Airflow Summit 2024

ASTRONOMER

A deep dive into Airflow Configuration options for scalability

Ephraim Anierobi

Senior Software Engineer at Astronomer
Committer & PMC member at Apache Airflow
Airflow Core Release Manager

ASTRONOMER

Airflow Configuration

Customizes Airflow's behavior to meet specific operational needs.

Where to set the configuration:

- ❏ airflow.cfg: The main configuration file for Airflow.
- ❏ Environment Variables: Override configuration settings

Configuration areas:

- ❏ core, scheduler, database, executors(celery, kubernetes), logging etc

Why Configuration matters

- Optimizes Resource Utilization
  - Proper configuration ensures efficient use of CPU, memory, and other resources.
- Enhances Performance
  - Fine-tuning settings like parallelism, max_tis_per_query, and executor type can significantly improve task execution speed.
- Supports Scalability
  - As your data pipelines become more complex, properly configuring Airflow ensures that it can handle the increased workload smoothly and efficiently.

How to set Airflow configurations

```
[core]
# The folder where your airflow pipelines live, most likely a
# subfolder in a code repository. This path must be absolute.
#
# Variable: AIRFLOW__CORE__DAGS_FOLDER
#
dags_folder = /Users/ephraimbuddy/airflow/dags
```

airflow.cfg

Environment Variables

AIRFLOW__CORE__DAGS_FOLDER=path/to/your/dags/folder

To make troubleshooting easier, it's best to clear the default values in the airflow.cfg file. Keep only the settings that are different from the defaults in the file. This way, it's easier to see what has been customized and identify any potential issues.

Important performance-related configurations

[core] parallelism

[core] max_tis_per_query

[core] max_active_runs_per_dag

[core] max_active_task_per_dag

[core] default_pool_task_slot_count

[scheduler] use_row_level_locking

[scheduler] max_dagruns_per_loop

[scheduler]max_dagruns_per_loop_to_schedule

[scheduler]schedule_after_task_execution

[scheduler] file_parsing_sort_mode

[scheduler] min_file_process_interval

[scheduler] dag_dir_list_interval

[scheduler] pool_metrics_interval

[core] parallelism

Maximum number of task instances that can run concurrently per scheduler, regardless of the worker count

About [core] parallelism

Determines the total running/queued tasks in an Airflow deployment.

Scheduler count affects the overall concurrency.

> If parallelism is 32 and you have 2 schedulers, the total task instances that can run in the deployment is 64

Unlimited Parallelism

[core] parallelism = 0

```
if parallelism == 0:
    parallelism = sys.maxsize
```

Desirable when using kubernetes executor and you don’t know the optimal parallelism for the cluster.

[scheduler] max_tis_per_query

Maximum number of task instances to examine ‘for scheduling/queuing’ in each scheduler loop

Usages of [scheduler] max_tis_per_query in code

```
        )
    dag_run.schedule_tis(schedulable_tis, session, max_tis_per_query=self.job.max_tis_per_query)
```

```
if not (self.task_instance.test_mode or is_deferral):
    if conf.getboolean( section: "scheduler",  key: "schedule_after_task_execution", fallback=True):
        self.task_instance.schedule_downstream_tasks(max_tis_per_query=self.job.max_tis_per_query)
```

```
    if self.job.max_tis_per_query == 0:
        max_tis = parallelism - num_occupied_slots
    else:
        max_tis = min(self.job.max_tis_per_query, parallelism - num_occupied_slots)
```

[core] parallelism & [core] max_tis_per_query

- For 1 scheduler with parallelism = 32
  - The default(16) is not bad for max_tis_per_query

- If parallelism is 200:
  - 1 scheduler => 150(max_tis_per_query)
  - 2 scheduler => 250…300(max_tis_per_query)

More on [core] max_tis_per_query

Too high a value can result in the scheduler not heartbeating on time.

SQL Query predicate could become complex with excessive locking which would impact your database

[core] default_pool_task_slot_count

Limits the execution parallelism of your tasks regardless of your parallelism setting or scheduler count.

The default pool slot count can only be increased through the UI, CLI or REST API in an existing deployment

You can create your own pools

[scheduler] schedule_after_task_execution

```
def _schedule_downstream_tasks(
        for schedulable_ti in schedulable_tis:
            if getattr(schedulable_ti, "task", None) is None:
                schedulable_ti.task = task.dag.get_task(schedulable_ti.task_id)

        num = dag_run.schedule_tis(schedulable_tis, session=session, max_tis_per_query=max_tis_per_query)
        cls.logger().info( msg: "%d downstream tasks scheduled from follow-on schedule check",  *args: num)
```

(mini scheduler)

Schedules downstream tasks of a task after the task's execution.

Makes same DAG to execute fast in the expense of other DAGs.

A bit about the scheduler loop

Does a SELECT .. FOR UPDATE query on DagModels, DagRuns, and TaskInstances before performing actions on them

- Locks DagModel to create dagruns
- Locks DagRun to move it to running, schedule its taskinstances or check the succes/failure
- Locks TaskInstances before sending them to executors

Runs timed events that are checked at regular intervals.

Scheduler HA (Running Multiple Schedulers)

[scheduler] use_row_level_locking: Should the scheduler lock queries for updates. Ensure that this is set to True(default) for HA.

Set [scheduler] max_dagruns_to_create_per_loop lower to distribute work across the schedulers

Also set [scheduler] max_dagruns_per_loop_to_schedule low for the same reason above

Timed events in the scheduler

- orphaned_tasks_check_interval: No harm in not detecting this for a while.
- trigger_timeout_check_interval: If you are not using triggers, set this to a high value.
- pool_metrics_interval: Set this to match your StatsD roll-up period.
- parsing_cleanup_interval: If you baked your DAG into the image or not using datasets, you can set this real high.

DAG Parsing

[scheduler] dag_dir_list_interval (5mins): If you are not adding new DAG files often, this should be set high and if DAG is baked in image, set this very high

[scheduler] min_file_process_interval: Setting this low increases CPU usage.

[scheduler] file_parsing_sort_mode : For HA mode and not using standalone dag processor, random_seeded_by_host is preferred. Other modes: modified_time, alphabetical

[scheduler] parsing_processes : You can have more than two parsing processes

Thank you!
Any questions?