北京 2019
遨游“视”界 做你所想
Explore World, Do What You Want

基于Airflow的视频编码平台

Hotstar Beijing

出品: LiveVideoStack 音视频技术社区 CSDN

深圳 2019

遨游“视”界 做你所想
Explore World, Do What You Want

LiveVideoStackCon 2019 深圳

2019.12.13-14

成为讲师： speaker@livevideostack.com

成为志愿者： volunteer@livevideostack.com

赞助、商务合作： kathy@livevideostack.com

出品：LiveVideoStack 音视频技术社区 CSDN

关于Hotstar

Hotstar 是迪士尼（Disney）和福克斯（Fox）旗下网络视频平台，是东南亚地区最大的版权视频网站。峰值流量超过印度全国 CDN的 70%，拥有12.6万小时版权视频，刚创造了2530万用户在线观看直播的最高纪录，峰值日活跃用户超过 1 亿。

1. 视频数量庞大，编码工作流管理艰巨
2. 视频类别繁多，需要引入AI处理
3. 头部内容集中，需要极致编码优化

我们的痛点

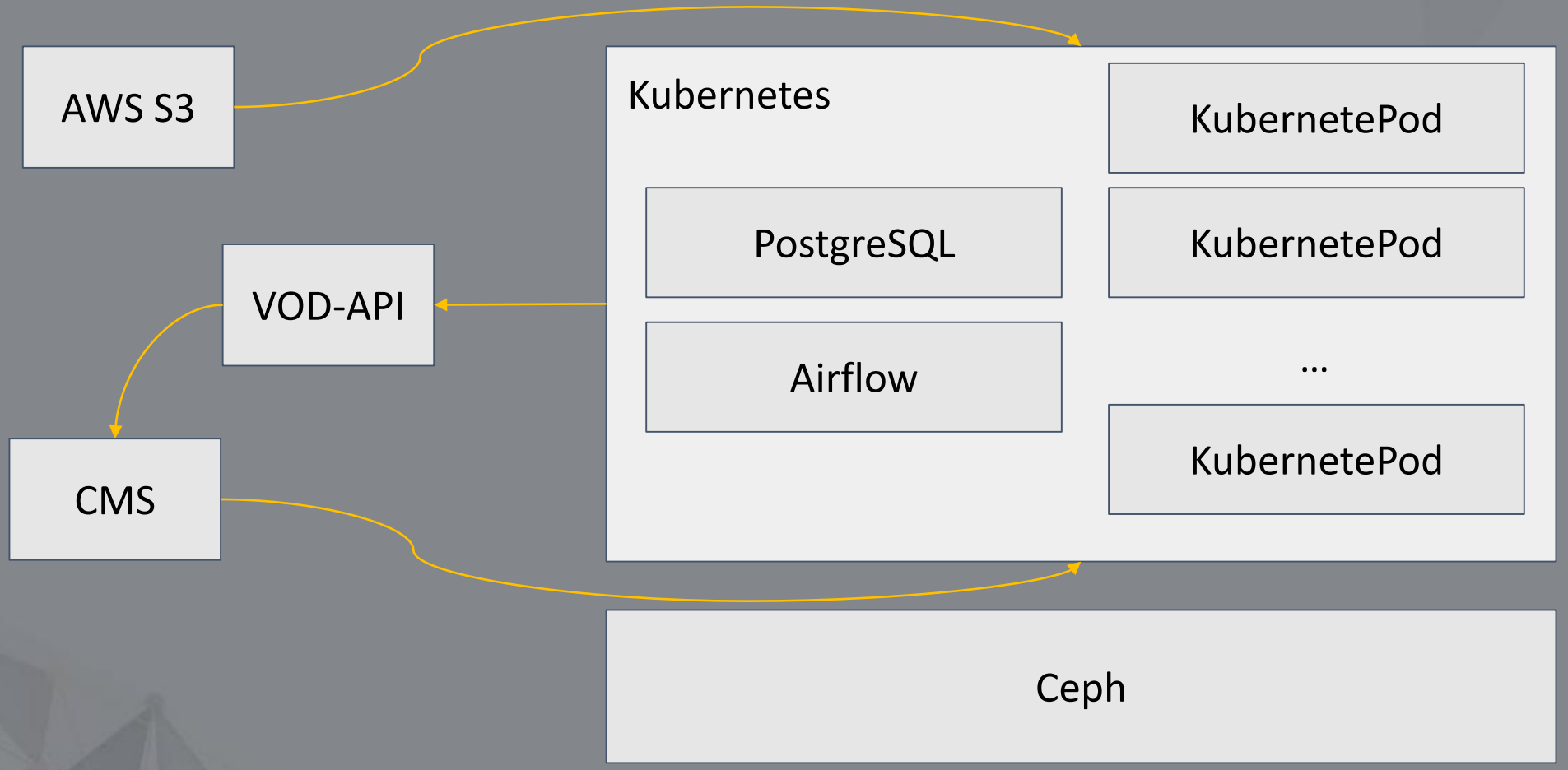
海量视频编码工作如何可追溯、可视化?

计算资源有限情况下如何调度编码任务?

如何统计分析视频编码时间开销?

怎样实现工作流动态执行?

