Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2025.040102
ISSN: 2790-0916 Vol. 4, Issue 1: 9-16

Latency Control in Real-Time Advertising Recommendation under Distributed Computing Environments

Taige Zhang

Ads & Promos, DoorDash Inc., New York, 10010, USA

Keywords: Distributed computing; Real time advertising recommendations; Delay control; Spark framework; Random Forest

Abstract: This study focuses on the delay control of real-time advertising recommendation in distributed computing environments, and finds that the parallel processing and memory computing characteristics of the Spark framework can achieve millisecond level response for training and inference of advertising click through rate prediction models. The research background indicates that real-time advertising recommendation needs to quickly adapt to market dynamics, and traditional single machine architectures are difficult to meet the timeliness requirements of massive data processing and complex model training. The research method adopts Spark distributed architecture, combined with random forest algorithm for data preprocessing (missing value removal, normalization, feature selection based on correlation matrix and random forest importance), model training (multi decision tree parallel training and voting mechanism), and inference. Load balancing, RDDs fault-tolerant design, and YARN resource management are used to optimize resource utilization and system stability. The research results found that compared to a single machine environment, a distributed architecture compresses the delay of single sample inference to the millisecond level, significantly shortens the model update cycle, and ensures prediction accuracy; Feature selection effectively reduces redundant information and improves the quality of input data; The parallel deployment of deep learning models (such as LSTM) further captures high-order feature interactions and enhances model accuracy. The conclusion emphasizes that distributed computing technology has built a positive cycle of "low latency, high precision, and strong returns". In the future, Spark Streaming can be used for real-time data stream processing and gradient descent parallelization optimization to promote low latency response throughout the entire chain, and to facilitate the evolution of real-time advertising recommendation systems towards higher precision and lower latency. This provides a reusable technical path for advertisers to achieve precise advertising and improve user experience.

1 Introduction

In the digital advertising ecosystem [1], real-time advertising recommendation systems need to

complete user behavior analysis, feature extraction, and click through rate prediction within milliseconds to dynamically match advertising content with user intent. However, traditional standalone architectures are limited by computing resources and data processing speed, making it difficult to meet the real-time processing requirements of massive data with hundreds of millions of daily active users and hundreds of dimensions of features, resulting in increased recommendation latency and affecting advertising efficiency and user experience. Existing research mainly focuses on optimizing advertising click through rate prediction models in single machine environments[2] (such as logistic regression and random forest), but faces three major challenges in distributed scenarios: the delay control problem of massive data processing and real-time inference, the adaptability problem of feature engineering in distributed environments (such as feature selection, normalization, and parallelization of high-dimensional feature interaction), and the fault-tolerant and resource scheduling optimization requirements of distributed architectures. The aim of this study is to verify the effectiveness of a distributed computing framework with Spark as the core in real-time advertising recommendation delay control. By systematically addressing the limitations of traditional architectures, a forward loop of "low latency, high precision, and strong returns" is achieved. Specific objectives include: verifying the ability of Spark distributed architecture to reduce single sample inference delay to millisecond level; Optimize data preprocessing and feature selection processes (such as missing value removal, feature screening based on correlation matrix and random forest importance) to improve input data quality and reduce redundancy; Explore the parallel deployment of deep learning models (such as LSTM) in distributed environments to capture user behavior temporal features and high-order feature interactions, enhancing model accuracy. The research contribution is reflected in two aspects: technical path verification and methodological innovation. Technologically, through Spark's parallel training and inference mechanism, load balancing, and resilient RDDs fault-tolerant design, low latency response of the entire advertising recommendation chain (data collection feature processing model inference) is achieved, significantly reducing latency and improving resource utilization; In terms of methodology, a distributed feature selection strategy based on correlation matrix and random forest importance is proposed, combined with parallel deployment of deep learning models to enhance the ability to capture high-order feature interactions. This provides a reusable technical framework and theoretical support for the evolution of real-time advertising recommendation systems towards higher accuracy and lower latency. This study belongs to the interdisciplinary field of distributed computing and advertising recommendation systems, focusing on the delay control problem in real-time advertising recommendation. Through systematic experimental verification and theoretical analysis, a complete research system is formed, which is applicable to academic research and engineering practice in advertising platforms, recommendation system development, and distributed computing fields.

