

#### 1.4.5 ASM

应用程序管理器 ASM 负责管理整个系统中所有<mark>应用程序</mark>,包括应用程序提交、与调度器协商资源以启动 MRAppMaster、监控 MRAppMaster 运行【态并在失败时重新启动它等

#### 1.4.6 Scheduler

这里的应用程序是指所有的MR Application

调度器根据应用程序的资源需求进行资源分配,不参与应用程序具体的执行和监控等工作资源分配的单位就是 Container,调度器是一个可插拔的组件,用户可以根据自己的需求实现自己的调度器。YARN 本身为我们提供了多种直接可用的调度器,比如 FIFO,Fair Scheduler和 Capacity Scheduler等

#1



#### 录目

1.1、YARN 概述       1         1.2、原 MapReduce 框架的不足       2         1.3、新版 YARN 架构的优点       2         1.4、YARN 的重要概念       2         1.4.1、ResourceManager       2         1.4.2、NodeManager       3         1.4.3、MRAppMaster       3         1.4.4、Container       3         1.4.5、ASM       3         1.4.6、Scheduler       3         1.5、YARN 架构及各角色职责       4         1.6、YARN 作业执行流程       4	1	YAKN	1
1.3、新版 YARN 架构的优点       2         1.4、YARN 的重要概念       2         1.4.1、ResourceManager       2         1.4.2、NodeManager       3         1.4.3、MRAppMaster       3         1.4.4、Container       3         1.4.5、ASM       3         1.4.6、Scheduler       3         1.5、YARN 架构及各角色职责       4		1.1、YARN 概述	1
1.4、YARN 的重要概念       2         1.4.1、ResourceManager       2         1.4.2、NodeManager       3         1.4.3、MRAppMaster       3         1.4.4、Container       3         1.4.5、ASM       3         1.4.6、Scheduler       3         1.5、YARN 架构及各角色职责       4		1.2、原 MapReduce 框架的不足	2
1.4.1、ResourceManager       2         1.4.2、NodeManager       3         1.4.3、MRAppMaster       3         1.4.4、Container       3         1.4.5、ASM       3         1.4.6、Scheduler       3         1.5、YARN 架构及各角色职责       4		1.3、新版 YARN 架构的优点	2
1.4.2、NodeManager       3         1.4.3、MRAppMaster       3         1.4.4、Container       3         1.4.5、ASM       3         1.4.6、Scheduler       3         1.5、YARN 架构及各角色职责       4		1.4、YARN 的重要概念	2
1.4.2、NodeManager       3         1.4.3、MRAppMaster       3         1.4.4、Container       3         1.4.5、ASM       3         1.4.6、Scheduler       3         1.5、YARN 架构及各角色职责       4		1.4.1 ResourceManager	2
1.4.3、MRAppMaster       3         1.4.4、Container       3         1.4.5、ASM       3         1.4.6、Scheduler       3         1.5、YARN 架构及各角色职责       4			
1.4.5、ASM		1.4.3、MRAppMaster	3
1.4.6、Scheduler3 1.5、YARN 架构及各角色职责4		1.4.4、Container	3
1.5、YARN 架构及各角色职责4		1.4.5 \ ASM	3
		1.4.6 Scheduler	3
1.6、YARN 作业执行流程4		1.5、YARN 架构及各角色职责	4
		1.6、YARN 作业执行流程	4

# 1 YARN

## 1.1、YARN 概述

YARN (Yet Another Resource Negotiator)

YARN 是一个资源调度平台,负责为运算程序提供服务器运算资源,相当于一个分布式的操作系统平台,而 MapReduce 等运算程序则相当于运行于操作系统之上的应用程序

YARN 是 Hadoop2.x 版本中的一个新特性。它的出现其实是为了解决第一代 MapReduce 编程框架的不足,提高集群环境下的资源利用率,这些资源包括内存,磁盘,网络,IO等。Hadoop2.X 版本中重新设计的这个 YARN 集群,具有更好的扩展性,可用性,可靠性,向后兼容性,以及能支持除 MapReduce 以外的更多分布式计算程序

