Open Talk 杭州站

Flink 批处理实践

张召 又拍云 CDN 运维开发工程师

又拍云

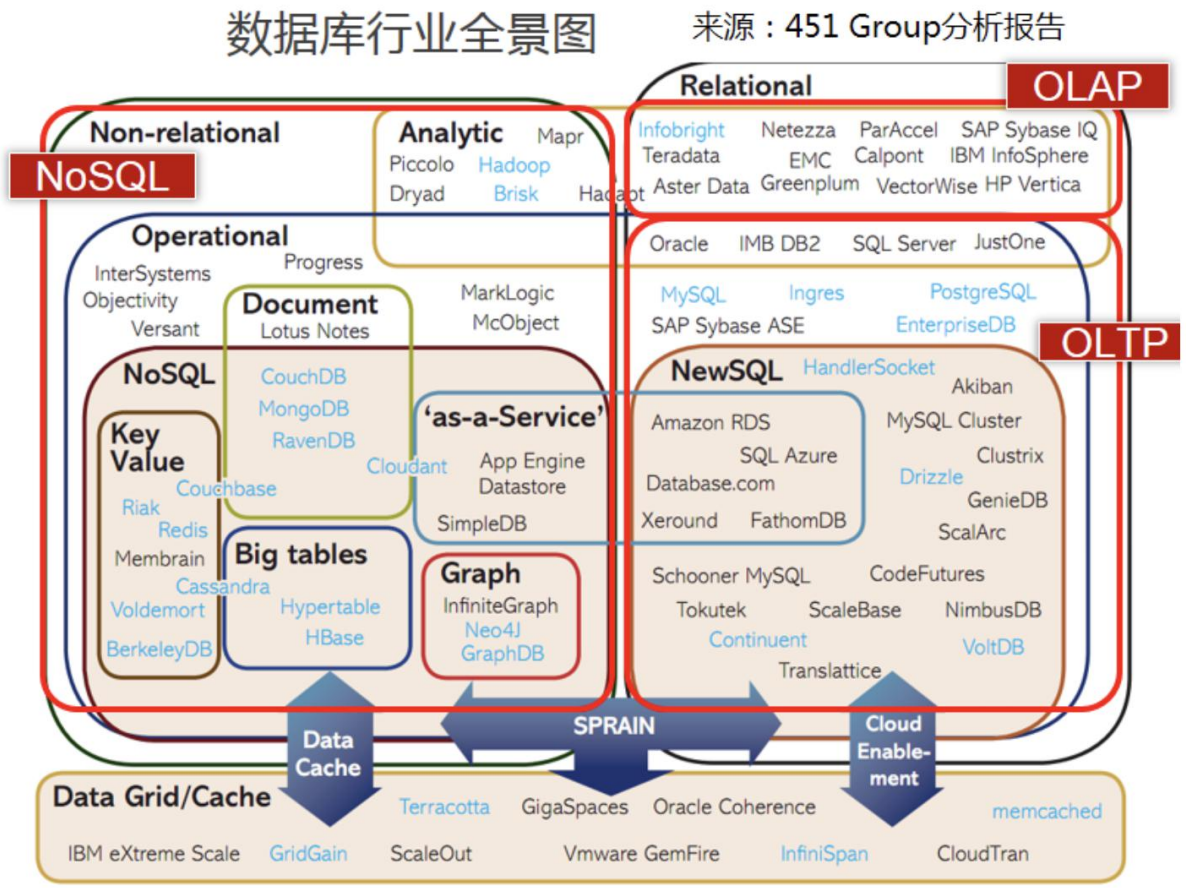
分享内容

- 为什么选择 Flink
- 关于 Flink 基础知识
- 使用 Flink 过程中遇到的问题

业务需求及痛点

- 日志的量比较大，每小时日志在百 G 左右
- 单机处理，速度慢，扩容不方便
- 代码抽象程度低，对时间排序，字段统计等需求重复编码
- 流程复杂，数据需要在多个服务间流转

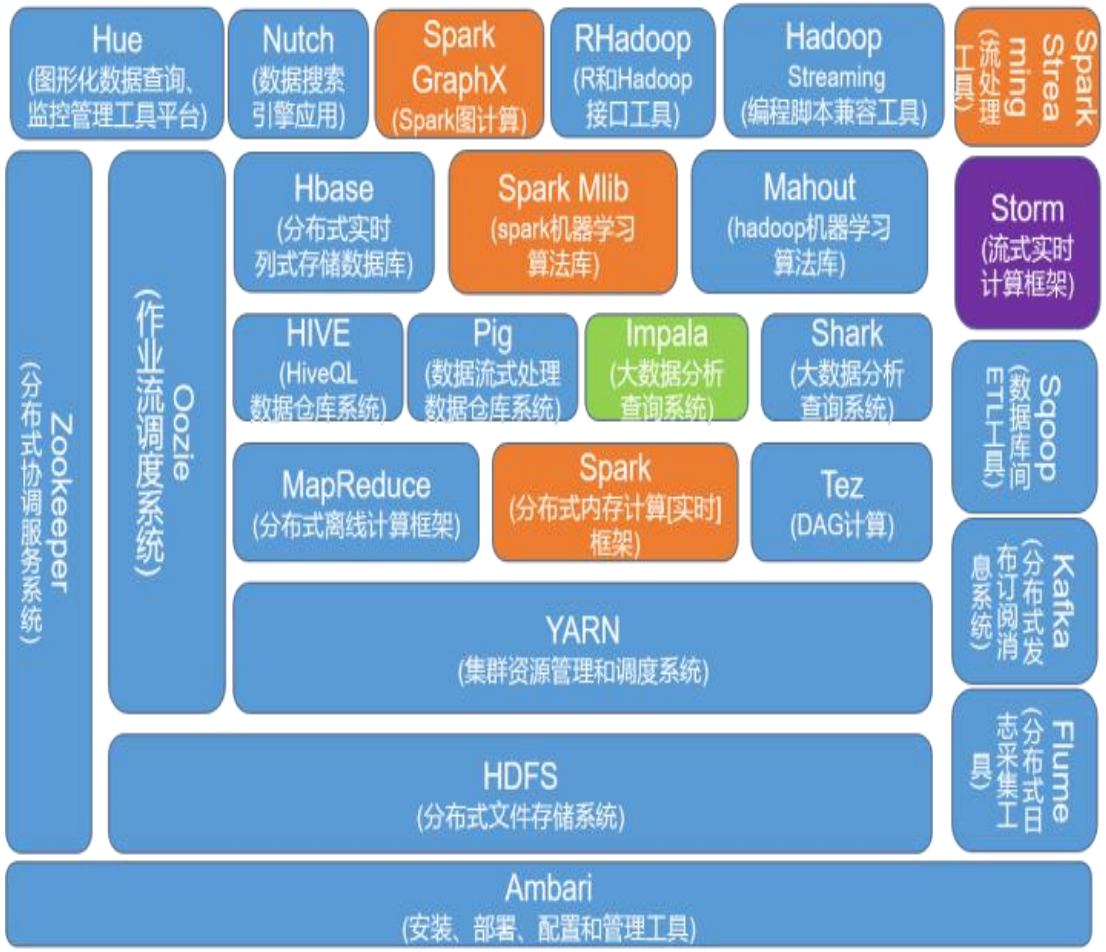
方案1: 我们需要一个数据库吗?



