Apache Spark 的高性能一定程度上取决于它采用的异步并发模型（这里指 server/driver 端采用的模型），这与 Hadoop 2.0（包括 YARN 和 MapReduce）是一致的。Hadoop 2.0 自己实现了类似 Actor 的异步并发模型，实现方式是 epoll+状态机，而 Apache Spark 则直接采用了开源软件 Akka，该软件实现了 Actor 模型，性能非常高。尽管二者在 server 端采用了一致的并发模型，但在任务级别（特指 Spark 任务和 MapReduce 任务）上却采用了不同的并行机制：Hadoop MapReduce 采用了多进程模型，而 Spark 采用了多线程模型。

注意，本文的多进程和多线程，指的是同一个节点上多个任务的运行模式。无论是 MapReduce 和 Spark，整体上看，都是多进 程：MapReduce 应用程序是由多个独立的 Task 进程组成的；Spark 应用程序的运行环境是由多个独立的 Executor 进程构建的临时资源池构 成的。

多进程模型便于细粒度控制每个任务占用的资源，但会消耗较多的启动时间，不适合运行低延迟类型的作业，这是 MapReduce 广为诟病的原因之一。 而多线程模型则相反，该模型使得 Spark 很适合运行低延迟类型的作业。总之，Spark 同节点上的任务以多线程的方式运行在一个 JVM 进程中，可带来以 下好处：

1）任务启动速度快，与之相反的是 MapReduce Task 进程的慢启动速度，通常需要 1s 左右；

2）同节点上所有任务运行在一个进程中，有利于共享内存。这非常适合内存密集型任务，尤其对于那些需要加载大量词典的应用程序，可大大节省内存。

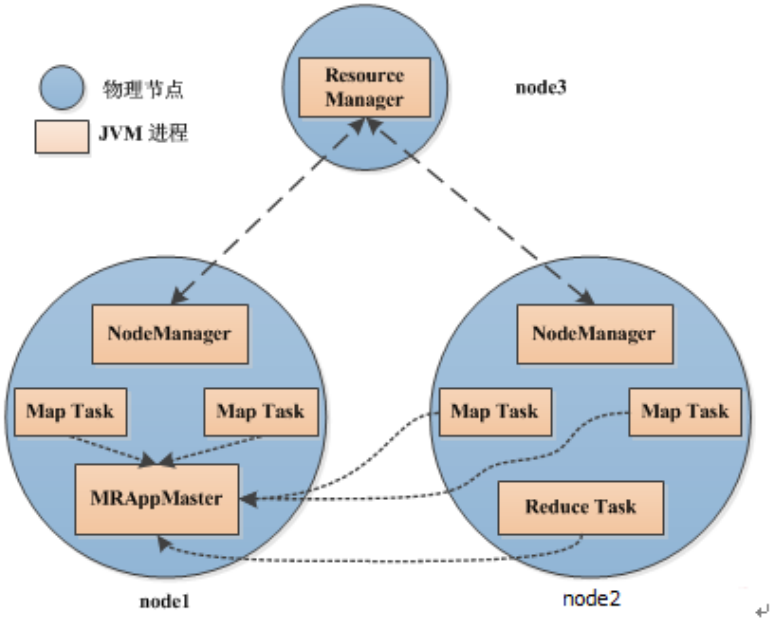
3）同节点上所有任务可运行在一个 JVM 进程(Executor)中，且 Executor 所占资源可连续被多批任务使用，不会在运行部分任务后释放 掉，这避免了每个任务重复申请资源带来的时间开销，对于任务数目非常多的应用，可大大降低运行时间。与之对比的是 MapReduce 中的 Task：每个 Task 单独申请资源，用完后马上释放，不能被其他任务重用，尽管 1.0 支持 JVM 重用在一定程度上弥补了该问题，但 2.0 尚未支持该功能。

