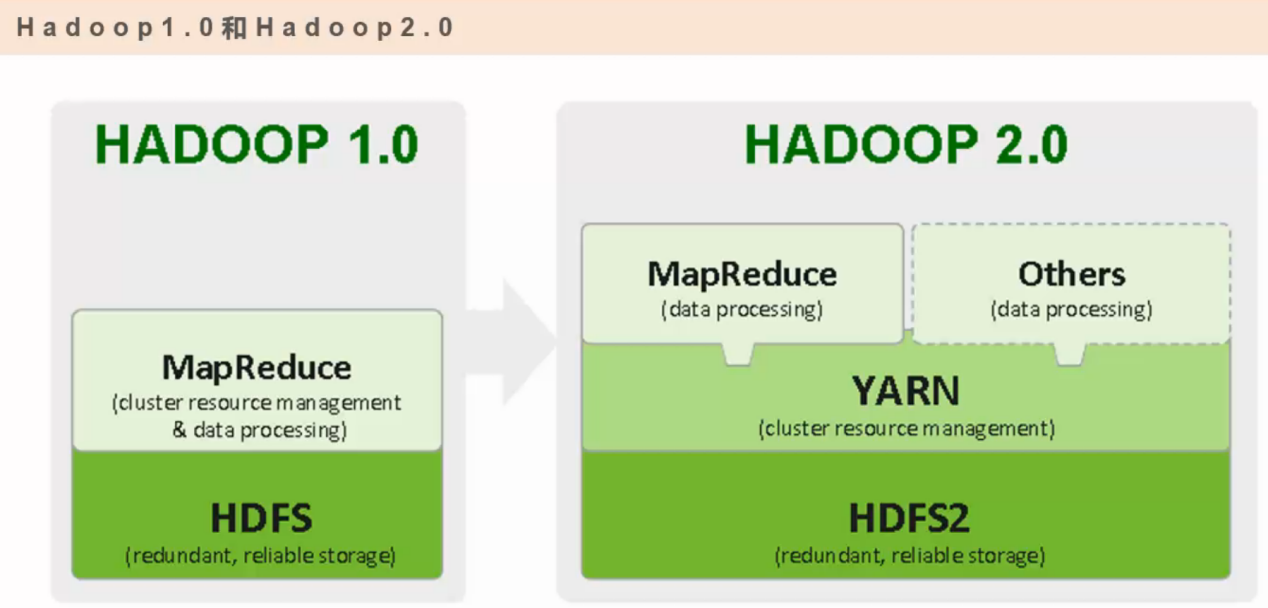
开始进入hadoop2.0平台或者是环境然后做了一个升级，这样的话大家过度可以有一个对比，然后进行一个学习，之前我们大概学了hadoop整个的生态架构里面的一些主流或者主要的一些模块和工具，那通过操作之后其实大概大家也掌握了一些不管是开发还是分析工具还是一些工具在某些场合在整个架构里面它处于什么功能，它是角色是怎么样，大概是有一个心里的一个预期了是吧？

然后这个时候带着这样的一个鲜艳的知识开始看一下我们2.0里面到底对之前这些知识到底有没有再哪个环节做了一些个不一样的地方。

那hadoop对于我们作为一个用户来说它提供了一些工具和应用程序并且它这个工具应用程序在hadoop内部它已经可以构建出一个相对来说广阔的一个生态系统是吧？那我们不管是hive,hbase，还是你开发一个mapreduce等等你都离不开你的hadoop对吧？我们已经做过大量的有关模块工具学习和使用之后，其实会有一个更深刻的体会。

那我们复习一下，就hadoop里面有两个最主要的组件是什么呢？一个是分布式文件系统和分布式文件的分布式集成框架对吧？文件系统就是hdfs，然后集成框架就是你的mapreduce是我们在hadoop1.0里面最主要的。

那我们看一下Yarn是什么呢？可以把它想象成mapreduce组件的一个新的替换，但是Yarn它又不是一个集成框架，但是它在某种程度又替换了你的mapreduce这么一个角色（如下图1所示）

图1：  


左侧的是一个Hadoop1.0的一个架构我们已经是很熟悉了是吧？那右侧的架构是给左侧的架构做了一次升级，升级完的架构就是变成右侧这个样子，那从这个右图的升级上来看变化还是很大的对吧？其中我们看一下有哪些变化有着大的变化，从左侧图底层HDFS，那原来在hadoop1.0里面，那这个hdfs我们暂且把它叫做hdfs1.0，在这个hdfs1.0里面其实它存在了明显的缺点和不足，缺点就主要在namenode上对吧？而且你namenode当时我们讲的1.0架构的时候，namenode里面保存着集群里面所有的元数据信息对吧？如果一旦出现namenode挂掉的话，然后有可能会丢数据的风险，而且它虽然有这个数据也有这种避免出现namenode挂掉之后丢数据的这么一个风险，但是从我们理解上来看或者是从原理来看，其实还是仍然避免不了理论上存在丢风险的这么一个可能，然后对于这样的一个问题，对于hadoop2.0里面或者HDFS2.0里面对这些部分就做了一定的完善！特别是针对刚才说的namenode存在这个单点问题，那在那个HDFS里面，它就不再存在这样的一个问题了，那怎么去解决这个问题呢？

