## 简介

Apache Hadoop 2.0 包含 YARN，它将资源管理和处理组件分开。基于 YARN 的架构不受 MapReduce 约束。本文将介绍 YARN，以及它相对于 Hadoop 中以前的分布式处理层的一些优势。本文将了解如何使用 YARN 的可伸缩性、效率和灵活性增强您的集群。

## Apache Hadoop 简介

Apache Hadoop 是一个开源软件框架，可安装在一个商用机器集群中，使机器可彼此通信并协同工作，以高度分布式的方式共同存储和处理大量数据。最初，Hadoop 包含以下两个主要组件：Hadoop Distributed File System (HDFS) 和一个分布式计算引擎，该引擎支持以 MapReduce 作业的形式实现和运行程序。

MapReduce 是 Google 推广的一个简单的编程模型，它对以高度并行和可扩展的方式处理大数据集很有用。MapReduce 的灵感来源于函数式编程，用户可将他们的计算表达为 map 和 reduce 函数，将数据作为键值对来处理。Hadoop 提供了一个高级 API 来在各种语言中实现自定义的 map 和 reduce 函数。

Hadoop 还提供了软件基础架构，以一系列 map 和 reduce 任务的形式运行 MapReduce 作业。*Map 任务*在输入数据的子集上调用 map 函数。在完成这些调用后，*reduce 任务* 开始在 map 函数所生成的中间数据上调用 reduce 任务，生成最终的输出。 map 和 reduce 任务彼此单独运行，这支持并行和容错的计算。

最重要的是，Hadoop 基础架构负责处理分布式处理的所有复杂方面：并行化、调度、资源管理、机器间通信、软件和硬件故障处理，等等。得益于这种干净的抽象，实现处理数百（或者甚至数千）个机器上的数 TB 数据的分布式应用程序从未像现在这么容易过，甚至对于之前没有使用分布式系统的经验的开发人员也是如此。

## Hadoop 的黄金时代

尽管 MapReduce 模型存在着多种开源实现，但 Hadoop MapReduce 很快就变得非常流行。Hadoop 也是全球最令人兴奋的开源项目之一，它提供了多项出色的功能：高级 API、近线性的可伸缩性、开源许可、在商用硬件上运行的能力，以及容错。它已获得数百（或许已达数千）个公司的成功部署，是大规模分布式存储和处理的最新标准。

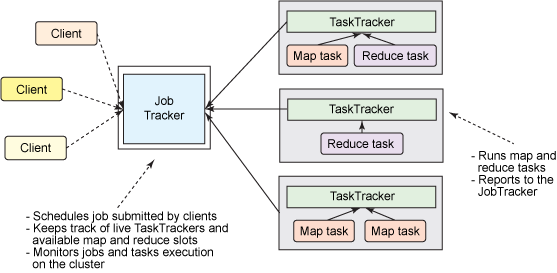
一些早期的 Hadoop 采用者，比如 Yahoo! 和 Facebook，构建了包含 4,000 个节点的大型集群，以满足不断增长和变化的数据处理需求。但是，在构建自己的集群后，他们开始注意到了 Hadoop MapReduce 框架的一些局限性。

## 经典 MapReduce 的局限性

经典 MapReduce 的最严重的限制主要关系到可伸缩性、资源利用和对与 MapReduce 不同的工作负载的支持。在 MapReduce 框架中，作业执行受两种类型的进程控制：

* 一个称为 *JobTracker* 的主要进程，它协调在集群上运行的所有作业，分配要在 TaskTracker 上运行的 map 和 reduce 任务。
* 许多称为 *TaskTracker* 的下级进程，它们运行分配的任务并定期向 JobTracker 报告进度。

##### Apache Hadoop 的经典版本 (MRv1)



大型的 Hadoop 集群显现出了由单个 JobTracker 导致的可伸缩性瓶颈。依据 Yahoo!，在集群中有 5,000 个节点和 40,000 个任务同时运行时，这样一种设计实际上就会受到限制。由于此限制，必须创建和维护更小的、功能更差的集群。

此外，较小和较大的 Hadoop 集群都从未最高效地使用他们的计算资源。在 Hadoop MapReduce 中，每个从属节点上的计算资源由集群管理员分解为固定数量的 map 和 reduce slot，这些 slot 不可替代。设定 map slot 和 reduce slot 的数量后，节点在任何时刻都不能运行比 map slot 更多的 map 任务，即使没有 reduce 任务在运行。这影响了集群的利用率，因为在所有 map slot 都被使用（而且我们还需要更多）时，我们无法使用任何 reduce slot，即使它们可用，反之亦然。

