EP1 - 调度平台

Keyword: DAG | Scheduler | Worker

Date 10/30/2025

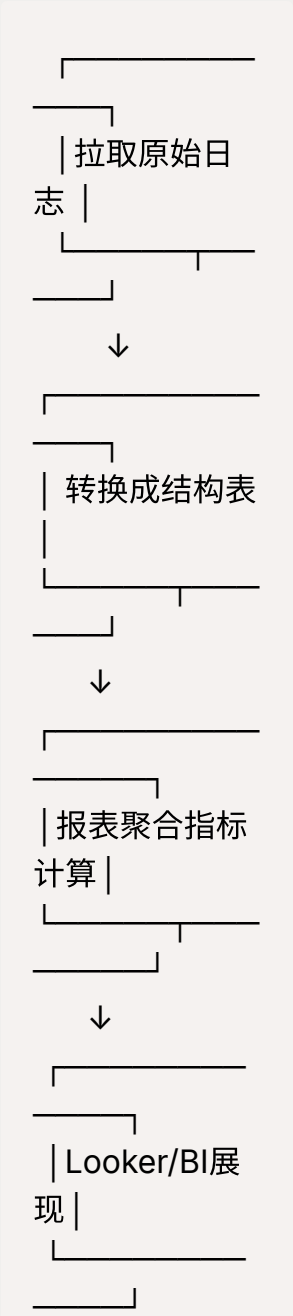
不做数据处理本身，但掌控：

- 谁先跑、谁后跑（依赖关系）
- 什么时候跑（定时触发）
- 出错怎么办（失败处理、报警）
- 哪里跑（指定服务器或集群资源）

DAG（Directed Acyclic Graph - 有向无环图）

描述"先做A，再做B，最后做C"的依赖关系
Direct：每个任务"有方向"， A→B表示A执行完才能到B；
Acyclic：所有任务不能回头，不能A→B→A
Graph：任务之间像图形一样连接，形成执行路径，A→B→C

📌 典型DAG



Scheduler（调度器）

定时触发 DAG 的运行，控制所有任务的启动/暂停

四个核心职责：

1. 定时触发任务（设置Cron — 描述定时触发规则）
2. 判断依赖是否满足（Task B需要等Task A成功)
3. 管理执行状态（如果失败了重复几次？是否跳过？）
4. 分配Worker（哪个任务分给哪个机器跑）

Worker（执行器）

实际执行任务代码或脚本的计算资源节点（比如执行 SQL、Python 脚本）

关键能力点：

1. 任务执行（跑代码，比如执行 `run_report.py`,连DB抽数）
2. 资源隔离（每个任务再自己的进程/容器跑，互不干扰）
3. 并发控制（控制同一时间最多跑几个任务，比如 `max_threads=5` ）
4. 日志记录（保存输出执行结果）
5. 失败自动汇报（返回exit code和报错内容给调度器）

使用场景

使用场景	你在其中的位置	技术行为
已有任务跑崩了	查失败日志 → 判断数据源还是依赖炸了	会读调度平台日志
想增加某个ETL流程	跟调度管理员说：我想在B之后插一个清洗脚本	能描述“依赖关系”和“调度时间”
想优化分析报表刷新速度	找到上游调度瓶颈点，比如某个 Hive 任务慢	会定位 DAG 中“最长路径”任务并协商优化

DAG

不同平台调度器

平台	调度器机制	说明
Airflow	有一个持续运行的 `airflow scheduler` 进程	实时检测 DAG 状态变化，控制 DAG Run 的生成
Azkaban	内建 `Executor + Scheduler`	把 Project 上传后，由调度器决定执行时间
xx-job	调度中心 + 执行器模式	中心负责定时调度，执行器跑任务

常见Advanced能力

能力	用处
并发控制	限制同一时间运行多少任务（防资源爆）
Slot Pooling	控制哪些任务抢哪些机器资源
Dynamic DAG	根据日期自动生成不同的DAG结构
SLA 监控	任务超时报警（比如该5分钟跑完的结果跑了1小时）

Worker

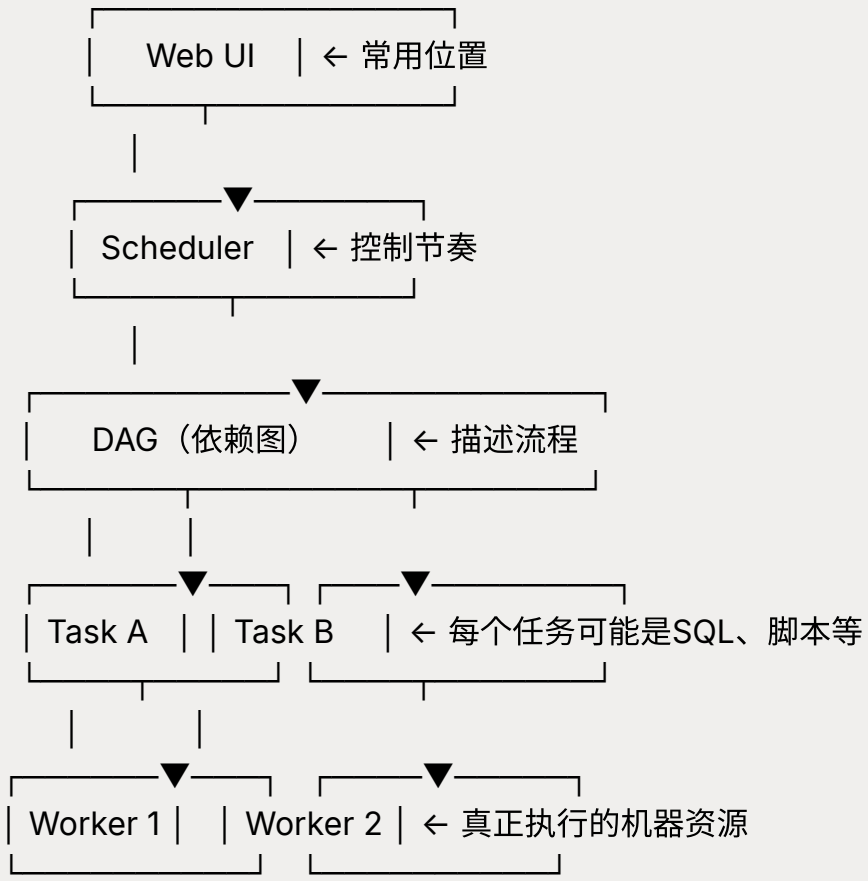
Worker常见样式（取决于平台和架构）

平台	Worker 形态	常见部署方式
Airflow	Python 进程 + Celery/Kubernetes	多台机器、容器分布式并行
Azkaban	Executor Server	通常是独立机器或进程
xx-job	Java agent	通常是跑在业务服务端的轻量执行器

常见问题

问题	可能对应 Worker 原因
报表没更新	Worker 跑 SQL 的机器炸了 / 连接失败
数据源链接断了	Worker 网络配置问题
某个任务总是失败	代码逻辑 or 环境依赖在 Worker 上缺失
日志为空 or 执行特别慢	Worker 本身CPU爆了 / 机器配置不足

Summary


Web UI ← 常用位置
Scheduler ← 控制节奏
DAG（依赖图） ← 描述流程
Task A
Task B ← 每个任务可能是SQL、脚本等
Worker 1
Worker 2 ← 真正执行的机器资源