在hdfs2.0里面它通过一个zookeeper的方式对namenode做了一个故障转移，然后它也有主也有从，所以通过这样的方式避免了你namenode挂掉之后导致你的数据突然的丢失这么一个风险，这是一个存储层面，如果我们看一下这个计算框架层面也就是mapreduce层面，那在hadoop1.0里面，这个mapreduce叫做mapreduce1.0，那在这个1.0里面发现这个mapreduce它在整个的架构里面它处于一个非常非常核心的一个角色，那如果说你把你的这个HDFS比喻成肉体的话，那这个mapreduce就是绝对的灵魂，但是在hadoop2.0里面就变了，变成了你这个人类的灵魂太单一了是吧？我想要更丰富的生活，比如说我喜欢读什么书就读什么书，想吃什么就吃什么，生活多姿多彩是吧？这个Hadoop2,0就想打造这么一个环境！目的就是说之前这个生活过的太单一了，我怎么能够让我的这个系统变的更丰富更强大，然后让我的这个系统变得扩展能力更强。

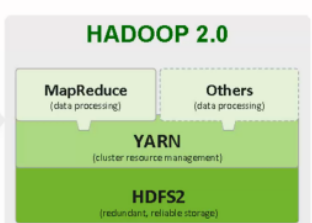
就是说我在1.0时代里面我不仅仅在我这个hadoop平台上，我不仅仅能够让mapreduce来跑我的代码，因为在hadoop里面，你想对hadoop编程，你只能走mapreduce对吧？并且你想走spark的话你就要单独再搭建一个集群，你想用这个流式计算框架storm，你得自己去搭建一个storm集群对不对？那相当于这几个集群之间是相互独立相互封闭，彼此之间你如果不去刻意打通的话，彼此之间是相互独立的，没有共享资源这么一说。

但是hadoop2.0里面想搞一件事情就是说我的平台不变，我的存储不变，但是我想让上面除了让mapreduce集成框架能够用我这个平台之外呢，我想让更多的其他的集成框架都在我的平台上能够运转，就是说我向我这个系统能够利用率最大化，我用同一套硬件可以同时运行mapreduce任务，也可以运行你的storm任务，也可以提交你的spark任务，这个时候你要这样的需求，你在hadoop1.0里面是做不到的。

于是针对这样的需求Yarn就诞生了！所以你从这个hadoop2.0这个相对比较复杂的结构里面，这个Yarn的角色就是非常的重要了，你从角色来看，其实你可以把这个Yarn能够比喻成一个操作系统，不管是什么操作系统，你底层都是磁盘，都是存数据地方是不是？然后在这个磁盘基础之上你再去格式化一层你的文件系统，这个文件系统可以被你的操作系统来管理一些设备是吧？然后你可以在这操作系统上面去执行你的代码，去跑你的程序对不对？

那相当于在Yarn这个操作系统上这个mapreduce就是一个非常非常普通的一个程序，那类似于这种集成框架的程序有很多，你可以扩展很多很多其他的一些计算框架，也就是说hadoop2.0里面你可以想象成底层是存储，中间是操作系统，上层是一些集成框架。（如下图2所示）