1. OLTP, Online Transaction Processing. OLTP 数据库最大的特点是支持事务, 增删查改等功能强大, 适合需要被频繁修改的"热数据".
2. OLAP, Online Analytical Processing, 数据分析为主. 不支持事务, 或对事务的支持有限.
3. NewSQL. ClickHouse, TiDB 等

小结：日志数据有什么特点?

方案2: Hadoop 生态圈



日志数据的特点:

数据库存储的数据都是二维的, 有行和列两个维度，但是日志只有行一个维度

MapReduce 没有落地的原因:

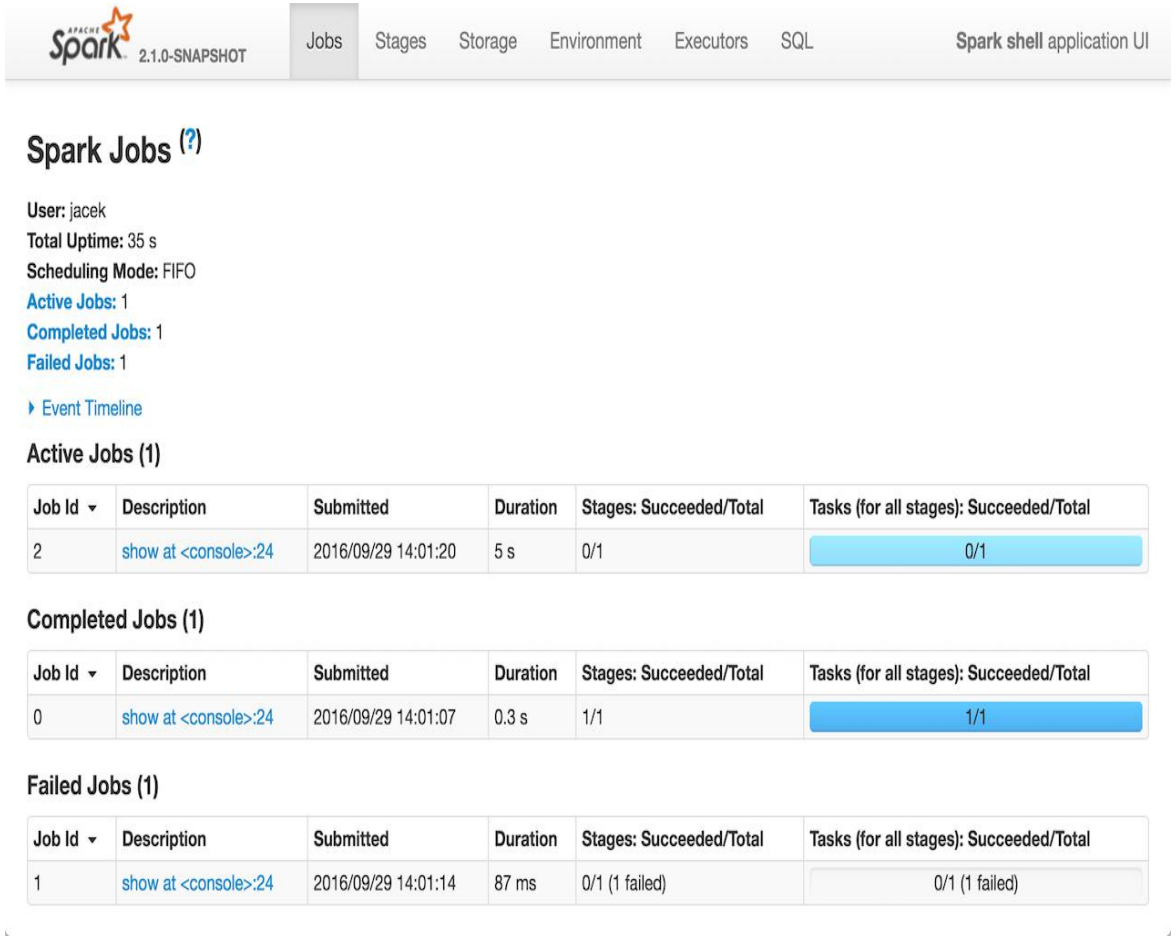
- 机器资源
- 存储分离是趋势

小结：计算引擎 + 云上部署

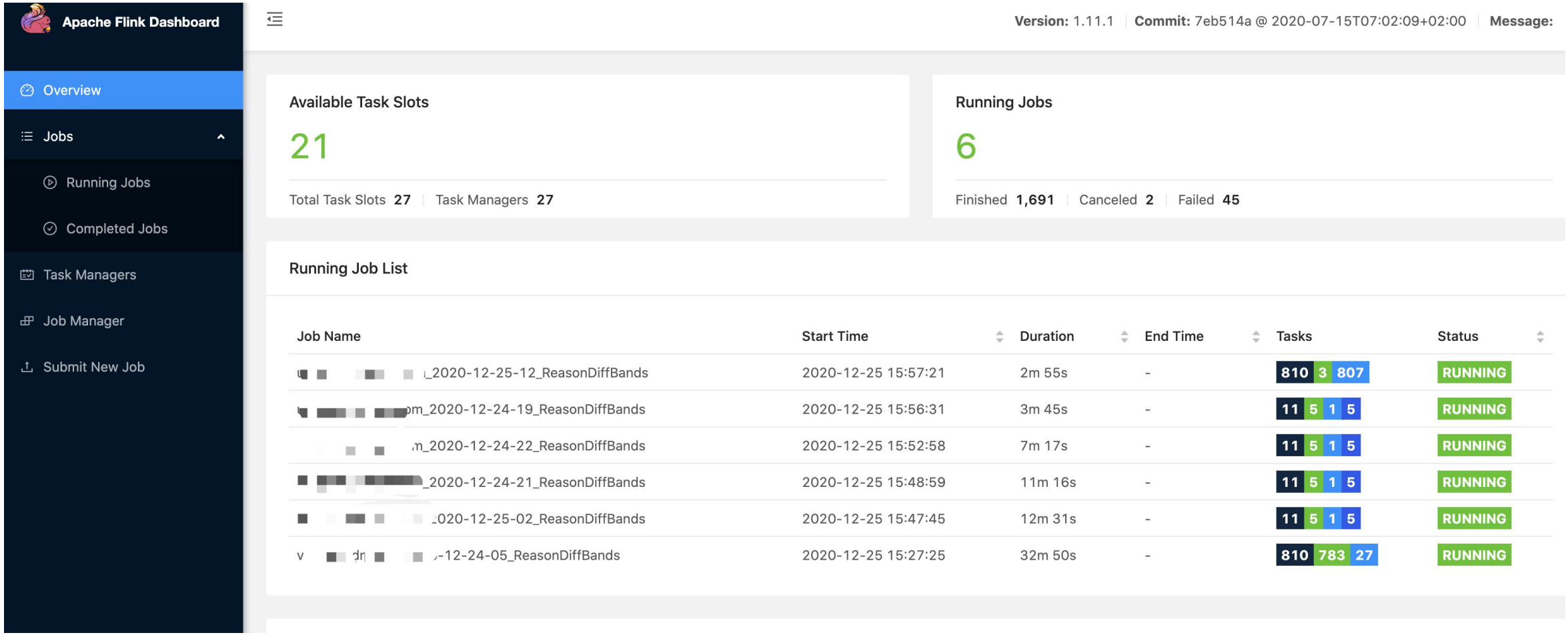
方案3：Spark

存在较多问题

- 提交任务：Restful API 接口
- Web 控制台：容器的虚拟 IP



Flink Web UI


Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Submit New Job
Version: 1.11.1 | Commit: 7eb514a @ 2020-07-15T07:02:09+02:00 | Message:
Available Task Slots
21
Total Task Slots 27 | Task Managers 27
Running Jobs
6
Finished 1,691 | Canceled 2 | Failed 45
Running Job List
Job Name
Start Time
Duration
End Time
Tasks
Status
_2020-12-25-12_ReasonDiffBands
2020-12-25 15:57:21
2m 55s
-
810 3 807
RUNNING
m_2020-12-24-19_ReasonDiffBands