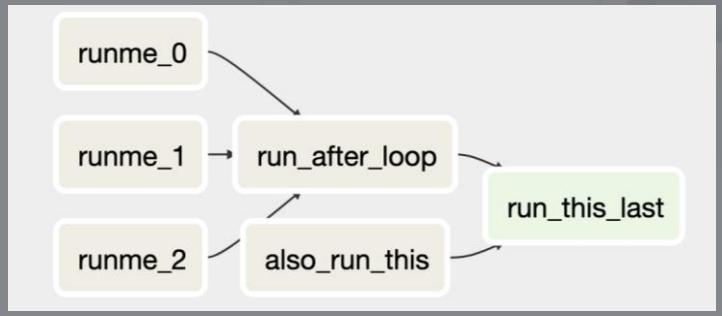
平台架构

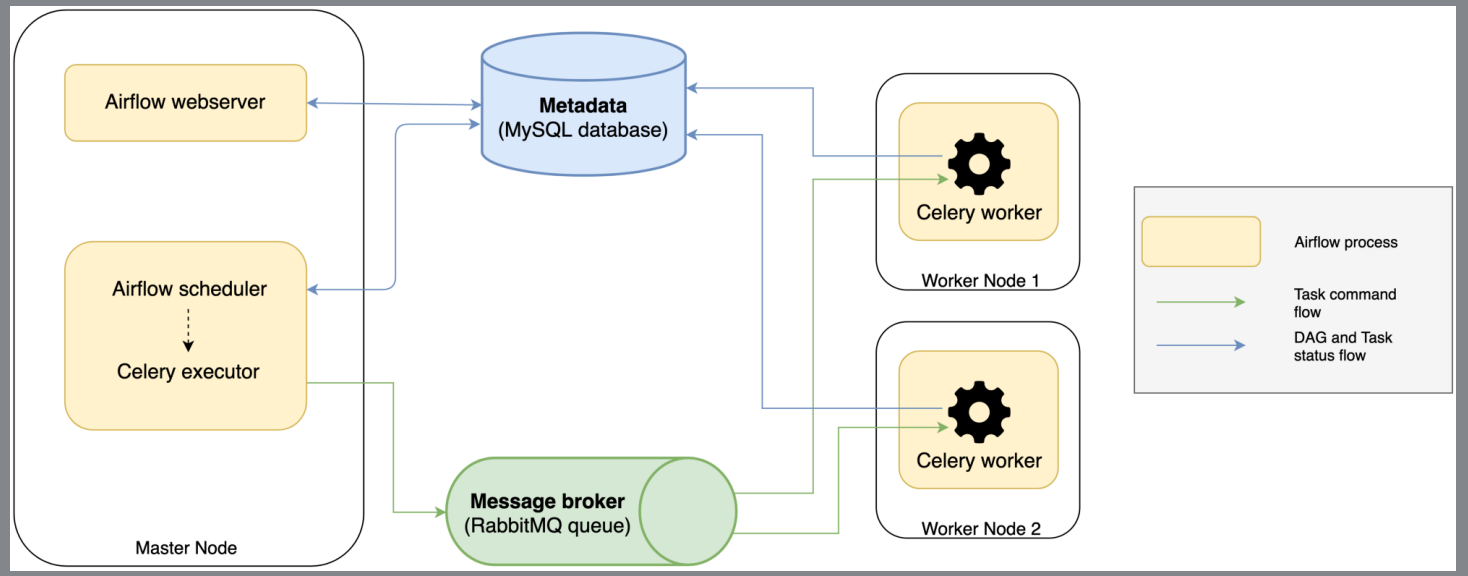

AWS S3
VOD-API
CMS
Kubernetes
PostgreSQL
Airflow
KubernetePod
KubernetePod
...
KubernetePod
Ceph

基本概念

- DAG (Dag run)：有向无环图
- Operator（Task）：有向无环图中的节点

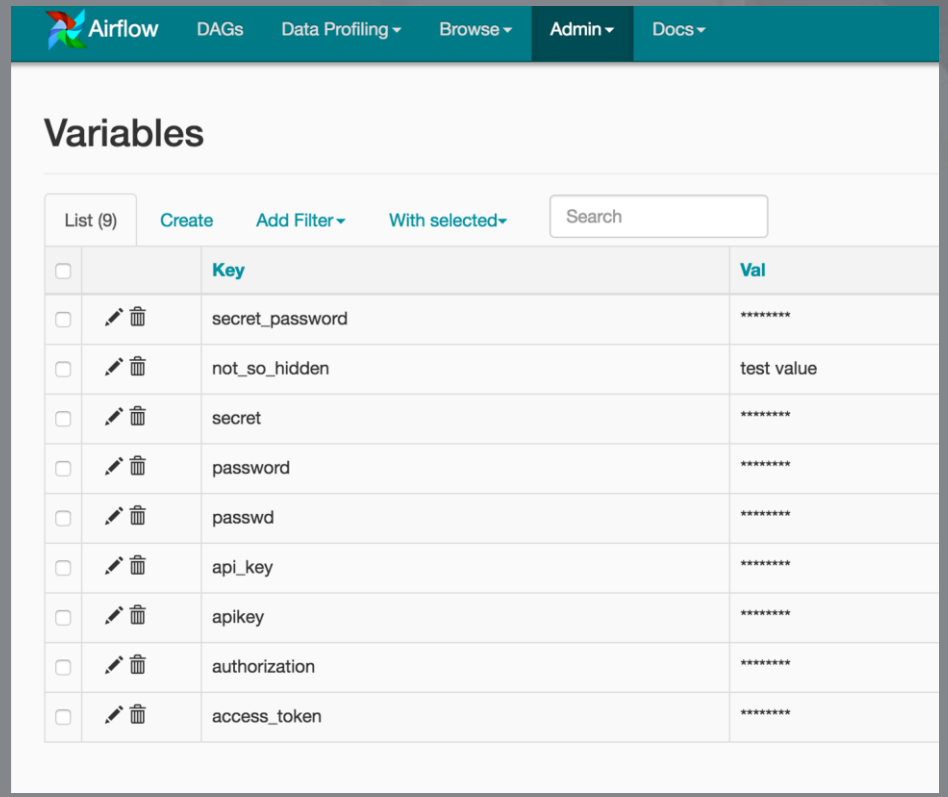
  Bash, Docker, SSH, KubernetesPod...





