A Python-based file manager utility along with Hadoop and Spark Integration

Ankan Mazumdar

Illinois Institute of Technology Chicago, United States [amazumdar@hawk.iit.edu](mailto:amazumdar@hawk.iit.edu)

***ABSTRACT:***

The GitHub repository <https://github.com/deepaknairrpf/file-manager> hosts an existing file manager utility designed to manage and manipulate files. However, this utility suffers from several limitations, including limited file format support, inefficient data processing, and a lack of advanced file manipulation features. These limitations hinder users' ability to effectively manage and analyze their data, leading to reduced productivity and increased time spent on data preparation tasks. To address these limitations, we propose to enhance the existing file manager utility with the following key improvements:

Objectives-

* The primary objectives of this project are to:
* Expand file format support to include a wide range of commonly used formats, apart from existing Parquet, JSON such as xlsx, zip
* Integrate Apache Spark, a powerful distributed data processing framework, into the file manager to enable efficient in-memory data processing tasks.
* Introduce advanced file manipulation features, such as data filtering, sorting, and aggregation, to provide users with greater control over their data.

Next Steps

* File Format Support: Implement dedicated file handling modules for each supported format to enhance file format compatibility.
* Apache Spark Integration: Create a SparkContext instance within the file manager and develop Spark-based data processing functionalities to enable efficient in-memory data processing.
* Advanced File Manipulation Features: Implement transformation operations such as File format conversion, copying, moving, removing, compression, etc.
* Testing and Evaluation: Conduct rigorous testing to ensure the proper implementation of the enhancements and evaluate their impact on the file manager's performance and usability.

Documentation and Deployment: Provide comprehensive documentation for the enhanced file manager and deploy the updated version to users.

I. Overview:

* The current file manager utility lacks comprehensive file format support, efficient data processing capabilities, and advanced file manipulation features. These limitations hinder users' ability to effectively manage and analyze their data, leading to reduced productivity and increased time spent on data preparation tasks.

*Solution Outline:*

* We propose to fix the open issues related to file reading and enhance the existing file manager utility by introducing the following key improvements:
* Expanded File Format Support: Expand the file manager's ability to handle a wider range of file formats, including .xlsx, .zip, and more.
* Apache Spark Integration: Integrate Apache Spark, a powerful distributed data processing framework, into the file manager. This integration will enable users to perform efficient in-memory data processing tasks directly within the file manager.
* Advanced File Manipulation Features: Introduce advanced file manipulation features, such as data filtering, sorting, aggregation, format conversion, copying, moving, removing, line counting, searching, and calculating size.

*Relevant Literature*:

* Spark: Cluster Computing with Working Sets Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica University of California, Berkeley
* "Converting CSV to Parquet with Spark" by Zaharia et al. (2014) explored using Apache Spark for efficient CSV to Parquet conversion, highlighting its effectiveness in processing large datasets.
* <https://github.com/apache/parquet-format>
* <https://towardsdatascience.com/parquet-file-format-everything-you-need-to-know-4eed5c0019e7>

II. Design and Architecture

* The enhanced file manager utility will consist of the following software components:
* Enhanced File Manager Utility: The core component of the system, responsible for managing files, interacting with users, and providing a user-friendly interface.
* Apache Spark Integration Layer: A bridge between the enhanced file manager utility and Apache Spark, enabling the system to leverage Spark's distributed data processing capabilities.
* VS Code Integration: A plugin or extension that integrates the enhanced file manager utility into VS Code, providing users with a familiar development environment.
* Python Packages: A collection of Python packages that provide additional functionality to the system, such as data visualization and file compression libraries.
* *Approach and Testing*
* The following approach will be followed for development and testing:
* Code Cloning and Setup: Clone the GitHub repository code into the local development environment.
* Local Testing: Run the code in the Integrated Development Environment (IDE) and conduct thorough testing to identify and resolve any existing issues.
* Issue Resolution: Address the reported reading operation issues by developing new functions or modifying existing code.
* Sample Data Testing: Validate the modified code's functionality using sample Parquet, CSV, and JSON files to ensure proper handling of various file formats.
* Apache Spark Deployment: Deploy the code into the Apache Spark environment.
* Apache Spark Testing: Conduct comprehensive testing in the Apache Spark environment to verify the code's compatibility, performance, and integration with Spark.

III .Code & Data –

*Documentation-*

* I have uploaded the documentation on my GitHub link - Reference Documentation. I have explained the code, the setup, testing, and the challenges. Please look into <https://github.com/ankan-mazumdar/file-manager>.
* "Data Sources: Sample open-source files have been utilized for testing purposes.
* Test Case for File Operations:
* This test case is designed to verify that the file operations in the code are completed successfully. It includes a check to ensure that the expected number of records is loaded during file operations.
* Test Case for File Conversion:
* This test case aims to validate the correct functioning of file conversion functions within the code. It ensures that files are converted as expected without errors.
* Test Hadoop Job Execution**:**
* The purpose of this test case is to confirm the successful execution of Hadoop jobs. It verifies that the Hadoop jobs run without encountering errors.
* Test Cases for AWS Apache Spark**:**
* This set of test cases focuses on checking the successful execution of Spark jobs specifically on the AWS Elastic MapReduce (EMR) platform.
* Issue:
* The reported issue involves Apache Spark jobs hanging. In such cases, it has been mentioned that skipping the input method might be a potential solution. Reference to a Stack Overflow solution is provided at <https://stackoverflow.com/questions/40910869/python-script-hangs-on-input-method-when-running-spark>.
* Explanation for the Issue:
* Apache Spark jobs hanging can be a challenging problem to diagnose. The provided Stack Overflow solution suggests that the issue might be related to the interaction with the input method. The suggested workaround involves skipping the input method to potentially resolve the hanging problem during Spark job execution. It is advisable to review the provided solution and implement it to address the reported issue.

IV. CONCLUSION

* Through the implementation of the proposed enhancements, we have successfully expanded the file manager's file format support, integrated Apache Spark for efficient data processing, and introduced advanced file manipulation features. These enhancements significantly improve the file manager's capabilities and empower users to manage and analyze their data more effectively.
* However, we have encountered an unexpected challenge when attempting to execute the enhanced file manager utility using the spark-submit command. The execution hangs indefinitely, preventing the utility from functioning as intended. Initial investigations suggest that this issue may be related to the spark-submit command's behavior when interacting with Python code that expects user input.

V. Bibliography and References-

* Spark: Cluster Computing with Working Sets Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica University of California, Berkeley
* "Converting CSV to Parquet with Spark" by Zaharia et al. (2014) explored using Apache Spark for efficient CSV to Parquet conversion, highlighting its effectiveness in processing large datasets.
* <https://github.com/apache/parquet-format>
* <https://towardsdatascience.com/parquet-file-format-everything-you-need-to-know-4eed5c0019e7>