最后但同样重要的是，Hadoop 设计为仅运行 MapReduce 作业。随着替代性的编程模型（比如 Apache Giraph 所提供的图形处理）的到来，除 MapReduce 外，越来越需要为可通过高效的、公平的方式在同一个集群上运行并共享资源的其他编程模型提供支持。

2010 年，Yahoo! 的工程师开始研究一种全新的 Hadoop 架构，用这种架构来解决上述所有限制并增加多种附加功能。

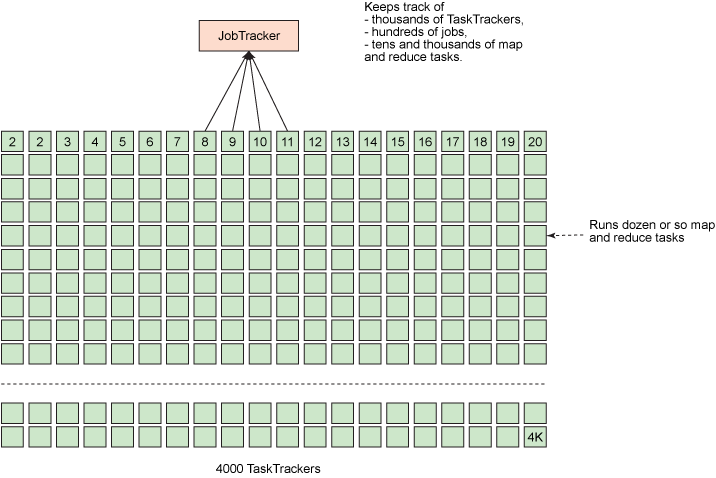
## 解决可伸缩性问题

在 Hadoop MapReduce 中，JobTracker 具有两种不同的职责：

* 管理集群中的计算资源，这涉及到维护活动节点列表、可用和占用的 map 和 reduce slots 列表，以及依据所选的调度策略将可用 slots 分配给合适的作业和任务
* 协调在集群上运行的所有任务，这涉及到指导 TaskTracker 启动 map 和 reduce 任务，监视任务的执行，重新启动失败的任务，推测性地运行缓慢的任务，计算作业计数器值的总和，等等

为单个进程安排大量职责会导致重大的可伸缩性问题，尤其是在较大的集群上，JobTracker 必须不断跟踪数千个 TaskTracker、数百个作业，以及数万个 map 和 reduce 任务。下图演示了这一问题。相反，TaskTracker 通常近运行十来个任务，这些任务由勤勉的 JobTracker 分配给它们。

##### 大型 Apache Hadoop 集群 (MRv1) 上繁忙的 JobTracker



为了解决可伸缩性问题，一个简单而又绝妙的想法应运而生：我们减少了单个 JobTracker 的职责，将部分职责委派给 TaskTracker，因为集群中有许多 TaskTracker。在新设计中，这个概念通过将 JobTracker 的双重职责（集群资源管理和任务协调）分开为两种不同类型的进程来反映。

不再拥有单个 JobTracker，一种新方法引入了一个集群管理器，它惟一的职责就是跟踪集群中的活动节点和可用资源，并将它们分配给任务。对于提交给集群的每个作业，会启动一个专用的、短暂的 JobTracker 来控制该作业中的任务的执行。有趣的是，短暂的 JobTracker 由在从属节点上运行的 TaskTracker 启动。因此，**作业的生命周期的协调工作分散在集群中所有可用的机器上**。得益于这种行为，更多工作可并行运行，可伸缩性得到了显著提高。

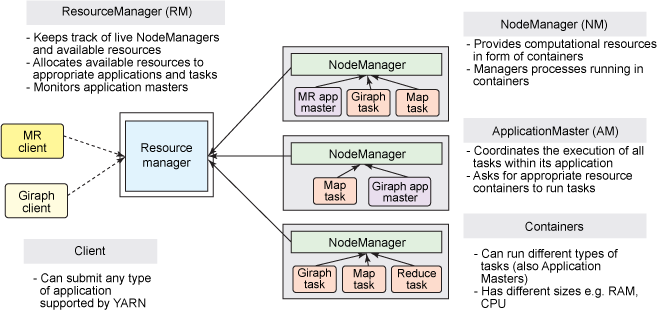
## YARN：下一代 Hadoop 计算平台

我们现在稍微改变一下用辞。以下名称的改动有助于更好地了解 YARN 的设计：

* ResourceManager 代替集群管理器
* ApplicationMaster 代替一个专用且短暂的 JobTracker
* NodeManager 代替 TaskTracker
* 一个分布式应用程序代替一个 MapReduce 作业