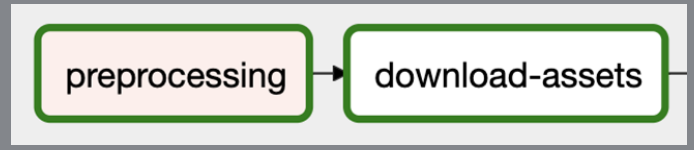
如何管理并传递全局变量?
Variable

1. Key-Value形式，Value可为字符串或json
2. 可在 Web UI中配置，也可在DAG中修改
3. 可加密存储Token，避免硬编码，更安全
4. 也可用于不同部署环境下的环境变量配置



如何在Operator间传递上下文?
XCOM

Xcom “cross-communication” 的缩写。主要用于同一个DagRun的Task间交换数据。交换数据的Task不必相邻，可跨多个Task传递。




```
"""\
  python download.py --work_directory {{ti.xcom_pull(task_ids='preprocessing').work_dir}} \
  --file_path "{{ti.xcom_pull(task_ids='preprocessing').input_key}}" \
  --source_bucket {{ti.xcom_pull(task_ids='preprocessing').s3_bucket}}\
"""
```

```
python download.py --work_directory /mnt/a9e3776b997fbf963abaefdd_14-15-05
--file_path "PREPROD/13-08-2019/JETHSP_1_9--20190001209190--PG--2012.mov"
--source_bucket hotstar-source-media
```

如何操作Kubernetes?

KubernetesPodOperator

支持KubernetesPod大部分操作，通过resources参数控制为Task具体分配的资源

Pod使用的Image在这里定义

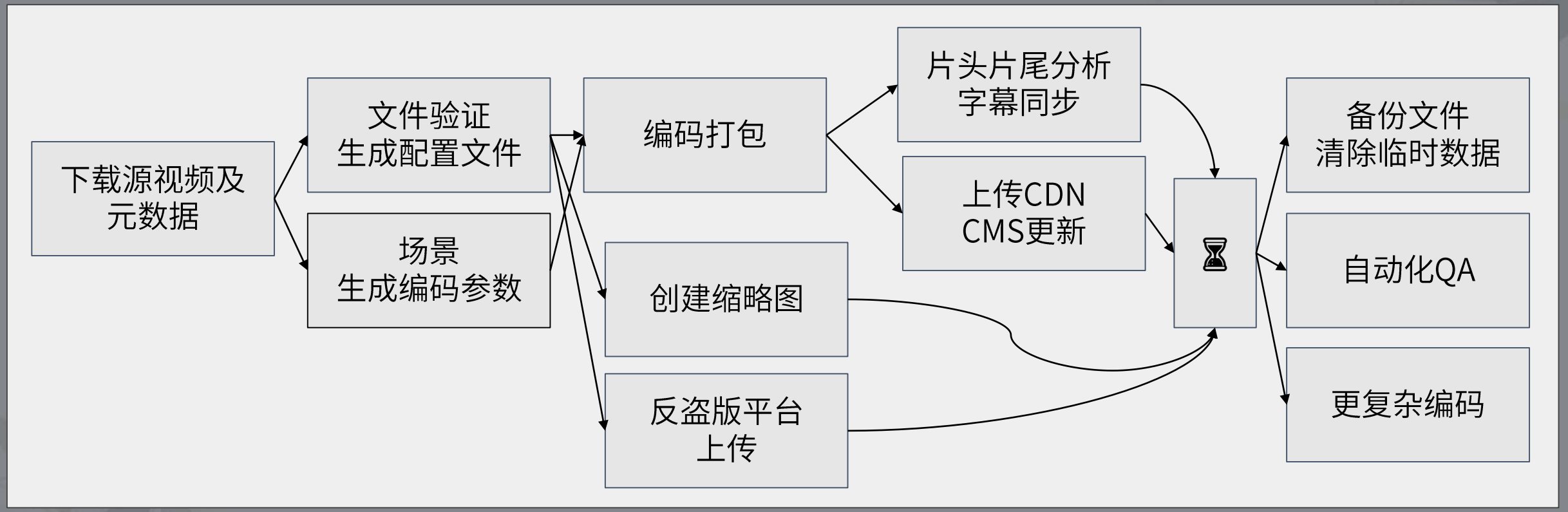
```
encode_task = KubernetesPodOperator(namespace='default',
                image="vod:release-1.0.2",
                cmds=["Python","-c"],
                arguments=["/vod/encode.py"],
                name="encode",
                task_id="encode-task",
                dag=dag,
                resources=Resources(
                        request_memory=REQUEST_MEMORY,
                        limit_memory=LIMIT_MEMORY,
                        request_cpu=REQUEST_CPU,
                        limit_cpu=LIMIT_CPU
                ))
```

