Luigi

Data workflow orchestration framework by Spotify
Used extensively by Crosswise (Oracle Data Cloud)
Boaz Menuhin
Crosswise identifies individuals across devices anonymously using Big-Data, Machine-Learning and other techniques.

We process 1.3 PB (gzipped) every week.

Our batch processing stack is made of Python, Scala, Java, AWS ElasticMapReduce, Hadoop, Tez, Spark, Pig and Luigi.

Part of Oracle Data Cloud.
What is a luigi?

- Data workflow orchestration framework
- Batch processing
- Handles the definition (a.k.a wiring, plumbing)
- Handles the execution (scheduling)
- DAG
What is a luigi?

cat /dev/random > file_a

cat file_a | grep "shakespeare" > file_b

cat file_a | grep "macbeth" > file_c

cat file_b | wc -l >> stats

cat file_c | wc -l >> stats
Pipeline - 2014 - 2Q

5 .py files

About 3K lines of code

10% of NYC
Pipeline - 2015 - 1Q

All available traffic of US
Pipeline - 2016 - 2Q

~100K loc
~700 .py files
~200 .pig files
~320 .java files
~50 .scala files
Run on US, GB, FR
Pipeline - 2017 - 2Q

~150K loc

~1100 .py files

~300 .pig files

~350 .java files

~100 .scala files

Run on US, GB, FR, IT, AU, ES, CA and soon BR

170 vertices
Data Flow Framework Requirements

- Modularity
- Dynamic task definition **by code** (not XML, YAML, JSON)
- Hadoop agnostic and support many technologies out of the box
- Enhance and restrict parallelization
- Same code for prod\dev\research
- Monitoring
- Scheduling
Many competitors

Airflow (airbnb)
Pinball (pinterest)
Oozie (apache)
Mrjob
...

Yet another workflow manager by every unicorn company
Key Concepts - Task

Task

- Logic <- Pig, Python, ScalaSpark, PySpark...
- Dependencies
- Parameters
- Outputs

Defined by code!
Key Concepts - Dependencies

Dependencies

- In what order tasks should be executed
- Termed “requirements”
Key Concepts - Parameters

Parameter

- Anything can be a parameter: file, date, number, string, task
- Affects the task signature
- Part of the task definition by code
- Notion of static typing
- Accessible from cli: --task-parameter-a=1
- Have configuration
- Can be injected from instantiating task
Key Concepts - Output\Target

Target

- A python object which implements “exists” method

Successful task execution: \( \forall \) target: target.exists() == true

Luigi will schedule a task: \( \exists \) target: target.exists() == false

Our targets: files, s3 objects, DB entries, deployment, cluster

* combined with the task signature
Key Concepts
Build in features

- Running tasks in parallel
- Resources
- Priorities
- Central Scheduling
- Wrapper Tasks <- used for abstraction and modularity
- CLI
```python
class TopArtists(luigi.Task):
    date = luigi.DateParameter()
    period = luigi.DateIntervalParameter()

def requires(self):
    return {
        'artists_feed': [ArtistFeed(date=d) for d in self.period]
    }

def run(self):
    inputs = self.input()
    self.run_hadoop_job(read_from=inputs['artist_feed'], write_to=self.output())

def output(self):
    return S3Target("s3://bucket-that-will-have/some/data/TopArtists/\%s" % self.date)
```
Example - Crosswise’s Luigi code

class AnotherTopArtistsInPig(PigTask):
    some_pig_input = PigInputParameter()
    any_parameter = PigPropertyParameter()
    outputs = ['filtered', 'filtered_into_parquet']

    # language=pig
    text = ""
    data = load_data('SOME_PIG_INPUT');
    filtered = FILTER data BY specific_field == 'ANY_PARAMETER';

    STORE some INTO '$OUTPUT/filtered.gz' USING PigStorage();
    STORE some INTO '$OUTPUT/filtered_into_parquet.gz' USING ParquetStorage();
    """
Crosswise’s Luigi
Luigi by Crosswise

Used for:
- Processing
- Preprocessing
- Getting data
- Deployments
- Deliveries
Luigi by crosswise

- Visibility
- Usability
- Execution
Visibility
Extended Execution

- Optimized code (Task.requires, Task.output)
- Optimized s3 access
- Deploy & publish underline
- Retry, Recovery and Reconnect of running hadoop jobs
- Remote jobs runs remotely!
Usability

- Runs only what you need - signatures are the thing!
- Smart task instantiation (local-remote, deployments).
- Scheduling from Jenkins
Lessons learned

- Using signature for proof-of-run or an equivalently granular property
- Research\prod should run the same code on same environment
- Do not use for resource allocation
Crosswise with and without luigi
Thank you

(We are hiring)