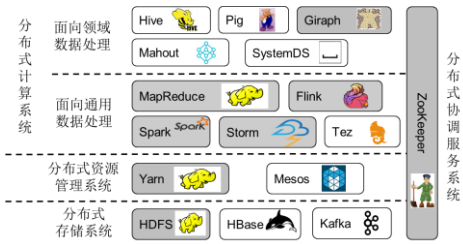
第五章　资源管理系统Yarn

徐 辰
cxu@dase.ecnu.edu.cn
華東師範大學

分布式计算系统生态圈



Hadoop 2.0

MapReduce:
是google MapReduce的开源实现。是对并行计算的封装，使用户通过一些简单的逻辑即可完成复杂的并行计算。将一个大的运算任务分解到集群每个节点上，充分运用集群资源，缩短运行时间。

Map/Reduce
Yarn
DFS

Yarn:
Yet Another Resource Negotiator(另一种资源协商)，是一个资源调度系统

DFS: 分布式文件系统，支持HDFS，S3

Yarn发展历史

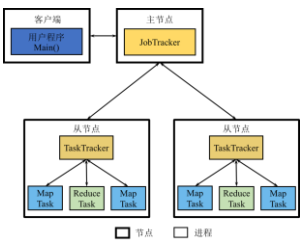
- 2010年，雅虎的工程师开始考虑MapReduce的新架构
- 2012年8月，Yarn成为Apache Hadoop的一个子项目

大纲

- 设计思想
  - 作业与资源管理
  - 平台与框架
- 体系架构
- 工作原理
- 容错机制
- 典型示例

MapReduce 1.0的JobTracker

- 作业管理：状态监控、信息汇总、任务调度等
- 资源管理：



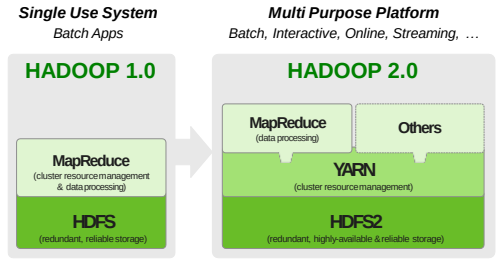
MapReduce 1.0的缺陷

7

- 资源管理与作业紧密耦合
  - 资源管理不单是MapReduce系统所需要的，而是通用的
- 作业控制管理高度集中
  - JobTracker需要维护所有作业的元信息
  - 若因某一作业运行过程中的错误信息造成JobTracker进程故障，则也会导致系统中所有作业发生故障

Hadoop as Next-Gen Platform

8



大纲

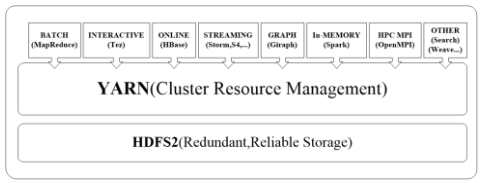
9

- 设计思想
  - 作业与资源管理
  - 平台与框架
- 体系架构
- 工作原理
- 容错机制
- 典型示例

平台 vs. 框架

10

- 系统
  - 平台（Platform）：具有提供资源功能的系统
  - 框架（Framework）：运行在平台上的系统



Yarn应用

11

- Yarn管理的粒度是应用
  - 但并不一定就是框架中的应用
  - 运行在Yarn这个平台上的框架可以将应用或作业映射为Yarn的应用
- Spark: application = 一个或多个Job

Yarn	Spark	MapReduce
应用	Application	Job

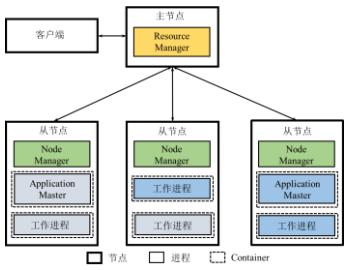
大纲

12

- 设计思想
- 体系架构
  - 架构图
  - 应用程序执行流程
- 工作原理
- 容错机制
- 典型应用

架构图

13



ResourceManager

14

- 资源管理器：负责整个系统的资源管理和分配
  - 资源调度器(Resource Scheduler)：分配Container并进行资源调度
  - 应用程序管理器(Application Manager)：管理整个系统中运行的所有应用
    - 应用程序提交
    - 与调度器协商资源以启动ApplicationMaster
    - 监控ApplicationMaster运行状态

NodeManager

15

- 节点管理器：负责每个节点资源和任务管理
  - 定时地向RM汇报本节点的资源使用情况和Container运行状态
  - 接受并处理来自AM的Container启动/停止等各种请求

ApplicationMaster

16

- 当用户基于Yarn平台提交一个框架应用，Yarn均启动一个 AM用于管理该应用
  - AM与RM调度器协商以获取资源（以Container表示），将获取的资源进一步分配给作业内部的任务
  - AM与NM通信以启动/停止任务，监控所有任务运行状态， 并在任务发生故障时重新申请资源来重启任务

