YARN的设计理念和基本框架

YARN(Yet Another Resouce Negotiator)是Hadoop 2.0新增加的一个子项目，与Common, Mapreduce和YARN三个分支并行。它的引入使得Hadoop分布式计算系统进入了平台化时代，即各种计算框架可以运行在一个集群中，由资源管理系统YRAN进行统一的管理和调度，从而共享整个集群资源、提高资源利用率。

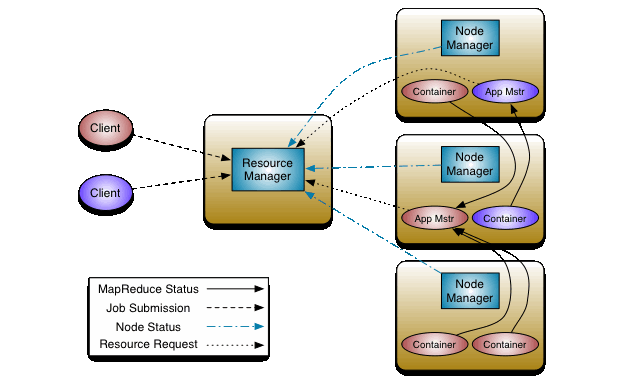
### 1. MRv1的局限性

YARN的出现是为了弥补MapReduce v1(MRv1)中的种种不足，具体表现在：  
****(1). 扩展性差****。MRv1的jobtracker同时具备了资源管理和作业控制两个功能，是一个系统瓶颈所在。严重制约了Hadoop集群扩展性。  
****(2). 可靠性差****。MRv1采用了master/slave结构，其中，master存在单点故障问题。  
****(3). 资源利用率低****。MRv1采用了基于粗粒度的“槽位(slot)”的资源分配模型，通常一个任务不会用完一个槽位的资源，而其他任务也无法使用这些空闲的资源。此外，槽位分为Map slot和reduce slot两种，他们之间不能相互共享，这样就导致了一种槽位资源紧张而另一种闲置。  
****(4). 无法支持其他计算框架****。MR框架基于磁盘数据的离线分析，满足不了应用需求。而内存计算框架、流失计算框架和迭代式计算框架也不能应用于MRv1的hadoop平台上。

hadoop的下一代计算框架MRv2将资源管理功能抽象成一个通用系统YARN，而MRv1的jobtracker和tasktrack也不复存在，计算框架(MR, storm, spark在)同时运行在YARN之上，使得hadoop进入了多计算框架的弹性计算平台时代。这样带来了种种好处：  
****(1). 资源利用率高****。在某些时间，有些资源计算框架的集群资源紧张，而另外一些集群资源空闲。那么这些框架共享使用一个集群则可以大大提高资源利用率。  
****(2). 运维成本低****。由多个集群集中为一个集群从而降低了运维成本。  
****(3). 数据共享****。避免了集群之间数据移动。

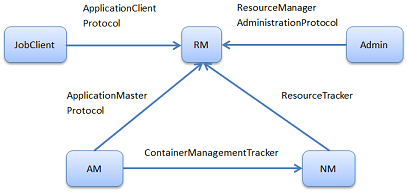
### 2. YARN的基础架构

YARN的基本思想是将MRv1的jobtracker拆分成了两个独立的服务：一个全局的资源管理器ResouceManager和每个应用程序独有的Application Master。前者负责整个系统的资源管理和分配，后者负责单个应用的管理。YARN总体上也是Master/slave架构——ResourceManager/NodeManager。前者(RM)负责对各个NodeManager(NM)上的资源进行统一管理和调度。它由两个组件构成：  
****(1). 调度器(Scheduler)****: 根据容量、队列等限制条件, 将系统资源分配给正在运行的应用程序。  
****(2). 应用程序管理器(ApplicationsManager, 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ApplicationMnager(AM)、监控AM运行状态并在失败的时候重启。

  
(pic by IBM developer)  
每一个被提交的应用都会有一个对应的AM，并向RM申请NodeManager的计算资源。不同应用的Applicaion Master可以运行在不同的NodeManager中，互相不干扰。AM主要功能包括:  
****(1)****. 与RM scheduler协商以获得container资源。  
****(2)****. 将得到的任务进一步分配给内部的任务。  
****(3)****. 与NodeManager(NM)通信以启动/停止任务。  
****(4)****. 监控所有任务的运行状态。  
NodeManager的作用则是负责接收并启动应用的container、而向RM回报本节点上的应用Container运行状态和资源使用情况。

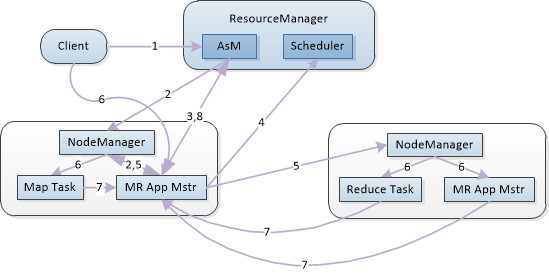
### 3. YARN之间的通信协议

RPC协议将各个组件连接起来，每个需要通信的组件只有一个RPC协议，且有固定”客户端”和”服务器端”，每次都是由客户端向服务器端主动发起请求。

YARN使用了google的protocol buffers来保证hadoop的兼容性。YARN中主要的通信协议如下：  
JobClient -> AM : ApplicationClientProtocol  
Admin -> AM : ResourceManagerAdministrationProtocol  
AM -> RM : ApplicationMasterProtocol  
AM -> NM : ContainerManagermentProtocol – 启停container，获得各个container的使用状态信息。  
NM -> RM : ResouceTracker – NM向RM注册、并定时发送心跳信息汇报当前节点的资源使用状况和container运行情况。  
[](http://debugo.com/wp-content/uploads/2014/04/2.png)

### 4. YARN的工作流程

运行在YARN上的两类应用，二者虽然作用不同，但在YARN上的运行流程是相同的：  
****短应用程序****：一定时间（无论是秒级、小时级甚至更长时间内）可以完成并正常退出的程序。如MapReduce作业、Tez DAG作业等。  
****长应用程序****：一般情况下永远不会退出的应用。通常指像storm service, HBase service(HMaster和RegionServer两类服务）等。它们本身作为一个框架提供API供其他用户使用。

YARN的工作流程如下:  
****(1)****. 用户向YARN ResouceManager(RM)提交应用程序，其中包括Application Master(AM)程序、启动AM命令、用户程序。  
****(2)****. RM为应用程序分配第一个Container，并与对应的NodeManager通信，要求它在这个container中启动应用程序的AM。  
****(3)****. AM先向RM进行注册，用户就可以在RM中查看应用程序的运行状态，然后它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4-7）。  
****(4)****. AM采用轮询的方式通过RPC协议向RM申请和领取资源。  
****(5)****. 一旦AM申请到资源后，便与对应的NodeManager（NM）通信，要求它启动任务。  
****(6)****. NM为任务设置好环境（环境变量、jar包、二进制程序等），然后将任务启动命令写到一个脚本上并执行它。  
****(7)****. 各个任务通过RPC向AM汇报自己的状态和进度。让AM随时掌握各个任务的运行状态，从而可以在任务失败时重新启动任务。同样的，用户也可以随时向AM查询当前程序运行状态。  
****(8)****. 应用程序运行完成后，AM向RM注销并退出。  
[](http://debugo.com/wp-content/uploads/2014/04/YARN.png)