yarn 编辑

Apache Hadoop YARN （Yet Another Resource Negotiator，另一种资源协调者）是一种新的 Hadoop 资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。

中文名	YARN	外文名	YARN

目录

术语解释 编辑

YARN的基本思想是将JobTracker的两个主要功能（资源管理和作业调度/监控）分离，主要方法是创建一个全局的ResourceManager（RM）和若干个针对应用程序的ApplicationMaster（AM）。这里的应用程序是指传统的MapReduce作业或作业的DAG（有向无环图）。

YARN 分层结构的本质是 ResourceManager。这个实体控制整个集群并管理应用程序向基础计算资源的分配。ResourceManager 将各个资源部分（计算、内存、带宽等）精心安排给基础 NodeManager（YARN 的每节点代理）。ResourceManager 还与 ApplicationMaster 一起分配资源，与 NodeManager 一起启动和监视它们的基础应用程序。在此上下文中，ApplicationMaster 承担了以前的 TaskTracker 的一些角色，ResourceManager 承担了 JobTracker 的角色。

ApplicationMaster 管理一个在 YARN 内运行的应用程序的每个实例。ApplicationMaster 负责协调来自 ResourceManager 的资源，并通过 NodeManager 监视容器的执行和资源使用（CPU、内存等的资源分配）。请注意，尽管目前的资源更加传统（CPU 核心、内存），但未来会带来基于手头任务的新资源类型（比如图形处理单元或专用处理设备）。从 YARN 角度讲，ApplicationMaster 是用户代码，因此存在潜在的安全问题。YARN 假设 ApplicationMaster 存在错误或者甚至是恶意的，因此将它们当作无特权的代码对待。

NodeManager 管理一个 YARN 集群中的每个节点。NodeManager 提供针对集群中每个节点的服务，从监督对一个容器的终生管理到监视资源和跟踪节点健康。MRv1 通过插槽管理 Map 和 Reduce 任务的执行，而 NodeManager 管理抽象容器，这些容器代表着可供一个特定应用程序使用的针对每个节点的资源。YARN 继续使用 HDFS 层。它的主要 NameNode 用于元数据服务，而 DataNode 用于分散在一个集群中的复制存储服务。

要使用一个 YARN 集群，首先需要来自包含一个应用程序的客户的请求。ResourceManager 协商一个容器的必要资源，启动一个 ApplicationMaster 来表示已提交的应用程序。通过使用一个资源请求协议，ApplicationMaster 协商每个节点上供应用程序使用的资源容器。执行应用程序时，ApplicationMaster 监视容器直到完成。当应用程序完成时，ApplicationMaster 从 ResourceManager 注销其容器，执行周期就完成了。 [1]

基本缺陷 编辑

MapReduce 的第一个版本既有优点也有缺点。MRv1 是目前使用的标准的大数据处理系统。但是，这种架构存在不足，主要表现在大型集群上。当集群包含的节点超过 4,000 个时（其中每个节点可能是多核的），就会表现出一定的不可预测性。其中一个最大的问题是级联故障，由于要尝试复制数据和重载活动的节点，所以一个故障会通过网络泛洪形式导致整个集群严重恶化。

但 MRv1 的最大问题是多租户。随着集群规模的增加，一种可取的方式是为这些集群采用各种不同的模型。MRv1 的节点专用于 Hadoop，所以可以改变它们的用途以用于其他应用程序和工作负载。当大数据和 Hadoop 成为云部署中一个更重要的使用模型时，这种能力也会增强，因为它允许在服务器上对 Hadoop 进行物理化，而无需虚拟化且不会增加管理、计算和输入/输出开销。

主要优点 编辑

大大减小了 JobTracker（也就是现在的 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 ApplicationsMasters(注意不是 ApplicationMaster)，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。

核心思想

编辑

将JobTracker和TaskTracker进行分离，它由下面几大构成组件：

a. 一个全局的资源管理器 ResourceManager

b.ResourceManager的每个节点代理 NodeManager

c. 表示每个应用的 ApplicationMaster

d. 每一个ApplicationMaster拥有多个Container在NodeManager上运行 [2]

主要架构

编辑

ResourceManager（RM）：RM是一个全局的资源管理器，负责整个系统的资源管理和分配。它主要由两个组件构成：调度器（Scheduler）和应用程序管理器（Applications Manager，ASM）。

