GEORGE BROWN PRESS
JOURNALS, BOOKS & PROCEEDINGS

Journal of Computer, Signal, and System Research

Article

Research on the Application of Spark Technology in Natural Resource Data Management

Jialu Yan [1,*]

[1] Decoded Advertising, New York, 10005, USA
* Correspondence: Jialu Yan, Decoded Advertising, New York, 10005, USA

Abstract: With the rapid growth of natural resource data and its complex structure, traditional data management technologies are facing numerous challenges, such as storage bottlenecks, difficulties in data integration, and insufficient processing efficiency. In this context, Spark, as a powerful distributed computing system, has shown great application prospects in the field of natural resource data management with its excellent in memory computing capabilities, real-time data processing capabilities, and outstanding scalability. This article explores the framework and significant advantages of Spark technology, and delves into its specific applications in natural resource data storage, real-time processing, modeling and analysis. It also explores how to enhance system performance and ensure information security through optimization strategies, in order to provide technical assistance and operational references for natural resource management practices.

Keywords: Spark technology; natural resource data; distributed computing; data management; real-time processing

1. Introduction

With the rapid advancement of big data technology, effective management of natural resource data has become particularly crucial, playing a core role in improving resource utilization efficiency and assisting decision-making. However, the current data processing methods are unable to cope with the challenges of managing data from numerous sources, multiple perspectives, and high frequencies. Spark, as a novel data processing framework, provides an efficient solution to the challenges in natural resource data management due to its distributed memory processing capabilities and the integration of batch and stream processing. This article will explore the unique attributes of Spark technology and analyze its specific applications and improvement methods in the field of natural resource management.

Received: 11 March 2025
Revised: 19 March 2025
Accepted: 16 April 2025
Published: 20 April 2025

2. Overview of Spark Technology

2.1. Architecture and Core Components of Spark

Spark is a popular open-source big data processing tool with excellent in-memory computing capabilities, capable of both batch processing and real-time streaming processing. The core components of the framework include Driver, Executor, and Cluster Manager. Driver is responsible for executing user code, regulating task scheduling, and resource allocation; Executor runs the computing units on each work node, specifically executing tasks and managing data storage. Cluster Manager is responsible for unified scheduling and management of cluster resources, supporting multiple modes such as standalone, YARN, or Mesos.

The core components of Spark include Elastic Distributed Data Resources (RDD), Spark SQL, Spark Streaming, MLlib, and GraphX. RDD provides powerful capabilities for distributed storage and operation of large-scale data, greatly simplifying the fault-tolerant

and parallel processing of data, and improving efficiency. Spark SQL supports querying and efficient manipulation of structured data. Spark Streaming excels at capturing and processing data streams to ensure real-time response. The MLlib library provides numerous machine learning algorithms. GraphX is used for graph data processing and analysis. These core modules constitute Spark's outstanding data processing capabilities.

2.2. Technical Advantages of Spark

Spark demonstrates outstanding technological features. Its efficient memory processing technology greatly accelerates data processing. Compared with the traditional MapReduce architecture, Spark can directly store intermediate data in memory, reducing the frequency of disk I/O usage and thus improving overall computational efficiency. In addition, Spark can flexibly support the integration of batch processing and streaming processing, achieving real-time data analysis through Spark Streaming, which is particularly suitable for real-time monitoring and analysis of natural resource data. Spark also has excellent scalability and error recovery capabilities, allowing for flexible addition of computing units as needed and automatic recovery in case of task errors, ensuring the robustness and credibility of the system. The rich libraries provided by Spark, such as Spark SQL, MLlib, and GraphX, provide powerful tools for processing structured data, machine learning tasks, and graph computing, bringing diverse and flexible processing strategies for data management of natural resources.

3. Problems in Natural Resource Data Management

3.1. Bottlenecks in Data Storage and Computing Resources

The data involved in natural resources covers various aspects such as spatial dimensions, spatiotemporal geographic data, climate data, etc. Its data scale is huge and constantly expanding, making traditional storage solutions inadequate in terms of efficient storage and convenient access. The complexity of data processing requires extremely high computing power. Due to limitations in computing resources and parallel processing capabilities in traditional systems, data parsing often consumes a considerable amount of time, especially when dealing with large-scale and complex datasets. The specific data storage and computing scale can be found in Table 1.

Table 1. Storage and Calculation Scale of Typical Natural Resource Data.

