Flink架构

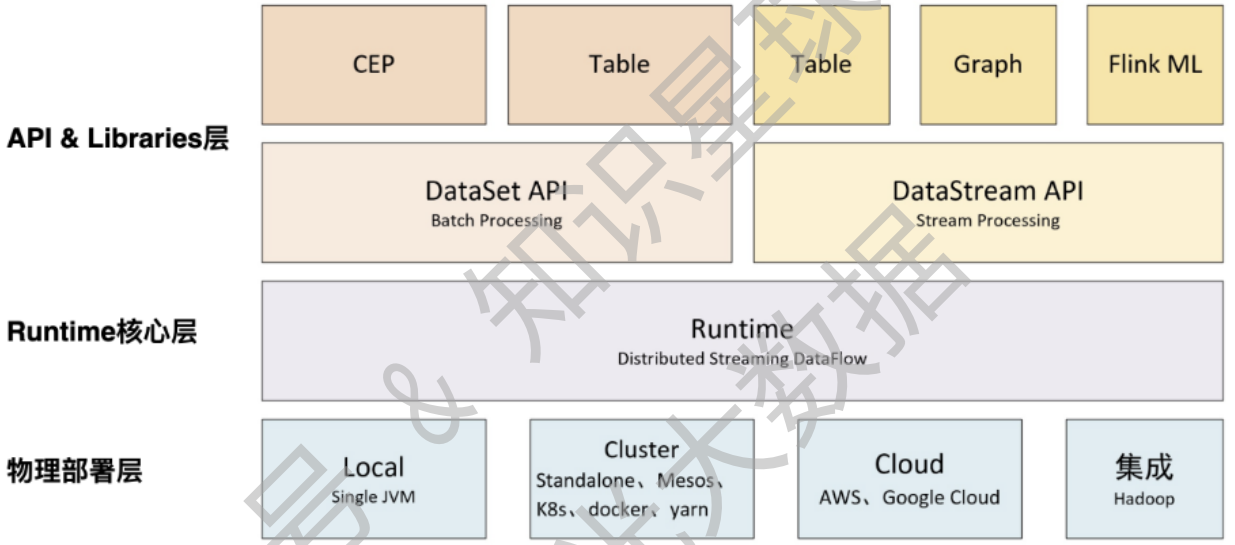
可回答：1）Flink组件；2）Flink的分层（回答Flink组件架构）

问过的一些公司：字节实习(2021.03)，有道(2021.03)，字节日常实习(2020.11)

参考答案：

1、Flink组件架构

在Flink整个软件架构体系中，遵循了分层的架构设计理念，在降低系统耦合度的同时也为上层用户构建Flink应用提供了丰富且友好的接口。



Flink的架构体系基本上可以分为以下三层：

1）API & Libraries层

作为分布式数据处理框架，**Flink同时提供了支持流计算和批计算的接口，同时在此基础之上抽象出不同的应用类型的组件库**，如基于流处理的CEP（复杂事件处理库）、SQL&Table库和基于批处理的FlinkML（机器学习库）、Gelly（图处理库）等。

API层包括构建流计算应用的DataStream API和批计算应用的DataSet API，两者都提供给用户丰富的数据处理高级API，例如Map、FlatMap操作等，同时也提供比较低级的Process Function API，用户可以直接操作状态和时间等底层数据。

2）Runtime核心层

该层主要负责对上层不同接口提供基础服务，也是Flink分布式计算框架的核心实现层，支持分布式Stream作业的执行、JobGraph到ExecutionGraph的映射转换、任务调度等。