YARN 是下一代 Hadoop 计算平台，如下所示。

##### YARN 的架构



在 YARN 架构中，一个全局 ResourceManager 以主要后台进程的形式运行，它通常在专用机器上运行，在各种竞争的应用程序之间仲裁可用的集群资源。ResourceManager 会追踪集群中有多少可用的活动节点和资源，协调用户提交的哪些应用程序应该在何时获取这些资源。ResourceManager 是惟一拥有此信息的进程，所以它可通过某种共享的、安全的、多租户的方式制定分配（或者调度）决策（例如，依据应用程序优先级、队列容量、ACLs、数据位置等）,注：ACL:[访问控制列表](https://baike.baidu.com/item/%E8%AE%BF%E9%97%AE%E6%8E%A7%E5%88%B6%E5%88%97%E8%A1%A8/1844390" \t "https://baike.baidu.com/item/ACL/_blank)（Access Control List，ACL） 是[路由器](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1%E5%99%A8/108294" \t "https://baike.baidu.com/item/ACL/_blank)和[交换机](https://baike.baidu.com/item/%E4%BA%A4%E6%8D%A2%E6%9C%BA/103532" \t "https://baike.baidu.com/item/ACL/_blank)接口的指令列表，用来[控制端口](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E7%AB%AF%E5%8F%A3/1688757" \t "https://baike.baidu.com/item/ACL/_blank)进出的数据包。

在用户提交一个应用程序时，一个称为 ApplicationMaster 的轻量型进程实例会启动来协调应用程序内的所有任务的执行。这包括监视任务，重新启动失败的任务，推测性地运行缓慢的任务，以及计算应用程序计数器值的总和。这些职责以前分配给所有作业的单个 JobTracker。ApplicationMaster 和属于它的应用程序的任务，在受 NodeManager 控制的资源容器中运行。

NodeManager 是 TaskTracker 的一种更加普通和高效的版本。没有固定数量的 map 和 reduce slots，NodeManager 拥有许多动态创建的资源容器。容器的大小取决于它所包含的资源量，比如内存、CPU、磁盘和网络 IO。目前，仅支持内存和 CPU (YARN-3)。未来可使用 cgroups 来控制磁盘和网络 IO。一个节点上的容器数量，由配置参数与专用于从属后台进程和操作系统的资源以外的节点资源总量（比如总 CPU 数和总内存）共同决定。

有趣的是，ApplicationMaster 可在容器内运行任何类型的任务。例如，MapReduce ApplicationMaster 请求一个容器来启动 map 或 reduce 任务，而 Giraph ApplicationMaster 请求一个容器来运行 Giraph 任务。您还可以实现一个自定义的 ApplicationMaster 来运行特定的任务，进而发明出一种全新的分布式应用程序框架，改变大数据世界的格局。您可以查阅 Apache Twill，它旨在简化 YARN 之上的分布式应用程序的编写。

在 YARN 中，MapReduce 降级为一个分布式应用程序的一个角色（但仍是一个非常流行且有用的角色），现在称为 MRv2。MRv2 是经典 MapReduce 引擎（现在称为 MRv1）的重现，运行在 YARN 之上。

## 一个可运行任何分布式应用程序的集群

ResourceManager、NodeManager 和容器都不关心应用程序或任务的类型。所有特定于应用程序框架的代码都转移到它的 ApplicationMaster，以便任何分布式框架都可以受 YARN 支持 — 只要有人为它实现了相应的 ApplicationMaster。

得益于这个一般性的方法，Hadoop YARN 集群运行许多不同工作负载的梦想才得以实现。想像一下：您数据中心中的一个 Hadoop 集群可运行 MapReduce、Giraph、Storm、Spark、Tez/Impala、MPI 等。

单一集群方法明显提供了大量优势，其中包括：

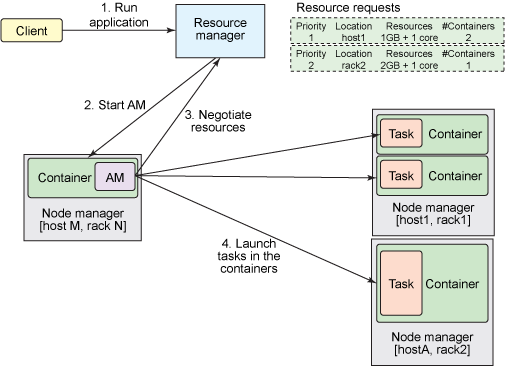
* 更高的集群利用率，一个框架未使用的资源可由另一个框架使用
* 更低的操作成本，因为只有一个 “包办一切的” 集群需要管理和调节
* 更少的数据移动，无需在 Hadoop YARN 与在不同机器集群上运行的系统之间移动数据