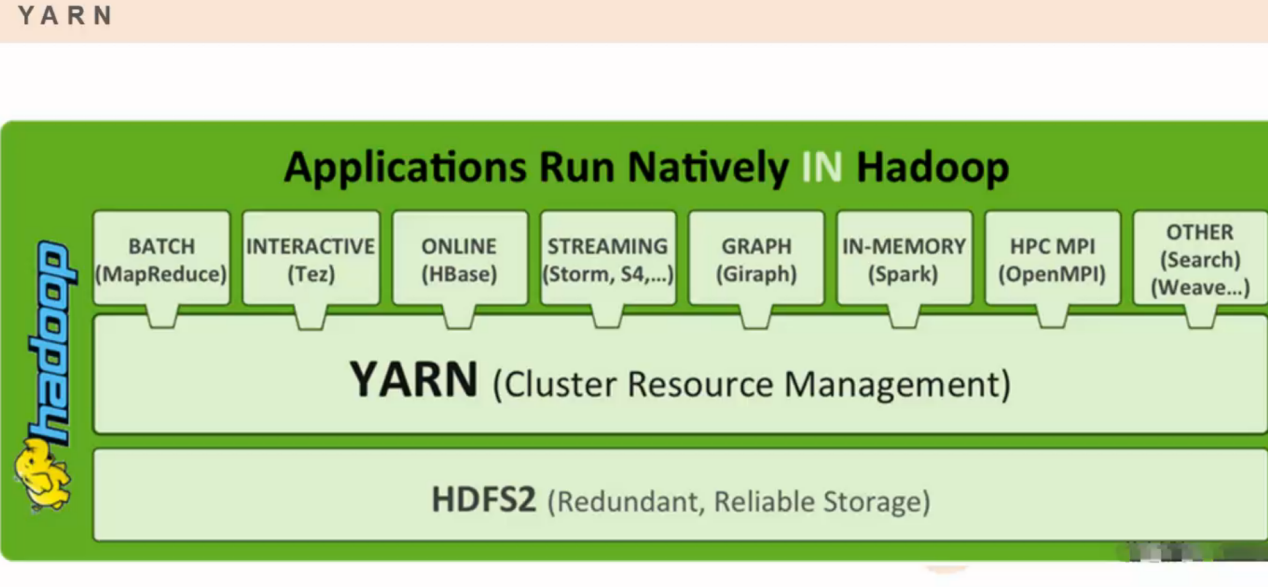
图2：



所以从这个角度来说你的hadoop2.0感觉比hadoop1.0来说，简直是丰富了很多的内容，那对比一看1.0这个hadoop太单薄了，这个2.0就变得更丰富了，而且它的扩展能力更强，这个时候你的mapreduce就变得很普通很普通的插件了，那之后你想在这上面去跑spark，那么spark也是当作插件直接插入进来就可以了，外部应用都通过这种插件的形式管理起来。

刚才我们看上图1和上图2都知道整个Yarn的一个角色和定位，Yarn的角色和定位处于操作系统这么一个级别，那么在下图3中也是一样

图3：



那对于Yarn来说你外围有很多很多的模块，那这些模块统统可以当作插件来对待，然后这个Yarn为各种应用程序，比如这里面有mapreduce的，有hbase的等等，它的目的就是Yarn就是为这些应用程序提供了一个最基本的服务来更好的利用这些底层的一些最基本的设施资源，怎么把这个资源用起来呢？尽可能用最大利用率的方式，就是尽可能不浪费任何有空闲的资源都把整体的那个服务调度起来，还有一个是动态的，就是说一个动态资源分配的一个过程，就相当于之前我们学mapreduce1.0时代，如果你这个mapreduce任务包含了一个map一个reduce，那你的map执行完之后，紧接着执行reduce，但是你reduce没有执行完的时候你的map所占用的资源它不释放，其实这个资源完成可以重新吐出来然后再重新的利用给其他的资源或者任务，所以这个就是一个资源提供的一个优化的一个点，当然还有其他，目的就是说能够尽可能大的利用起一些资源，所以从一个比较高的层次来看，其实Yarn就是一个集群操作系统。

所以你可以看到这个Yarn在整个的hadoop集群上它运行着不仅仅是mapreduce任务了，它还具备了其他的优势，有着更好的扩展性和集群利用率，让整个的系统能够扩展能力更强。

在这个Yarn里面引入了一些新的核心服务（如下图4所示）

图4：



这个核心服务有点类似于我们之前学的那个mapreduce1.0的时代，里面有两个核心的进程分别是JobTracker和TaskTracker这两个进程，咱现在说的这个Yarn跟这个HDFS你可以认为是没有什么关系，我们仅仅是从Yarn和mapreduce对比这么去看，所以在mapreduce1.0里面就是JobTracker和TaskTracker这是两个非常核心的服务，但是在Yarn里面它也有一些核心的服务，但这个核心的服务就不是再叫job和抓捕踹了，就是说这两个已经成为历史了，之后在hadoop2.0里面或者在Yarn里面就不存在这两个进程了。其中首先给大家介绍很重要的一个进程或者是模块，这个模块叫做ResourceManger，它是个进程。

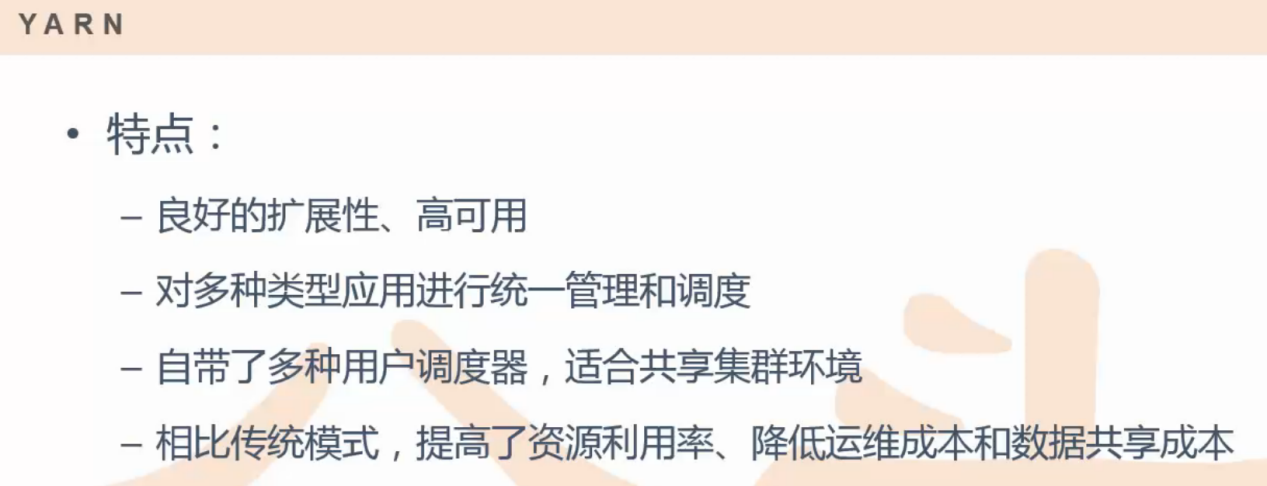
从这个名字就可以看得出来，就是资源管理器，那为了简单的记忆你就把它当作一个RM，它是集群资源的唯一的一个仲裁者，那大家想一想它的这种资源管理能力回到我们mapreduce1.0里面，哪一个进程可以做这样的事情？那肯定是JobTracker是不是？它在老的系统里面就是做这个资源管理分配的。