Pod具体执行脚本

请求资源分配

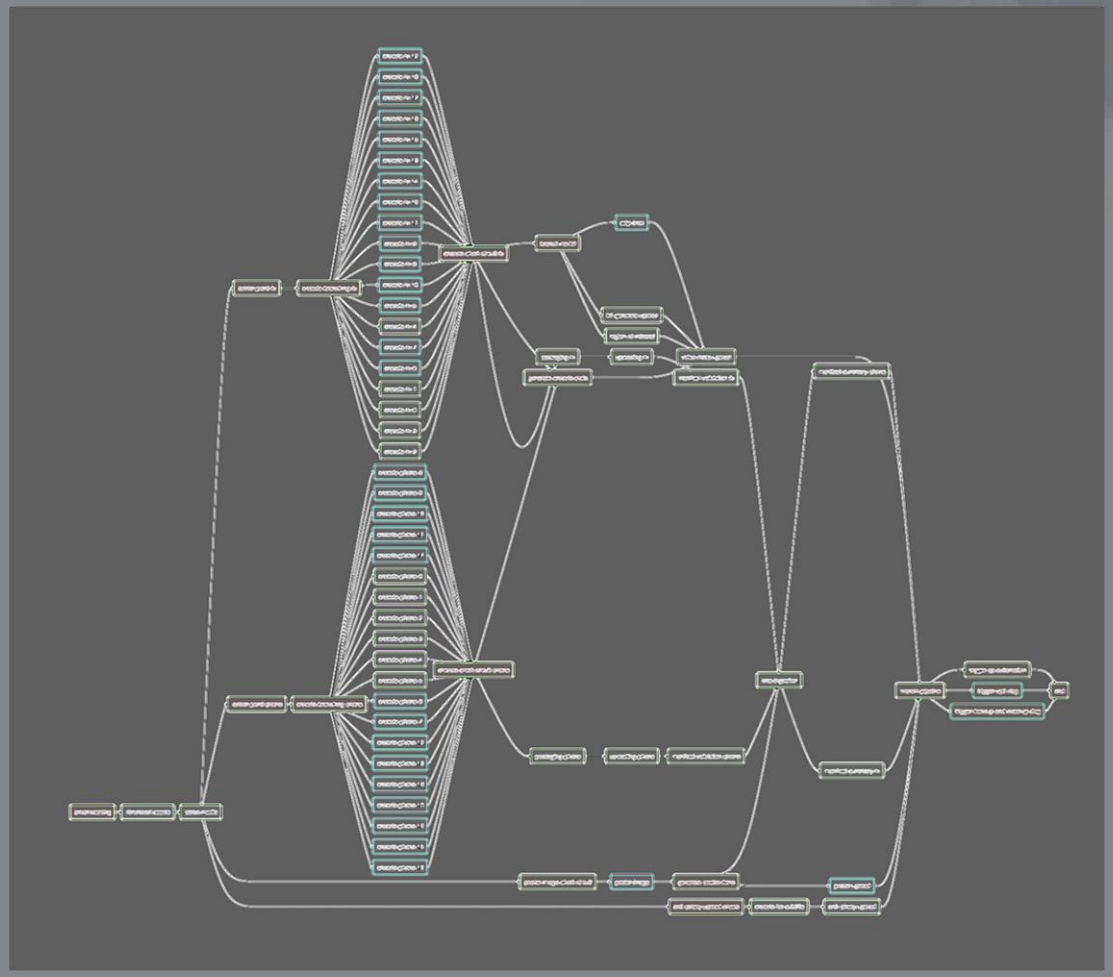
Using the KubernetesPodOperator （Google）

