ОБЗОР АРХИТЕКТУРЫ APACHE AIRFLOW

А.В. Свирновский
Научный руководитель – Алексеев В.Ф.
канд. техн. наук, доцент
Белорусский государственный университет информатики и радиоэлектроники

Продукт Apache Airflow как платформа доминирует в разработке потоков данных. В его возможности входит планирование и перезапуск как частичных, так и целых потоков данных.

Направленный ациклический граф [1].

Большинство менеджеров рабочих процессов требуют, чтобы графы были ацикличными, что означает, что они не могут содержать циклы. Ациклические рабочие процессы гарантируют начало и конец, запуск которых можно запланировать на определенное время. Из-за этого ациклического свойства рабочие процессы моделируются как направленные ациклические графы (НАГ). Airflow позволяет создавать и планировать

потоки данных путем создания НАГ. НАГ состоит из операторов и зависимостей между ними. Операторы могут выполнять любую задачу с использованием любой технологии, необходимой для Airflow.

Пакетная обработка [2].

Airflow использует пакетную обработку данных. Пакет данных – серия конечных задач с четко определенным началом и концом, запускаемых через определенные интервалы или посредством триггеров. Фреймворк Apache Spark, часто используется как одна задача в рабочем процессе Airflow.

Весь НАГ определяется скриптом Python. Задачи в НАГ могут быть сгенерированы из различных источников, таких как список имен файлов, в результате чего создаются небольшие динамические сценарии. Платформа представляет собой фреймворк Python с открытым исходным кодом, который позволяет легко реализовывать собственные операторы и хуки. Такая гибкость позволила Airflow набрать популярность.

Airflow отличается от других систем рабочих процессов своей «полнотой функций» – он содержит длинный и постоянно растущий список операторов и хуков, может работать распределенно, имеет простой в использовании пользовательский интерфейс для управления рабочими процессами и позволяет создавать динамические рабочие процессы скриптами Python.

На рисунке 1 представлена общая архитектура Airflow. Веб–сервер обслуживает интерфейс, планировщик обеспечивает состояние всех рабочих процессов и задач, рабочие выполняют фактическую работу.

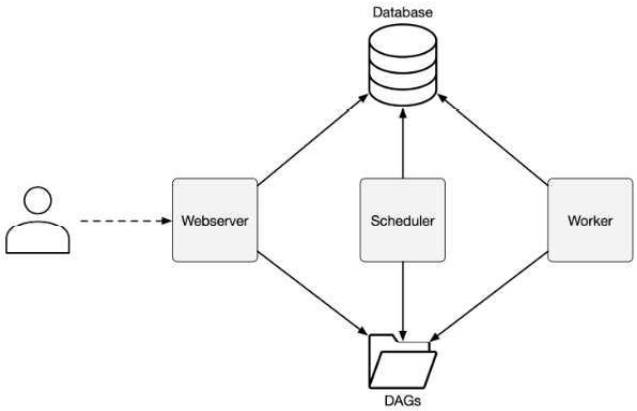

Рисунок 1 – Общая архитектура Airflow

Поскольку рабочие процессы Airflow определены в коде Python, это позволяет создавать гибкие и динамические рабочие процессы. В результате имеется возможность применить те же принципы программирования, что и для любого другого кода. Также имеется возможность хранить конфигурацию в том же репозитории, стиле и соглашениях, что и логика приложения. Отслеживание изменений с помощью кода с контролем версий и откат в случае возникновения проблем делают код повторяемым в любой момент времени.

Планирование и обратное заполнение [1].

Рабочие процессы Airflow можно запускать различными способами: вручную, с помощью внешних триггеров или по расписанию. Во всех случаях при каждом запуске рабочего процесса выполняется ряд задач. В какой–то момент времени может появиться необходимость в изменении логики,

например, если необходимо, чтобы ежедневный рабочий процесс по–разному вычислял выражения трех входных таблиц.

В данном случае можно изменить рабочий процесс для вычисления новых значений и с этого момента запускать новый рабочий процесс. Было бы целесообразно также запустить новую логику для всех ранее завершенных рабочих процессов, для всех исторических данных. Это возможно в Airflow с механизмом, называемым обратным заполнением, запускающим рабочие процессы в прошлом. Помимо обратного заполнения, Airflow предоставляет несколько конструкций для управления жизненным циклом и выполнением задач и рабочих процессов.

Библиографический список

1. Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee Learning Spark, 2nd Edition – O'Reilly Media, Inc, 2020. – 300 p.

2. Towards data science [Электронный ресурс]: база данных. – Режим доступа: https://towardsdatascience.com/a-complete-introduction-to-apache-airflow-b7e238a33df. – Дата доступа: 10.10.2020.