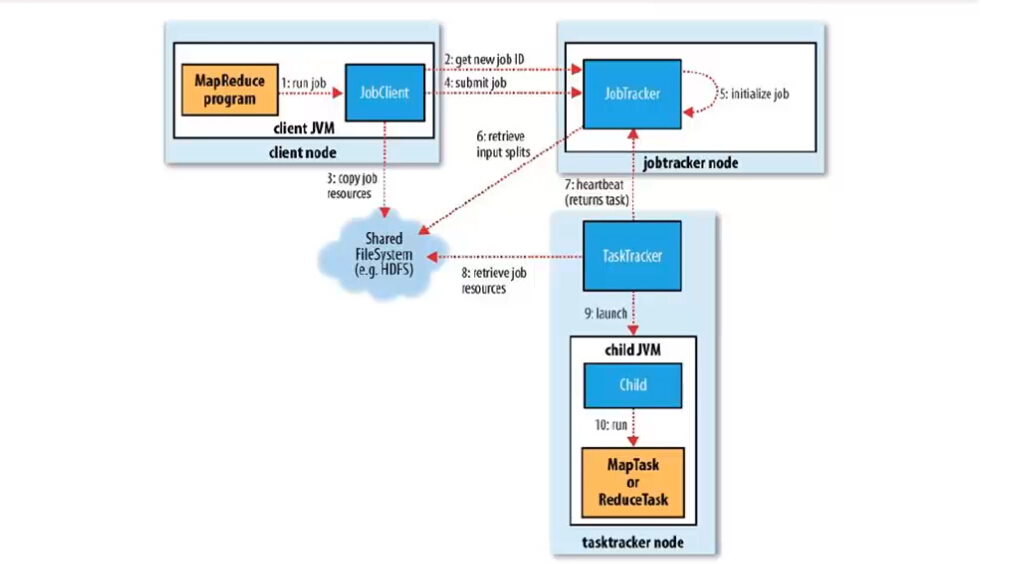
# MR1



工作流程主要分为6个步骤

## 作业的提交

1)客户端向jobtracker请求一个新的作业ID(通过JobTracker的getNewJobId()方法获取，见第2步

2)计算作业的输入分片，将运行作业所需要的资源(包括jar文件、配置文件和计算得到的输入分片)复制到一个以ID命名的jobtracker的文件系统中(HDFS)，见第3步

3)告知jobtracker作业准备执行，见第4步

## 作业的初始化

4)JobTracker收到对其submitJob()方法的调用后，会把此调用放入一个内部队列中，交由作业调度器进行调度，并对其初始化，见第5步

5)作业调度器首先从共享文件系统HDFS中获取客户端已经计算好的输入分片，见第6步

6)为每个分片创建一个map任务和reduce任务，以及作业创建和作业清理的任务。

## 任务的分配

7)tasktracker定期向jobtracker发送“心跳”，表明自己还活着。见第7步

8)jobtracker为tasktracker分配任务，对于map任务，jobtracker会考虑tasktracker的网络位置，选取一个距离其输入分片文件最近的tasktracker，对于reduce任务，jobtracker会从reduce任务列表中选取下一个来执行。

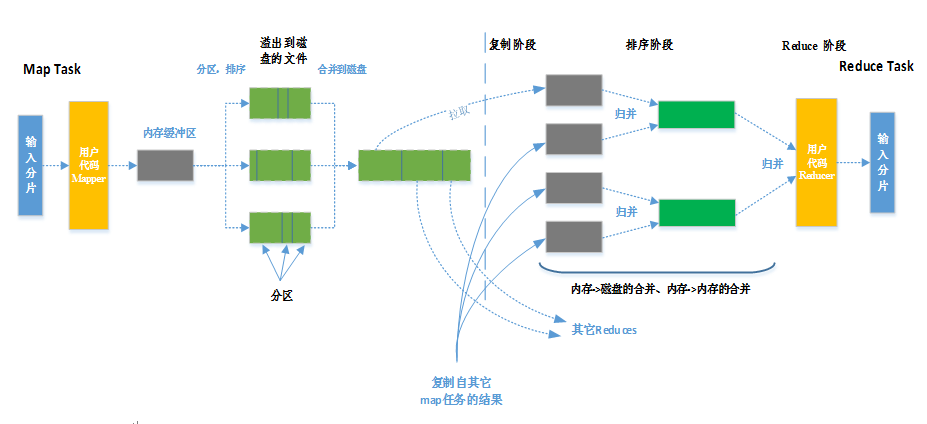
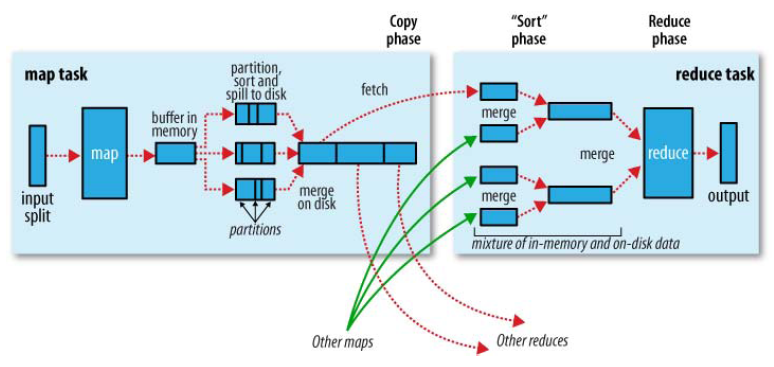
## 任务的执行