基于Airflow设计一个编码工作流

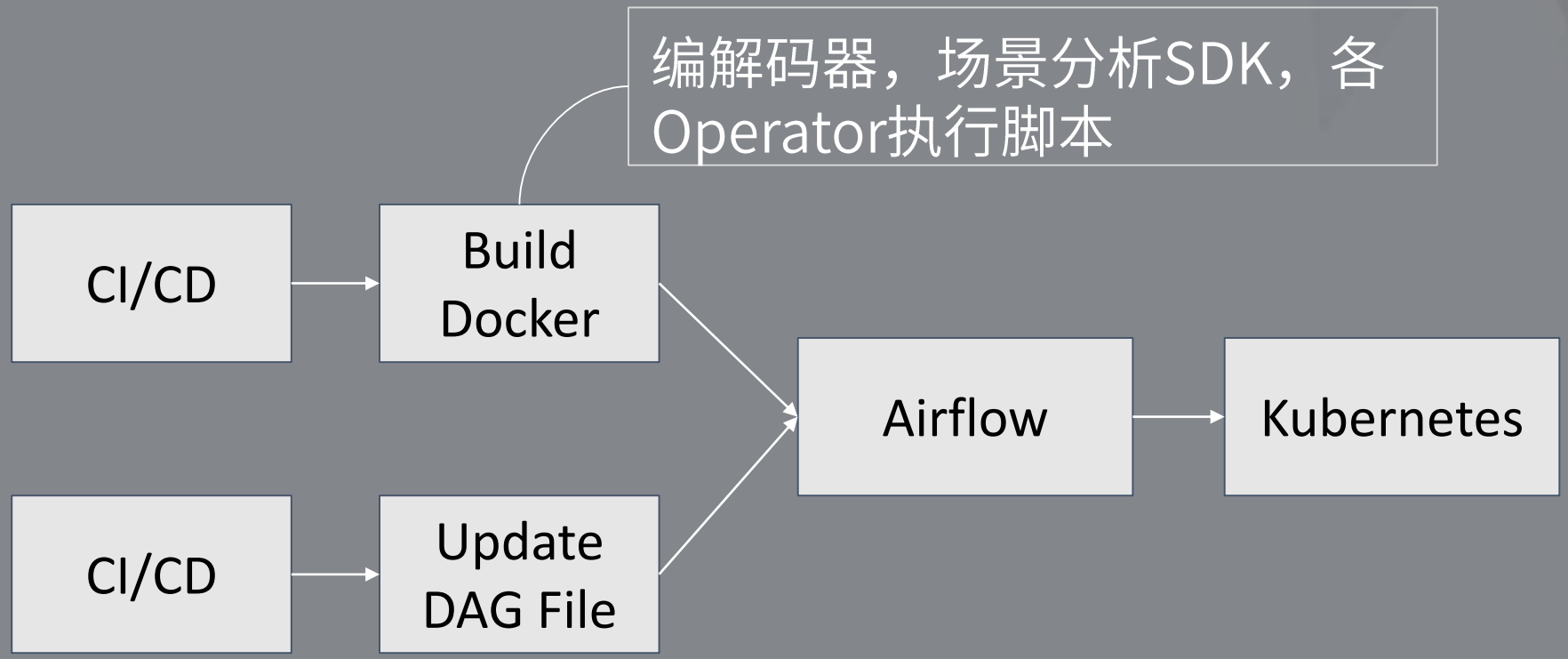


基于Airflow设计一个编码工作流


```
dag = DAG(
    dag_id=DAG_ID,
    default_args=args,
    schedule_interval=None,
    concurrency=int(Variable.get('MOVIE_LADDER_CONCURRENCY', 60)),
    max_active_runs=int(Variable.get('MOVIE_LADDER_MAX_ACTIVE_RUN', 30)),
    user_defined_filters={'fromjson': lambda s: json.loads(s), 'basename': lambda s: os.path.basename(s),
                          'tojson': lambda s: json.dumps(s),
                          'content_id': lambda catalog: os.path.splitext(os.path.basename(catalog))[0].split('-')[1],
                          'hls_manifest': lambda manifest_paths: hls_manifest(manifest_paths),
                          'convert_phone': lambda tv_urls: convert_phone(tv_urls)
                         }
)


def prepare_params(**context):
    data = {}
    time_obj = datetime_parser(str(context['dag_run'].run_id).split('_')[-1])
    run_id = time_obj.strftime('%Y-%m-%dT%H-%M-%S')
    data['content_type'] = context['dag_run'].conf.get('content_type', 'movie').lower()
    data['content_id'] = context['dag_run'].conf.get('content_id')
    data['id'] = context['dag_run'].conf.get('id', str(uuid.uuid4())) + '_' + run_id
    data['work_dir'] = os.path.join(TARGET_DIR, data['id'])
    data['output_dir'] = os.path.join(data['work_dir'], 'output')
    data['metadata'] = os.path.join(data['output_dir'], 'metadata.json')
    data['catalog'] = os.path.join(data['work_dir'], 'catalog.xml')
    data['backup_s3_config'] = "s3://" + os.path.join(BACKUP_S3_BUCKET, BACKUP_DESTINATION_PREFIX, data['id'],'metadata.json')
    data['cms_s3_dir'] = CMS_S3_DIR
    return data


preprocessing = PythonOperator(task_id="preprocessing",
                               dag=dag,
                               python_callable=prepare_params,
                               provide_context=True,
                               xcom_push=True,
                               on_success_callback=started_callbacks,
                               priority_weight=MAX_PRIORITY_WEIGHT
                               )

##########################################CMS FETCH ###################################################
# MOVIE-REENCODE <
all_platform_list = TV_PLATFORMS.split(",")
all_platform_list.extend(PHONE_PLATFORMS.split(","))
all_platform_list = [ p for p in all_platform_list if p ]

country_list = COUNTRY_LIST.split(",")

cms_fetch_command = """\
                               python cms_fetch.py --content_id {{ti.xcom_pull(task_ids='preprocessing').content_id}} \
                               --cms_fetch_api "%s" \
                               --country_list %s \
                               --hotstarauth %s \
                               --work_dir {{ti.xcom_pull(task_ids='preprocessing').work_dir}}\
```