2020-12-25 15:56:31
3m 45s
-
11 5 1 5
RUNNING
m_2020-12-24-22_ReasonDiffBands
2020-12-25 15:52:58
7m 17s
-
11 5 1 5
RUNNING
_2020-12-24-21_ReasonDiffBands
2020-12-25 15:48:59
11m 16s
-
11 5 1 5
RUNNING
020-12-25-02_ReasonDiffBands
2020-12-25 15:47:45
12m 31s
-
11 5 1 5
RUNNING
-12-24-05_ReasonDiffBands
2020-12-25 15:27:25
32m 50s
-
810 783 27
RUNNING

小结：为什么选择 Flink

Flink 相关知识介绍

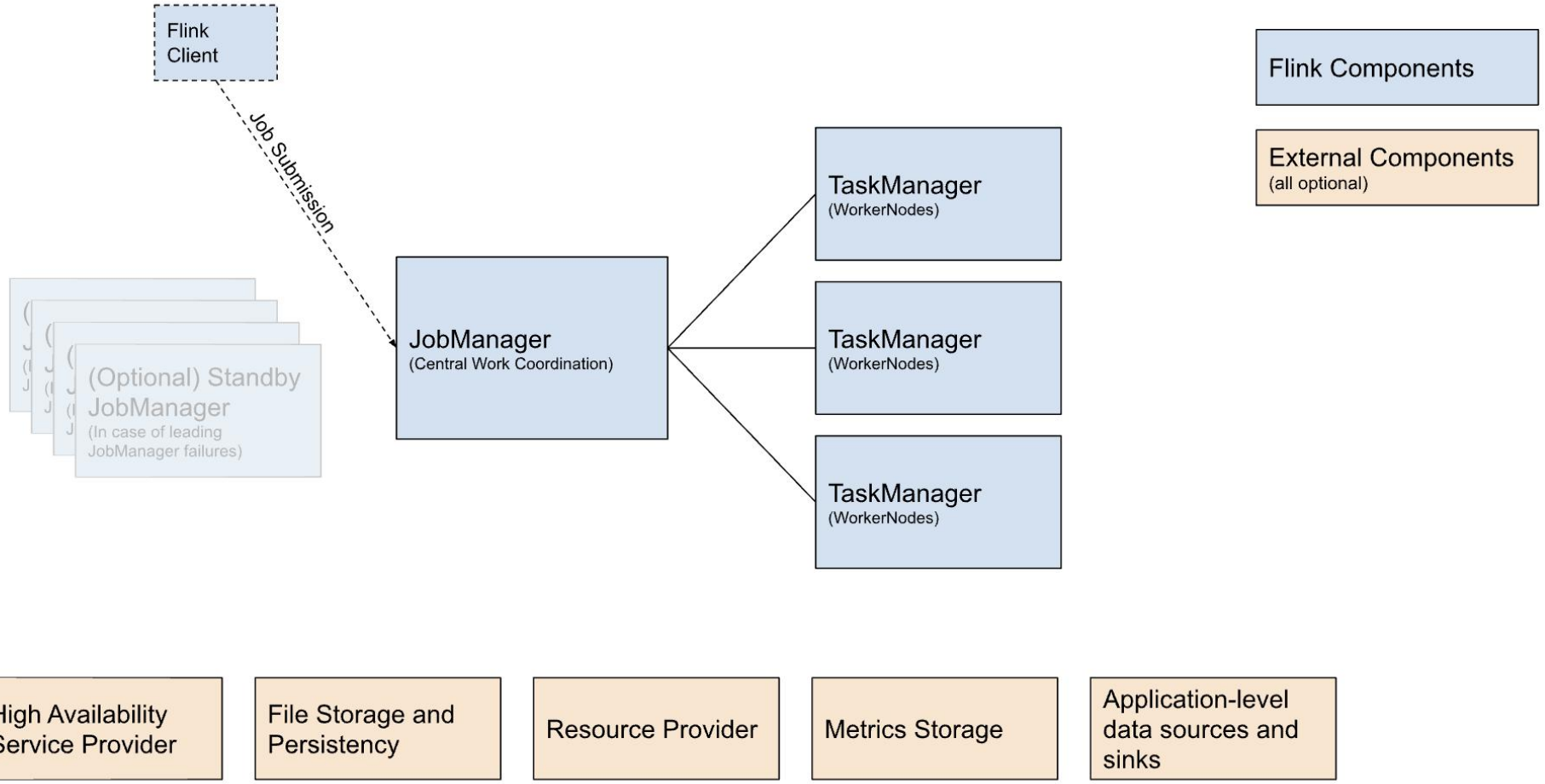
Flink 组件栈

JobManager 和 TaskManager

Flink 提交任务

Flink 部署

Flink 组件栈


Flink
Client
Job Submission
JobManager
(Central Work Coordination)
(Optional) Standby
JobManager
(In case of leading
JobManager failures)
TaskManager
(WorkerNodes)
TaskManager
(WorkerNodes)
TaskManager
(WorkerNodes)
Flink Components
External Components
(all optional)
High Availability
Service Provider
File Storage and
Persistency
Resource Provider
Metrics Storage
Application-level
data sources and
sinks