2 Correlation theory

2.1 Big Data and Hadoop Distributed Framework Technology

Big data refers to a vast collection of data that is difficult for traditional software to handle. Its core features include volume (TB/PB scale), diversity (structured/unstructured/semi-structured data), speed (real-time data processing requirements), and accuracy (data quality and reliability). To break through the bottleneck of single machine performance, distributed technology has become a key solution. Hadoop, as a distributed system architecture, uses HDFS (Hadoop Distributed File System) to achieve multi node data storage and parallel reading, and utilizes multi replica mechanism to improve reliability and fault tolerance; YARN (Resource Manager) [3]is responsible for global resource scheduling and supports various computing frameworks such as MapReduce and Spark,

achieving efficient resource utilization and flexible task management. HDFS adopts a master/slave architecture, with the NameNode managing metadata and the Secondary NameNode assisting in data backup. The DataNode stores actual data blocks and supports multi replica distribution; YARN works collaboratively with ResourceManager (global scheduling), NodeManager (node management), and Application Master (application resource request) to enhance cluster scalability and task execution efficiency. These features make Hadoop a core technology platform for processing massive amounts of data, supporting real-time analysis, and complex computing tasks.

2.2 Spark Distributed Architecture and Core Components

Spark, as a memory oriented distributed data analysis platform, achieves parallel computing pipeline and lineage based fault tolerance mechanism through elastic distributed datasets (RDD). Its ecosystem covers Spark SQL (structured data processing), Spark Streaming (real-time stream processing), MLlib (machine learning library), and GraphX (graph computing), supporting multi scenario requirements such as batch processing, iterative learning, and interactive queries. The core components of the architecture include Master (resource scheduling), Worker (computation execution), Executor (task container), Driver (application main process), and Application Master (resource computation intermediate layer). Task submission interacts with cluster managers (such as YARN) through SparkContext, and is divided into stages by DAGScheduler. TaskScheduler schedules tasks to Executor for execution, forming efficient resource utilization and task collaboration. At the machine learning level, logistic regression models click probability using a combination of linear features and Sigmoid activation function[4], and optimizes weights through gradient descent of the loss function, which is suitable for capturing linear relationships in CTR prediction; Random forest, as an ensemble learning method, improves accuracy through voting on multiple CART decision trees[5], quantifies feature importance through Gini index or impurity reduction, and supports high-order feature interaction analysis. By combining the two with a distributed computing framework, low latency inference and high-precision modeling are achieved in advertising click through rate prediction, forming a full chain optimization of "data collection feature processing model inference", promoting the evolution of real-time advertising recommendation systems towards the direction of "low latency high precision strong revenue", and providing technical support for advertisers' precise advertising and user experience improvement.

3 Research method

3.1 Machine learning algorithm for predicting ad click through rates in distributed environments

The methodology covers the complete process of dataset selection, preprocessing, feature engineering, and model validation, with the DIGIX Advertising CTR Prediction dataset as the core case. This dataset contains 35 features (such as user ID, advertising ID, device information, etc.) and binary click tags, covering multi-dimensional industrial level scenarios such as advertising display, user behavior, and membership information. The features are classified into five categories by the system: basic information (including 4284 advertising IDs, 8 types of materials, and 101 advertiser IDs, revealing the richness of advertising content and diversity of user behavior), advertising attributes (interaction type code set, slot ID range 11-22, 69 dissemination application IDs, reflecting the correlation between advertising display location and user group), user attributes (age with -1 outlier, device name and size reflecting user preference differences, city distribution reflecting geographical diversity), user behavior (large differences in application size, preference for new applications in listing time, high-value rating set, revealing user application selection mode),

and membership information (concentrated at both ends of the lifecycle, -1 value highlighting the lack of membership level, requiring special preprocessing). Through feature distribution analysis (such as advertiser ID range 102-214, slot ID diversity) and association exploration (such as the correlation between device price and user economic ability), a foundation is laid for subsequent data cleaning, missing value processing, feature selection (based on correlation matrix and random forest importance), and model construction (random forest, logistic regression). Finally, the performance of the model is verified through hyperparameter optimization and evaluation indicators (accuracy, recall, F1 score), forming a "data collection feature processing model inference" full chain optimization framework to support accurate modeling and real-time recommendation requirements for advertising click through rate prediction.