调度器 调度器根据容量、队列等限制条件（如每个队列分配一定的资源，最多执行一定数量的作业等），将系统中的资源分配给各个正在运行的应用程序。需要注意的是，该调度器是一个“纯调度器”，它不再从事任何与具体应用程序相关的工作，比如不负责监控或者跟踪应用的执行状态等，也不负责重新启动因应用执行失败或者硬件故障而产生的失败任务，这些均交由应用程序相关的ApplicationMaster完成。调度器仅根据各个应用程序的资源需求进行资源分配，而资源分配单位用一个抽象概念“资源容器”（Resource Container，简称Container）表示，Container是一个动态资源分配单位，它将内存、CPU、磁盘、网络等资源封装在一起，从而限定每个任务使用的资源量。此外，该调度器是一个可插拔的组件，用户可根据自己的需要设计新的调度器，YARN提供了多种直接可用的调度器，比如Fair Scheduler和Capacity Scheduler等。

应用程序管理器应用程序管理器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ApplicationMaster、监控ApplicationMaster运行状态并在失败时重新启动它等。

ApplicationMaster（AM）：用户提交的每个应用程序均包含一个AM，主要功能包括：

与RM调度器协商以获取资源（用Container表示）；

将得到的任务进一步分配给内部的任务(资源的二次分配)；

与NM通信以启动/停止任务；

监控所有任务运行状态，并在任务运行失败时重新为任务申请资源以重启任务。

当前YARN自带了两个AM实现，一个是用于演示AM编写方法的实例程序distributedshell，它可以申请一定数目的Container以并行运行一个Shell命令或者Shell脚本；另一个是运行MapReduce应用程序的AM—MRAppMaster。

注：RM只负责监控AM，在AM运行失败时候启动它，RM并不负责AM内部任务的容错，这由AM来完成。

NodeManager（NM）：NM是每个节点上的资源和任务管理器，一方面，它会定时地向RM汇报本节点上的资源使用情况和各个Container的运行状态；另一方面，它接收并处理来自AM的Container启动/停止等各种请求。

Container：Container是YARN中的资源抽象，它封装了某个节点上的多维度资源，如内存、CPU、磁盘、网络等，当AM向RM申请资源时，RM为AM返回的资源便是用Container表示。YARN会为每个任务分配一个Container，且该任务只能使用该Container中描述的资源。

注：1. Container不同于MRv1中的slot，它是一个动态资源划分单位，是根据应用程序的需求动态生成的。

2. 现在YARN仅支持CPU和内存两种资源，且使用了轻量级资源隔离机制Cgroups进行资源隔离。

YARN的资源管理和执行框架都是按主/从范例实现的——Slave ---节点管理器（NM）运行、监控每个节点，并向集群的Master---资源管理器(RM)报告资源的可用性状态，资源管理器最终为系统里所有应用分配资源。

特定应用的执行由ApplicationMaster控制，ApplicationMaster负责将一个应用分割成多个任务，并和资源管理器协调执行所需的资源，资源一旦分配好，ApplicationMaster就和节点管理器一起安排、执行、监控独立的应用任务。

需要说明的是， YARN不同服务组件的通信方式采用了事件驱动的异步并发机制，这样可以简化系统的设计。

主要架构

编辑

集中式架构

集中式调度器(Monolithic Scheduler)的特点是，资源的调度和应用程序的管理功能全部放到一个进程中完成，开源界典型的代表是MRv1 JobTracker的实现。这样设计的缺点很明显，扩展性差：首先，集群规模受限；其次，新的调度策略难以融入到现有代码中，比如之前仅支持MapReduce作业，现在要支持流式作业，而将流式作业的调度策略嵌入到中央调度其中是一项很难的工作。

双层调度架构

为了克服集中式调度器的不足，双层调度器是一种很容易被想到的解决之道，它可看作是一种分而治之的机制或者是策略下放机制：双层调度器仍保留一个经简化的集中式资源调度器，但具体任务相关的调度策略则下放到各个应用程序调度器完成。这种调度器的典型代表是Mesos。Mesos调度器由两部分组成，分别是资源调度器和框架(应用程序)调度器,其中，资源调度器负责将集群中的资源分配给各个框架(应用程序),而框架(应用程序)调度器负责将资源进一步分配给内部的各个任务，用户很容易将一种框架或者系统接入Mesos.

双层调度器的特点是：各个框架调度器并不知道整个集群资源使用情况，只是被动地接受资源；资源调度器仅将可用的资源推送给各个框架，而由框架自己选择是使用还是拒绝这些资源；一旦框架接受到新资源，再进一步将资源分配给其内部的任务，进而实现双层调度。然而这种调度器也是有缺点，主要表现在以下两个方面:1.各个框架无法知道整个集群的实时资源使用情况；采用悲观锁，并发粒度小。

参考资料

1. Hadoop 新 MapReduce 框架 Yarn 详解 ．IBM开发社区．2013-01-17[引用日期2015-07-09]

2. 将 Hadoop YARN 发扬广大 ．IBM developerWorks．2013-11-05[引用日期2015-07-09]

词条标签： 采矿工程