Flink 组件栈

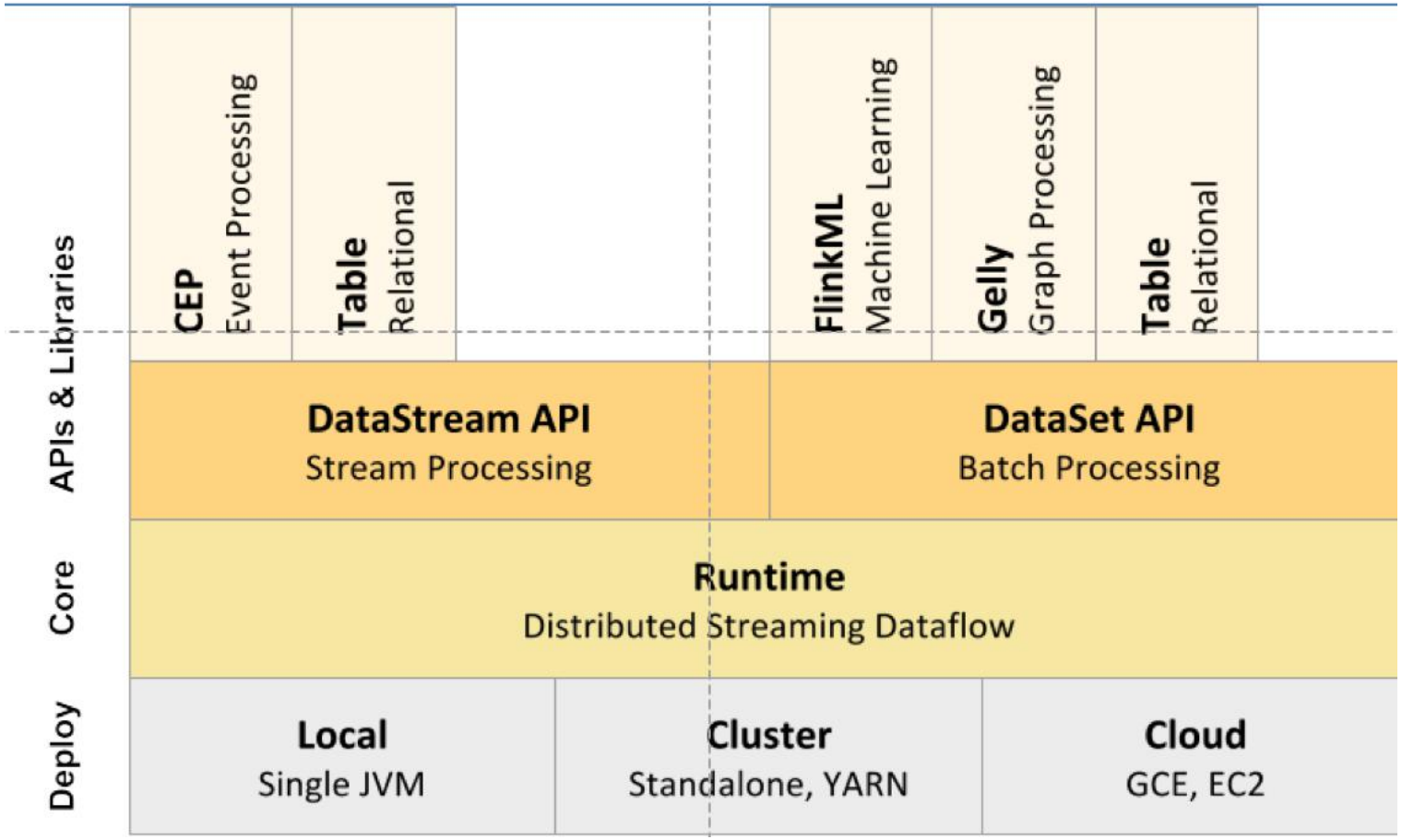
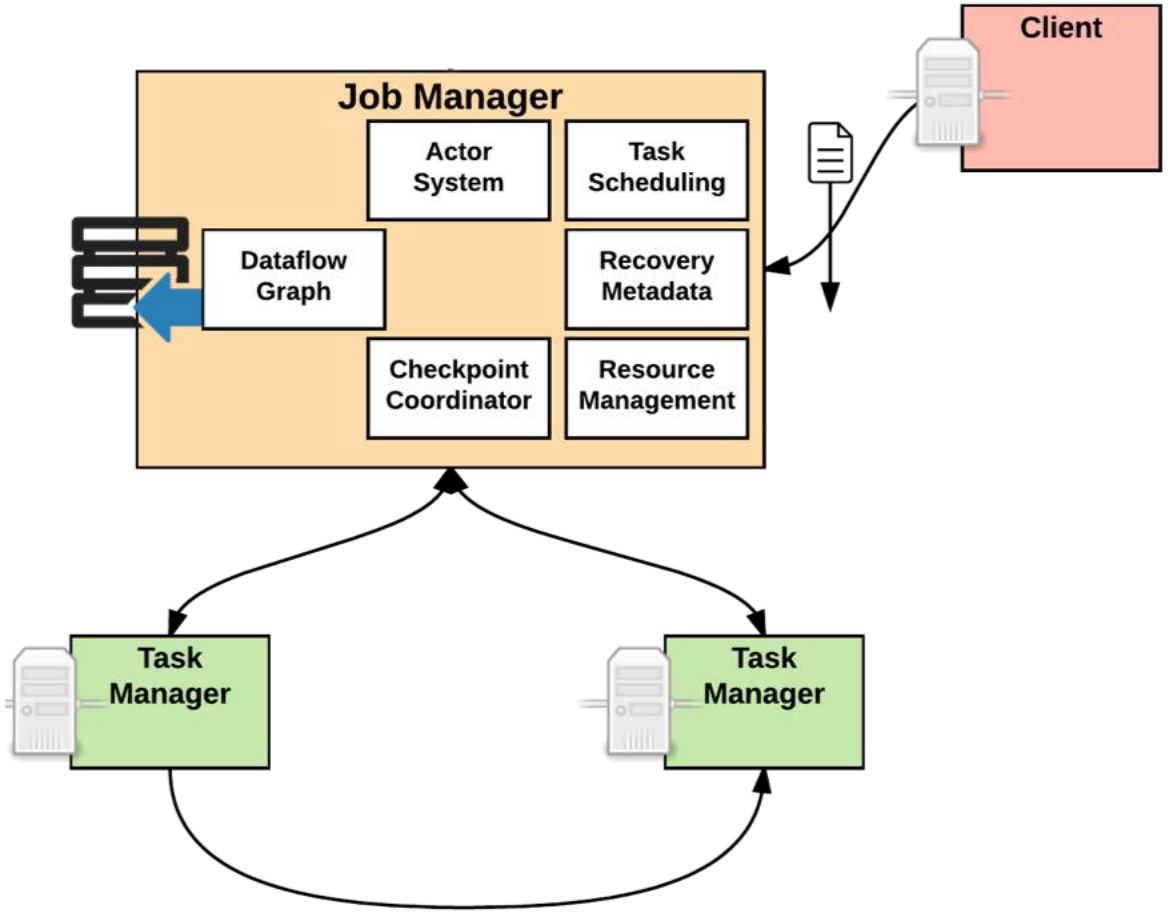