Container

17

- Container是资源的抽象表示，包含CPU、内存等资源，是一个动态资源划分单位
- 当AM向RM申请资源时，RM向AM返回以Container表示的资源

YARN: Yet Another Resource Negotiator

18

- 资源管理与计算相分离
  - MapReduce 1.0既是计算系统，也是资源管理系统
  - Yarn是独立出来的**资源管理系统**

MapReduce 1.0	功能	Yarn
JobTracker TaskTracker	资源管理	ResourceManager
	作业管理	ApplicationMaster
	节点管理	NodeManager
Task	执行计算	Container

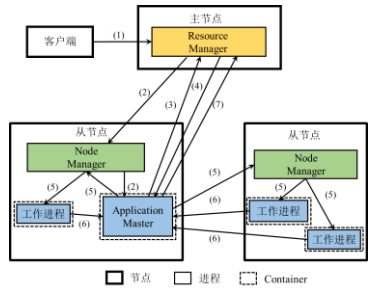
大纲

19

- 设计思想
- 体系架构
  - 架构图
  - 应用程序执行流程
- 工作原理
- 容错机制
- 典型示例

执行流程图

20



应用程序执行流程

21

1. 用户编写客户端应用程序，向Yarn提交应用程序。
2. RM负责接收和处理来自客户端的请求，尝试为该程序分配第一个Container，若分配成功则在Container中启动应用程序的AM。
3. AM向RM注册，这样客户端可通过RM查看应用程序的资源使用情况。AM将应用解析为作业并进一步分解为若干任务，并向RM申请启动这些任务的资源。

应用程序执行流程

22

4. RM向提出申请的AM分配以Container形式表示的资源。一旦AM申请到资源后，在多个任务间进行资源分配。
5. AM确定资源分配方案后，便与对应的NM通信，在相应的Container中启动相应的工作进程用于执行任务。
6. 各个任务向AM汇报自己的状态和进度，以便让AM随时掌握各个任务的运行状态。
7. 随着任务执行结束，AM逐步释放所占用的资源，最终向 RM注销并关闭自己。

大纲

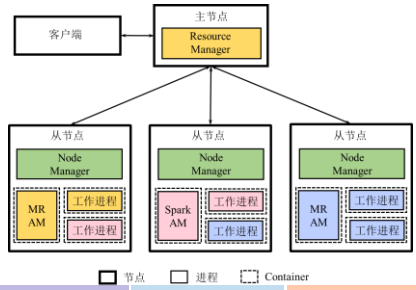
23

- 设计思想
- 体系架构
- 工作原理
  - 单平台多框架
  - 平台资源分配
- 容错机制
- 典型应用

一个平台多个框架

24

- 一个资源管理平台运行多个计算框架



大纲

25

- 设计思想
- 体系架构
- 工作原理
  - 单平台多框架
  - 平台资源分配
- 容错机制
- 典型应用

资源分配

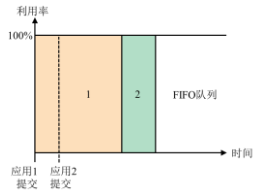
26

- Resource Manager中的调度器维护了一个或多个应用队列（queue），每个队列拥有一定量的资源，位于同一队列中的应用共享该队列所拥有的资源
- Yarn进行资源分配对象是应用，用户提交的每个应用会分配到其中一个队列当中，而队列决定了该应用能使用的资源上限
- 资源调度实际上是决定如何将资源分配给队列、以及如何分配给队列中应用的过程

资源分配策略-FIFO

27

- FIFO Scheduler只维护一个队列，该队列拥有集群中所有的资源，调度器的资源分配方式是先提交的应用先得到资源

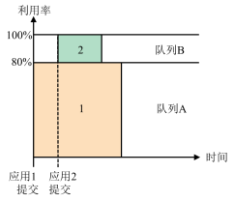


应用1占用所有资源，应用2需要等待应用1执行完毕后才会执行。

资源分配策略-Capacity

28

- Capacity Scheduler维护了层级式的队列，集群中的资源划分给这些队列，队列内部的资源分配方式是FIFO



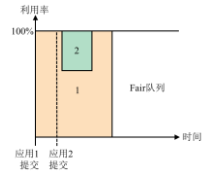
Capacity Scheduler调度方式可以避免某一长时间运行的应用独占集群资源而其它应用得不到运行的情况。

然而，我们也能观察到，在提交应用2之前队列B中的资源处于空闲状态，这造成了集群资源的浪费。

资源分配策略-Fair

29

- Fair Scheduler维护层级式的队列，集群中的资源划分给这些队列，但是这些队列可以共享资源，因而这些队列逻辑上可以看作是一个共享队列



当只有一个应用运行时，这个应用可以独占整个集群。但当其它应用提交到集群时，将空出部分资源给新的应用，最终所有的应用会根据所需使用内存的大小得到分配的资源。

大纲

30

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型示例

故障类型

31

- Resource Manager故障
- Node Manager故障
- Application Master故障：重启
- Container中的任务故障：重启

Resource Manager故障

32

- 如果Resource Manager发生故障，那么它在进行故障恢复时需要从某一持久化存储系统中恢复状态信息，所有应用将会重新执行
- 我们可以部署多个Resource Manager并通过ZooKeeper进行协调，从而保证Resource Manager的高可用性