尽管 Spark 的过线程模型带来了很多好处，但同样存在不足，主要有：

1）由于同节点上所有任务运行在一个进程中，因此，会出现严重的资源争用，难以细粒度控制每个任务占用资源。与之相反的是 MapReduce，它允 许用户单独为 Map Task 和 Reduce Task 设置不同的资源，进而细粒度控制任务占用资源量，有利于大作业的正常平稳运行。

下面简要介绍 MapReduce 的多进程模型和 Spark 的多线程模型。

（1） MapReduce 多进程模型

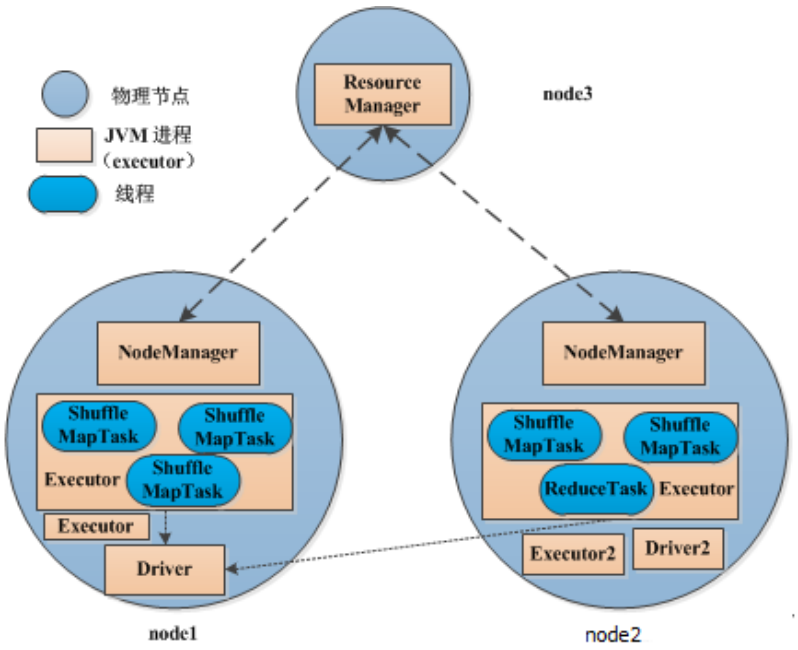


1） 每个 Task 运行在一个独立的 JVM 进程中；

2） 可单独为不同类型的 Task 设置不同的资源量，目前支持内存和 CPU 两种资源；

3） 每个 Task 运行完后，将释放所占用的资源，这些资源不能被其他 Task 复用，即使是同一个作业相同类型的 Task。也就是说，每个 Task 都要经历“申请资源—> 运行 Task –> 释放资源”的过程。

（2） Spark 多线程模型



1） 每个节点上可以运行一个或多个 Executor 服务；

2） 每个 Executor 配有一定数量的 slot，表示该 Executor 中可以同时运行多少个 ShuffleMapTask 或者 ReduceTask；

3） 每个 Executor 单独运行在一个 JVM 进程中，每个 Task 则是运行在 Executor 中的一个线程；

4） 同一个 Executor 内部的 Task 可共享内存，比如通过函数 SparkContext#broadcast 广播的文件或者数据结构只会在每个 Executor 中加载一次，而不会像 MapReduce 那样，每个 Task 加载一次；

5） Executor 一旦启动后，将一直运行，且它的资源可以一直被 Task 复用，直到 Spark 程序运行完成后才释放退出。

总体上看，Spark 采用的是经典的 scheduler/workers 模式，每个 Spark 应用程序运行的第一步是构建一个可重用的资源池，然后 在这个资源池里运行所有的 ShuffleMapTask 和 ReduceTask（注意，尽管 Spark 编程方式十分灵活，不再局限于编写 Mapper 和 Reducer，但是在 Spark 引擎内部只用两类 Task 便可表示出一个复杂的应用程序，即 ShuffleMapTask 和 ReduceTask），而 MapReduce 应用程序则不同，它不会构建一个可重用的资源池，而是让每个 Task 动态申请资源，且运行完后马上释放资源。