管理单个集群还会得到一个更环保的数据处理解决方案。使用的数据中心空间更少，浪费的硅片更少，使用的电源更少，排放的碳更少，这只是因为我们在更小但更高效的 Hadoop 集群上运行同样的计算。

## YARN 中的应用程序提交

本节讨论在应用程序提交到 YARN 集群时，ResourceManager、ApplicationMaster、NodeManagers 和容器如何相互交互。下图显示了一个例子。

##### YARN 中的应用程序提交



假设用户采用与 MRv1 中相同的方式键入 hadoop jar 命令，将应用程序提交到 ResourceManager。ResourceManager 维护在集群上运行的应用程序列表，以及每个活动的 NodeManager 上的可用资源列表。ResourceManager 需要确定哪个应用程序接下来应该获得一部分集群资源。该决策受到许多限制，比如队列容量、ACL 和公平性。ResourceManager 使用一个可插拔的 Scheduler。Scheduler 仅执行调度；它管理谁在何时获取集群资源（以容器的形式），但不会对应用程序内的任务执行任何监视，所以它不会尝试重新启动失败的任务。

在 ResourceManager 接受一个新应用程序提交时，Scheduler 制定的第一个决策是选择将用来运行 ApplicationMaster 的容器。在 ApplicationMaster 启动后，它将负责此应用程序的整个生命周期。首先也是最重要的是，它将资源请求发送到 ResourceManager，请求运行应用程序的任务所需的容器。资源请求是对一些容器的请求，用以满足一些资源需求，比如：

* 一定量的资源，目前使用 MB 内存和 CPU 份额来表示
* 一个首选的位置，由主机名、机架名称指定，或者使用 \* 来表示没有偏好
* 此应用程序中的一个优先级，而不是跨多个应用程序

如果可能的话，ResourceManager 会分配一个满足 ApplicationMaster 在资源请求中所请求的需求的容器（表达为容器 ID 和主机名）。该容器允许应用程序使用特定主机上给定的资源量。分配一个容器后，ApplicationMaster 会要求 NodeManager（管理分配容器的主机）使用这些资源来启动一个特定于应用程序的任务。此任务可以是在任何框架中编写的任何进程（比如一个 MapReduce 任务或一个 Giraph 任务）。NodeManager 不会监视任务；它仅监视容器中的资源使用情况，举例而言，如果一个容器消耗的内存比最初分配的更多，它会结束该容器。

ApplicationMaster 会竭尽全力协调容器，启动所有需要的任务来完成它的应用程序。它还监视应用程序及其任务的进度，在新请求的容器中重新启动失败的任务，以及向提交应用程序的客户端报告进度。应用程序完成后，ApplicationMaster 会关闭自己并释放自己的容器。

尽管 ResourceManager 不会对应用程序内的任务执行任何监视，但它会检查 ApplicationMaster 的健康状况。如果 ApplicationMaster 失败，ResourceManager 可在一个新容器中重新启动它。您可以认为 ResourceManager 负责管理 ApplicationMaster，而 ApplicationMasters 负责管理任务。

## 有趣的事实和特性

YARN 提供了多种其他的优秀特性。介绍所有这些特性不属于本文的范畴，我仅列出一些值得注意的特性：

* 如果作业足够小，*Uberization* 支持在 ApplicationMaster 的 JVM 中运行一个 MapReduce 作业的所有任务。这样，您就可避免从 ResourceManager 请求容器以及要求 NodeManagers 启动（可能很小的）任务的开销。
* 与为 MRv1 编写的 MapReduce 作业的二进制或源代码兼容性 (MAPREDUCE-5108)。
* 针对 ResourceManager 的高可用性 (YARN-149)。此工作正在进行中，已由一些供应商完成。
* 重新启动 ResourceManager 后的应用程序恢复 (YARN-128)。ResourceManager 将正在运行的应用程序和已完成的任务的信息存储在 HDFS 中。如果 ResourceManager 重新启动，它会重新创建应用程序的状态，仅重新运行不完整的任务。此工作已接近完成，社区正在积极测试。它已由一些供应商完成。
* 简化的用户日志管理和访问。应用程序生成的日志不会留在各个从属节点上（像 MRv1 一样），而转移到一个中央存储区，比如 HDFS。在以后，它们可用于调试用途，或者用于历史分析来发现性能问题。
* Web 界面的新外观。