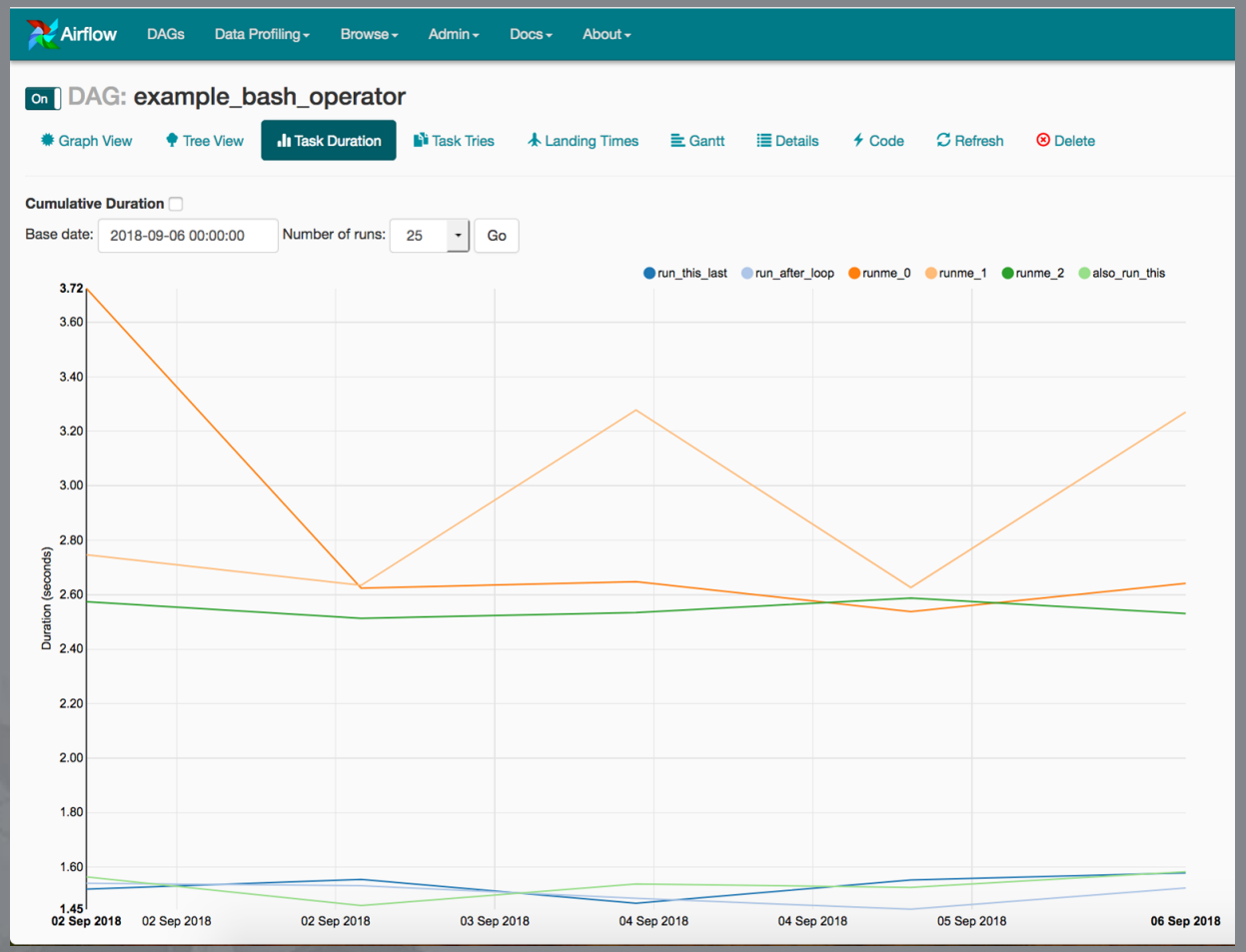


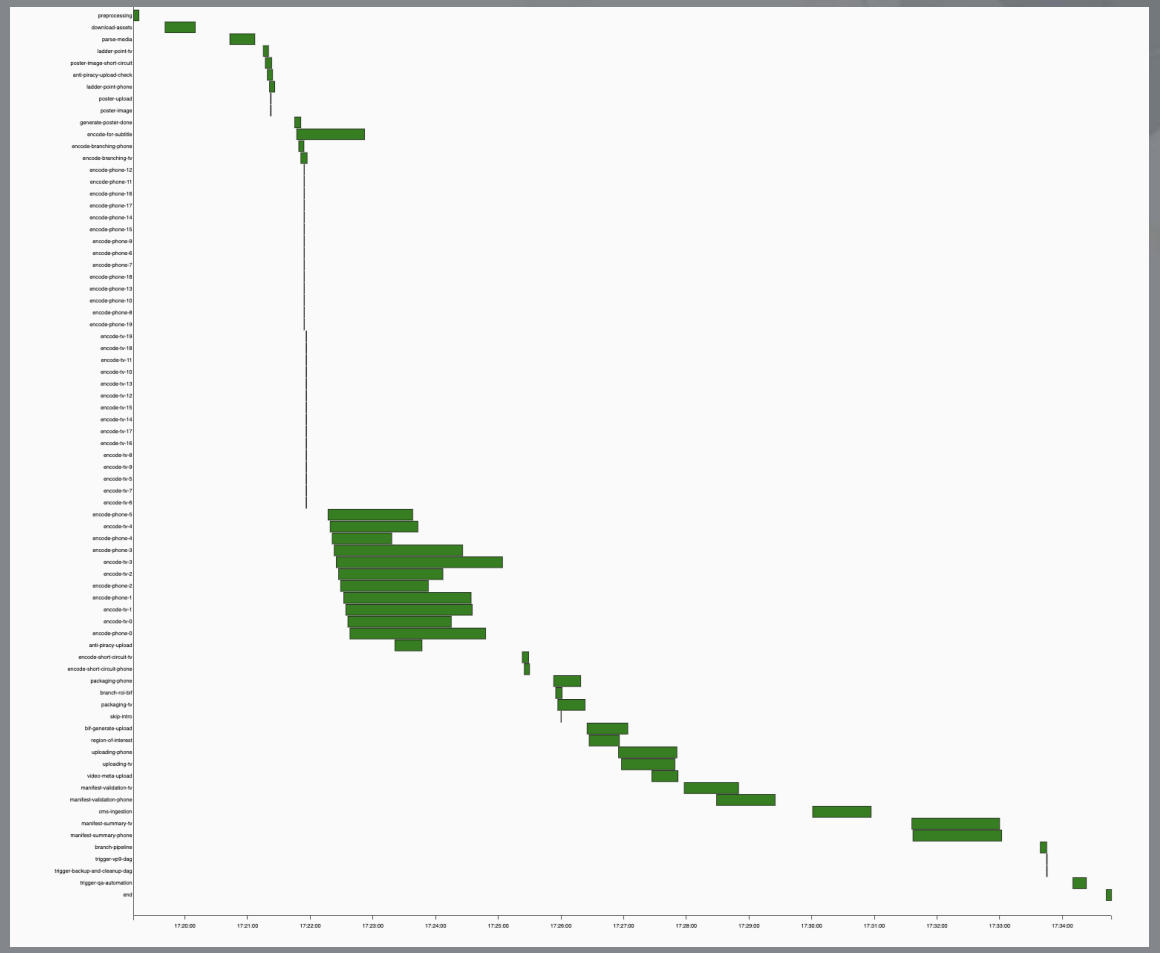
我们的发布工作流


编解码器，场景分析SDK，各Operator执行脚本
CI/CD
Build Docker
CI/CD
Update DAG File
Airflow
Kubernetes

