}, x_t] + b_f) \qquad (4)$$

（2）Input gate: It is to control at time t, the information component of the input x_t and the previously short-term state h_{t-1} after passing i_t are kept in the current c_t, expressed as $i_t \otimes \tilde{c}_t$. The realization formula of i_t is:

$$\begin{cases} i_t = \sigma(\omega_i^{\mathrm{T}} \cdot [h_{t-1}, x_t] + b_i) \\ c_t = \phi(\omega_c^{\mathrm{T}} \cdot [h_{t-1}, x_t] + b_c) \end{cases} \qquad (5)$$

Where $\sigma(\cdot)$ is the Sigmoid function and $\phi(\cdot)$ is the Tanh function.

（3）Output gate: It is to control at time t, the information component of the current long-term state c_t after passing o_t is retained in the current h_t, expressed as $o_t \otimes \phi(c_t)$, and the realization formula of o_t is:

$$\begin{cases} h_t = y_t = o_t \otimes \phi(c_t) \\ c_t = i_t \otimes \tilde{c}_t + f_t \otimes c_{t-1} \\ o_t = \sigma(\omega_o^{\mathrm{T}} [h_{t-1}, x_t] + b_o) \end{cases} \qquad (6)$$

For data sets with relatively high data levels, this article proposes an algorithm parallelization scheme for LSTM. For a training set with a total sample size of Q, it is divided into N groups for parallel calculation, and each group runs in parallel. Serial calculation using traditional LSTM algorithm. After calculating the result X in each group, Xs can form a new sequence (x_1, x_2,…, x_n). For this new sequence of length N, continue to repeat the above steps. After this cycle several times, the length of the new sequence itself will reach a threshold W. For this sequence of length W, the LSTM algorithm is used for serial calculation to calculate the final result. The grouping diagram is shown in Figure 4.

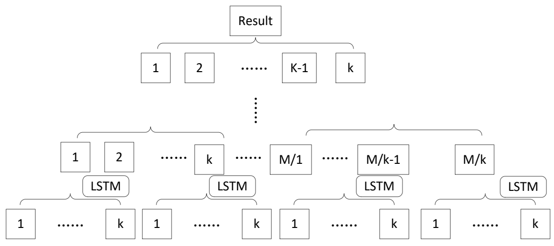


Fig. 4 The LSTM Schematic grouping diagram

3 Test results and analysis

In order to verify the performance of the parallelized LSTM algorithm model, an example simulation is carried out in this section. The example simulations were carried out on a single machine and a Spark distributed cluster. Build a Hadoop+Spark big data platform. In terms of hardware, it is built by six 8-core processor servers, each with 16G memory, and uses zookeeper components for distributed coordination and management. One server is selected as the leader, and the remaining 5 are Follower and Observer; in terms of software, the operating system is Ubuntu16.04, Hadoop and Spark cluster environments are installed and deployed, and open source software is used to form a fully distributed big data platform, which is more efficient than a single-node pseudo-distributed platform.

Select the Spark framework in the stand-alone mode and the MapReduce framework in the cluster mode to compare, process data sets of different sizes and record the time it takes to run a complete job. The result is shown in Figure 5.

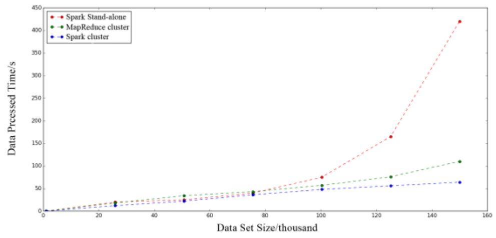
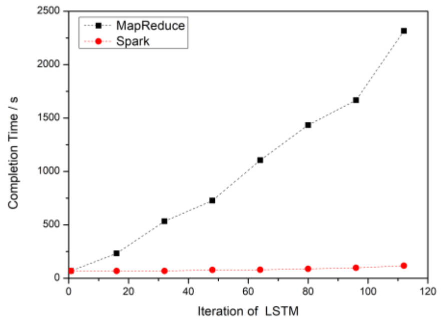