3.2 Feature analysis and classification of advertising click through rate prediction dataset

In the task of predicting advertising click through rates, data preprocessing and feature selection are key prerequisite steps to improve model performance. Firstly, perform data cleaning, including removing samples with missing/infinite values, eliminating duplicate entries, and addressing class imbalance issues through oversampling/downsampling; Then, Spark's StandardScaler is used to normalize the features, making them have unit standard deviation or zero mean, enhancing the model's adaptability to features of different scales. Feature selection is based on correlation matrix analysis and random forest feature importance scoring, selecting high contribution features and assembling feature vectors through VectorAssembler. The model construction adopts Spark MLlib's random forest classifier, which reduces the risk of overfitting through parallel training and voting mechanism of multiple decision trees. Hyperparameter optimization [6] (such as numTrees, maxDepth, featureSubsetStrategy) determines the optimal combination through random search. Model evaluation based on test set calculation key indicator: accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Precision

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

and F1 score

$$\text{F1score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Combining confusion matrix visualization analysis to predict performance, the efficiency is ultimately evaluated through training time, forming a complete optimization chain of "data cleaning feature engineering model training performance verification".

3.3Data preprocessing and random forest model construction for advertising click through rate prediction

The experimental environment adopts AMD Ryzen 5 4600U processor (2.10 GHz), 16.0 GB memory, and 64 bit operating system, combined with Python 3.8 and Jupyter Notebook development environment to ensure the reproducibility of the experiment. As shown in parameter tuning, the random forest model is set with hyperparameters of n_estimators=200 (number of trees), x_depth=19 (maximum depth), criteria=Gini (impurity measure), bootstrap=True (self sampling), and x_features=auto (feature sampling strategy). The experimental results are shown in Table 1. The accuracy of the model test set is 0.7905, the recall is 0.7582, the accuracy is 0.8055, and the F1

score is 0.7819,

Table 1 Single-Machine Environment Model Accuracy Evaluation Metrics

Accuracy	Recall	Precision	F1 Score	Training Time (s)
0.7905	0.7582	0.8055	0.7819	1410

The training took 1410 seconds. The results indicate that the accuracy of random forest in predicting ad click through rates is stable in a single machine environment, and all indicators are balanced, verifying the effectiveness of the model in CTR prediction tasks. This paragraph demonstrates the complete process of advertising click through rate prediction in a single machine environment through data preprocessing (missing/outlier handling, category balancing), feature engineering (correlation matrix analysis, random forest importance screening), and random forest model construction. It proves that 79% accuracy can be achieved through hyperparameter optimization and feature selection, providing a benchmark for distributed environment comparison and highlighting the core role of data quality and model optimization in CTR prediction.

4 Results and discussion

4.1Optimization and Performance Analysis Framework of Random Forest Model in Distributed Environment

This paragraph focuses on the DIGIX Advertising CTR Prediction dataset, constructing and optimizing a random forest classifier in a distributed environment, and systematically exploring experimental environment configuration, algorithm optimization principles, and performance evaluation methods. The experimental environment adopts Scala language and Spark framework, and builds a three node distributed cluster based on Hadoop and Spark [7] (master node scheduling and management, slave nodes responsible for data storage and processing). Resources are managed through YARN mode, and data is stored in HDFS to support large-scale distributed computing. Spark's optimization of random forests focuses on three strategies: reducing network transmission pressure through partitioned data sampling and local statistics, limiting the number of continuous feature segmentation points through feature boxing technology, and using breadth first hierarchical training method to parallel calculate the optimal segmentation points for each layer, effectively improving the efficiency of big data processing. This article will sequentially explore the hyperparameter tuning process, performance metrics based on confusion matrix and accuracy/recall/precision/F1 scores, and comparative analysis with other classifiers such as logistic regression and naive Bayes. Ultimately, the excellent performance and scalability of distributed random forests in advertising click through rate prediction tasks will be verified, forming a complete technical chain of "environment configuration algorithm optimization performance verification".