比如来了一个请求，那我就联系我的TaskTracker看看哪些机器上或者哪些节点上有空闲的就给他分配一些资源，那相当于RM就类似于旧的系统里面的JobTracker这么一个角色。

Hadoop2.0里面它不仅仅是对这个mapreduce做了彻底的重构，而且把这个mapreduce升级成了新的版本变成了MRv2或者是Yarn，原先的版本是MRv1，但是其实这个mapreduce你可以暂且把它当作为MRv2或者是Yarn这两个等价来看，其实这两个完成不是一回事。然后另外一个这个Yarn它为整个集群提供了资源管理和调度的功能，其实它完成这个工作后完成这个任务的大概一半的功劳都是来自于RM，另外一半的功劳暂且先不说，先跟大家留一个悬念，我们之后就揭晓。

然后它可以在这个Yarn上可以使得多个计算框架可以同时运行在一个集群上，然后以往我们提交一个mapreduce就是提交一个任务，但是现在在hadoop2.0里面这个名称就变了，就不再叫做job了，叫做application了。就是说你写了mapreduce或者你写了一个spark代码，你想在这个Yarn上跑一圈，那我提交了一个任务，其实也可以但是现在标准的描述就是我在你这个集群里面跑了一个application任务。

所以Yarn它在这里就有一些特点，我们在这里就简单的介绍一下（如下图5所示）

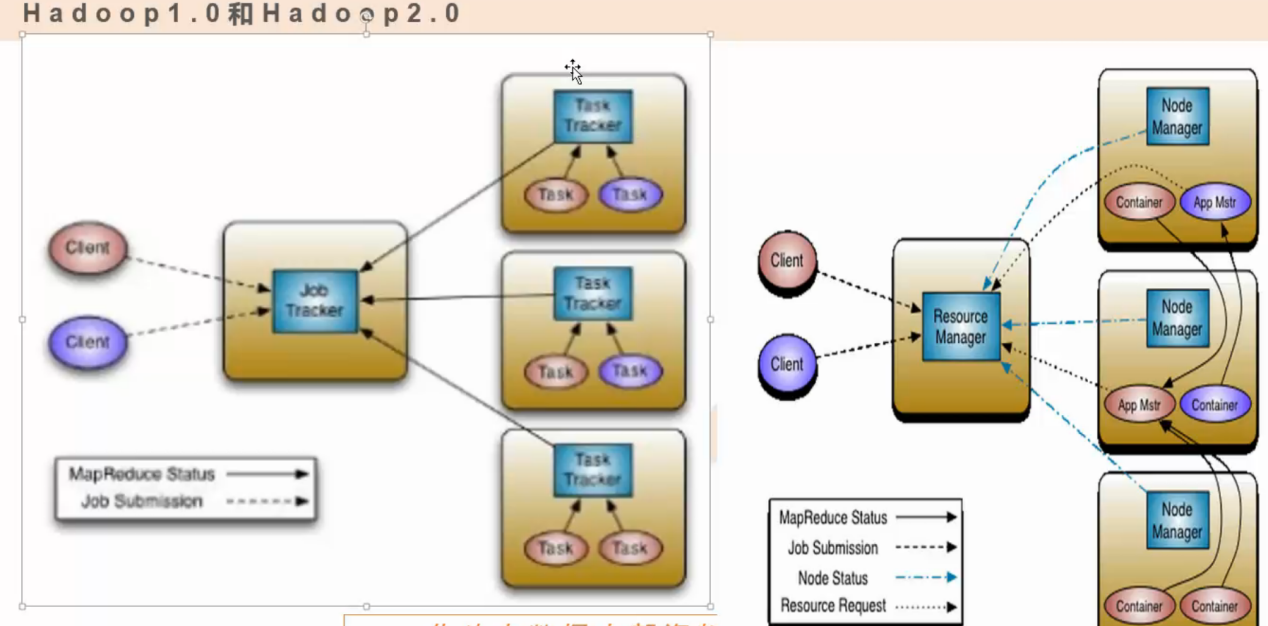
图5：  


其实这些都比较好理解，我们来说一下多种用户调度器，这个是干嘛用的呢？它的主要的工作就是你提交了一个任务，比如说我有一个集群，大家都在这个集群上提交任务，那有的人提交的任务可能是一个大的任务比较耗时，有的人提交一个小的任务，那怎么能够把这个合理的资源能够分配出来并且让大家都觉得都是公平的呢？是这么一个出发点，所以在Yarn里面它有一个公平用户调度器，这个我们后面还会说的。

