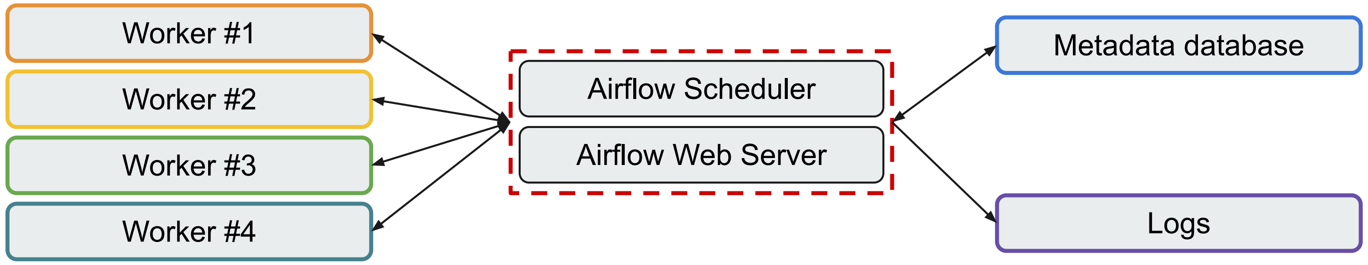
Sequential Executor With SQLite

The default configuration

A Quick Reminder

Apache Airflow can work in a distributed environment. Tasks are scheduled according to their dependencies defined into a DAG and the Workers pick up and run jobs with their load balanced for performance optimization. All task information is stored into the Metadatabase which is regularly updated. The following schema represents Apache Airflow in a distributed environment (where every border color represents a machine) :



What is SQLite

- SQLite is a relational database.
- SQLite is not a client-server database engine, it is embedded into the end program.
- SQLite is ACID-compliant (Atomicity, Consistency, Isolation, Durability).
- It implements most of the SQL standard.
- It requires almost no configuration to run.
- It supports an unlimited number of simultaneous readers, **but only one writer at any instant in time**.
- Limited in size up to 140 TB and the **entire database is stored into a single disk file**.

A Quick Reminder

An Executor is fundamentally a **message queue process** which determines the Worker processes that execute each scheduled task.

A Worker is a process where the task is executed.

What is a Sequential Executor

- It is the most basic executor to use.
- This executor only run one task at a time (Sequential), useful for debugging.
- It is the only executor that can be used with SQLite since SQLite doesn’t support multiple writers.
- It is the default executor you get when you run Apache Airflow for the first time.

The Configuration File

- `executor=SequentialExecutor`
    - The executor class that Airflow should use (Either SequentialExecutor, LocalExecutor, CeleryExecutor, DaskExecutor).
- `sql_alchemy_conn=sqlite:////home/airflow/airflow/airflow.db`
    - The SqlAlchemy connection string used to connect airflow to the metadatabase. SqlAlchemy supports many different database engine.
- `sql_alchemy_pool_enabled=True`
    - If SqlAlchemy should pool database connections. A connection pool is a standard technique used to maintain long running connections in memory for efficient re-use as well as to provide management for the total number of connections an application might use simultaneously.

Concurrency vs Parallelism

A very important concept to understand before going forward is the difference between concurrency and parallelism in programming. Here is a very nice explanation:

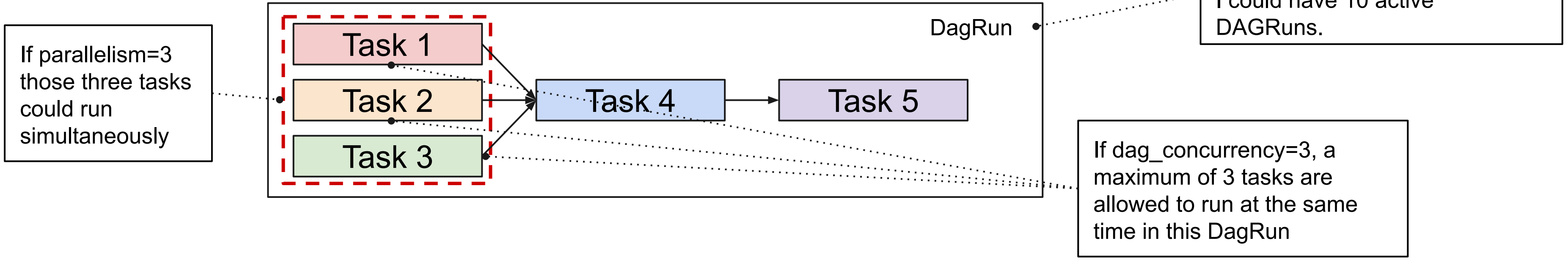
A system is said to be concurrent if it can support two or more actions **in progress** at the same time. A system is said to be parallel if it can support two or more actions **executing simultaneously.** The key concept and difference between these definitions is the phrase “in progress”.