## 结束语

YARN 是一个完全重写的 Hadoop 集群架构。它似乎在商用机器集群上实现和执行分布式应用程序的方式上带来了变革。

与第一版 Hadoop 中经典的 MapReduce 引擎相比，YARN 在可伸缩性、效率和灵活性上提供了明显的优势。小型和大型 Hadoop 集群都从 YARN 中受益匪浅。对于最终用户（开发人员，而不是管理员），这些更改几乎是不可见的，因为可以使用相同的 MapReduce API 和 CLI 运行未经修改的 MapReduce 作业。

没有理由不将 MRv1 迁移到 YARN。最大型的 Hadoop 供应商都同意这一点，而且为 Hadoop YARN 的运行提供了广泛的支持。如今，YARN 已被许多公司成功应用在生产中，比如 Yahoo!、eBay、Spotify、Xing、Allegro 等。

## 一、Yarn简介

Yarn是Hadoop集群的资源管理系统。Hadoop2.0对MapReduce框架做了彻底的设计重构，我们称Hadoop2.0中的MapReduce为MRv2或者Yarn。在介绍Yarn之前，我们先回头看一下Hadoop1.x对MapReduce job的调度管理方式（可参考：[Hadoop核心之MapReduce架构设计](http://blog.csdn.net/suifeng3051/article/details/49364677" \t "http://blog.csdn.net/suifeng3051/article/details/_blank)），它主要包括两部分功能：

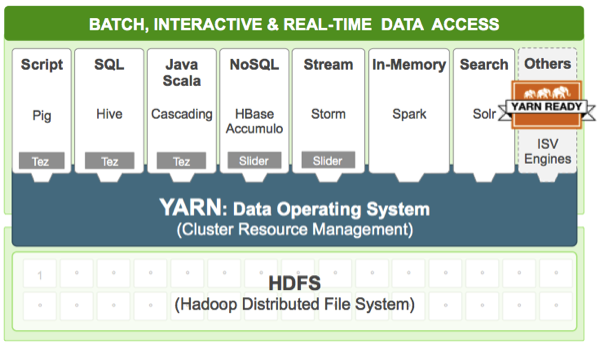
1. ResourceManagement 资源管理

2. JobScheduling/JobMonitoring 任务调度监控

到了Hadoop2.x也就是Yarn，它的目标是将这两部分功能分开，也就是分别用两个进程来管理这两个任务：

1. ResourceManger

2. ApplicationMaster

需要注意的是，在Yarn中我们把job的概念换成了application，因为在新的Hadoop2.x中，运行的应用不只是MapReduce了，还有可能是其它应用如一个DAG（有向无环图Directed Acyclic Graph，例如storm应用）。Yarn的另一个目标就是拓展Hadoop，使得它不仅仅可以支持MapReduce计算，还能很方便的管理诸如Hive、Hbase、Pig、Spark/Shark等应用。这种新的架构设计能够使得各种类型的应用运行在Hadoop上面，并通过Yarn从系统层面进行统一的管理，也就是说，有了Yarn，各种应用就可以互不干扰的运行在同一个Hadoop系统中，共享整个集群资源，如下图所示：   


## 二、Yarn的组件及架构

Yarn主要由以下几个组件组成：

1. ResourceManager：Global（全局）的进程

2. NodeManager：运行在每个节点上的进程

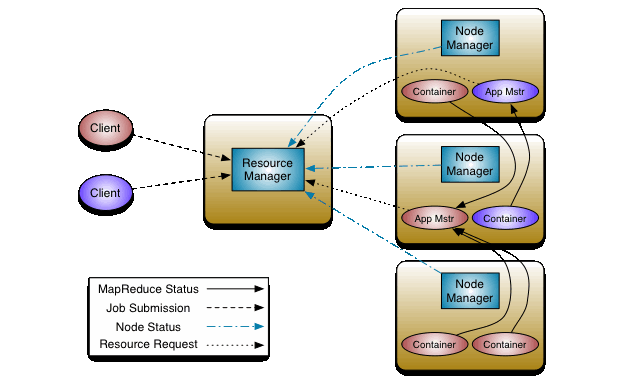
3. ApplicationMaster：Application-specific（应用级别）的进程

