使用Python开发数据流水线任务智能调度系统

阳曙光 @ 猎聘网 DIG

20161015

Agenda

- 背景
- 系统架构
- 现有系统
- 技术选择

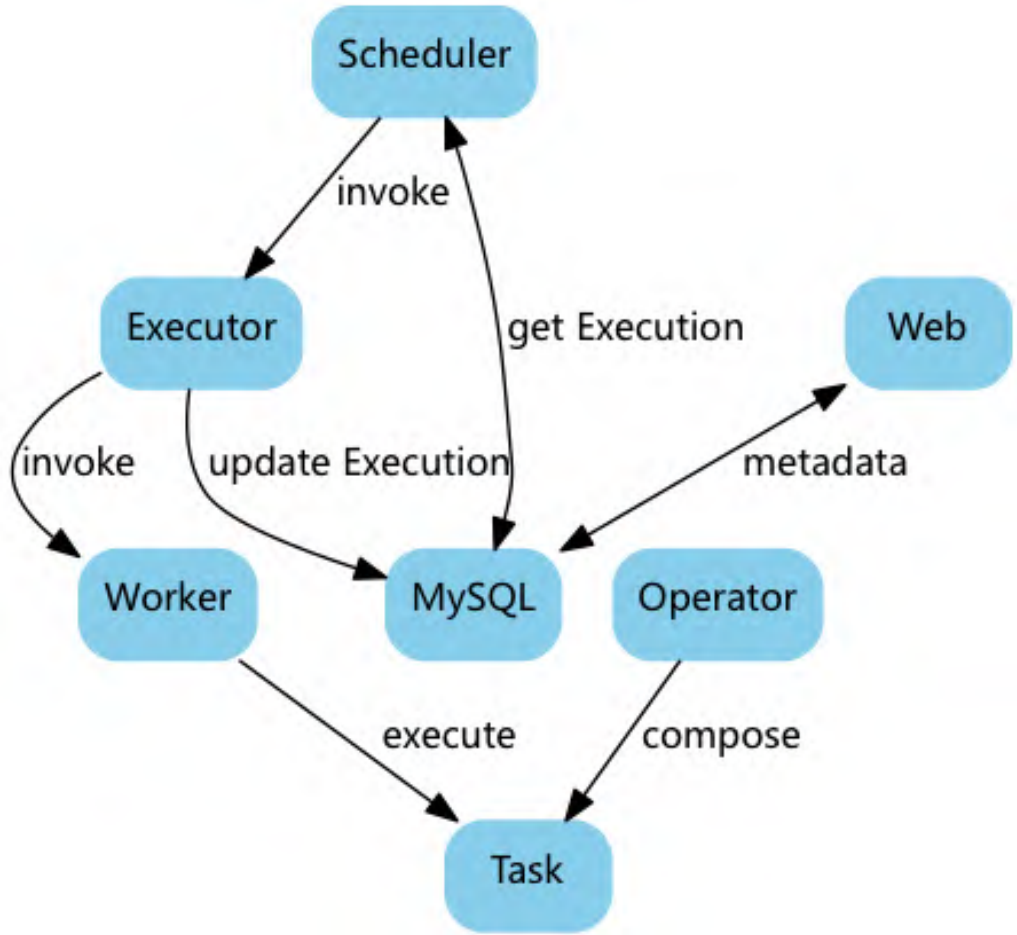
背景

- 典型的数据流水线任务
  - 执行Hive SQL
  - 稽核数据
  - 导出数据（邮件、MySQL、龙猫）
  - 发送相关通知
- 任务特点
  - 周期性执行（按小时、日、周、月）
  - 分层依赖
  - 执行时间长

背景

- 系统必要的功能
  - 定义任务
  - 按依赖关系执行任务
  - 按优先级执行任务
  - 并发执行任务
  - 定时调度
  - 非开发人员提交任务
  - 监控和提醒

系统架构


Scheduler
invoke
Executor
get Execution
Web
invoke
update Execution
metadata
Worker
MySQL
Operator
execute
compose
Task

现有系统

- Luigi
  - spotify
  - github stars 5499
- airflow
  - airbnb
  - github stars 3628
  - apache incubator project
- korin
  - 猎聘网
  - 内部使用中