9)从HDFS中把作业的jar文件复制到tasktracker所在的文件系统，实现jar文件本地化，同时，tasktracker将应用程序所需的全部文件从分布式缓存中复制到本地磁盘，见第8步，并且tasktracker为任务新建一个本地工作目录，并把jar文件的内容解压到这个文件夹下，然后新建一个taskRunner实例运行该任务

10)TaskRunner启动一个新的JVM(见第9步)来运行每个任务(见第10步)

## 进度和状态的更新

11)任务运行期间，对其进度progress保持追踪。对map进度是已经处理输入所占的比例。对于reduce任务，分三部分，与shuffle的三个阶段相对应。Shuffle是系统执行排序的过程。是mapreduce的心脏。



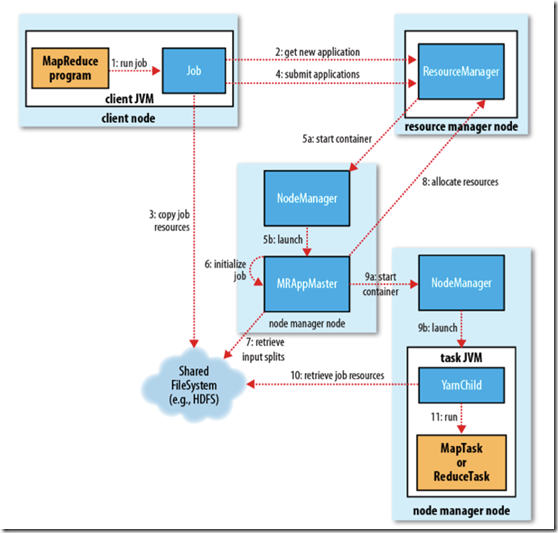
对于map端而言：每个map任务都有一个环形内存缓冲区，默认是0.8，当缓冲区达到阈值时便开始把内容溢出spill到磁盘，在写入磁盘之前，线程会根据数据最终要传的reducer把数据划分成相应的分区，每个分区中，按键值进行内排序，如果有combine (使结果更紧凑)，会在combine完成之后再写入磁盘。

对于reducer端而言，map的输出文件位于tasktracker的本地磁盘，每个map任务完成的时间可能不同，只要有一个完成，就会复制其输出(这就是复制阶段)，然后把map的输出进行merge合并，然后直接把数据输入到reduce函数，完成输出。

shuffle具体过程参见《hadoop问题整理》

## 作业的完成

# MR2（YARN）



## 工作流程主要分为以下6步执行：

作业的提交

1)、客户端向ResourceManager请求一个新的作业ID，ResourceManager收到后，回应一个ApplicationID，见第2步

2)、计算作业的输入分片，将运行作业所需要的资源(包括jar文件、配置文件和计算得到的输入分片)复制到一个(HDFS)，见第3步

3)、告知ResourceManager作业准备执行，并且调用submitApplication()提交作业，见第4步

## 作业的初始化

4)、ResourceManager收到对其submitApplication()方法的调用后，会把此调用放入一个内部队列中，交由作业调度器进行调度，并对其初始化，然后为该其分配一个contain容器，见第5步

5)、并与对应的NodeManager通信见第5a步，要求它在Contain中启动ApplicationMaster见第5b步

6)、ApplicationMaster启动后，会对作业进行初始化，并保持作业的追踪见第6步.

7)、ApplicationMaster从HDFS中共享资源，，接受客户端计算的输入分片为每个分片。见第7步

## 任务的分配

8)、ApplicationMaster向ResourceManager注册，这样就可以直接通过RM查看应用的运行状态，然后为所有的map和reduce任务获取资源，见第8步

## 任务的执行

9)、ApplicationMaster申请到资源后，与NodeManager进行交互，要求它在Contain容器中启动执行任务。见第9a、9b步

## 进度和状态的更新

10)、各个任务通过RPC协议umbilical接口向ApplicationMaster汇报自己的状态和进度，方便ApplicationMaster随时掌握各个任务的运行状态。用户也可以向ApplicationMaster查询运行状态。

## 作业的完成

1. 、应用完成后，ApplicationMaster向ResourceManager注销并关闭自己。

# 针对MR1的缺点，YARN解决了什么

MapReduce由以下缺点：

★ JobTracker挂掉，整个作业挂掉，存在单点故障

★ JobTracker既负责资源管理又负责作业控制，当作业增多时，JobTracker内存是扩展的瓶颈

★ map task全部完成后才能执行reduce task，造成资源空闲浪费

YARN设计考虑以上缺点，对MapReduce重新设计：

★ 将JobTracker职责分离，ResouceManager全局资源管理，ApplicationMaster管理作业的调度

★ 对ResouceManager做了HA设计

★ 设计了更细粒度的抽象资源容器Container

# 参考

<https://www.cnblogs.com/snowbook/p/5660572.html>