- \*Scheduler：是ResourceManager的一个组件\*

- \*Container：节点上一组CPU和内存资源\*

****Container****是Yarn对计算机计算资源的抽象，它其实就是*一组CPU和内存资源*，所有的应用都会运行在Container中。****ApplicationMaster****是对运行在Yarn中某个应用的抽象，它其实就是某个类型应用的实例，ApplicationMaster是应用级别的，它的主要功能就是向****ResourceManager****（全局的）申请计算资源（Containers）并且和NodeManager交互来执行和监控具体的task。****Scheduler****是ResourceManager专门进行资源管理的一个组件，负责分配NodeManager上的Container资源，****NodeManager****也会不断发送自己Container使用情况给ResourceManager。

ResourceManager和NodeManager两个进程主要负责系统管理方面的任务。ResourceManager有一个Scheduler，负责各个集群中应用的资源分配。对于每种类型的每个应用，都会对应一个ApplicationMaster实例，ApplicationMaster通过和ResourceManager沟通获得Container资源来运行具体的job，并跟踪这个job的运行状态、监控运行进度。

下面我们看一下整个Yarn的架构图：   


## 三、Yarn的组件详解

### 3.1 Container

****Container****是Yarn框架的计算单元，是具体执行应用task（如map task、reduce task）的基本单位。Container和集群节点的关系是：一个节点会运行多个Container，但一个Container不会跨节点。

一个Container就是一组分配的系统资源，现阶段只包含两种系统资源（之后可能会增加磁盘、网络等资源）：

1. CPU core

2. Memory in MB

既然一个Container指的是具体节点上的计算资源，这就意味着Container中必定含有计算资源的位置信息：计算资源位于哪个机架的哪台机器上。所以我们在请求某个Container时，其实是向某台机器发起的请求，请求的是这台机器上的CPU和内存资源。

任何一个job或application必须运行在一个或多个Container中，在Yarn框架中，ResourceManager只负责告诉ApplicationMaster哪些Containers可以用，ApplicationMaster还需要去找NodeManager请求分配具体的Container。

### 3.2 Node Manager

****NodeManager****进程运行在集群中的节点上，每个节点都会有自己的NodeManager。NodeManager是一个slave服务：它负责接收ResourceManager的资源分配请求，分配具体的Container给应用。同时，它还负责监控并报告Container使用信息给ResourceManager。通过和ResourceManager配合，NodeManager负责整个Hadoop集群中的资源分配工作。ResourceManager是一个全局的进程，而NodeManager只是每个节点上的进程，管理这个节点上的资源分配和监控运行节点的健康状态。下面是NodeManager的具体任务列表：

- 接收ResourceManager的请求，分配Container给应用的某个任务

- 和ResourceManager交换信息以确保整个集群平稳运行。ResourceManager就是通过收集每个NodeManager的报告信息来追踪整个集群健康状态的，而NodeManager负责监控自身的健康状态。

- 管理每个Container的生命周期

- 管理每个节点上的日志

- 执行Yarn上面应用的一些额外的服务，比如MapReduce的shuffle过程

当一个节点启动时，它会向ResourceManager进行注册并告知ResourceManager自己有多少资源可用。在运行期，通过NodeManager和ResourceManager协同工作，这些信息会不断被更新并保障整个集群发挥出最佳状态。

NodeManager只负责管理自身的Container，它并不知道运行在它上面应用的信息。负责管理应用信息的组件是ApplicationMaster，在后面会讲到。

### 3.3 Resource Manager

ResourceManager主要有两个组件：Scheduler和ApplicationManager。

Scheduler是一个资源调度器，它主要负责协调集群中各个应用的资源分配，保障整个集群的运行效率。Scheduler的角色是一个纯调度器，它只负责调度Containers，不会关心应用程序监控及其运行状态等信息。同样，它也不能重启因应用失败或者硬件错误而运行失败的任务

Scheduler是一个可插拔的插件，它可以调度集群中的各种队列、应用等。在Hadoop的MapReduce框架中主要有两种Scheduler：Capacity Scheduler和Fair Scheduler，关于这两个调度器后面会详细介绍。

另一个组件ApplicationManager主要负责接收job的提交请求，为应用分配第一个Container来运行ApplicationMaster，还有就是负责监控ApplicationMaster，在遇到失败时重启ApplicationMaster运行的Container。

### 3.4 Application Master

ApplicationMaster的主要作用是向ResourceManager申请资源并和NodeManager协同工作来运行应用的各个任务然后跟踪它们状态及监控各个任务的执行，遇到失败的任务还负责重启它。