然后相比传统模式，就是在Yarn里面会提高了整个资源的一个利用率，然后降低了运维成本和数据共享成本，就是说你比如像我们之前在跑那个mapreduce任务的时候，好了我们在hadoop1.0里面跑，那你跑完了之后可能这个集群就比较空闲了或者就是说你这个集群资源就有限的，那你集群就不仅仅是可以跑你的mapreduce，也可以跑你的流式计算，然后这个时候你比如说这个集群只要是资源没有达满，那我这个底层的这些存储还是比如这些cpu也好还是你的内存也好其实这些资源同时都可以给你的流式处理用也可以给你批量任务用，都是共享的，不是只是把这些资源仅限给某一个计算框架来用

然后我们再看一张图（如下图6所示）

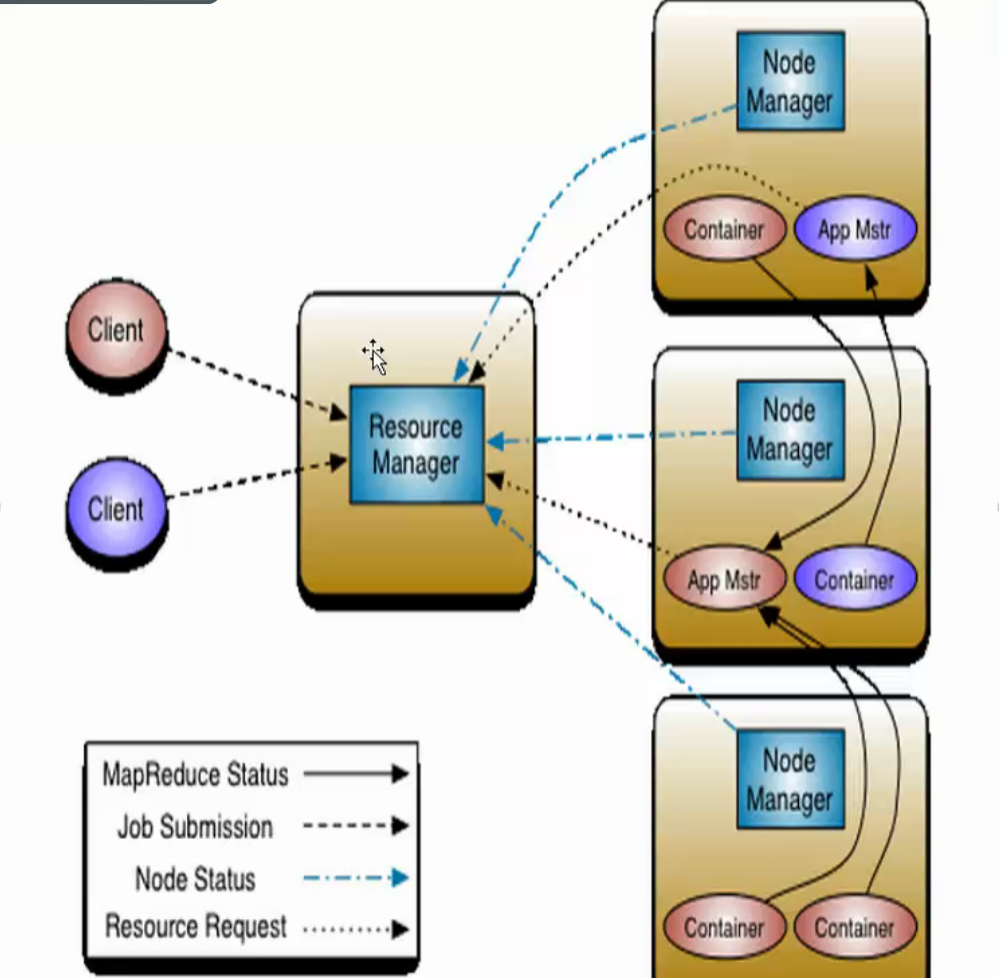
图6：



左边是一个hadoop1.0的一个架构，中间是一个JobTracker，右边都是一个从节点TaskTracker，然后每一个从节点上你跑的一个任务都是一个进程对吧？都是一个map或者是reduce这样的一个进程。

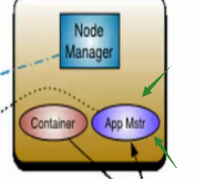
那么右边这个图就变的很复杂了，那为了理解，我们整个Yarn的这个处理流程，那其实你先对这个1.0如果是有一个比较深入的认识的话，你再去认识一个新的架构来说是一个比较

容易的事情。（如下图7所示）

图7：  


为此这个Yarn就是对你这个jobTracker拥有的绝对权力进行了肢解，就是说你在这个1.0里面你jobTracker是一个绝对的独裁份子，你这个手里掌握的权力太大了，你这个jobTracker不仅仅是一个资源分配或者是资源调度，就是说我可以给你任务可以提供哪些资源，它不仅仅提供资源，它还管你的任务的监控，就是jobTracker有两个功能一个就是资源管理和作业的一个调度和监控。那么资源关联和作业调度完全不是一个层次的东西，资源管理相当于是你对这个CPU内存，磁盘这些都是资源，相当于硬件资源这么一个层面，作业调度就相当于是一个进程是一个程序，这两个完全不是一个层次的东西对不对？好了你不仅要管资源你还要管作业调度，你这个权力有点太大了导致了你这个jobTracker这个任务有点太重了，一旦你这个jobTracker出现问题之后，你这个集群就很紧张了，所以你可以这么理解，就是说你在hadoop2.0里面在Yarn里面我想办法怎么把你这个1.0的jobTracker独裁者的权力给你肢解。