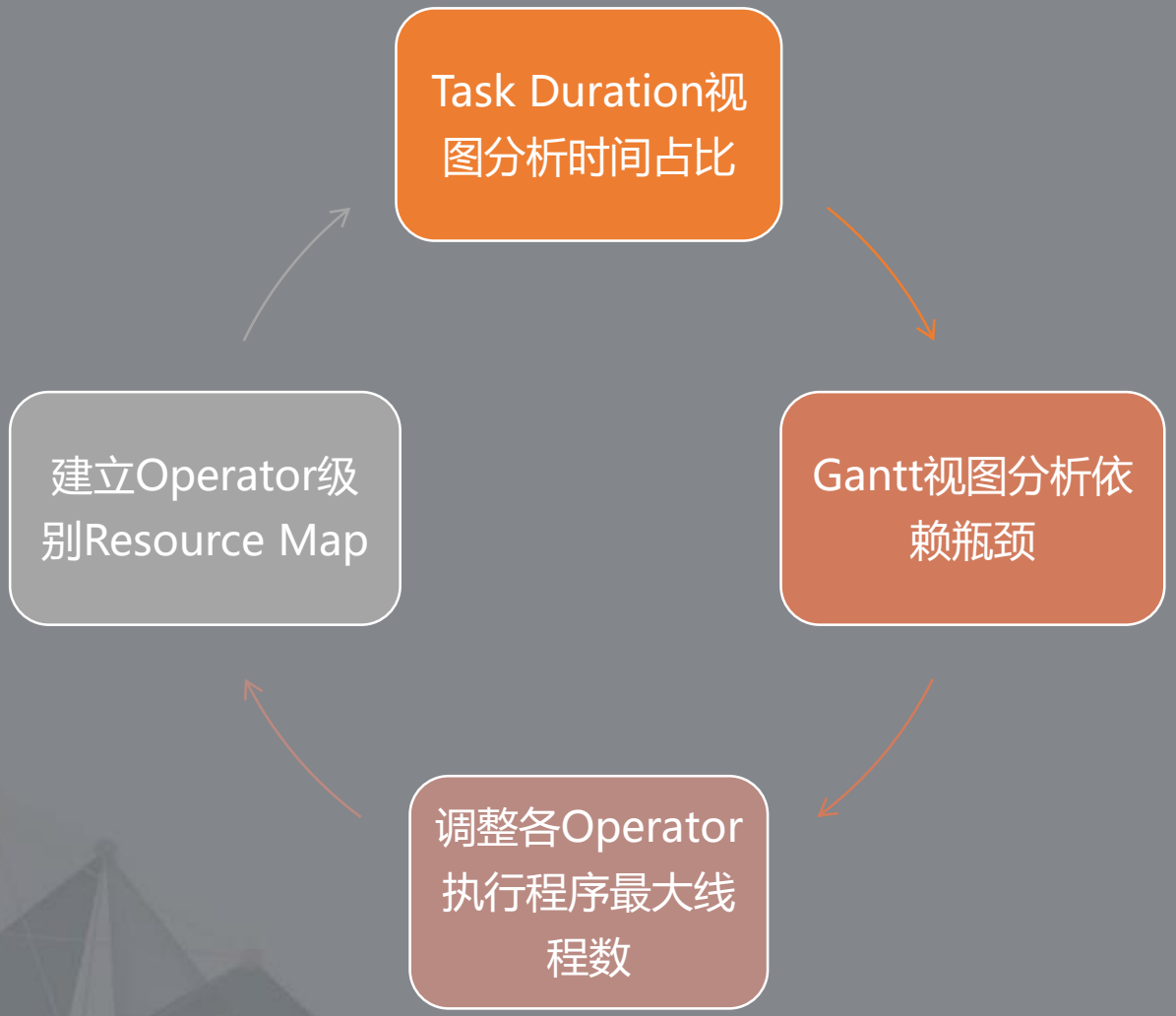
如何统计分析视频编码时间开销?

