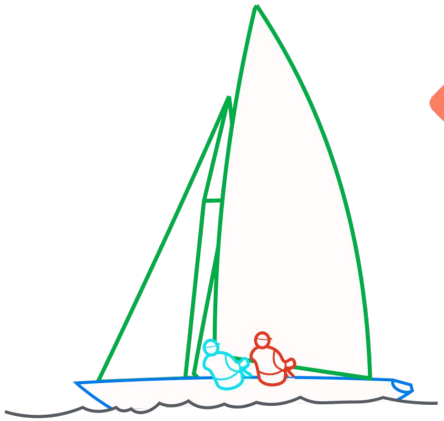
DAG Parsing Optimization

Raphaël Vandon



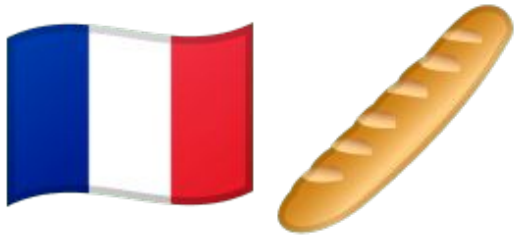
Airflow Summit

Let's flow together

September 19-21, 2023,
Toronto, Canada

About Me

Raphaël Vandon
he/him

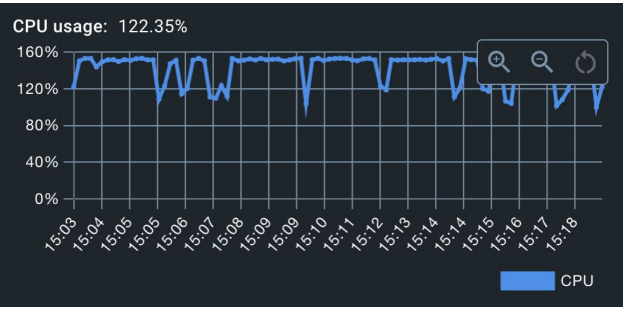


Genesis: “Scheduler Performance Issue”

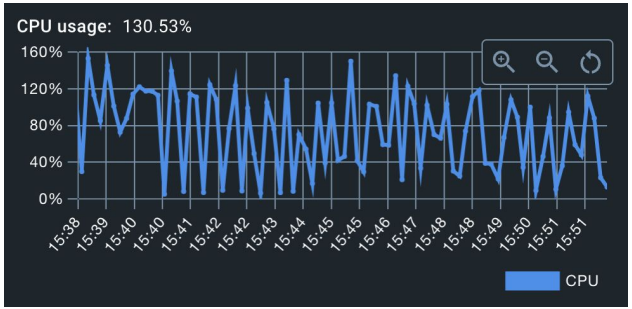
- 300 DAGs generated from one python file is OK
- 300 DAGs from 300 python files overwhelms the Scheduler
- WHY ?

Experimenting to understand the problem

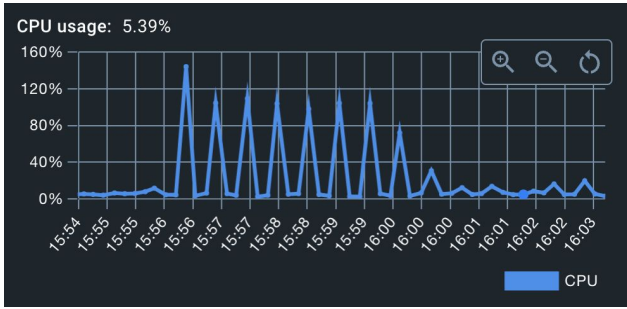
- Hypothesis: Scheduler is slow because it has many files to parse
- Experiment: Vary the number of dags



200



100



25

- Understand the results:
  - Parsing time: ~300ms/file (`logging.dag_processor_log_target=stdout`)
  - Done every 30s in 2 processes means >200 DAGs is too much

We already have solutions!

- Increase the parsing interval

  `scheduler.min_file_process_interval=120`
- More processes

  `scheduler.parsing_processes=4`
- Run the DAG processor separately

  `scheduler.standalone_dag_processor=True`

  `airflow dag-processor`

```python
from airflow.decorators import dag, task
from airflow.providers.postgres.operators.postgres import PostgresOperator
from airflow.sensors.s3_key_sensor import S3KeySensor
from airflow.models import Variable
from airflow.utils.dates import days_ago

POSTGRES_CONN_ID=Variable.get("POSTGRES_CONN_ID", default_var="postgres_redshift")
S3_BUCKET = Variable.get("S3_BUCKET", default_var="my-dag-bucket")
S3_KEY = Variable.get("S3_KEY", default_var="dags/")
SCHEDULE_INTERVAL = Variable.get("SCHEDULE_INTERVAL", default_var="0 * * * *")
DAG_RETRIES = int(Variable.get("DAG_RETRIES", default_var="3"))

default_args = {
    'start_date': days_ago(1),
    'retries': DAG_RETRIES,
}

@dag(
    dag_id="demo",
    schedule_interval=SCHEDULE_INTERVAL,
    default_args=default_args,
    catchup=False,
)
def update_table_dag():
    s = S3KeySensor(task_id="check_s3", bucket_key=S3_KEY, wildcard_match=True, bucket_name=S3_BUCKET)
    t = PostgresOperator(task_id="query_t", sql="select * from my-table;", postgres_conn_id=POSTGRES_CONN_ID)
    s >> t

update_table_dag()
```

What is happening ?