于是在Yarn里面就把这资源管理和作业调度这两个功能拆分了，就不再有同一个进程来管理了，那这个时候jobTracker是一个进程对不对？我这个时候竟然把这个功能拆解出来了，那我这不可能再有一个进程来管理了嘛对吧？那么这个时候资源管理是由2.0的RM来进行管理，然后另外一个AM来管理作业调度及监控，那么我们看看AM在图7中所在的位置在哪里（如下图8所示）

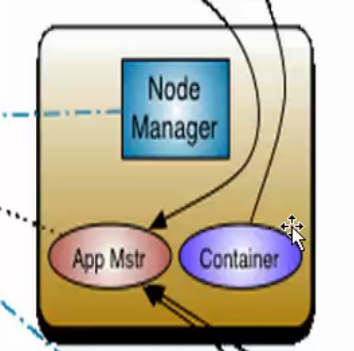
图8：  


它是在图中显得很小对吧？它叫做app master，，其实就叫application master，其实就是你应用的主节点，应用就是你任务的主节点，你的任务就是你的作业对吧？那由这个主来去控制你的作业和调度和监控，那大家想象一下我jobTracker也有这样的作业调度功能，但是我的jobTracker就在这个主上单节点上对不对？，所有的从节点上所有运行的任务你必须都向我汇报是吧？都必须把所有的这种情况或者你的任务的运转情况都得向我汇报，但是现在在2.0里面不一样了，这个AM不再是在master节点上去了，这个AM就是来管理你这个任务监控和调度的这个主是分布在不同的从节点上了，这是一个不一样的地方，这样做有什么好处呢？就是说你原来你这个1.0里面你这个所有的任务都必须和你这个单台机器去建立关系，然后使得对你这台机器要求就比较高对吧？

现在就不一样了，现在就是说你主可以晒满了整个的一个从节点上去了，那你这个从节点上这个主可以管理很多的你自己下面的下属，这个下属汇报的时候就没有必要再往那个大的主节点汇报了，就直接往你相应的从节点的主（application master）汇报就行了，相当于把整个的集群的这种性能负载都给你平庸化了，压力就减小了。

还有一个是NM，它的具体名字叫NodeManager，就是它这个工作的话有点像我们1.0里面的TaskTracker，TaskTracker是从节点的一个进程对吧？那么NodeManager也是每一个从节点上都要有这么一个进程来管理你当前本地的一些资源。

还有一个是RM的可以插拔的调度组件Scheduler，可以插拔就是说你可以换，就是这个组件它有很多种的调度组件策略，你可以通过一个非常简单的方式去让这个集群就变得另外一种调度的方式，就相当于更方便，然后这个Scheduler是负责运行中的各种应用分配资源，刚才我们说过了这个RM是你的资源管理系统，但是实际上它是有一个叫Scheduler来去做这样的事情，就相当于更细节的一些认识，但是这个Scheduler它是一个纯粹的调度器，你不要给它分配不属于调度的任务工作，它就是一个调度器，所以它不负责应用程序的监控和状态跟踪，它只给你分配资源，但是资源给你分配好了，你这个任务去跑了，你的任务是跑的好与不好，成不成功这个Scheduler调度器是完全不关心的，那Scheduler是根据你的应用需求进行一个资源调度，那么什么是资源呢？这个资源就是抽象起来就是Container，Container类似于一个容器对不对？（如下图9所示）

图9：  


这个Container是上面跑你真正的任务，比如你写一个mapreduce，这个Container就在Container这里面跑，然后这个Container肯定是跟你这1.0里面的Task一样的，它是在不同的从节点上启动相应的进程来去运行，这个Container它主要有两类，分别是CPU和内存。

好了有了这些资源之后，当然了Container是一个进程，那么这个进程是谁来启动的呢？是由你的Node Manager来启动，那为啥不是AM来启动呢？其实AM本质也是一个Container，这样就感觉特别通用了是吧？你这个Container里面跑的都是真正运行你的作业是不是？你的Node Manager是来启动这些资源的，Node Manager是用来做资源管理的，也就是管理你的Container，就是在合适的位置去启动一个资源Container，这个Container是由你的Node Manager来管理的，你的AM虽然本质是一个Container，你可以认为AM是一个很特殊的Container，肯定跟普通的Container不一样是不是？那也就是说它这个Container是所有普通的Container的一个管理者来监控用的，举一个例子，为了让大家更好的理解。