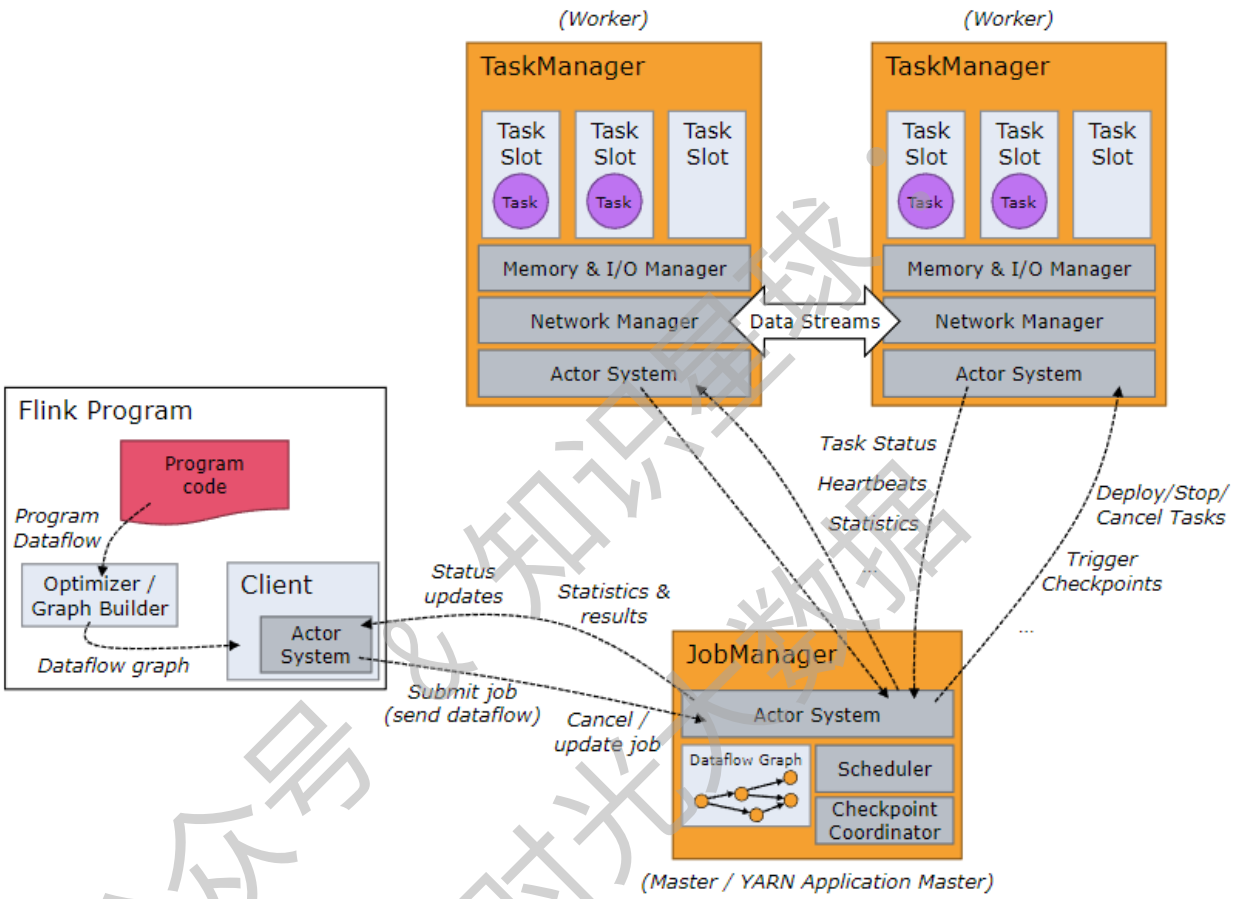
3）物理部署层

该层主要涉及Flink的部署模式，目前Flink支持多种部署模式：本地、集群（Standalone/YARN）、云（GCE/EC2）、Kubenetes。Flink能够通过该层能够支持不同的部署，用户可以根据需要选择使用对应的部署模式。

2、Flink运行时架构

Flink运行时系统主要由以下四个组件组成：

- JobManager（任务管理器）
- TaskManager（作业管理器）
- ResourceManger（资源管理器）
- Dispatcher（分发器）



可以通过多种方式启动 JobManager 和 TaskManager：直接在机器上作为standalone集群启动、在容器中启动、或者通过YARN等资源框架管理并启动。TaskManager 连接到 JobManagers，宣布自己可用，并被分配工作。

