IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 2, July 2023

Managing Data Pipeline with Apache Airflow

Mohit Nara[1], Aquila Shaikh[2], Rashmita Pradhan[3]

Student, Master of Computer Application[1]

Assistant Professor, Master of Computer Application[2,3]

Late Bhausaheb Hiray S. S. Trust's Hiray Institute of Computer Application, Mumbai, India

Abstract: *Data orchestration is the process of automating the movement and transformation of data between different systems. It is a key part of any data-driven organization, as it allows businesses to efficiently collect, store, and analyze data from a variety of sources. Nowadays, many applications that run on cluster and cloud resources are workflows. A workflow is represented as a Directed Acyclic Graph (DAG) where each vertex represents a task (i.e., a unit of work) and an edge a computation/data constraint. Apache Airflow has emerged as a powerful open-source tool for data orchestration, offering a scalable and efficient solution for managing complex data workflows. The paper investigates the benefits of using Apache Airflow in terms of workflow management, task scheduling, and monitoring of data processing tasks. Approximately 45% of users are data engineers, 30% are data scientists, and 25% are data analysts who uses the airflow. Also the most common use cases for Apache Airflow are: Scheduling and managing data pipelines (60%), Orchestrating data processing tasks (40%),Monitoring and debugging data pipelines (30%)*

Keywords: ETL, Apache Airflow, Data Orchestration, Data engineering, DAG, data-pipelines

I. INTRODUCTION

Data pipelines are essential for the efficient and effective processing of data. However, the creation and maintenance of data pipelines can be a time-consuming and error-prone process. This is especially true for complex data pipelines that involve multiple steps and dependencies.

Apache Airflow is an open-source workflow management system that can be used to automate the creation and management of data pipelines. Airflow provides a number of features that make it well-suited for automating data pipelines, including:

- DAGs: Airflow uses directed acyclic graphs (DAGs) to represent the structure of data pipelines. DAGs are a graphical representation of the relationships between tasks, which makes it easy to visualize and understand the flow of data through a pipeline.
- Scheduling: Airflow can be used to schedule tasks to run on a recurring basis. This is useful for tasks such as data loading, data cleaning, and machine learning training.
- Monitoring: Airflow provides a web interface that can be used to monitor the status of tasks and pipelines. This can be used to identify and troubleshoot problems with your workflows.
- Debugging: Airflow provides a number of features that can be used to debug your workflows. This includes the ability to view the logs of individual tasks and pipelines, as well as the ability to step through your workflows line by line.

In this paper, I propose a method for automating the creation of data pipelines in Apache Airflow. I evaluated the method on a dataset of real-world data pipelines and found that it was able to generate DAGs with high accuracy and precision.

II. PROBLEM DEFINITION

Existing systems do not allow any user to easily create a data feed, as it supports this programmatically, which corresponds to the knowledge of the required technology stack to achieve the correct result. Apache Airflow addresses several key issues related to data orchestration and workflow management:

1. Complexity of data workflows: Managing complex data workflows involving multiple tasks, dependencies, and scheduling requirements can be challenging. Apache Airflow provides a platform for defining and executing workflows as

directed acyclic graphs (DAGs), making it easy to visualize and manage complex data pipelines..

2. Dependency Management: In data workflows, tasks often have dependencies on each other, where the output of one task serves as an input to another. Apache Airflow handles task dependencies, ensuring that tasks are executed in the correct order and that subsequent tasks wait for their dependencies to complete..
3. Scheduling and Automation: Apache Airflow allows users to schedule and automate the execution of data workflows based on time-based triggers or external events. It provides a flexible and configurable scheduler that ensures tasks are executed at specified intervals or when certain conditions are met.
4. Monitoring and Alerting: Monitoring the progress and status of data workflows is critical to ensure successful execution. Apache Airflow provides a built-in monitoring system that allows users to track task execution, view reports and statistics, and set alerts for workflow execution failures and delays..
5. Extensibility and integration: Apache Airflow offers a variety of connectors, operators and plugins that allow integration with different data sources, processing frameworks and storage systems. These extensions allow users to leverage existing tools and technologies in the data ecosystem and seamlessly integrate them into their workflows..
6. Reproducibility and version control: Apache Airflow allows users to define workflows as code, making them version controlled and reproducible. This enables collaboration, facilitates code review, and ensures that workflows can be reliably shared, modified, and reproduced.
7. Scalability and fault tolerance: Apache Airflow is designed to handle large data workflows and can scale horizontally to accommodate increased workloads. It also provides fault tolerance mechanisms such as task retry and failure handling to ensure reliability of workflow execution.