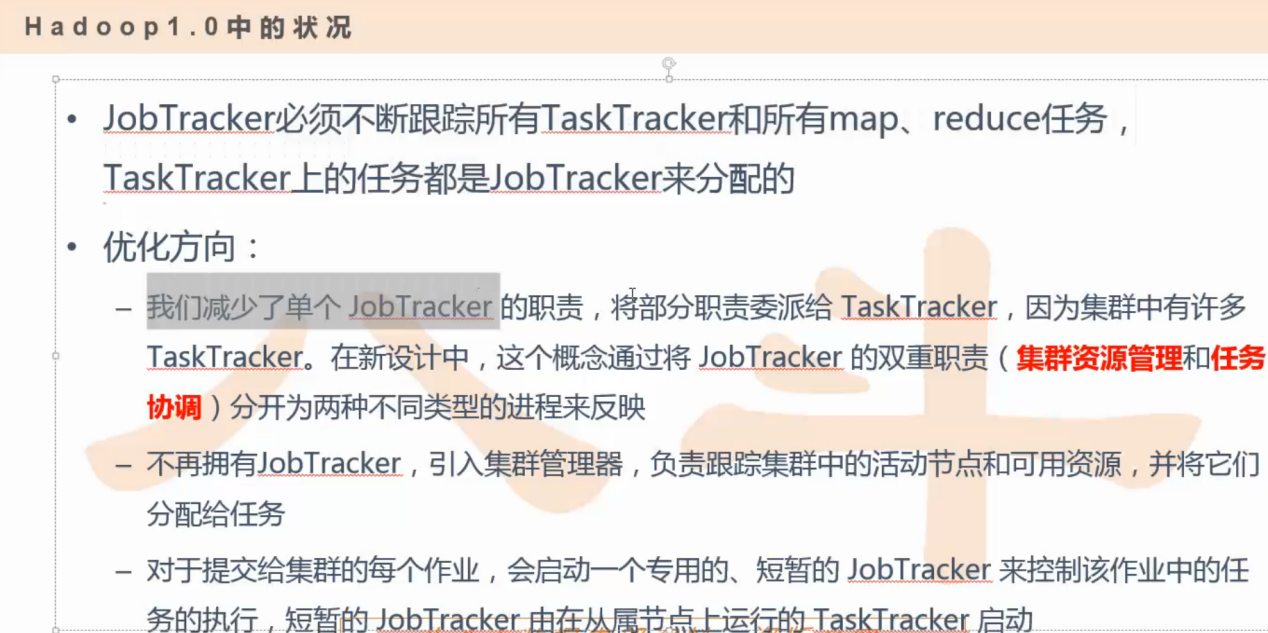
假设说这个RM是一个租房的房产中介，然后这个RM它有一个中介平台，在这个中介平台上有很多的一些租房信息，好了这个时候来了一个大客户，这个大客户想租房子开一个美容院，那么他得选位置对不对？那它要是开美容院的话肯定不会去只开一家，它肯定得开多家对不对？所以比如说在北京多个商圈里面人比较密集区，都设置相应的一个商铺的地点，而且这每一个店面它肯定是位置都设置在同一个地区对不对？然后每个商圈都有一个，然后你虽然这个每一个商场里面或者每一个商圈你都设了一个美容院，但是这个美容院的背后它肯定是有一个总公司有一个集团的这么一个概念，它必须有一个总部，这个总部必须得有一个办公楼或者办公室，而且这个办公楼它并不是那个对接普通的那些来美容的一些顾客来这做美容的，它就是来管理者的每一个商圈里面的美容店的一个总部，好了那这不管总部也好还是一些美容院的下者的一些分店也好，它们都得有一个租房需求是吧？起码有一个落脚的地方是吧？好了那这个不管是办公地点也好还是你的这个商铺也好它都是一个资源，起码你这个占地面积空间然后关于这个整个的一些个设计等等，这些所有都是资源，都是你的硬件资源是吧？

好了这个时候美容院的总部就相当于是一个AM，就是说来了一个大客户，这个大客户它需要租房但是它租房之前它连AM都没有，所以这里面就有一个问题了，就是说你的这个AM不是一直处于启动状态，就是它这个大客户来了，首先把你这个总部先定到某一个位置才能在不同的商圈才开你的分店对吧？好了这个时候你的AM你得先租个房子先租这么一个Container，然后RM是一个中间平台，根据你的AM需求我就提供相关资源，你比如说你想开十个分店，那么我根据你的需求根据你的选举我在我的中间平台里面去找适合你需求的租房信息资源对不对？找到了之后，我每一个商铺它都有一个地区经理是不是？你比如说朝阳，朝阳有一个总经理或者总销售，那么这个销售就是你的NM，你的NM是直接来服务你当前这个区域对接这区域的客户销售，那这个时候你的总部也有了，然后你这个分部也都起来了，那这个时候你的这个美容院就可以正式开业了对吧？但是这个时候虽然美容院已经开始运转起来了，但是你的NM你也不能闲着，你必须做点事情是不是？你必须得监控这些管辖区的这些Containe，就看一下我虽然我这个房子租给你开美容院，但是我的房子不能被打隔断是不是？，另外一个是你只能开美容院不能在里面去做一些个有明火的一些危险要求对吧？我必须要保证我这个房子是完整的，不能给我去做二次性的破坏，好了这个NM就是来管我自己的房子的。