当 Flink 集群启动后，首先会启动一个 JobManger 和一个或多个的 TaskManager。由 Client 提交任务给 JobManager，JobManager 再调度任务到各个 TaskManager 去执行，然后 TaskManager 将心跳和统计信息汇报给 JobManager。TaskManager 之间以流的形式进行数据的传输。上述三者均为独立的 JVM 进程。

`Client` 不是运行时和程序执行的一部分，而是用于准备数据流并将其发送给 JobManager。之后，客户端可以断开连接（分离模式），或保持连接来接收进程报告（附加模式）。客户端可以作为触发执行 Java/Scala 程序的一部分运行，也可以在命令行进程./bin/flink run ...中运行。

`JobManager`

Flink遵循Master-Slave（主从）架构设计原则，JobManager为Master节点，TaskManager为Slave节点，并且所有组件之间的通信都借助Akka，包括任务的状态以及CheckPoint（检查点）触发等信息。

- 作为主进程（Master Process），JobManager控制着单个应用程序的执行，也就是每个应用都由一个不同的JobManager管理。
- JobManager可以接受需要执行的应用，该应用会包含一个所谓的Job Graph（任务图），即逻辑Dataflow Graph（数据流图），以及一个打包了全部所需类、库以及其他资源的JAR文件。

- JobManager将JobGraph转化为名为Execution Graph（执行图）的物理Dataflow Graph，其中包含了所有可以并发实行的任务。
- JobManager会从ResourceManager申请执行任务的必要资源——TaskManager slot，一旦它收到了足够数量的TaskManager slot，它就会将Execution Graph中的任务分发给TaskManager来执行。在执行过程中，JobManager还要负责所有需要集中协调的操作，如创建CheakPoint等。

`TaskManager`

- TaskManager是Flink的工作进程（Worker Process），在Flink的搭建过程中要启动多个TaskManager。每个TaskManager提供一定数量的slot（处理槽），slot的数量限制了TaskManager可执行的任务数。
- TaskManager在启动之后会向ResourceManager注册它的slot，当接收到ResourceManager的指示时，TaskManager会向JobManager提供一个或者多个slot。之后JobManager就可以向slot中分配任务来执行。
- 在执行过程中，运行同一应用的不同任务的TaskManager之间会产生数据交换。

`ResourceManger`

- Flink为不同的环境和资源提供者（如YARN、Kubernetes、Stand-alone）提供了不同的ResourceManger。
- ResourceManger负责管理Flink的处理资源单元——TaskManager Slot。
- 当JobManager申请TaskManager slot时，ResourceManger会指示一个拥有空闲slot的TaskManager将其slot提供给JobManager。如果ResourceManger的slot数无法满足JobManager的请求，则ResourceManger可以与资源提供者通信，让他们提供额外的容器来启动更多的TaskManager进程。同时，ResourceManger还负责终止空闲进程的TaskManager以释放计算资源。

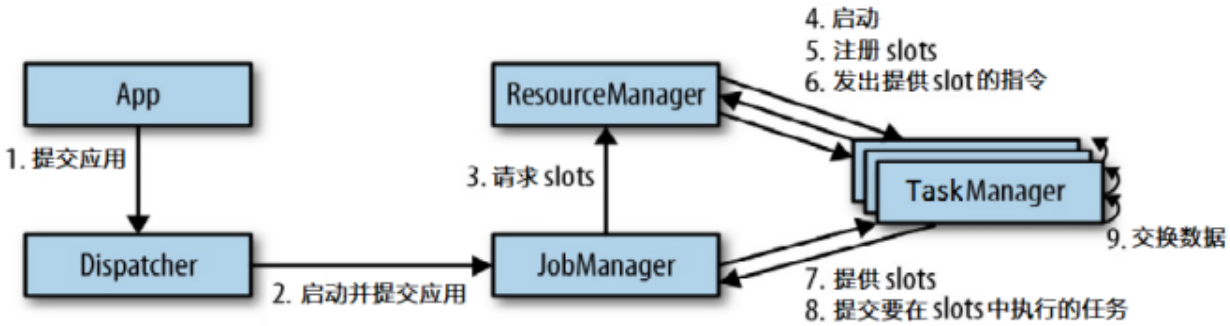
`Dispatcher`

- Dispatcher在会跨多个作业运行，它提供了一个REST接口来让我们提交需要执行的应用，一旦某个应用提交执行，则Dispatcher会启动一个JobManager并将应用转交给它。
- REST接口意味着Dispatcher这一集群的HTTP入口可以受到防火墙的保护。
- Dispatcher同时还会启动一个Web UI，用来展示和监控有关作业执行的信息。
- Dispatcher并不是必需的组件，某些应用提交执行的方式可能用不到Dispatcher。