Data category	Data scale	Processing time requirements
Geographic Information Data	50TB	Need efficient computing resource support
meteorological data	70TB	High real-time requirements
Land use data	100TB	Long term processing and storage requirements

3.2. Difficulties in Data Integration and Sharing

There are various channels for collecting natural resource data, including geographic location information, satellite imaging data, climate data, and other diverse content. These data are produced by numerous organizations and platforms, and their standardization levels vary greatly, causing significant obstacles to data integration and circulation. Due to the lack of unified standards and protocols for heterogeneous data, the integration process becomes complicated, often requiring extensive preprocessing and adjustments. Numerous natural resource data are scattered in various regions, forming a phenomenon of data silos, making effective sharing across departments and regions extremely difficult. The obstacles to data sharing also include network architecture, regulations related to data security and privacy protection, which limit the full utilization and collaborative management of natural resource information.

3.3. Insufficient Data Processing Efficiency and Analytical Ability

In natural resource data management, due to low computational efficiency and the diversity of processing tasks, conventional methods often fail to meet the demands of high-performance computing. When the system needs to deal with tasks that require extremely high timeliness, such as weather forecasting and real-time environmental monitoring, its lower processing speed cannot meet the demand for timely response, which in turn affects the timeliness of adjustments. Due to the difficulty of traditional computing architectures in matching the highly parallel processing requirements of big data analysis and complex numerical calculations, delays in analysis tasks and computational limitations often occur. In the processes of data cleaning, merging, and model construction, significant time and resources are often required, which directly affects the timeliness and accuracy of data analysis.

4. Application of Spark Technology in Natural Resource Data Management

4.1. Data Storage and Distributed Computing Optimization

Spark technology demonstrates excellent performance optimization capabilities in handling natural resource data storage and distributed computing tasks, especially when dealing with massive geographic information data. In the application of Spark for storing and parallelizing remote sensing image data, it effectively solves the inefficiency problem of traditional centralized processing methods in the early data preparation stage, such as image cropping, combination, and various statistical operations. Through Spark technology, the efficiency of these processing steps has been greatly improved.

When processing a large-scale task involving 1TB remote sensing image data, it is necessary to crop the data and screen key features. According to conventional methods, it takes approximately 20 hours for a single computer to complete this task. However, with Spark's efficient distributed computing architecture, data can be divided into 100 independent small blocks and allocated to various Executor nodes for processing, greatly improving computational efficiency. Spark utilizes distributed storage systems (such as HDFS) for data management and uses the following formula to achieve data partitioning and parallel processing:

$$N = \frac{T}{P}$$

Among them, N is the partition data volume of each node, T is the total data volume, and P is the number of nodes in the cluster. Through reasonable partition design and the application of parallel computing technology, the data processing process that originally took 20 hours has been shortened to only 2 hours, greatly improving the overall efficiency of data processing. The data processing system relies on Spark's powerful memory computing capabilities and distributed architecture, which not only greatly improves the speed of data processing, but also significantly reduces resource consumption, making data management of natural resources more efficient.

4.2. Real Time Data Processing and Streaming Analysis

The urgent need for real-time data processing and streaming analysis is significant in natural resource data management. The streaming processing technology of Apache Spark endows the system with powerful means of real-time data processing, ensuring that the natural resource monitoring system can perform real-time analysis and processing at the source of data generation. The data streams output by meteorological information and environmental monitoring instruments can be instantly parsed by Spark Streaming to achieve real-time monitoring and warning objectives.

When performing environmental monitoring tasks, sensors collect nearly a thousand pieces of information per second (such as air quality, temperature and humidity, precipitation status, etc.). Real-time analysis of these data is crucial to determine whether the preset thresholds have been breached. If a conventional batch processing flow is used, it

will cause time delays and not meet the real-time monitoring requirements. With the help of Spark Streaming technology, the data flow from sensors is seamlessly integrated into the Spark ecosystem and processed in small batches per second, ensuring real-time monitoring of environmental conditions. Real time monitoring formula:

$$R = \frac{\sum_{i=1}^{n} Di}{T}$$

Among them, represents the average rate of real-time processing, represents the processing time of the first batch of data, and represents the time interval of the data stream.

In this environmental monitoring system, Spark Streaming technology divides the data stream generated per second into multiple small batches to ensure that its processing speed always exceeds the speed of data generation, thereby achieving zero latency real-time data analysis. By utilizing the advantages of this formula, the application of Spark Streaming in monitoring systems effectively improves the timeliness of data processing. Once any parameter exceeds the predefined threshold, an alarm can be quickly triggered, providing efficient feedback and assistance for the effective management and decision-making of natural resources.

4.3. Data Analysis and Modeling

Spark technology has demonstrated excellent computational efficiency and strong flexibility in processing natural resource data analysis and building models. With the help of Spark's machine learning library, it is possible to efficiently process massive amounts of data, build predictive models, and conduct trend research. In the process of predicting land use changes, by deeply mining historical data and constructing models, it is possible to predict the future direction of land use, thereby providing strong data support for the rational planning of resources.