现有系统：技术栈


multiprocessing
celery
pythondeamon
sqlalchemy
tornado
dill
schedule
kazoo
flask
impyla
PyYAML
pyhs2
pyHive
kafka
jinja2
croniter

技术选择

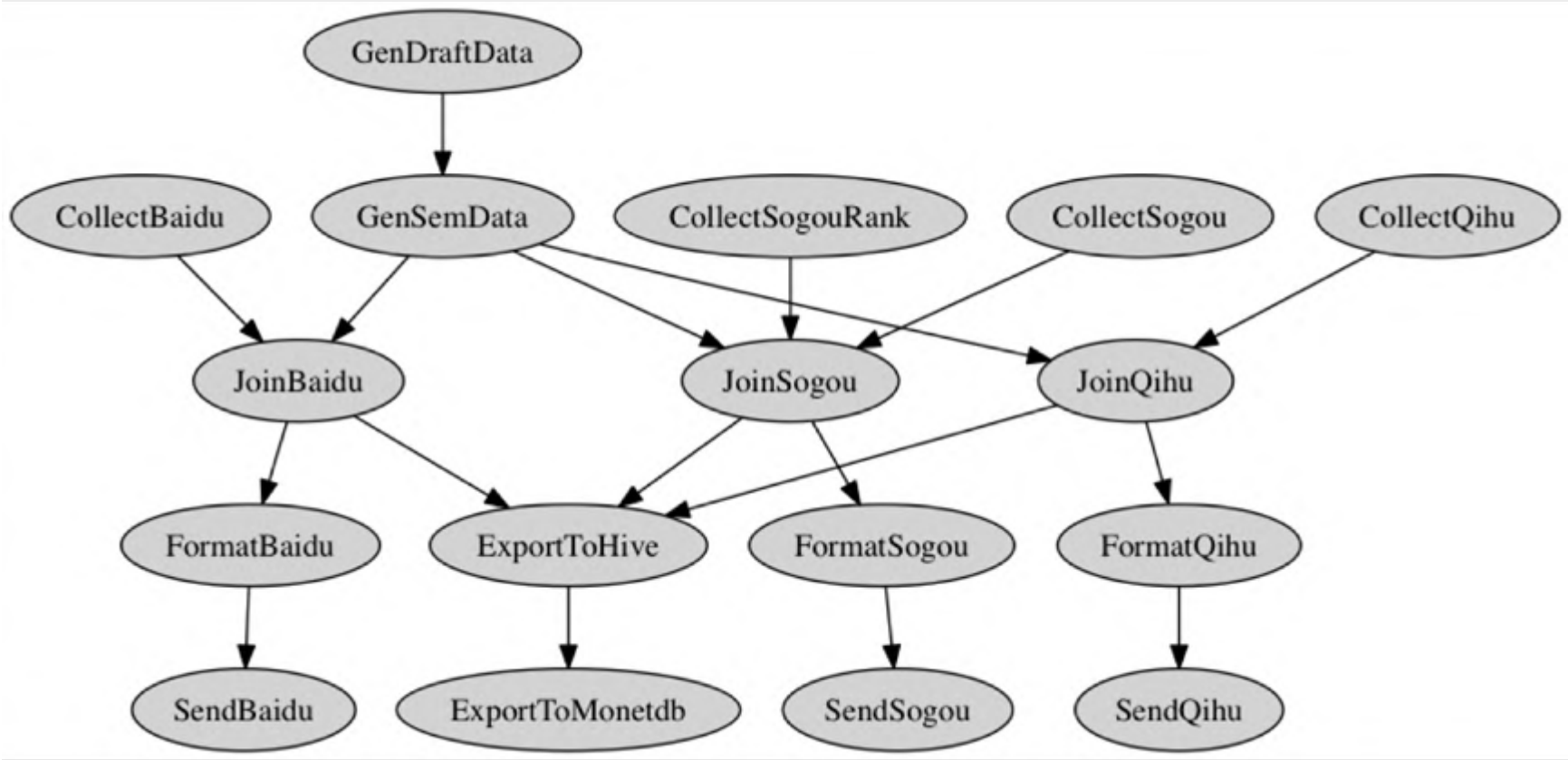
- 任务依赖
- 任务定义
- 执行调度
- Web
- 大数据

技术选择

- 任务依赖
- 执行调度
- 任务定义
- Web
- 大数据

任务依赖

- DAG


GenDraftData
CollectBaidu
GenSemData
CollectSogouRank
CollectSogou
CollectQihu
JoinBaidu
JoinSogou
JoinQihu
FormatBaidu
ExportToHive
FormatSogou
FormatQihu
SendBaidu
ExportToMonetdb
SendSogou
SendQihu