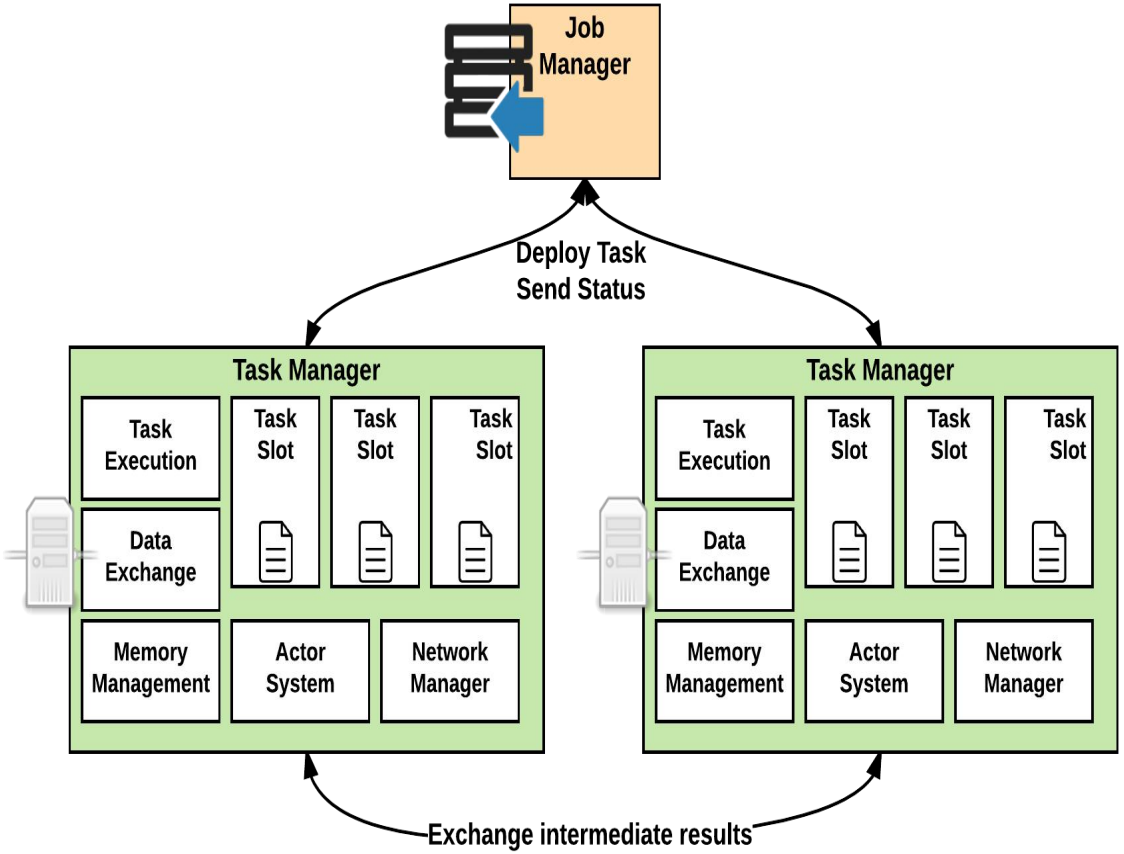
Table 等 API 不稳定, bug 多, 很多功能尚未实现

流处理和批处理在 API 层是分开的；在 Flink 1.12.0 中, API 层开始合并

实现了 UpyunStoreOutputFormat 等功能, 上传到又拍云存储

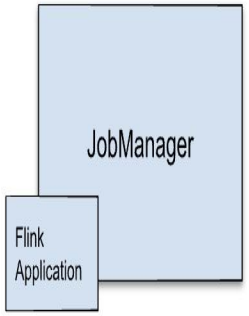
JobManager 和 TaskManager





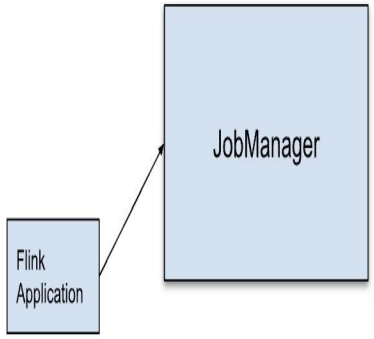
Flink 部署

Application Mode



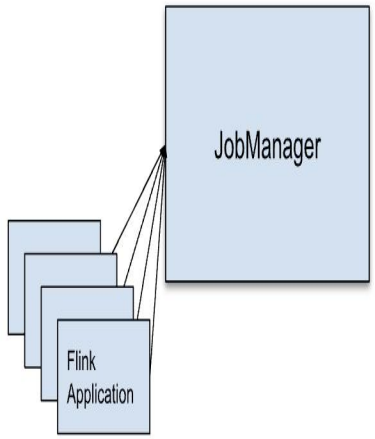
A dedicated JobManager is started for submitting the job. The JobManager will only execute this job, then exit.
The Flink Application runs on the JobManager.

Per-Job Mode



A dedicated JobManager is started for submitting the job. The JobManager will only execute this job, then exit.
The Flink Application runs on the client submitting the per-job cluster

Session Mode

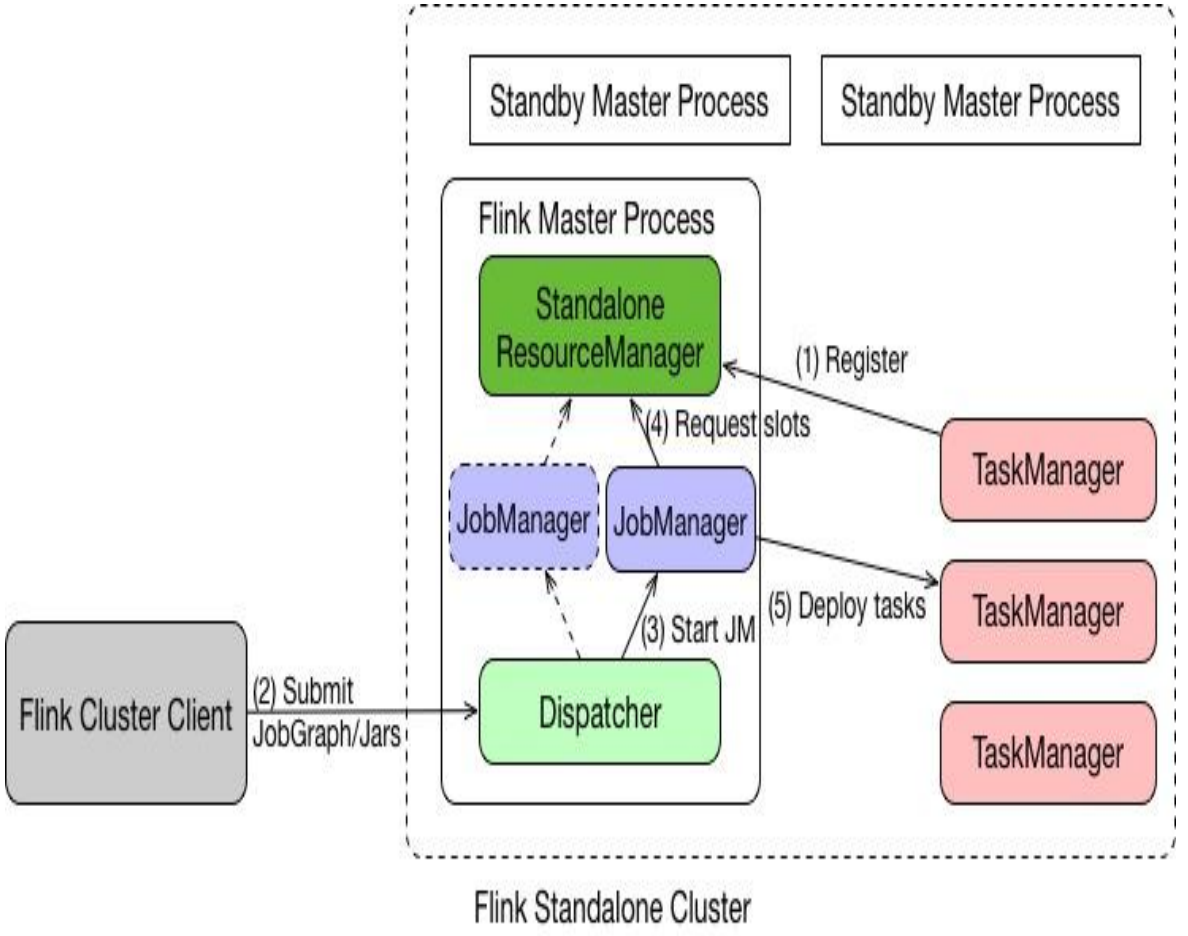


Multiple jobs share one JobManager.