For the task of predicting the trend of land use change in a certain region in the next decade, detailed geographic information covering population distribution, land use status, and economic dynamics over the past five years was collected. A predictive model was constructed using Spark MLlib, which relies on Spark's powerful distributed processing capabilities and can complete the entire process from model construction to prediction in just a few hours. Compared to traditional methods, the efficiency is significantly improved. During this process, a multiple regression model was used for prediction, and its calculation formula is as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \in$$

Among them, y represents the amount of land use change, $x_1, x_2, \ldots, x_n$ is the variable that affects land use (such as population density, economic activity intensity, etc.), β_0 is the intercept, β_i is the coefficient of the variable, and $\in$ is the error term. Utilizing Spark MLlib for machine learning model construction and parameter adaptation, significantly enhancing computational efficiency and model accuracy through decentralized data processing technology. With the concurrent processing feature of Spark, the model can quickly iterate and refine, accurately predict the trend of land resource utilization, and contribute key data support for ecological environment protection and regional planning decisions.

4.4. Data Visualization and Query Optimization

In natural resource data management, data visualization and fast query functions play a key role in in-depth analysis and decision-making. The Spark SQL platform has excellent data query capabilities. Once combined with data visualization software, it can present complex natural resource information in an intuitive way and achieve fast retrieval. The following example demonstrates how to track and visualize the use of ecological resources in real-time.

A certain region aims to continuously track and graphically display cross year water resource utilization, involving data content such as water consumption, river changes,

and climate conditions. Spark SQL is used to quickly query data stored in distributed file systems and visualize the obtained information, enabling decision-makers to instantly understand real-time changes in resource usage. The seamless integration of visualization software such as Tableau and Power BI with Spark ensures real-time data updates and display.

In order to further improve retrieval speed, data partitioning and index optimization techniques were adopted, which significantly reduced query latency across distributed datasets:

$$T = \frac{Q}{P \times I}$$

Among them, Q is the amount of query data, P is the number of partitions, and I is the index factor. By adding data partitioning and optimizing retrieval, the information query time has been greatly reduced. In terms of execution details, Spark SQL performs partition management on large databases and utilizes indexing technology to quickly lock target information, thereby significantly improving query performance. By combining Spark SQL with graphical interface tools, managers can directly grasp the dynamic use of water resources through various visual methods such as charts and heat maps, helping to implement precise resource allocation and policy formulation. This visual analysis and query optimization significantly enhance data processing efficiency and decision support capabilities.

5. Optimization Strategies of Spark in Natural Resource Data Management

5.1. Performance Optimization

In natural resource data management, the optimization strategies for Spark performance mainly involve key elements such as data partitioning techniques, caching mechanisms, task parallelization, and broadcast variables. Accurately adjusting data partitioning is the core of improving computational parallelization efficiency. By utilizing the repartition and coalesce functions provided by Spark, the number of data partitions can be easily adjusted. Increasing the number of partitions can shorten the processing time per unit and thereby enhance overall execution speed for large-scale datasets that require frequent operations. For datasets with lower access frequency, reducing the number of partitions can lower memory consumption and enable more efficient use of system resources.

The data caching mechanism can effectively reduce duplicate calculations and accelerate processing speed. By using caching or persist techniques, frequently used data is temporarily stored in memory, thereby reducing frequent reads from the hard drive. Parallel processing of tasks is particularly crucial in Spark performance optimization. By adjusting the spark, default, parallelism configuration parameter, the number of parallel tasks assigned can be determined, and the parallelism can be dynamically adjusted based on the amount of data, which can significantly improve the efficiency of data processing. In addition, Configuring the memory and CPU cores of the executor appropriately can promote more efficient parallel execution of tasks and prevent unnecessary resource consumption. By cleverly applying Spark optimization techniques, its performance and execution efficiency in processing natural resource data have been significantly improved.

5.2. Data Security and Access Control

Ensuring data security and access control are core components in natural resource data management. The security features and corresponding configuration methods inherent in the Spark platform can effectively maintain the security of data. Spark's authentication system can be enabled and the use of Kerberos authentication method can be employed to verify user identity, ensuring that only authorized users can access resources in the cluster. Set the spark. Authenticate parameter to true to initiate the authentication process and prevent unauthorized access. In addition, Hadoop's security mechanism can be utilized to set detailed access permissions and ACLs (Access Control Lists) for stored data

in the HDFS file system, in order to accurately divide and restrict the usage permissions of different users.

