spark：master、worker、executor和driver


刘栉风 关注
0.166 2019.07.25 11:08:15 字数 870 阅读 2,393

首先说一句，master和worker是物理节点，driver和executor是进程。

1，master和worker节点

搭建spark集群的时候我们就已经设置好了master节点和worker节点，一个集群有多个master节点和多个worker节点。

master节点常驻master守护进程，负责管理worker节点，我们从master节点提交应用。

worker节点常驻worker守护进程，与master节点通信，并且管理executor进程。

PS：一台机器可以同时作为master和worker节点（举个例子：你有四台机器，你可以选择一台设置为master节点，然后剩下三台设为worker节点，也可以把四台都设为worker节点，这种情况下，有一个机器既是master节点又是worker节点）

2，driver和executor进程

driver进程就是应用的main()函数并且构建sparkContext对象，当我们提交了应用之后，便会启动一个对应的driver进程，driver本身会根据我们设置的参数占有一定的资源（主要指cpu core和memory）。下面说一说driver和executor会做哪些事。

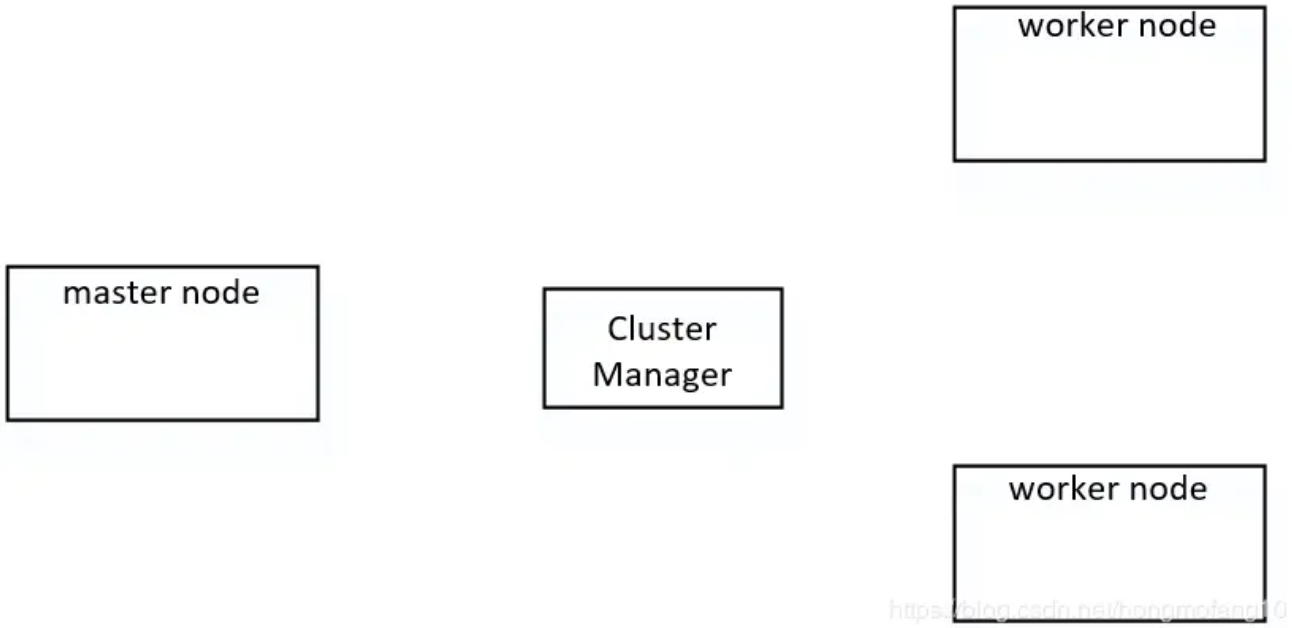
driver可以运行在master上，也可以运行worker上（根据部署模式的不同）。driver首先会向集群管理者（standalone、yarn，mesos）申请spark应用所需的资源，也就是executor，然后集群管理者会根据spark应用所设置的参数在各个worker上分配一定数量的executor，每个executor都占用一定数量的cpu和memory。在申请到应用所需的资源以后，driver就开始调度和执行我们编写的应用代码了。driver进程会将我们编写的spark应用代码拆分成多个stage，每个stage执行一部分代码片段，并为每个stage创建一批tasks，然后将这些tasks分配到各个executor中执行。

executor进程宿主在worker节点上，一个worker可以有多个executor。每个executor持有一个线程池，每个线程可以执行一个task，executor执行完task以后将结果返回给driver，每个executor执行的task都属于同一个应用。此外executor还有一个功能就是为应用程序中要求缓存的 RDD 提供内存式存储，RDD 是直接缓存在executor进程内的，因此任务可以在运行时充分利用缓存数据加速运算。