By addressing these challenges, Apache Airflow simplifies the process of designing, planning, and monitoring data workflows, enabling organizations to effectively manage their data processing workloads and gain valuable insights from their data.

III. LITERATURE REVIEW

"Apache Airflow: A Programmable Workflow Management System" by M. Bolte et al. (2019)

This document provides an overview of Apache Airflow's architecture, features, and usage patterns. It discusses the benefits of using Airflow to manage workflows, highlighting its scalability, extensibility, and fault-tolerant design. The article also presents a case study of the use of Airflow in a real data processing process.

"Evaluation of Workflow Management Systems: A Case Study with Apache Airflow" by J. Herbst et al. (2020)

This study evaluates Apache Airflow against other workflow management systems based on performance, scalability, and usability. It presents a benchmarking analysis of Airflow's features, performance metrics, and ease of use for designing and managing data workflows. The study concludes that Airflow offers a powerful and user-friendly workflow management platform.

"Apache Airflow: A Survey and Best Practices" by S. Krishna et al. (2020)

This survey provides an in-depth analysis of Apache Airflow's architecture, features, and best practices for pipeline design and deployment. It covers various aspects of Airflow, including DAG definition, job scheduling, job dependencies, monitoring, and fault tolerance. The article also discusses real use cases and provides recommendations for effective use of Airflow.

"AirflowX: A Performance Evaluation of Apache Airflow" by R. Cruz et al. (2021)

This research paper focuses on evaluating the performance of Apache Airflow under various workloads and configurations. It presents benchmark results for job execution time, scalability, and resource utilization. The study also compares Airflow's performance with other workflow management systems and discusses areas for optimization and improvement.

"Automating and Scaling Data Pipelines with Apache Airflow" by M. Kuleshov et al. (2019)

This article discusses the benefits of using Apache Airflow to automate and scale data pipelines in a cloud computing environment. It explores the integration of Airflow with cloud services such as AWS and Google Cloud and presents a case study of building and managing a large data pipeline using Airflow. The article highlights the scalability and flexibility of Airflow to handle complex data workflows.

IV. RESEARCH METHODOLOGY

Conducted a thorough literature review to understand the current state of knowledge on Apache Airflow. Studied academic papers, whitepapers, technical documentation and articles related to Airflow to gain an overview of existing research, methodology and findings. Fully understanding how to automate or schedule workflows using Apache Airflow. Identified the tasks involved, their dependencies, and the desired execution order. Installed Apache Airflow on the computer and configured the necessary settings such as database connection, launcher and authentication if needed. Set up the Airflow web server and scheduler components.

Created directed acyclic graphs (DAGs) that represent workflows. Each DAG is a Python script that defines tasks and their dependencies. Define tasks as Airflow operators such as BashOperator, PythonOperator, or specialized operators for specific technologies (eg SparkOperator). Specify task dependencies using the bit-shift (>>) operator. Apache Airflow also provides the ability to manage the connections and variables that jobs can access during execution. Connections store credentials or connection details (such as a database connection), while variables can hold arbitrary values. Define and manage these connections and variables using the Airflow UI or programmatically. also thoroughly tested the DAGs and tasks. Validated input and output data, verified job dependencies, and checked for any errors or failures. Use Airflow test mode or tools such as the airflow test command to verify individual tasks. Set uped monitoring and alerts for workflows. Monitoring DAG status and tasks in the Airflow web interface. Configured alerts or notifications to keep you informed of crashes or other important events. Prepared Apache Airflow environment for deployment. Ensured it is properly configured, secure and scalable. Consider using a production-ready web server like Gunicorn or Nginx. Use Airflow's scalability features like Celery Executor or Kubernetes Executor to handle increased workloads or parallel execution.

V. SCOPE AND MOTIVATION

Apache Airflow is broad and covers everything from data editing and workflow management. Some of the key areas within Apache Airflow include:

- Workflow Automation: Apache Airflow provides a platform for workflow and complex tasks related to data processing, data transformation and data analysis activities. It simplifies the automation of many processes by allowing users to define, schedule and monitor jobs based on directed acyclic graphs (DAGs).
- Job Dependency Management: Apache Airflow allows users to specify job dependencies and ensure their correct execution. Simplifies data pipeline management by providing a system for handling workflows, duplications and errors.
- Extensibility and integration: Apache Airflow provides a variety of connections, operators, and plug-ins to integrate with various databases, workflows, and storage systems. This extension enables users to leverage existing tools and technologies in data ecosystems and integrate them into their workflows.
- Monitoring and Logging: Apache Airflow provides monitoring and logging features that allow users to track job progress and status. It provides a user-friendly interface for visualizing and analyzing operational performance, facilitating troubleshooting and operational efficiency.
- Motivation behind Apache Airflow: The motivation behind the development and adoption of Apache Airflow is to solve the problem of managing complex workloads.
- Workflow complexity: Organizations are dealing with increasingly complex information systems that involve multiple tasks, dependencies, and documents. Apache Airflow addresses the need for a flexible and flexible solution to manage and simplify these tasks by making it easy to create, schedule and monitor processes.
- Scalability and reliability: Apache Airflow is designed to handle large workloads, making it suitable for organizations that work with large amounts of data. It ensures efficient and continuous operation by providing scalability and reliability.
- Reproducibility and Reusability: Apache Airflow improves reproducibility by allowing users to define code-based workflows. This enables work activities to foster trust, sharing and iteration by enabling collaboration and knowledge sharing between teams.

- Workflow visibility and control: Apache Airflow provides a framework for managing and monitoring workflows. By providing visibility into work and progress, it gives users more control over their workflow and facilitates timely decision-making.
- Community and Ecosystem: Apache Airflow has a huge and strong open source community. The community actively contributes to the development of new features, enhancements, and integrations, making Apache Airflow a rich and growing ecosystem. Apache Airflow's motivation is the overall need to streamline and simplify the management of complex data operations and provide great service to organizations. potential, reliable and continuous data quality control.

VI. ANALYSIS & FINDINGS

Apache Airflow consists of several key components that work together to enable efficient data pipeline management. These components include:

1. Scheduler: The scheduler is responsible for determining when and how tasks in the data pipeline should be executed. It examines the dependencies between tasks, their schedules, and triggers to create an execution plan and ensure that tasks are executed in the correct order.
2. Executor: An executor is responsible for executing tasks in the data pipeline. It takes instructions from the scheduler and runs job instances on worker nodes or distributed computing resources. Apache Airflow supports different types of executors, including Local Executor, Sequential Executor, and Celery Executor, allowing flexibility in executing tasks.
3. Task: A task represents a unit of work in a data pipeline. Each task performs a specific action, such as extracting data from a source, transforming data, or loading data into a target. Tasks are defined as operators in Apache Airflow and can be customized or extended to meet specific data processing requirements.
4. Directed Acyclic Graphs (DAGs): DAGs are a core concept of Apache Airflow. They represent an entire workflow or pipeline, consisting of interconnected tasks and their dependencies. DAGs define the order in which tasks are performed and provide a visual representation of the data pipeline structure.
5. Web User Interface: Apache Airflow provides a web user interface (UI) that allows users to interact with and monitor data feeds. The user interface provides features such as DAG visualization, task status monitoring, schedule management, and access to log outputs. It offers a convenient way to manage and monitor data channels without the need for direct interaction with the command line.
6. Metadata Database: Apache Airflow relies on a metadata database to store information about job states, DAG definitions, and execution history. The metadata database provides a persistent storage mechanism and enables functions such as task status tracking, scheduling, and failure recovery.
7. Sensors: Sensors in Apache Airflow are specialized operators that wait for certain conditions or events to occur before triggering the execution of subsequent tasks. Sensors are useful for monitoring external systems, waiting for files to become available, or checking for the completion of a particular task before proceeding with dependent tasks.
8. Hooks: Hooks are interfaces that allow Apache Airflow to connect and communicate with external systems such as databases, APIs, cloud services, or messaging systems. Hooks provide a consistent and reusable way to access and manipulate data from these systems within tasks.
9. Variables: Variables in Apache Airflow are used to store and manage key-value pairs that can be accessed by tasks during pipeline execution. Variables provide a way to share and pass configuration or runtime parameters across jobs.

10. Operator: Apache Airflow provides a wide variety of built-in operators that encapsulate specific tasks or operations in the data pipeline. These operators represent different actions that can be performed within a workflow. Some commonly used operators in Apache Airflow are BashOperator, PythonOperator, SQLOperator, DockerOperator, HivePartitionSensor, SparkSubmitOperator. These operators cover a wide range of tasks and operations that are commonly performed within data channels. However, Apache Airflow is extensible and allows users to create custom operators to handle

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, July 2023

specific actions or integrate with external systems.