Spark uses SSL/TLS protocol to encrypt communication between nodes by activating transmission encryption function. Users need to set the spark, sell, enabled parameter and specify the certificate storage path to ensure the confidentiality of data during transmission and prevent critical information from being stolen or tampered with. Meanwhile, Spark SQL provides access control for rows and columns, enabling precise data permission allocation for various user groups. By utilizing external tools such as Apache Ranger, permission policies can be further optimized to achieve user role-based permission configuration and data access auditing, thereby ensuring data security and compliance. This series of multi-level security protection and access control measures provide a solid foundation for the protection of sensitive information in natural resource data management systems.

5.3. System Scalability and Resource Allocation

To enhance the system scalability and resource allocation efficiency of Spark in natural resource data management, comprehensive improvement measures must be implemented. For cluster scalability, by adjusting node size in real-time, it can flexibly adapt to fluctuations in data traffic. Utilize the YARN cluster administrator or Mesos platform integrated with Spark to set up intelligent expansion solutions that automatically increase or decrease computing nodes based on task load, in order to maximize resource utilization efficiency. When facing high load conditions, nodes can be expanded in real-time to enhance processing performance, while nodes can be reduced during low load periods to save costs.

The optimization of resource allocation involves the rational allocation of memory and CPU. By adjusting the parameters of spark. executor. memory and spark. executor. cores, the memory capacity and number of cores of each execution unit can be controlled to ensure balanced use of resources. In the natural resource data processing stage, the setting of execution units is fine-tuned according to the size and complexity of the database, striving to complete each task within the memory range whenever possible, reduce the frequency of data input and output, and enhance the effectiveness of the processing flow. By configuring priority task queues and task scheduling strategies, different jobs can be matched for resources, effectively alleviating resource competition issues. Combining fixed partitioning with flexible resource allocation is beneficial for enhancing system scalability and flexibility when dealing with large-scale natural resource tasks. With the comprehensive application of these strategies, Spark has achieved significant improvement in system scalability and resource allocation efficiency in natural resource data management, thereby more effectively addressing the challenges of variable and complex data processing.

6. Conclusion

Faced with the surge and increasing complexity of natural resource data, traditional data management models seem inadequate. Spark, an excellent distributed computing tool, provides a solid foundation for natural resource data management with its in-memory computing, real-time processing, and scalability. With Spark-based optimization of data storage, real-time processing, data analysis, and visualization, the efficiency of resource data management has been significantly improved, and extraordinary potential has been demonstrated in decision assistance. In addition, the implementation of key strategies such as performance enhancement, data security, access control, and system scalability has enhanced Spark's effectiveness in handling complex data environments.

References

1. Z. Fu, M. He, Y. Yi, and Z. Tang, "Improving data locality of tasks by executor allocation in Spark computing environment," *IEEE Trans. Cloud Comput.*, vol. 12, no. 3, pp. 876–888, Jul.–Sep. 2024, doi: 10.1109/TCC.2024.3406041.
2. Y. Guo, "Application of Big Data Mining System Integrating Spectral Clustering Algorithm and Apache Spark Framework," *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 1, 2025, doi: 10.14569/IJACSA.2025.0160165.
3. H. Qu, L. Zhang, M. Shao, and Z. Yan, "Large-scale hydropower dispatching system based on cloud platform and its key technologies," *Energy Rep.*, vol. 12, pp. 2560–2572, 2024, doi: 10.1016/j.egyr.2024.08.051.
4. D. Fan, W. Jiabin, and L. Sheng, "Optimization of frequent item set mining parallelization algorithm based on Spark platform," *Discover Comput.*, vol. 27, no. 1, pp. 1–19, 2024, doi: 10.1007/s10791-024-09470-5.
5. P. Sewal and H. Singh, "Analyzing distributed Spark MLlib regression algorithms for accuracy, execution efficiency and scalability using best subset selection approach," *Multimed. Tools Appl.*, vol. 83, no. 15, pp. 44047–44066, 2024, doi: 10.1007/s11042-023-17330-5.
6. L. Theodorakopoulos, A. Karras, and G. A. Krimpas, "Optimizing Apache Spark MLlib: Predictive performance of large-scale models for big data analytics," *Algorithms*, vol. 18, no. 2, Art. no. 74, 2025, doi: 10.3390/a18020074.
7. L. Qin, X. Wang, L. Yin, and Z. Jiang, "A distributed evolutionary based instance selection algorithm for big data using Apache Spark," *Appl. Soft Comput.*, vol. 159, Art. no. 111638, 2024, doi: 10.1016/j.asoc.2024.111638.