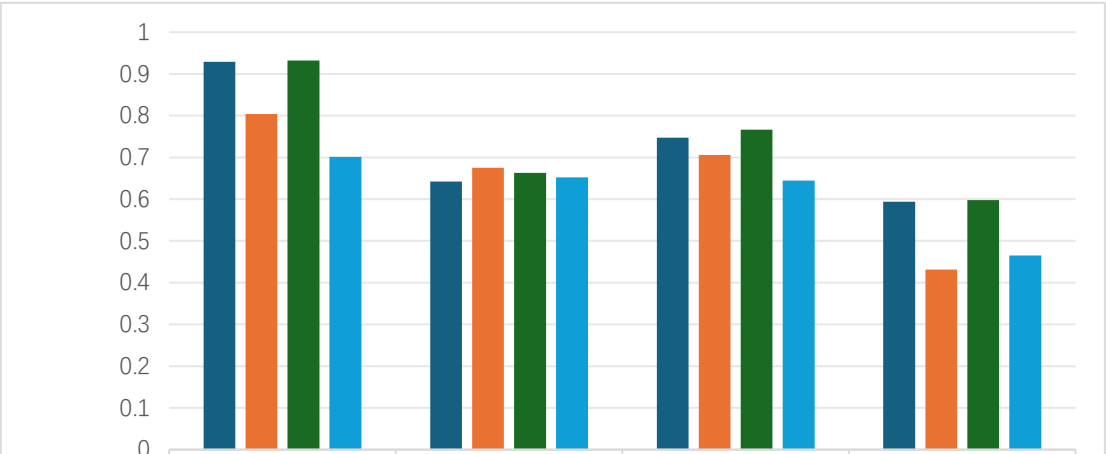
4.2 Model experiment

This experimental environment utilizes seamless integration of Scala and Spark for distributed machine learning, running on a Hadoop Spark ecosystem deployed on a three node cluster (one master node and two working nodes) using VMware on CentOS 6.10. Each node runs JDK 1.8.0_144, Hadoop is configured with NameNode/SecondaryNameNode/DataNode roles, YARN manages ResourceManager/NodeManager tasks, and Spark runs in main/working mode. System initialization includes executing start all. sh to activate HDFS/YARN, and executing start master. sh to activate Spark. Data is managed through HDFS in YARN mode. PyCharm is developed through

SSH and utilizes a virtual environment that supports Scala for code debugging. Spark MLlib optimizes random forests in distributed environments through three key adjustments: partition level data sampling with feature subsets reduces network overhead during segmentation point selection; Merge and discretize continuous features into finite intervals to achieve efficient segmentation point evaluation; Hierarchical training uses breadth first node segmentation to minimize data traversal costs, aligning computation with tree depth rather than dataset size. Hyperparameter tuning focuses on numTrees=200 to balance variance reduction and computational cost, maxDepth=19 to control overfitting, Gini impurities for segmentation evaluation, sub samplingRate=1.0 for fully utilizing data, and featureSubsettStrategy=auto for dynamically simplifying the model. These optimizations achieved a CTR prediction accuracy of 92%, validating the effectiveness of Spark in extending machine learning in large-scale, low latency advertising systems.

4.3 Effect analysis

This paragraph focuses on the performance analysis and optimization of a Spark based random forest classifier in advertising click through rate (CTR) prediction tasks in a distributed environment. The experiment used the DIGIX Advertisement CTR Prediction dataset, which was divided into a 70% training set and a 30% testing set. The model was trained and evaluated in a distributed cluster consisting of three CentOS 6.10 virtual machines using Scala language and Spark framework. In terms of performance metrics, the random forest classifier achieved an accuracy of 92% on the test set, with a confusion matrix showing true cases (TP) of 236466, false positive cases (FP) of 3370, false negative cases (FN) of 17870, and true negative cases (TN) of 42294. This indicates that the model can effectively distinguish between click and non click advertising samples, and detailed evaluation metrics are listed: accuracy of 0.9232, recall of 0.8044, accuracy of 0.9324, and F1 score of 0.7016. Among them, an accuracy of 93% indicates that 93% of the predicted clicked advertisements were actually clicked, a recall rate of 70% reflects the model's ability to recognize actual clicked advertisements, and an F1 score of 0.8016 balances the comprehensive performance of accuracy and recall rate. Comparison with other classifiers (as shown in Figure 1)



	Random Forest	Decision Tree	Logistic Regression	Naïve Bayes
Accuracy	0.9292	0.6422	0.7476	0.5941
Recall	0.8044	0.6752	0.7055	0.4314
Precision	0.9324	0.6633	0.7667	0.5975
F1 Score	0.7016	0.6524	0.6442	0.4652

Figure 1 Performance Metrics of Classification Models on CTR Prediction

Random forest outperforms decision tree (accuracy 0.6422), logistic regression (accuracy 0.7476), and Na ï ve Bayes (accuracy 0.5941) in all metrics. The ROC curve shows that the AUC value of the random forest is the highest, indicating its ability to capture complex nonlinear relationships and effectively handle user interaction characteristics influenced by multiple factors; The AUC value of logistic regression is low but higher than the baseline, requiring additional feature engineering to improve performance; Decision trees perform poorly due to high variance or simplicity; Naive Bayes has weak adaptability in CTR prediction due to its assumption of feature independence.