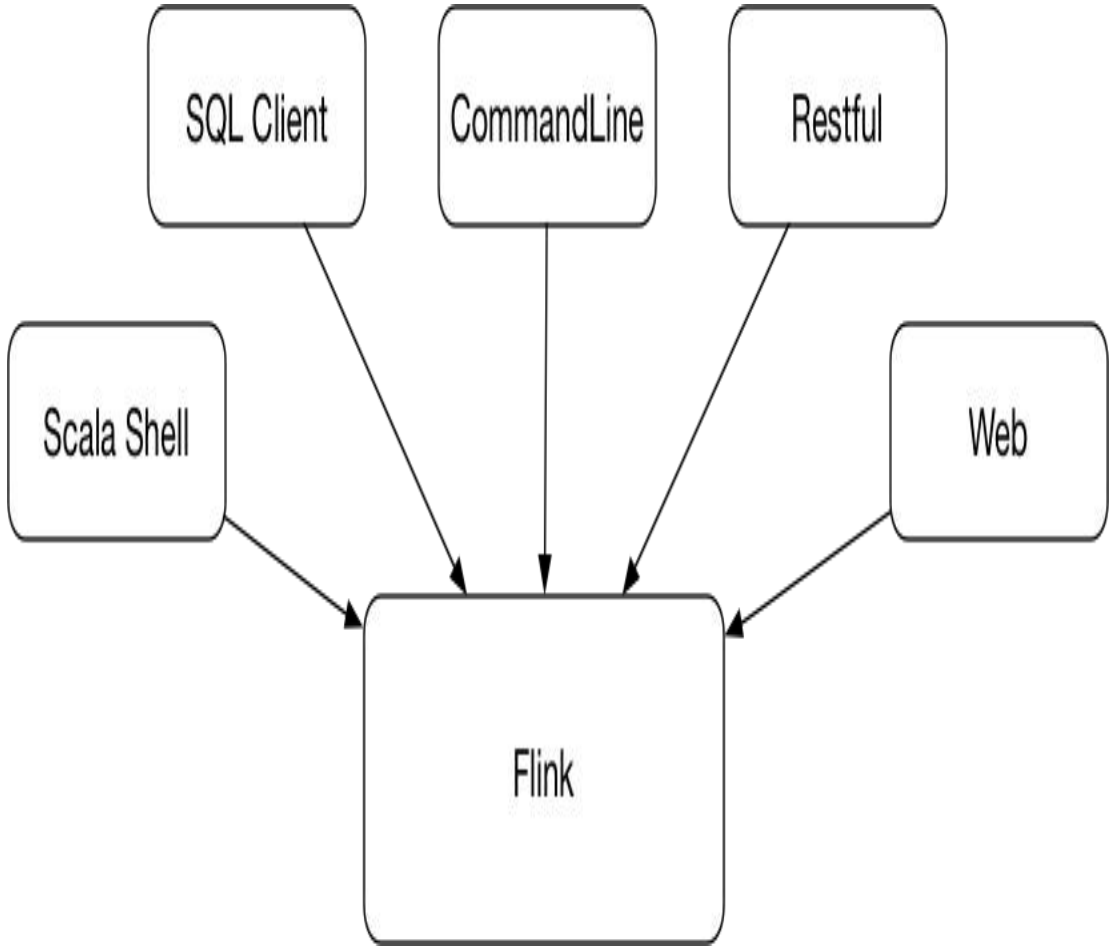
Component	Purpose	Implementations
Flink Client	Compiles batch or streaming applications into a dataflow graph, which it then submits to the JobManager.	• Command Line Interface • REST Endpoint • SQL Client • Python REPL • Scala REPL
JobManager	JobManager is the name of the central work coordination component of Flink. It has implementations for different resource providers, which differ on high-availability, resource allocation behavior and supported job submission modes. JobManager modes for job submissions: • **Application Mode**: runs the cluster exclusively for one application. The job's main method (or client) gets executed on the JobManager. Calling `execute`/`executeAsync` multiple times in an application is supported. • **Per-Job Mode**: runs the cluster exclusively for one job. The job's main method (or client) runs only prior to the cluster creation. • **Session Mode**: one JobManager instance manages multiple jobs sharing the same cluster of TaskManagers	• Standalone (this is the barebone mode that requires just JVMs to be launched. Deployment with Docker, Docker Swarm / Compose, non-native Kubernetes and other models is possible through manual setup in this mode) • Kubernetes • YARN • Mesos
TaskManager	TaskManagers are the services actually performing the work of a Flink job.	

Standalone on Docker 模式

- Standalone 模式下，Master 和 TaskManager 可以运行在同一台机器或者不同的机器上。
- Master 进程中，Standalone ResourceManager 的作用是对资源进行管理。当用户通过 Flink Cluster Client 将 JobGraph 提交给 Master 时，JobGraph 先经过 Dispatcher。
- 当 Dispatcher 收到请求，生成 JobManager。接着 JobManager 进程向 Standalone ResourceManager 申请资源，最终再启动 TaskManager。
- TaskManager 启动后，经历注册后 JobManager 将具体的 Task 任务分发给 TaskManager 去执行。



