Automating Workflows for Analytics Pipelines

Sadayuki Furuhashi
Sadayuki Furuhashi

An open-source hacker.

A founder of Treasure Data, Inc. located in Silicon Valley.

Github: @frsyuki

OSS projects I founded:

- fluentd
- embulk
- digdag
- MessagePack
What's Workflow Engine?

- Automates your manual operations.
  - Load data → Clean up → Analyze → Build reports
  - Get customer list → Generate HTML → Send email
  - Monitor server status → Restart on abnormal
  - Backup database → Alert on failure
  - Run test → Package it → Deploy (Continuous Delivery)
Challenge: Multiple Cloud & Regions

Different API, Different tools, Many scripts.

Microsoft Azure

RED HAT OPENSHIFT

On-Premises

Amazon Web Services

Google Cloud Platform

openstack
Challenge: Multiple DB technologies

- Amazon S3
- Amazon Redshift
- Google BigQuery
- Apache Spark
- Amazon EMR
- MongoDB
- MySQL
- Presto
- Cassandra
- Hadoop
- Microsoft SQL Server
Challenge: Multiple DB technologies

Hi!
I'm a new technology!

DynamoDB
Azure Cosmos DB
Challenge: Modern complex data analytics

**Ingest**
- Application logs
- User attribute data
- Ad impressions
- 3rd-party cookie data

**Enrich**
- Removing bot access
- Geo location from IP address
- Parsing User-Agent
- JOIN user attributes to event logs

**Model**
- A/B Testing
- Funnel analysis
- Segmentation analysis
- Machine learning

**Load**
- Creating indexes
- Data partitioning
- Data compression
- Statistics collection

**Utilize**
- Recommendation API
- Realtime ad bidding
- Visualize using BI applications
Traditional "false" solution

- Poor error handling
- Write once, Nobody reads
- No alerts on failure
- No alerts on too long run
- No retrying on errors
- No resuming
- No parallel execution
- No distributed execution
- No log collection
- No visualized monitoring
- No modularization
- No parameterization
Solution: Multi-Cloud Workflow Engine

- Poor error handling
- Write once, Nobody reads
- No alerts on failure
- No alerts on too long run
- No retrying on errors
- No resuming
- No parallel execution
- No distributed execution
- No log collection
- No visualized monitoring
- No modularization
- No parameterization
Example in our case

1. Dump data to BigQuery
2. Load all tables to Treasure Data
3. Run queries
4. Create reports on Tableau Server (on-premises)
5. Notify on slack
Workflow constructs
Unite Engineering & Analytic Teams

Powerful for Engineers
> Comfortable for advanced users

Friendly for Analysts
> Still straight forward for analysts to understand & leverage workflows

```plaintext
+wait_for_arrival:
  s3_wait>:
    bucket/www_${session_date}.csv

+load_table:
  redshift>:
    scripts/copy.sql
```
Unite Engineering & Analytic Teams

Powerful for Engineers
> Comfortable for advanced users

Friendly for Analysts
> Still straightforward for analysts to understand & leverage workflows

+ is a task
> is an operator
${...}$ is a variable

```
+wait_for_arrival:
s3_wait>:
  bucket/www_${session_date}.csv
+load_table:
  redshift>:
    scripts/copy.sql
```
Operator library

Standard libraries
redshift>: runs Amazon Redshift queries
emr>: create/shutdowns a cluster & runs steps
s3_wait>: waits until a file is put on S3
pg>: runs PostgreSQL queries
td>: runs Treasure Data queries
td_for_each>: repeats task for result rows
mail>: sends an email

Open-source libraries
You can release & use open-source operator libraries.

_export:
  td:
    database: workflow_temp

+task1:
  td>: queries/open.sql
  create_table: daily_open

+task2:
  td>: queries/close.sql
  create_table: daily_close
Parallel execution

Tasks under a same group run in parallel if _parallel option is set to true.

+load_data:
  _parallel: true

+load_users:
  redshift>: copy/users.sql

+load_items:
  redshift>: copy/items.sql
Loops & Parameters

Parameter
A task can propagate parameters to following tasks

Loop
Generate subtasks dynamically so that Digdag applies the same set of operators to different data sets.

```plaintext
+send_email_to_active_users:
  td_for_each>: list_active.sql
  _do:
  +send:
    email>: tempalte.txt
    to: ${td.for_each.addr}
```
Grouping workflows...
Grouping workflows

- Ingest
  - tasks
  + task
  + task
  + task

- Enrich
  - task

- Model
  - learn
    - task

- Basket analysis
  - task
  - task

- Load
  - task
  - task

- Utilize
Pushing workflows to a server with Docker image

Digdag server
> Develop on laptop, push it to a server.
> Workflows run periodically on a server.
> Backfill
> Web editor & monitor

Docker
> Install scripts & dependences in a Docker image, not on a server.
> Workflows can run anywhere including developer's laptop.

```
schedule:
  daily>: 01:30:00

timezone: Asia/Tokyo

_export:
  docker:
    image: my_image:latest

+task:
  sh>: ./run_in_docker
```
Demo
Digdag is production-ready

It's just like a web application.
Digdag is production-ready

Stateless servers + Replicated DB

Visual UI

Digdag client

HTTP Load Balancer

API & scheduler & executor

Digdag server

Digdag server

All task state

PostgreSQL

PostgreSQL

HA
Digdag is production-ready

Isolating API and execution for reliability

- **Digdag client**
- **HTTP Load Balancer**
- **Digdag server**
- **Digdag server**
- **Digdag server**

All task state

- **PostgreSQL**
- **PostgreSQL**

scheduler & executor

**HA**
Digdag at Treasure Data

- 850 active workflows
- 3,600 workflows run every day
- 28,000 tasks run every day
- 400,000 workflow executions in total
Digdag & Open Source
Learning from my OSS projects

• Make it pluggable!

- *fluentd*
  700+ plugins in 6 years
  input/output, and filter

- *embulk*
  200+ plugins in 3 years
  input/output, parser/formatter, decoder/encoder, filter, and executor

- MessagePack
  70+ implementations in 8 years
Digdag also has plugin architecture

- 32 operators
- 7 schedulers
- 2 command executors
- 1 error notification module
Visit my website! https://digdag.io