5 Conclusion

Distributed computing technology, through its parallel processing and memory computing characteristics, has become the core solution for delay control in real-time advertising recommendation. This study uses the Spark framework as the core and combines machine learning algorithms such as random forest to verify its significant effectiveness in reducing latency and improving efficiency in predicting advertising click through rates. In the data preprocessing stage, by removing missing values, normalizing, and selecting features based on the importance of correlation matrix and random forest features, the quality of input data is optimized and redundant information is reduced, laying the foundation for low latency inference. Spark's distributed architecture supports parallel execution of data cleaning, feature assembly, and model training, especially in the distributed random forest model. Through parallel training and voting mechanisms of multiple decision trees, the single sample inference delay is compressed to milliseconds while ensuring prediction accuracy, meeting the timeliness requirements of real-time advertising recommendation. Compared to standalone environments, distributed frameworks avoid latency fluctuations caused by node failures through load balancing and fault-tolerant design, and optimize computing resource utilization through dynamic resource scheduling (such as YARN resource management) to further reduce overall latency. Although current research is based on a single high click through rate dataset (20%) and has limitations in verifying generalization in practical scenarios, it provides a reusable technical path for delay control: in the future, dynamic models can be constructed through real-time data stream processing[8] (such as Spark Streaming), combined with deep learning models (such as LSTM) to capture user behavior temporal characteristics, and distributed algorithms (such as gradient descent parallelization) can be optimized [9] to reduce model update latency, ultimately achieving low latency response for the entire "data collection feature processing model inference" chain[10], promoting the evolution of real-time advertising recommendation systems towards higher accuracy and lower latency.

References

[1] Carrasco Gubernatis D A. Structure of the Digital Advertising Ecosystem[M]. Apress, Berkeley, CA, 2024.

[2] Choi J, Lee J, Lee G, et al. MultiFile View: File-view-based Isolation in a Single-User Environment to Protect User Data Files[J]. IEEE Transactions on Dependable and Secure Computing, 2025:1-16. DOI:10. 1109/tdsc. 2025. 3541728.

[3] Rawlinson N. USE Windows 11's new tools in Windows 10[J]. Computer active, 2023(Dec. 19 TN. 672).

[4] Chen X. Research on the Integration of Voice Control Technology and Natural Language Processing in Smart Home Systems[J]. European Journal of Engineering and Technologies, 2025, 1(2): 41-48.

[5] *Argente-Garrido A, Zuheros C, Luzón, M. Victoria, et al. An Interpretable Client Decision Tree Aggregation process for Federated Learning[J]. 2024.*
[6] *Lyu N. Adaptive Generative AI Interfaces via EEG-based Cognitive State Recognition[J]. Advances in Computer and Communication, 2025, 6(4).*
[7] *Tang X, Wu X, Bao W. Intelligent Prediction-Inventory-Scheduling Closed-Loop Nearshore Supply Chain Decision System[J]. Advances in Management and Intelligent Technologies, 2025, 1(4).*
[8] *Wu X, Bao W. Research on the Design of a Blockchain Logistics Information Platform Based on Reputation Proof Consensus Algorithm[J]. Procedia Computer Science, 2025, 262: 973-981.*
[9] *Zhou, Y. (2025). Improvement of Advertising Data Processing Efficiency Through Anomaly Detection and Recovery Mechanism. Journal of Media, Journalism & Communication Studies, 1(1), 80-86.*
[10] *Chen M. Research on the Application of Privacy-enhancing Technologies in AI-driven Automated Risk Detection Systems[J]. Advances in Computer and Communication, 2025, 6(4).*
[11] *Zhou Y. Cost Control and Stability Improvement in Enterprise Level Infrastructure Optimization[J]. European Journal of Business, Economics & Management, 2025, 1(4): 70-76.*
[12] *Huang, J. (2025). Research on Resource Prediction and Load Balancing Strategies Based on Big Data in Cloud Computing Platform. Artificial Intelligence and Digital Technology, 2(1), 49-55.*
[13] *Gao Y. Research on Risk Identification in Legal Due Diligence and Response Strategies in Cross border Mergers and Acquisitions Transactions [J]. Socio-Economic Statistics Research, 2025, 6(2): 71-78.*
[14] *Chen X. Research on AI-Based Multilingual Natural Language Processing Technology and Intelligent Voice Interaction System[J]. European Journal of AI, Computing & Informatics, 2025, 1(3): 47-53.*
[15] *Li W. Building a Credit Risk Data Management and Analysis System for Financial Markets Based on Blockchain Data Storage and Encryption Technology[C]//2025 3rd International Conference on Data Science and Network Security (ICDSNS). IEEE, 2025: 1-7.*