- 1、YARN 并不清楚用户提交的程序的运行机制
- 2、YARN 只提供运算资源的调度(用户程序向 YARN 申请资源, YARN 就负责分配资源)
- 3、YARN 中的主管角色叫 ResourceManager
- 4、YARN 中具体提供运算资源的角色叫 NodeManager
- 5、这样一来,YARN 其实就与运行的用户程序完全解耦,就意味着 YARN 上可以运行各种类型的分布式运算程序(MapReduce 只是其中的一种),比如 MapReduce、Storm 程序,Spark程序,Tez ······
- 6、所以,Spark、Storm 等运算框架都可以整合在 YARN 上运行,只要他们各自的框架中有符合 YARN 规范的资源请求机制即可



7、yarn 就成为一个通用的资源调度平台,从此,企业中以前存在的各种运算集群都可以整合在一个物理集群上,提高资源利用率,方便数据共享

# 1.2、原 MapReduce 框架的不足

- 1、JobTracker 是集群事务的集中处理点,存在单点故障
- 2、JobTracker 需要完成的任务太多,既要维护 job 的状态又要维护 job 的 task 的状态,造成过多的资源消耗
- 3、在 TaskTracker 端,用 Map/Reduce Task 作为资源的表示过于简单,没有考虑到 CPU、内存等资源情况,当把两个需要消耗大内存的 Task 调度到一起,很容易出现 OOM
- 4、把资源强制划分为 Map/Reduce Slot, 当只有 MapTask 时, TeduceSlot 不能用; 当只有 Reduce Task 时, MapSlot 不能用, 容易造成资源利用不足。

#### 总结起来就是:

- 1、扩展性差
- 2、可靠性低
- 3、资源利用率低
- 4、不支持多种计算框架

# 1.3、新版 YARN 架构的优点

YARN/MRv2 最基本的想法是将原 JobTracker 主要的资源管理和 Job 调度/监视功能分开作为两个单独的守护进程。有一个全局的 ResourceManager(RM)和每个 Application 有一个 ApplicationMaster(AM),Application 相当于 MapReduce Job 或者 DAG jobs。ResourceManager 和 NodeManager(NM)组成了基本的数据计算框架。ResourceManager 协调集群的资源利用,任何 Client 或者运行着的 applicationMaster 想要运行 Job 或者 Task 都得向 RM 申请一定的资源。ApplicatonMaster 是一个框架特殊的库,对于 MapReduce 框架而言有它自己的 AM 实现,用户也可以实现自己的 AM,在运行的时候,AM 会与 NM 一起来启动和监视 Tasks。

# 1.4、YARN 的重要概念

## 1.4.1 Resource Manager

ASM)

ResourceManager 是基于应用程序对集群资源的需求进行调度的 YARN 集群主控节点,负责协调和管理整个集群(所有 NodeManager)的资源,响应用户提交的不同类型应用程序的解析,调度,监控等工作。ResourceManager 会为每一个 Application 启动一个 MRAppMaster,并且 MRAppMaster 分散在各个 NodeManager 节点

它主要由两个组件构成:调度器(Scheduler)和应用程序管理器(ApplicationsManager,

注意: ASM 是ApplicationsManager,而不是ApplicationMaster

AM 才是 ApplicationsManager,而不是ApplicationMaster

Stay hungry Stay foolish -- http://blog.csdn.net/zhongqi2513



YARN 集群的主节点 ResourceManager 的职责:

- 1、处理客户端请求
- 2、启动或监控 MRAppMaster
- 3、监控 NodeManager
- 4、资源的分配与调度

### 1.4.2 NodeManager

NodeManager 是 YARN 集群当中真正资源的提供者,是真正执行应用程序的容器的提供者,监控应用程序的资源使用情况(CPU,内存,硬盘,网络),并通过心跳向集群资源调度器 ResourceManager 进行汇报以更新自己的健康状态。同时其也会监督 Container 的生命周期管理,监控每个 Container 的资源使用(内存、CPU 等)情况,追踪节点健康状况,管理日志和不同应用程序用到的附属服务(auxiliary service)。