Node Manager故障

33

- Resource Manager认为Node Manager所在节点上所有容器运行的任务也都执行失败，并把执行失败的信息告诉Application Master
  - AM将向RM重新申请资源运行这些任务
  - RM将分配其它节点的Container执行这些任务
- 如果发生故障的Node Manager进行恢复，那么它将向Resource Manager重新注册，重置本地的状态信息

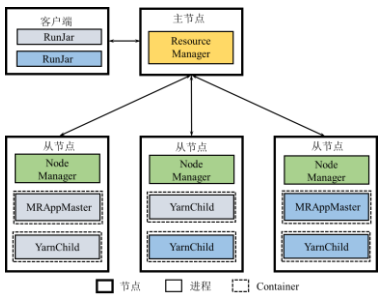
大纲

34

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型示例
  - Yarn平台运行MapReduce框架
  - Yarn平台运行Spark框架
  - Yarn平台运行MapReduce和Spark框架

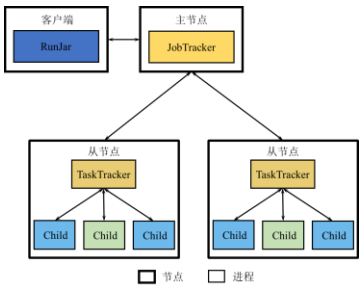
基于Yarn运行MapReduce

35



MapReduce 1.0

36



MapReduce 1.0与2.0

37

	MapReduce 1.0	MapReduce 2.0 + Yarn
资源管理	JobTracker TaskTracker	ResourceManager
		NodeManager
应用管理		MRAppMaster
任务执行	Child	YarnChild

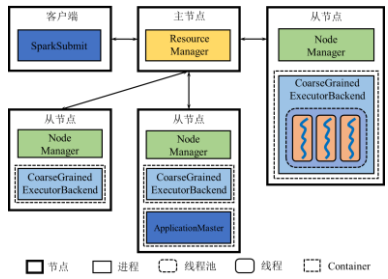
大纲

38

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型示例
  - Yarn平台运行MapReduce框架
  - Yarn平台运行Spark框架
  - Yarn平台运行MapReduce和Spark框架

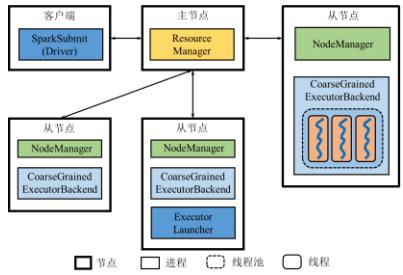
Yarn Cluster模式

39



Yarn Client模式

40



Yarn Client vs. Yarn Cluster

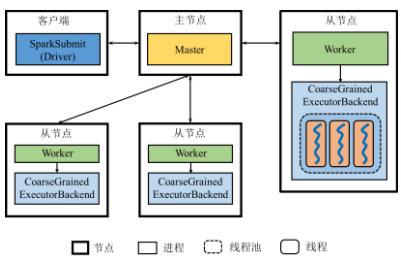
41

- Yarn Client
  - Driver：在客户端启动的进程中
  - ApplicationMaster：名为ExecutorLauncher，向ResourceManager申请资源，用container资源去链接其他的NodeManager，然后去启动executor
- Yarn Cluster
  - Driver存在于NodeManager上的某一个ApplicationMaster

Standalone Client

42

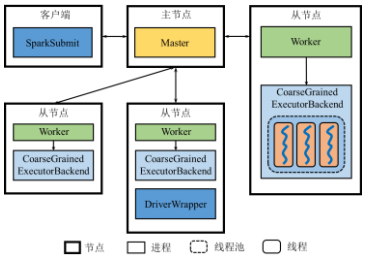
- Driver和客户端以同一个进程存在



Standalone Cluster

43

- 某一Worker启动一个进程作为Driver



Standalone模式 vs. Yarn模式

44

	Standalone		Yarn	
	Client	Cluster	Client	Cluster
资源管理	Master		ResourceManager	
	Worker		NodeManager	
应用管理	SparkSubmit	DriverWrapper	SparkSubmit	ApplicationMaster
任务执行	CoarseGrainedExecutorBackend			

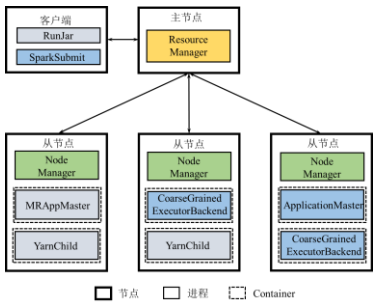
大纲

45

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型示例
  - Yarn平台运行MapReduce框架
  - Yarn平台运行Spark框架
  - Yarn平台运行MapReduce和Spark框架

基于Yarn运行MapReduce和Spark

46



课后阅读

47

- 论文
  - Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., … Saha, B. (2013). Apache Hadoop yarn: Yet another resource negotiator. In SoCC (pp. 5:1-5:16).

本章小结

48

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型示例

谢谢！Q&A