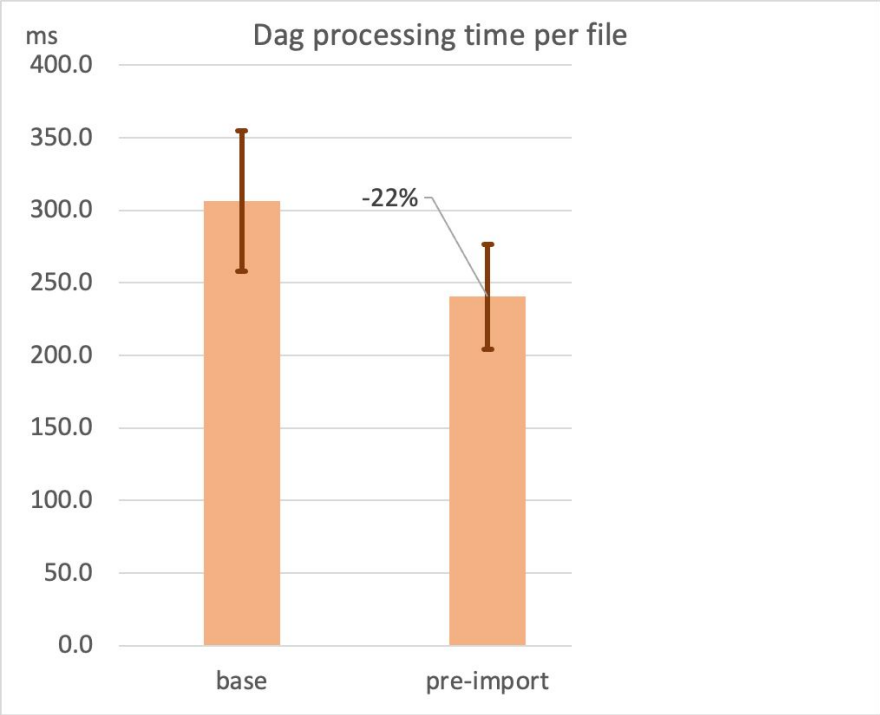
- 1 process per DAG parsed
- Working on a copy of the memory of the main process
- Imports are discarded when we’re done with the dag.

- Software optimisation is very often identifying what is repeated, and then finding a way to do it only once.
- We found what was repeated
- How to do it only once ?

Solutions

- Communication between processes ?
- Stop using 1 process per DAG ?
- Do the imports in the main thread before forking ?

Results



	avg ms on main	avg ms with preload	change %
example_sftp_to_wasb	632	192	-70%
example_azure_blob_to_gcs	696	216	-69%
example_local_to_adls	536	186	-65%
example_adls_delete	510	190	-63%
example_local_to_wasb	509	197	-61%
example_adf_run_pipeline	577	226	-61%
example_azure_service_bus	555	283	-49%
example_postgres	360	184	-49%
example_azure_cosmosdb	355	197	-45%
example_docker	337	187	-44%
example_taskflow_api_docker_virtualenv	773	441	-43%
example_docker_copy_data	350	200	-43%
example_azure_container_instances	316	185	-41%
example_azure_synapse	321	189	-41%
example_docker_swarm	351	207	-41%
example_fileshare	327	193	-41%
example_zendesk_custom_get	315	193	-39%
example_ftp	311	198	-37%
example_sql_column_table_check	259	173	-33%
example_sql_execute_query	270	187	-31%
example_github	278	199	-29%
example_s3	1291	942	-27%
example_sagemaker	1339	997	-26%
example_http	263	196	-25%

About those usages of Variable...

- Not too slow when using a local DB
- ...but as soon as there are network calls, it's bad

- Default settings: 300ms/DAG
- With AWS Secret Manager: 800ms/DAG

Solution: don’t do it ?

- Don’t use Variables in top level DAG code
- It’s bad practice
- The documentation says you shouldn’t do it

- ...
- Users do it anyway

Adding a cache always solves everything

(no)

(but yes)

There is now an optional cache on Variables

- Uses Python's `multiprocessing.Manager`
- `secrets.use_cache=true`
- Caveats!
- I can make arbitrarily good looking benchmarks

And Connections

Only during parsing

It's experimental

Conclusion

- You don’t have to be an expert to have an impact
- Just time and motivation to investigate

Questions?

Get in contact:
vandonr@gmail.com
linkedin.com/in/vandonr
or on Airflow Slack