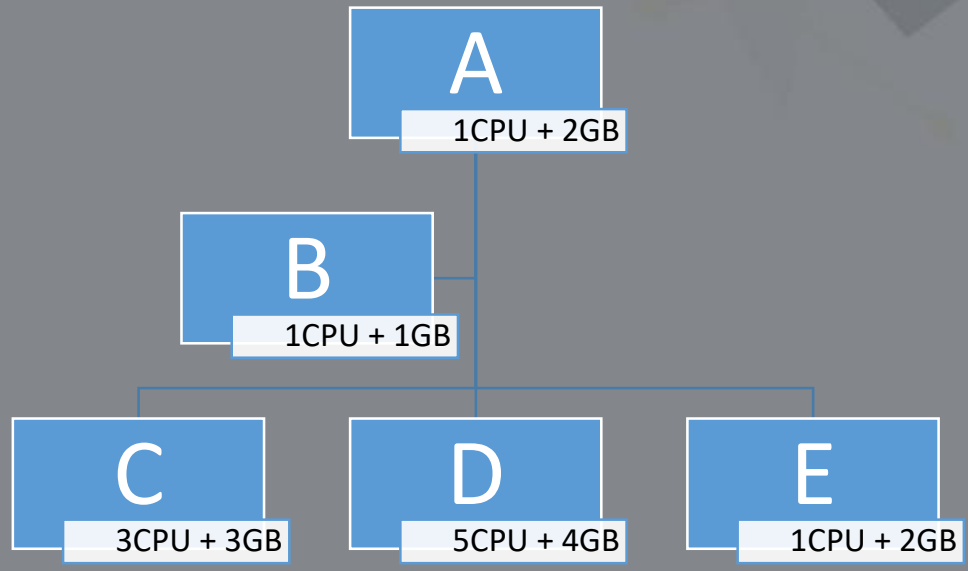
Gantt & Task Duration View





计算资源有限情况下如何调度?
Resource Map





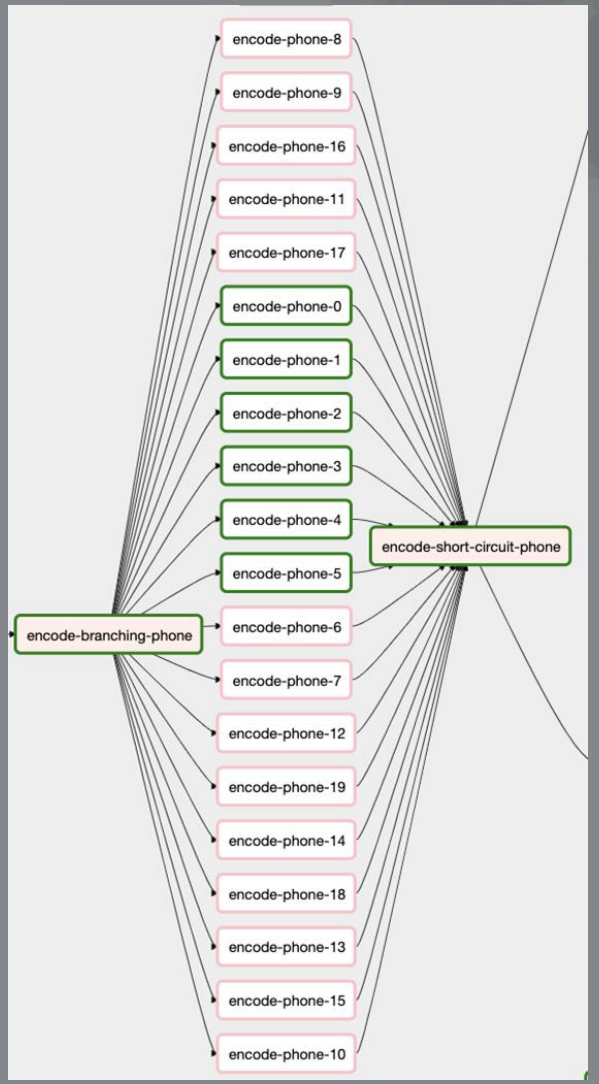
- 尽量使上游Task在相近时间完成
- 尽量使分配的CPU内存有效使用

怎样实现工作流动态执行?

Operator Placeholder+BranchOperator

DAG图是在DAG工作流开始前构建的，DAG运行时无法改变DAG拓扑。

不同视频需要输出分辨率组合不同。需要根据需求动态启动不同数量的编码Task。



解决方法：在DAG构建时生成足够的占位Operator，前置BranchOperator。 BranchOperator在运行时根据需求分配编码Task，并将冗余Task的状态设置为Skip。

踩过的坑

1. 当DAG更新时，已经完成的早期DAG Run按照新DAG视图显示，与原执行过程不匹配。

2. DAG代码是在DagRun正式启动前执行的，相当于Build time，不是执行完成一个Task再去构建下一个Task，只有Operator里面调用的代码相当于Runtime

踩过的坑

3. 当DAG内Operator数量过多、并行任务过于密集时，Scheduler效率非常低

4. Airflow Variable中只接受双引号JSON，使用单引号（Airflow Variable导出格式）会导致DAG加载失败

5. KubernetesPodOperator本地开发较麻烦，可以通过airflow plugin实现一个假的KubernetesPodOperator，让它实际调用DockerOperator

优点总结

1. 开发与运维解耦
2. 生态社区趋于成熟
3. 灵活易定制
4. 分析功能强大

本地快速部署尝鲜

1. 安装Docker Desktop并启用Kubernetes
2. brew install kubernetes-helm
3. helm install --namespace "airflow" --name "airflow" stable/airflow
4. export POD_NAME=$(kubectl get pods --namespace airflow -l "component=web,app=airflow" -o jsonpath="{.items[0].metadata.name}")
   echo http://127.0.0.1:8080
   kubectl port-forward --namespace airflow $POD_NAME 8080:8080
5. 打开http://127.0.0.1:8080

有待解决的问题

1. DAG并行数大时Scheduler存在效率瓶颈
2. Scheduler无官方热备方案
3. DAG任务 Tree View、Landing Time等分析功能较为耗时
4. Xcom、DagRun信息检索不方便
5. 异构分布式Worker及调度

北京
2019

遨游“视”界 做你所想
Explore World, Do What You Want

Thank you

出品：LiveVideoStack 音视频技术社区 CSDN