3，例图

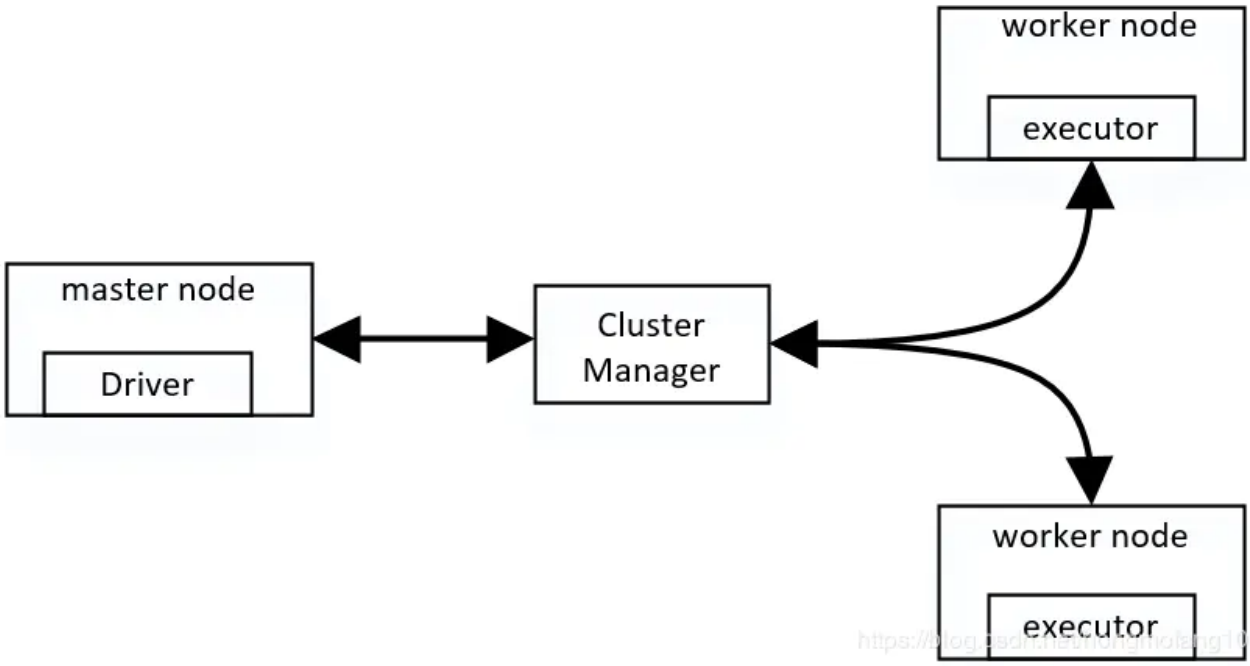
为了加深理解，我做了一个说明图，为了便于作图，假设了一种非常简单的集群模式。

（1）



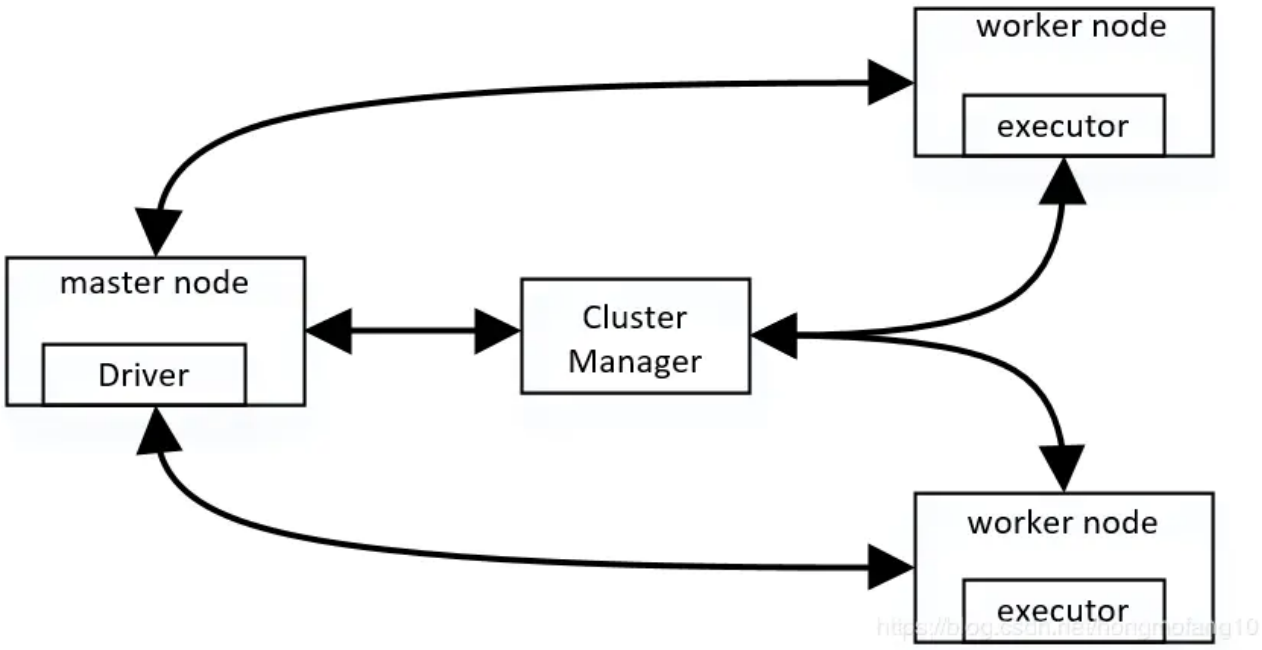
假设我们现在有一个集群，为了便于作图，我们设有一个master节点，两个worker节点，并且驱动程序运行在master节点上。

（2）



在master节点提交应用以后，在master节点中启动driver进程，driver进程向集群管理者申请资源（executor），集群管理者在不同的worker上启动了executor进程，相当于分配资源。

（3）



driver进程会将我们编写的spark应用代码拆分成多个stage，每个stage执行一部分代码片段，并为每个stage创建一批tasks，然后将这些tasks分配到各个executor中执行。