YARN 集群的从节点 NodeManager 的职责:

- 1、管理单个节点上的资源
- 2、处理来自 ResourceManager 的命令
- 3、处理来自 MRAppMaster 的命令

### 1.4.3 MRAppMaster

MRAppMaster 对应一个应用程序,职责是:向资源调度器申请执行任务的资源容器,运行任务,监控整个任务的执行,跟踪整个任务的状态,处理任务失败以异常情况

#### 1.4.4 Container

Container 容器是一个抽象出来的逻辑资源单位。容器是由 ResourceManager Scheduler 服务 动态分配的资源构成,它包括了该节点上的一定量 CPU,内存,磁盘,网络等信息,MapReduce 程序的所有 Task 都是在一个容器里执行完成的,容器的大小是可以动态调整的

# 1.4.5. ASM (Applications Manager)

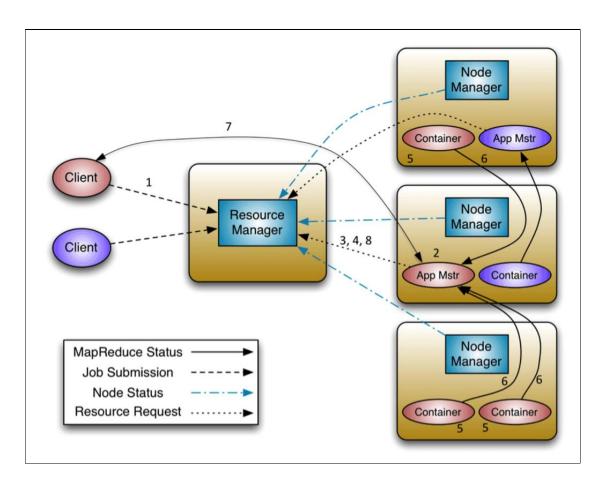
应用程序管理器 ASM 负责管理整个系统中所有应用程序, 包括应用程序提交、与调度器协商资源以启动 MRAppMaster、监控 MRAppMaster 运行 大态并在失败时重新启动它等

### 1.4.6 Scheduler

调度器根据应用程序的资源需求进行资源分配,不参与应用程序具体的执行和监控等工作资源分配的单位就是 Container,调度器是一个可插拔的组件,用户可以根据自己的需求实现自己的调度器。YARN 本身为我们提供了多种直接可用的调度器,比如 FIFO,Fair Scheduler和 Capacity Scheduler等



## 1.5、YARN 架构及各角色职责



# 1.6、YARN 作业执行流程

YARN 作业执行流程:

- 1、用户向 YARN 中提交应用程序,其中包括 MRAppMaster 程序,启动 MRAppMaster 的命令,用户程序等。
- 2、ResourceManager 为该程序分配第一个 Container,并与对应的 NodeManager 通讯,要求它在这个 Container 中启动应用程序 MRAppMaster。
- 3、MRAppMaster 首先向 ResourceManager 注册,这样用户可以直接通过 ResourceManager 查看应用程序的运行状态,然后将为各个任务申请资源,并监控它的运行状态,直到运行结束,重复 4 到 7 的步骤。
- 4、MRAppMaster 采用轮询的方式通过 RPC 协议向 ResourceManager 申请和领取资源。
- 5、一旦 MRAppMaster 申请到资源后,便与对应的 NodeManager 通讯,要求它启动任务。
- 6、NodeManager 为任务设置好运行环境(包括环境变量、JAR 包、二进制程序等)后,将任务启动命令写到一个脚本中,并通过运行该脚本启动任务。
- 7、各个任务通过某个 RPC 协议向 MRAppMaster 汇报自己的状态和进度,以让 MRAppMaster 随时掌握各个任务的运行状态,从而可以在任务败的时候重新启动任务。
- 8、应用程序运行完成后,MRAppMaster 向 ResourceManager 注销并关闭自己。