在MR1中，JobTracker即负责job的监控，又负责系统资源的分配。而在MR2中，资源的调度分配由ResourceManager专门进行管理，而每个job或应用的管理、监控交由相应的分布在集群中的ApplicationMaster，如果某个ApplicationMaster失败，ResourceManager还可以重启它，这大大提高了集群的拓展性。

在MR1中，Hadoop架构只支持MapReduce类型的job，所以它不是一个通用的框架，因为Hadoop的JobTracker和TaskTracker组件都是专门针对MapReduce开发的，它们之间是深度耦合的。Yarn的出现解决了这个问题，关于Job或应用的管理都是由ApplicationMaster进程负责的，Yarn允许我们自己开发ApplicationMaster，我们可以为自己的应用开发自己的ApplicationMaster。这样每一个类型的应用都会对应一个ApplicationMaster，一个ApplicationMaster其实就是一个类库。这里要区分ApplicationMaster\*类库*和ApplicationMaster*实例\*，一个ApplicationMaster类库何以对应多个实例，就行java语言中的类和类的实例关系一样。总结来说就是，每种类型的应用都会对应着一个ApplicationMaster，每个类型的应用都可以启动多个ApplicationMaster实例。所以，在yarn中，是每个job都会对应一个ApplicationMaster而不是每类。

Yarn 框架相对于老的 MapReduce 框架什么优势呢？

- 这个设计大大减小了 ResourceManager 的资源消耗，并且让监测每一个 Job 子任务 (tasks) 状态的程序分布式化了，更安全、更优美。

- 在新的 Yarn 中，ApplicationMaster 是一个可变更的部分，用户可以对不同的编程模型写自己的 AppMst，让更多类型的编程模型能够跑在 Hadoop 集群中，可以参考 hadoop Yarn 官方配置模板中的 ``mapred-site.xml`` 配置。

- 对于资源的表示以内存为单位 ( 在目前版本的 Yarn 中，没有考虑 cpu 的占用 )，比之前以剩余 slot 数目更合理。

- 老的框架中，JobTracker 一个很大的负担就是监控 job 下的 tasks 的运行状况，现在，这个部分就扔给 ApplicationMaster 做了，而 ResourceManager 中有一个模块叫做 ApplicationsManager，它是监测 ApplicationMaster 的运行状况，如果出问题，会将其在其他机器上重启。

- Container 是 Yarn 为了将来作资源隔离而提出的一个框架。这一点应该借鉴了 Mesos 的工作，目前是一个框架，仅仅提供 java 虚拟机内存的隔离 ,hadoop 团队的设计思路应该后续能支持更多的资源调度和控制 , 既然资源表示成内存量，那就没有了之前的 map slot/reduce slot 分开造成集群资源闲置的尴尬情况。

## 四、Yarn request分析

### 4.1 应用提交过程分析

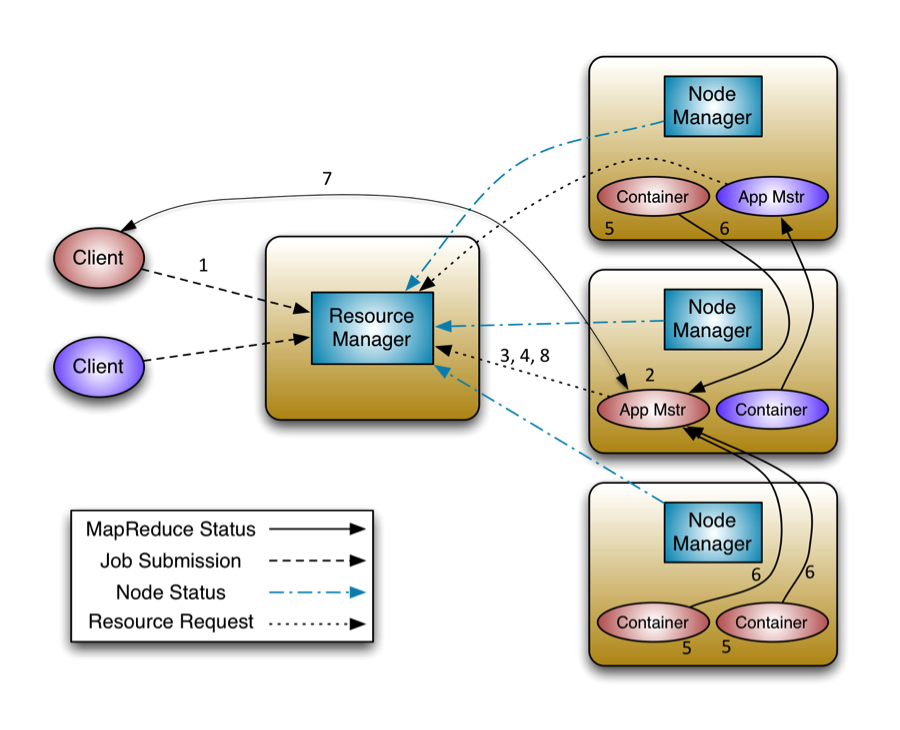
了解了上面介绍的这些概念，我们有必要看一下Application在Yarn中的执行过程，整个执行过程可以总结为三步：