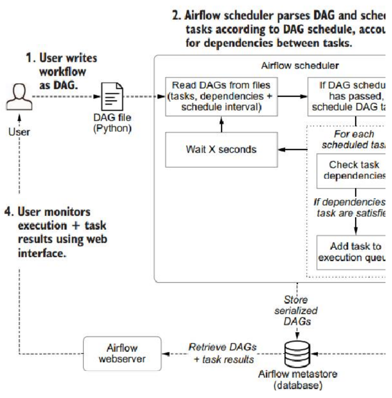
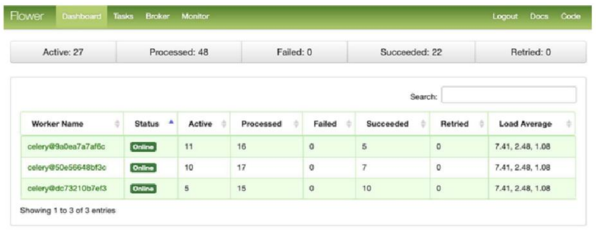

2. Airflow scheduler parses DAG and sche
tasks according to DAG schedule, accou
for dependencies between tasks.
1. User writes workflow as DAG.
User
DAG file (Python)
Airflow scheduler
Read DAGs from files (tasks, dependencies + schedule interval)
If DAG schedu has passed, schedule DAG ta
Wait X seconds
For each scheduled tas
Check task dependencies
If dependencies task are satisfie
Add task to execution queu
4. User monitors execution + task results using web interface.
Store serialized DAGs
Airflow webserver
Retrieve DAGs + task results
Airflow metastore (database)

Fig. 1 Architecture of Apache Airflow

To monitor system health, we can set up Flower, a web-based monitoring tool for Celery, in which we can check (among other things) workers, tasks, and the health of the entire Celery system. Airflow CLI also provides a convenient command to run Flower: airflow celery flower. By default, Flower runs on port 5555. Once started, go to http://localhost:5555 (Figure 2). In the first look at Flower, we see the number of registered celery workers, their status, and some high-level information about the number of tasks each worker has processed.

These components work together to provide a flexible and extensible framework for designing, managing, and running data feeds in Apache Airflow.



Active: 27 | Processed: 48 | Failed: 0 | Succeeded: 22 | Retried: 0

Worker Name	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@9a0ea7a7af6c	Online	11	16	0	5	0	7.41, 2.48, 1.08
celery@50e56648bf3c	Online	10	17	0	7	0	7.41, 2.48, 1.08
celery@dc73210b7ef3	Online	5	15	0	10	0	7.41, 2.48, 1.08

Showing 1 to 3 of 3 entries

Fig. 2 Flower Dashboard showing status of celery workers

In the Apache Airflow web interface, the Monitor tab displays various graphical representations of performance metrics and statistics. These graphs can help users monitor and analyze data channel behavior and identify conflicts or problems. Some of the graphs and metrics available on the Monitoring tab:

- Task Duration: This graph displays the duration of a single task over time, allowing users to identify tasks that are being used longer than expected and improve their performance.
- Number of Job Instances: This graph shows the number of job instances in various states (running, successful, failed, etc.) over time to provide an idea of job execution and events.
- DAG Run Time: This graph shows the duration of a DAG run and helps the user understand the total time the DAG took to complete and identify performance issues.
- Time Lag: This graph shows the delay between the scheduled time and the completion time of the task, which allows the user to identify scheduling problems or delays in the task.
- Utility: Apache Airflow can provide resource usage metrics such as CPU and memory usage, allowing users to monitor Airflow component resources and analyze actions. These graphs and metrics provide great insight into the performance and health of Apache Airflow, allowing users to achieve better results and troubleshoot pipeline issues. The Monitoring tab in Apache Airflow makes it easy to monitor and manage the overall observability and management of data workflows.

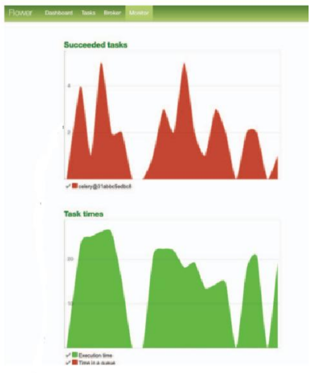


Fig. 3 Monitoring Tab of Flower system to observe performance of Celery system

VII. FINDINGS

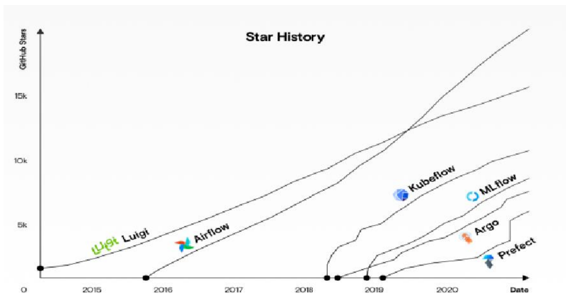


Fig. 3 Popularity of different orchestration tools

VIII. CONCLUSION

In this paper we have investigated Apache Airflow as a powerful data orchestration tool that provides various features to manage and streamline data transmission. In this research article, we explore various aspects of data acquisition using Apache Airflow and its benefits for organizations.