In concurrent systems, **multiple actions can be in progress (may not be executed) at the same time**, meanwhile, **multiple actions are simultaneously executed in parallel systems**.

takuti.me

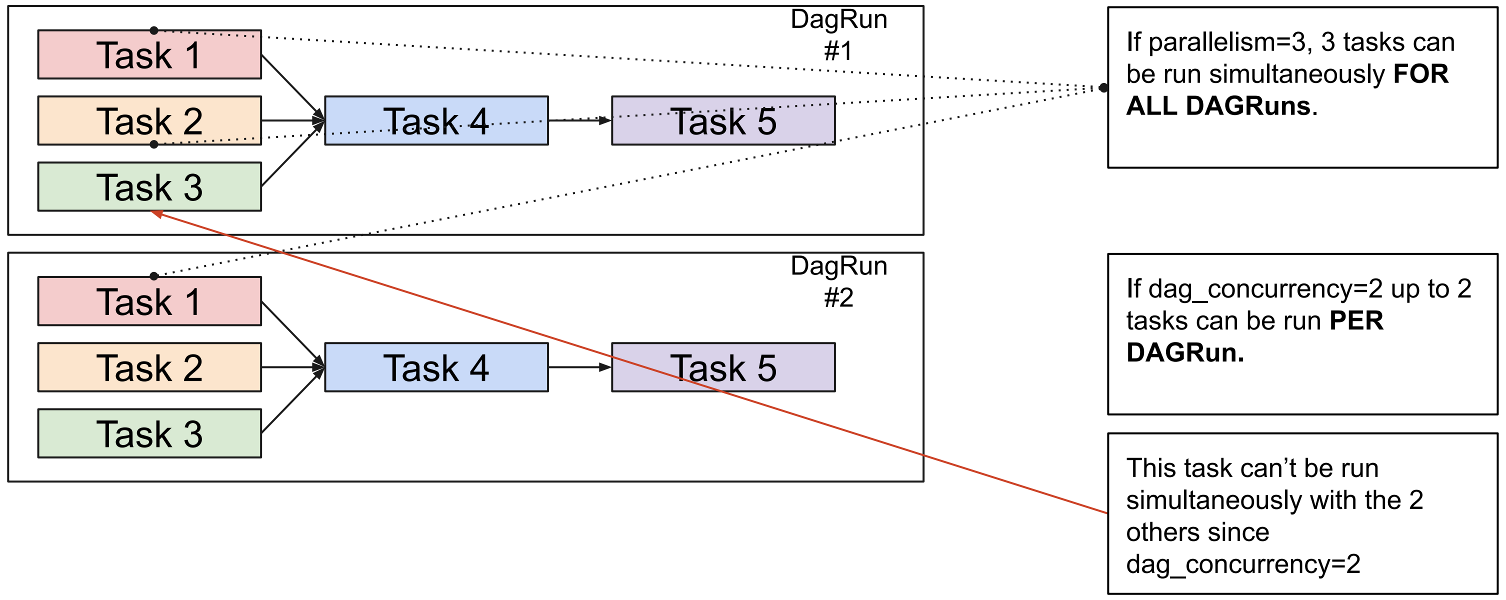
The Configuration File

- `parallelism=32`
  - The number of physical python processes (worker) the scheduler can run.
- `max_active_runs_per_dag=16`
  - The number of DAGRuns (per-DAG) to allow running at once.
- `dag_concurrency=16`
  - The number of task instances allowed to run PER DAGRun at once.



If parallelism=3 those three tasks could run simultaneously
Task 1
Task 2
Task 3
Task 4
Task 5
DagRun
If max_active_runs_per_dag=10 I could have 10 active DAGRuns.
If dag_concurrency=3, a maximum of 3 tasks are allowed to run at the same time in this DagRun

dag_concurrency vs parallelism Parameters



If parallelism=3, 3 tasks can be run simultaneously **FOR ALL DAGRuns.**

If dag_concurrency=2 up to 2 tasks can be run **PER DAGRun.**

This task can’t be run simultaneously with the 2 others since dag_concurrency=2

SQLite And Sequential Executor

- It’s basically the default configuration you get when you install Apache Airflow.
- Really suitable for debugging and testing.
- Do not use this configuration in Production since doesn’t scale.
- There is no parallelism and/or concurrency.

We actually already used SQLite Sequential Executor multiple times during the different DAGs we executed throughout the course. Let’s see a more exciting configuration..