Flink 提交任务

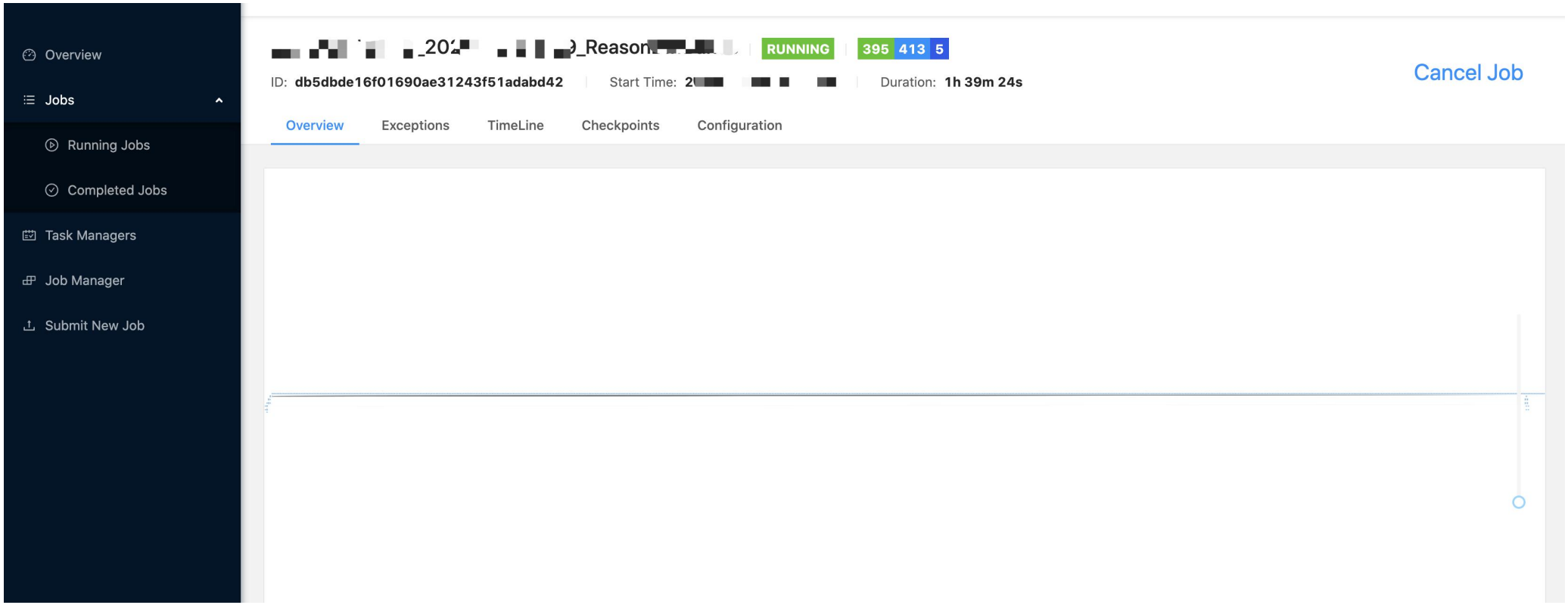


Flink 提供丰富的客户端操作提交任务和与任务进行交互，包括 Flink 命令行，Scala Shell，SQL Client，**Restful API** 和 Web

最重要的是命令行，其次是 SQL Client 用于提交 SQL 任务的运行，以及 Scala Shell 提交 Table API 的任务

还提供 Restful 服务，可以通过 http 方式进行调用。此外，还有 Web 的方式可以提交任务。

批处理场景下的网络问题



```
43      ... 1 more
44  Caused by: akka.pattern.AskTimeoutException: Ask timed out on [Actor[akka.tcp://flink@10.244.77.41:40807/user/rpc/taskmanager_0#1377840888]] after [30000 ms]. Message of type
    [org.apache.flink.runtime.rpc.messages.RemoteRpcInvocation]. A typical reason for `AskTimeoutException` is that the recipient actor didn't send a reply.
45      at akka.pattern.PromiseActorRef$$anonfun$2.apply(AskSupport.scala:635)
46      at akka.pattern.PromiseActorRef$$anonfun$2.apply(AskSupport.scala:635)
47      at akka.pattern.PromiseActorRef$$anonfun$1.apply$mcV$sp(AskSupport.scala:648)
48      at akka.actor.Scheduler$$anon$4.run(Scheduler.scala:205)
49      at scala.concurrent.Future$InternalCallbackExecutor$.unbatchedExecute(Future.scala:601)
50      at scala.concurrent.BatchingExecutor$class.execute(BatchingExecutor.scala:109)
51      at scala.concurrent.Future$InternalCallbackExecutor$.execute(Future.scala:599)
52      at akka.actor.LightArrayRevolverScheduler$TaskHolder.executeTask(LightArrayRevolverScheduler.scala:328)
53      at akka.actor.LightArrayRevolverScheduler$$anon$4.executeBucket$1(LightArrayRevolverScheduler.scala:279)
54      at akka.actor.LightArrayRevolverScheduler$$anon$4.nextTick(LightArrayRevolverScheduler.scala:283)
55      at akka.actor.LightArrayRevolverScheduler$$anon$4.run(LightArrayRevolverScheduler.scala:235)
56      ... 1 more
57
```

java.util.concurrent.TimeoutException: Heartbeat of TaskManager with id 0baf4590112de86da3d4dddb8943850c timed out.

Connection reset by peer

批处理场景下的网络问题

Flink on K8s 实践经验总结：我们都踩过哪些坑？

缓解网络问题

- 调大网络容错性, 也就是配置参数中 timeout 相关的部分
- 开启压缩
- 利用缓存, 如 `taskmanager.memory.network.fraction` 等
- 减少单个 task manager 下 task slots 的数量

Connection reset by peer

- 不要有异构网络环境（尽量不要跨机房访问）
- 云服务商的机器配置网卡多队列 (将实例中的网络中断分散给不同的CPU处理，从而提升性能)
- 选取云服务商提供的高性能网络插件：例如阿里云的 Terway
- Host network，绕开 K8s 的虚拟化网络（需要一定的开发量）