Flink官网：JobManager进程中包括ResourceManager、Dispatcher、JobMaster三个组件。

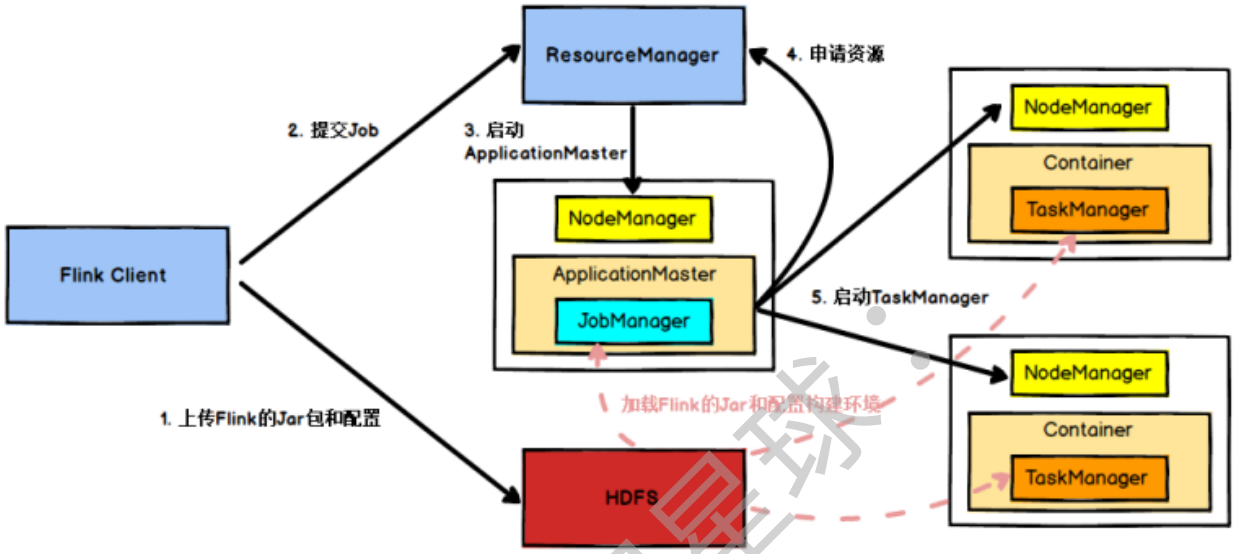
3、任务提交流程

我们来看看当一个应用提交执行时，Flink的各个组件是如何交互协作的：



上图是从一个较为高层级的视角，来看应用中各组件的交互协作。如果部署的集群环境不同（例如YARN，Mesos，Kubernetes，standalone等），其中一些步骤可以被省略，或是有些组件会运行在同一个JVM进程中。

具体地，如果我们将Flink集群部署到YARN上，那么就会有如下的提交流程：



Flink任务提交后，Client向HDFS上传Flink的Jar包和配置，之后向Yarn ResourceManager提交任务，ResourceManager分配Container资源并通知对应的NodeManager启动ApplicationMaster，ApplicationMaster启动后加载Flink的Jar包和配置构建环境，然后启动JobManager，之后ApplicationMaster向ResourceManager申请资源启动TaskManager，ResourceManager分配Container资源后，由ApplicationMaster通知资源所在节点的NodeManager启动TaskManager，NodeManager加载Flink的Jar包和配置构建环境并启动TaskManager，TaskManager启动后向JobManager发送心跳包，并等待JobManager向其分配任务。