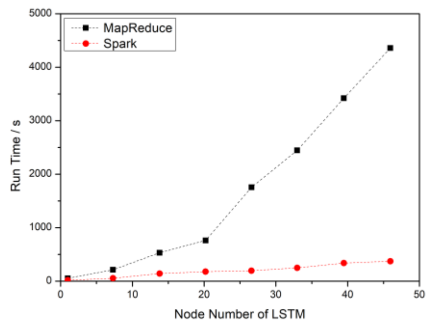
Fig. 5 Data processing time under different computing frameworks

It can be found that as the scale of data continues to increase, the data processing speed advantage of a fully distributed Spark framework computing cluster over the Spark framework in stand-alone mode and the MapReduce framework in cluster mode becomes more and more obvious. This is because Spark distributes tasks to multiple processors, and uses lazy evaluation to optimize the entire data preprocessing process, so that data processing can be completed efficiently and quickly. At the same time, because the Spark framework uses high-level data abstraction, flexible RDD sets, and high-reuse memory calculations, compared to Hadoop MapReduce, it saves data reading time and improves the real-time performance of data processing.

In addition, this paper selects 2.6G aero-engine's health data to inputs the parallelized PSO-LSTM neural network algorithm model, which is respectively calculated under the framework of Spark and MapReduce. The results are shown in Figure 6.



(a)



(b)

Fig. 6 The performance comparison with Spark and MapReduce

Figure (a) shows that as the number of iterations increases, the performance of Spark's computing performance is particularly improved. This is because Spark is based on memory computing, the intermediate results are stored in memory, and the parallelized LSTM network is used, which greatly improves Spark The ability of parallel computing reduces the completion time of iterative tasks. Figure (b) shows that as the number of GA-LSTM network nodes increases, the computational complexity of the system increases, and the running time of Spark and MapReduce will increase accordingly. After multiple verifications, the computational efficiency of Spark combined with parallel LSTM network algorithms is higher than that of MapReduce. An order of magnitude higher.

4 Conclusion

In this paper, the Spark platform is used to implement the parallelized LSTM algorithm model, which can effectively improve the efficiency of the prediction algorithm model. The use of load balancing strategies can further optimize the efficiency of each node and increase the degree of parallelization. However, the current research is only tested in terms of running time, and the accuracy of data mining has not been significantly improved. Subsequent research will focus on how to further enhance accuracy while improving operational efficiency.

Reference:

[1] Luo Bin. Research on fleet maintenance decision-making method based on structural fatigue life prediction[D]. Harbin: Harbin Institute of Technology, 2018.

[2] Hu C H, Hong P, Wang Z Q, et al. A new remaining useful life estimation method for equipment subjected to intervention of imperfect maintenance activities[J]. Chinese Journal of Aeronautics, 2018, 31(3):514-528.

[3] Sun Shaohui, Wang Huawei, Li Wei. Prediction of remaining life of aero-engine during potential failure period[J]. Aeronautical Computing Technology, 2012, 42(1): 8-11.

[4] Meng Xiaofeng, Kindness. Big Data Management: Concepts, Techniques and Challenges[J]. Computer Research and Development, 2013, 50(1):146-169.

[5] A. Svyatkovskiy, K. Imai, M. Kroeger, et al. Large-scale text processing pipeline with Apache Spark[C]//2016 IEEE International Conference on Big Data. IEEE, 2016.

[6] Yang Jixiang, Tan Guozhen, Wang Rongsheng. Overview of parallel and distributed computing dynamic load balancing strategies [J]. Chinese Journal of Electronics, 2010, 38(005): 1122-1130.

[7] Li Jingfeng, Chen Yunxiang, Xiang Huachun, et al. Residual life prediction of aero-engine based on LSTM-DBN[J]. Systems Engineering and Electronic Technology, 2020, 42(7): 211-218.