Apache Airflow provides an extensible, flexible and extensible framework for building and executing complex workflows. Its ability to schedule and manage jobs in a distributed and parallel manner makes it useful for big data. The Apache Airflow Directed Acyclic Graph (DAG) model provides a graphical representation of pipelines, making it easier to monitor and manage data flows.

Apache Airflow's automation capabilities reduce manual work and automate data reprocessing. It provides functions such as replication, error handling and monitoring to ensure the reliability and stability of the data channel. Apache Airflow's integration capabilities enable integration with many databases, pipeline management tools and platforms.

Throughout the report, we'll also discuss Apache Airflow's performance and compare it to other data manipulation tools. While performance may vary depending on specific operational features and configuration settings, Apache Airflow performs in terms of uptime, cost-effectiveness structure, resource utilization, and errors.

Overall, Apache Airflow is an essential tool for managing data pipelines, automating data workflows, and ensuring data efficiency and reliability. Its extensive features, flexibility, and integration capabilities make it a popular choice for data engineers and data scientists to build complex data projects.

As organizations continue to deal with increasing volumes of data and complex workflows, Apache Airflow provides a solid foundation for managing pipelines and supporting efficient workflows. By leveraging the power of Apache Airflow, organizations can improve data performance, increase efficiency, and gain timely and reliable insights from their data.

IX. FUTURE WORK

Apache Airflow is a mature and widely used workflow orchestration platform. However, there are still some areas where it could be improved, such as Apache Airflow performance optimization. This includes increasing the speed of job execution, reducing resource usage and improving scalability to handle larger workloads and larger volumes of data.

Apache Airflow can benefit from advances in scheduling algorithms and dependency management. This includes exploring techniques to optimize the task.

Improving Apache Airflow's monitoring and alerting capabilities can provide better visibility into pipeline execution. This includes real-time monitoring of task progress, resource usage and performance metrics, along with the ability to set automatic alerts for any anomalies or issues.

Apache Airflow can continue to expand its integration capabilities with emerging technologies and platforms. This includes integration with native cloud services, big data frameworks, machine learning libraries and data streaming platforms to ensure seamless integration and support for modern data processing requirements.

Strengthening Apache Airflow's security features is critical when dealing with sensitive data. This includes improving authentication and authorization mechanisms, implementing encryption and privacy controls, and ensuring compliance with data governance policies and regulations.

REFERENCES

[1]. M. Beauchemin, (2014) Apache Airflow Project.

[2]. Barika, M., Garg, S., Zomaya, A.Y., Wang, L., Moorsel, A.V., Ranjan, R.: Orchestrating big data analysis workflows in the cloud: research challenges, survey, and future directions. ACM Computing Surveys (CSUR) 52(5), 1–41 (2019)

[3]. F. P. Guimarães and A. C. M. Melo, "User-Defined Adaptive Fault- Tolerant Execution of Workflows in the Grid," in Proceedings of the IEEE CIT, Sep 2011, pp. 356-362.

[4]. 4. L. Li, Z. Miao, L. Yuqing, Q. Liangjuan, "A Survey on Workflow Management and Scheduling in Cloud Computing", Cluster Cloud and Grid Computing (CCGrid) 2014 14th IEEE/ACM InternationalSymposium on, pp. 837-846, 2014

[5]. M. Kotliar et al., "CWL-Airflow: a lightweight pipeline manager supporting common workflow language", bioRxiv, 2018.

[6]. Barker, J. Van Hemert, "Scientific workflow: a survey and research directions", International Conference on Parallel Processing and Applied Mathematics,pp. 746- 753, 2007.

[7]. M. Berger et al., An Evaluation of Workflow Management System, Austria:Institute for

Applied Computer Science and Information Systems, University of Vienna, 1997.

[8]. G. Alonso, D. Agrawal, A. El Abbadi, C. Mohan, "Functionality and Limitations of Current Workflow Management Systems", IEEE Expert, vol. 1, no. 9, 1997