资源浪费的问题

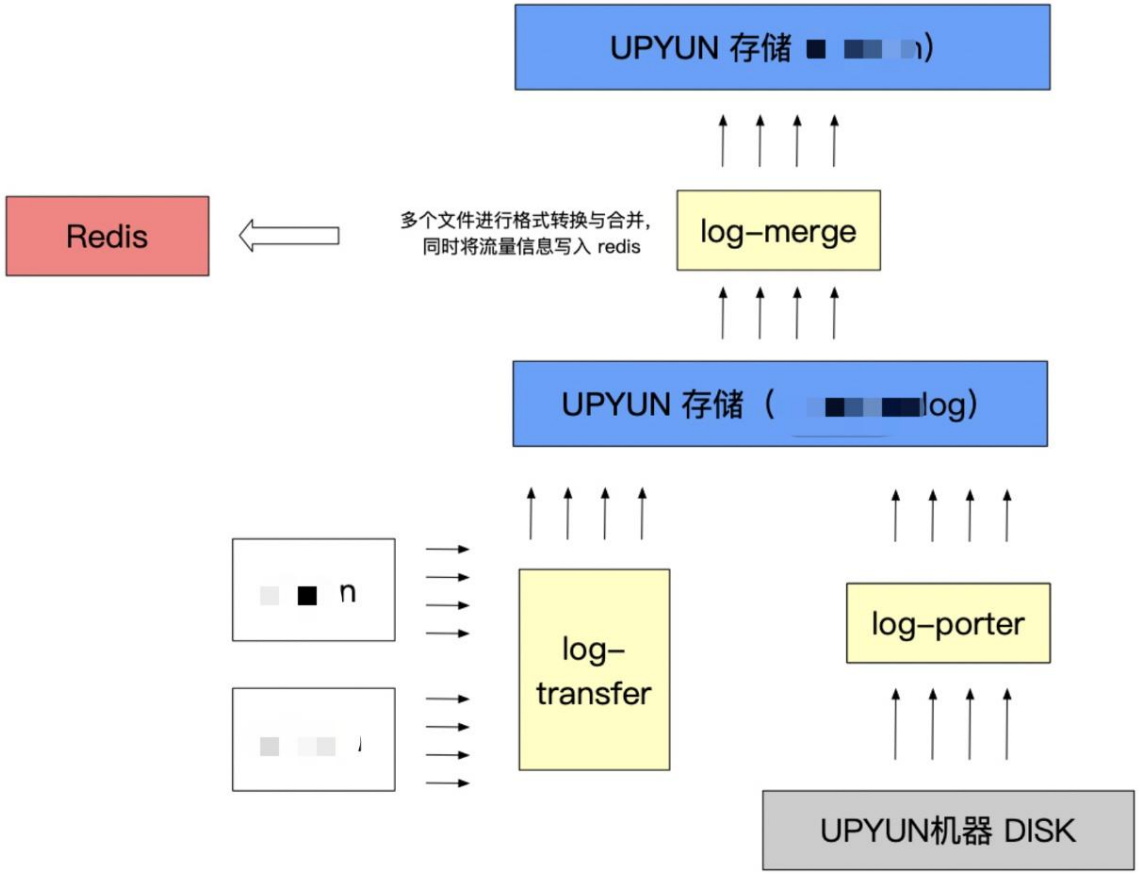
DataSource (at com.upai.jobs.FusionGifShow$$anonfun$2.apply...	FINISHED	0 B	0	160 MB	201,102	1	2020-12 1
DataSource (at com.upai.jobs.FusionGifShow$$anonfun$2.apply...	FINISHED	0 B	0	163 MB	205,240	1	2020-12 1
CHAIN Map (Map at com.upai.jobs.FusionGifShow$.handleLogSo...	RUNNING	149 GB	191,603,272	264 GB	191,603,267	1	2020-12 1
CHAIN Map (from: (remoteAddr, _1, remoteUser, timeLocal, meth...	RUNNING	132 GB	191,603,273	0 B	13	1	2020-12 1
CHAIN GroupReduce (groupBy: (EXPR$1), select: (EXPR$1, SU...	RUNNING	22 B	0	0 B	0	1	2020-12 1
Map (Map at com.upai.jobs.FusionGifShow$.mapRedisData(Fusi...	CREATED	◡	◡	◡	◡	1	- 1
DataSink (com.upai.jobs.format.RedisOutputFormat@2d70764c)	CREATED	◡	◡	◡	◡	1	- 1
CHAIN Map (Map at com.upai.jobs.FusionGifShow$.handleLogSo...	FINISHED	24.4 GB	31,408,111	43.2 GB	31,408,111	1	2020-12 1
CHAIN Map (from: (remoteAddr, _1, remoteUser, timeLocal, meth...	FINISHED	21.6 GB	31,408,111	0 B	12	1	2020-12 1
CHAIN GroupReduce (groupBy: (EXPR$1), select: (EXPR$1, SU...	FINISHED	268 B	12	0 B	12	1	2020-12 1
Map (Map at com.upai.jobs.FusionGifShow$.mapRedisData(Fusi...	FINISHED	244 B	12	0 B	12	1	2020-12 1
DataSink (com.upai.jobs.format.RedisOutputFormat@1e8710f4)	FINISHED	676 B	12	0 B	0	1	2020-12 1
Map (Map at com.upai.jobs.FusionGifShow$.run(FusionGifShow....	RUNNING	156 GB	225,930,370	150 GB	225,930,386	1	2020-12 1

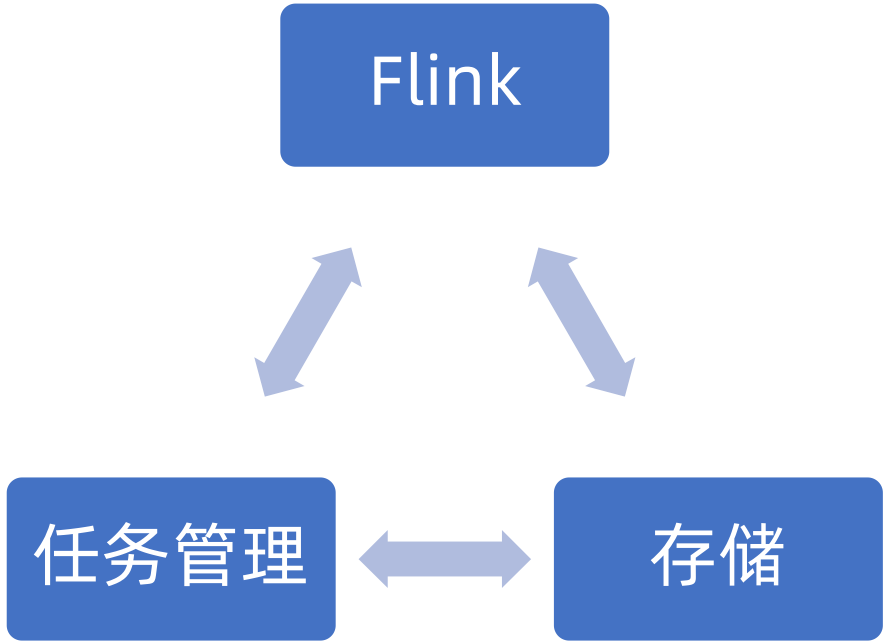
job 卡死

Running Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
[redacted]om	2020-12-22 14:59:38	1m 9s	-	12 6 1 5	RUNNING
[redacted]com	2020-12-22 14:57:35	3m 13s	-	12 2 5 5	RUNNING
[redacted]m	2020-12-22 14:44:19	16m 28s	-	1606 877 45 684	RUNNING
mergeComm	2020-12-22 14:29:42	31m 5s	-	16 2 14	RUNNING
mergeComm	2020-12-22 14:24:15	36m 32s	-	15 2 13	RUNNING
mergeComm	2020-12-22 14:15:31	45m 16s	-	16 2 14	RUNNING
mergeComm	2020-12-22 14:07:49	52m 59s	-	16 2 14	RUNNING
mergeComm	2020-12-22 14:01:39	59m 8s	-	16 2 14	RUNNING
mergeComm	2020-12-22 13:52:43	1h 8m 4s	-	16 2 14	RUNNING
[redacted]m	2020-12-22 11:57:41	3h 3m 7s	-	1606 808 368 430	RUNNING

引入 Flink 带来的收益



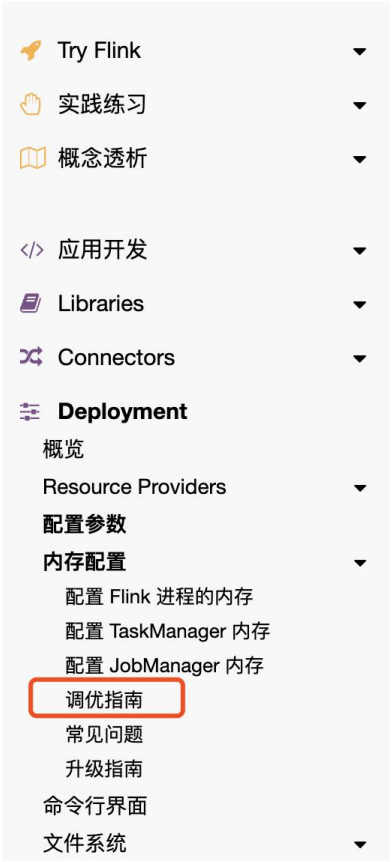


总结

以 Standalone on Docker 部署的 Flink，进行百G 以上的批处理任务是非常有挑战性的

这不是 Flink 的典型应用场景，很多配置也做不到 “开箱即用”，需要经验去调优

但是目前 Flink 的发展方向是 “流批一体化”，所以对 Flink 今后的发展还是非常有信心的



关注又拍云微信公众号，

获取更多干货！

THANKS！

又拍云