1. 应用程序提交

2. 启动应用的ApplicationMaster实例

3. ApplicationMaster实例管理应用程序的执行

下面这幅图展示了应用程序的整个执行过程：



客户端程序向ResourceManager提交应用并请求一个ApplicationMaster实例

ResourceManager找到可以运行一个Container的NodeManager，并在这个Container中启动ApplicationMaster实例

ApplicationMaster向ResourceManager进行注册，注册之后客户端就可以查询ResourceManager获得自己ApplicationMaster的详细信息，以后就可以和自己的ApplicationMaster直接交互了

在平常的操作过程中，ApplicationMaster根据resource-request协议向ResourceManager发送resource-request请求

当Container被成功分配之后，ApplicationMaster通过向NodeManager发送container-launch-specification信息来启动Container， container-launch-specification信息包含了能够让Container和ApplicationMaster交流所需要的资料

应用程序的代码在启动的Container中运行，并把运行的进度、状态等信息通过application-specific协议发送给ApplicationMaster

在应用程序运行期间，提交应用的客户端主动和ApplicationMaster交流获得应用的运行状态、进度更新等信息，交流的协议也是application-specific协议

一但应用程序执行完成并且所有相关工作也已经完成，ApplicationMaster向ResourceManager取消注册然后关闭，用到所有的Container也归还给系统

### 4.2 Resource Request和Container

Yarn的设计目标就是允许我们的各种应用以共享、安全、多租户的形式使用整个集群。并且，为了保证集群资源调度和数据访问的高效性，Yarn还必须能够感知整个集群拓扑结构。为了实现这些目标，ResourceManager的调度器Scheduler为应用程序的资源请求定义了一些灵活的协议，通过它就可以对运行在集群中的各个应用做更好的调度，因此，这就诞生了****Resource Request****和****Container****。

具体来讲，一个应用先向ApplicationMaster发送一个满足自己需求的资源请求，然后ApplicationMaster把这个资源请求以resource-request的形式发送给ResourceManager的Scheduler，Scheduler再在这个原始的resource-request中返回分配到的资源描述Container。每个ResourceRequest可看做一个可序列化Java对象，包含的字段信息如下：

<resource-name, priority, resource-requirement, number-of-containers>

* 1
* 2
* resource-name：资源名称，现阶段指的是资源所在的host和rack，后期可能还会支持虚拟机或者更复杂的网络结构
* priority：资源的优先级
* resource-requirement：资源的具体需求，现阶段指内存和cpu需求的数量
* number-of-containers：满足需求的Container的集合

number-of-containers中的Containers就是ResourceManager给ApplicationMaster分配资源的结果。Container就是授权给应用程序可以使用某个节点机器上CPU和内存的数量。

ApplicationMaster在得到这些Containers后，还需要与分配Container所在机器上的NodeManager交互来启动Container并运行相关任务。当然Container的分配是需要认证的，以防止ApplicationMaster自己去请求集群资源。

Yarn相关的其它文章：

[Yarn内存分配管理机制及相关参数配置](http://blog.csdn.net/suifeng3051/article/details/48135521" \t "http://blog.csdn.net/suifeng3051/article/details/_blank)   
[Yarn 调度器Scheduler详解](http://blog.csdn.net/suifeng3051/article/details/49508261" \t "http://blog.csdn.net/suifeng3051/article/details/_blank)

本文参考文档

[https://www.safaribooksonline.com/library/view/hadoop-the-definitive/9781491901687/ch04.html](https://www.safaribooksonline.com/library/view/hadoop-the-definitive/9781491901687/ch04.html" \t "http://blog.csdn.net/suifeng3051/article/details/_blank)   
[https://jiaminglin.gitbooks.io/hadoopconceptsnote/content/chapter\_3\_yarn\_and\_mapreduce/resource\_request.html](https://jiaminglin.gitbooks.io/hadoopconceptsnote/content/chapter_3_yarn_and_mapreduce/resource_request.html" \t "http://blog.csdn.net/suifeng3051/article/details/_blank)