这个时候AM是大客户的一个总经理，它这个总经理任务已经运转起来了，它美容院已经运转起来了，那这个总经理就要监督美容院的整个的产业是不是完好的运转对吧？比如说你这个朝阳的这个店面今天赚的钱有没有达到预期？这个美容院有没有跟顾客一些纠纷等等就是处于这种你真正生意相关的一些工作对不对？

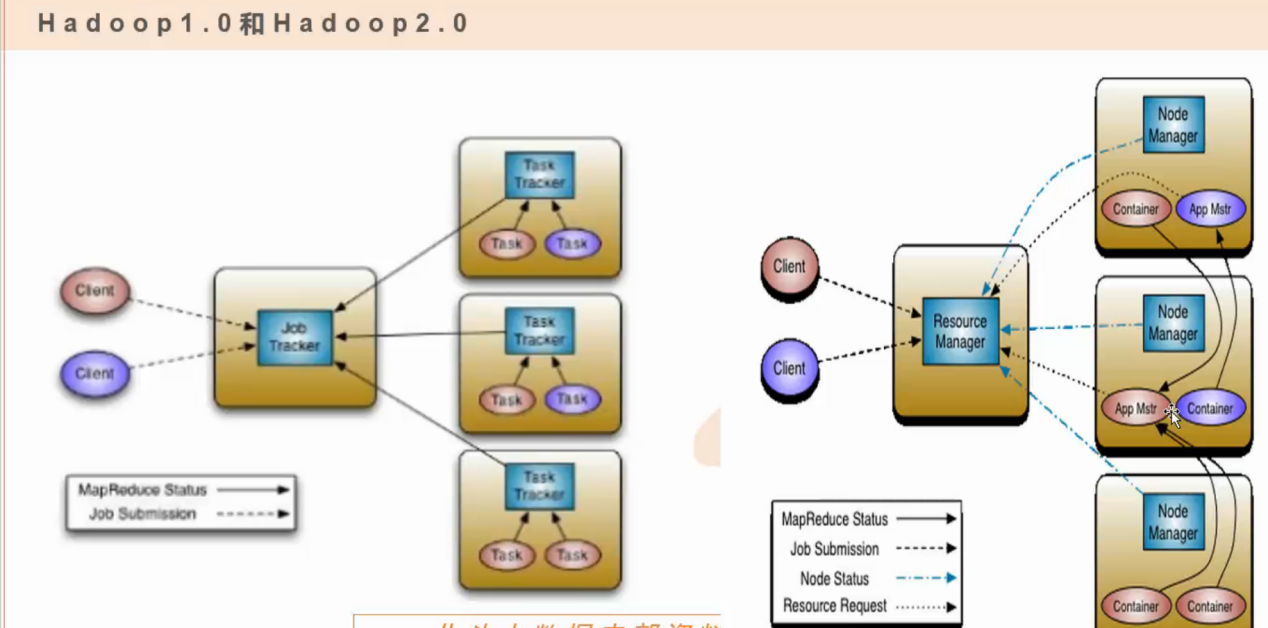
所以不管你是这个NM也好还是你这个AM也好，大家都不能闲着，NM是管理你自己的这个店面的，AM是真正管理你的这个任务的运转的，当然RM也不能闲着，就是这个NodeManager是一个大区的销售，它要定期的给你的这个总部就说我现在底下的这个区域这个房产是不是符合公司要求的一个运转情况，有没有被破坏，有什么异常情况或者是一些资源利用率都要上报给你的RM对不对？所以通过这个例子就可以很好的理解你这个RM，NM，Container，AM之间的一些个关系了。

然后hadoop1.0里面，这里面有一些东西刚才都跟大家说了（如下图10所示）

图10：  


这里就不再多说了哈。

你的AM都是在从上面启动，任务也是在从节点上运行（如下图11所示）

图11：  


那么这个任务统一，那么不管你是AM也好还是你的Container也好，那我们大家所有人在公司里面不管是你的待遇也好公司也好还是你的职位也好等等，所有关于工作职能相关的都是由你的HR来管是吧？这些都统一由HR来管对不对？所以这HR就是给你来分配工位或者是分配这些资源的。就是说不管你是AM坐在哪个角落，不管你这个AM是在哪个从节点上启动起来，你这个AM管理的这个团队你都可以随时去控制和支配它的，这个和你在这个运行在主节点上（RM）还是运行在从节点上，这个位置是没有关系的，只要是你这个人在，那你整个的这个项目整个团队都可以正常的进行一个运转。

然后接下来在2.0把所有的名词都告诉了大家了（如下图12所示）

图12：  


这RM,AM,NM三个是主要的核心进程，RM是主要来做资源管理的，AM是主要做一个作业管理和监控的对吧？NM是主要代替了TakTracker，是来管理你的资源，好了那么一个分布式应用程序就是代替了Mapreduce作业，就是你以前一个Mapreduce作业感觉是一个非常核心，是一个很高大上的一个角色，但是现在在2.0里面你这个mapreduce作业就变成了一个很普通的一个应用程序了。

所以我们最后有一句话就是说将你的JobTracker两个主要的功能分离成两个单独的组件，这两个功能就是一个资源管理和任务调度与监控，这个相当于算是整个Yarn里面的一个核心，主要的目的就是把你的这个JobTracker职能给你拆分了是吧？