任务依赖

- Luigi
  - make方式，待执行的任务启动依赖的任务
  - 检查output判断依赖是否完成
- airflow
  - Python代码定义DAG
- korin
  - 动态生成DAG
  - 检查依赖完成消息

技术选择

- 任务依赖
- 任务定义
- 执行调度
- Web
- 大数据

任务定义: Luigi

```python
import luigi

class MyTask(luigi.Task):
    param = luigi.Parameter(default=42)

    def requires(self):
        return SomeOtherTask(self.param)

    def run(self):
        f = self.output().open('w')
        print >>f, "hello, world"
        f.close()

    def output(self):
        return luigi.LocalTarget('/tmp/foo/bar-%s.txt' % self.param)

if __name__ == '__main__':
    luigi.run()
```

The business logic of the task

Where it writes output

What other tasks it depends on

Parameters for this task

任务定义：airflow

```python
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta


default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2015, 6, 1),
    'email': ['airflow@airflow.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    # 'queue': 'bash_queue',
    # 'pool': 'backfill',
    # 'priority_weight': 10,
    # 'end_date': datetime(2016, 1, 1),
}

dag = DAG(
    'tutorial', default_args=default_args, schedule_interval=timedelta(1))
```

任务定义：airflow

```
# t1, t2 and t3 are examples of tasks created by instantiating operators
t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag)

t2 = BashOperator(
    task_id='sleep',
    bash_command='sleep 5',
    retries=3,
    dag=dag)

templated_command = """
    {% for i in range(5) %}
        echo "{{ ds }}"
        echo "{{ macros.ds_add(ds, 7)}}"
        echo "{{ params.my_param }}"
    {% endfor %}
"""

t3 = BashOperator(
    task_id='templated',
    bash_command=templated_command,
    params={'my_param': 'Parameter I passed in'},
    dag=dag)
```

任务定义：airflow


```
t2.set_upstream(t1)
t3.set_upstream(t1)
```

任务定义：korin

- Operator 定义具体功能
- 多个Operator组合成一个任务
- 任务元信息存储在MySQL中
  - YAML字符串存储Operator组合
  - 任务dependencies属性保存所有依赖的任务列表

技术选择

- 任务依赖
- 任务定义
- 执行调度
- Web
- 大数据

执行调度

- Luigi
  - 无
- airflow
  - crontab语法
- korin
  - schedule

airflow: croniter


```python
from croniter import croniter
from datetime import datetime
base = datetime(2016, 10, 15, 13, 45)
cron_iter = croniter('*/5 * * * *', base)  # every 5 minites
print cron_iter.get_next(datetime)  # 2016-10-15 13:50:00
print cron_iter.get_next(datetime)  # 2016-10-15 13:55:00
print cron_iter.get_next(datetime)  # 2016-10-15 14:00:00

cron_iter = croniter('2 4 * * mon,fri', base)  # 04:02 on every Monday and Friday
print cron_iter.get_next(datetime)  # 2016-10-17 04:02:00
print cron_iter.get_next(datetime)  # 2016-10-21 04:02:00
print cron_iter.get_next(datetime)  # 2016-10-24 04:02:00
```

Korin: schedule


```python
import schedule
import time


def job():
    print("I'm working...")

schedule.every(10).minutes.do(job)
schedule.every().hour.do(job)
schedule.every().day.at("10:30").do(job)
schedule.every().monday.do(job)
schedule.every().wednesday.at("13:15").do(job)

while True:
    schedule.run_pending()
    time.sleep(1)
```

技术选择

- 任务依赖
- 任务定义
- 执行调度
- Web
- 大数据

技术选择

- Web
  - Luigi: tornado
  - airflow: flask + sqlalchemy
  - korin: flask + sqlalchemy

技术选择

- 任务依赖
- 执行调度
- 任务定义
- Web
- 大数据

大数据

- 访问Hive
  - pyhs2
  - pyHive
  - impyla
- 访问hdfs
  - snakebite
- kafka
  - kafka-python
- zookeeper
  - kazoo

Q&A