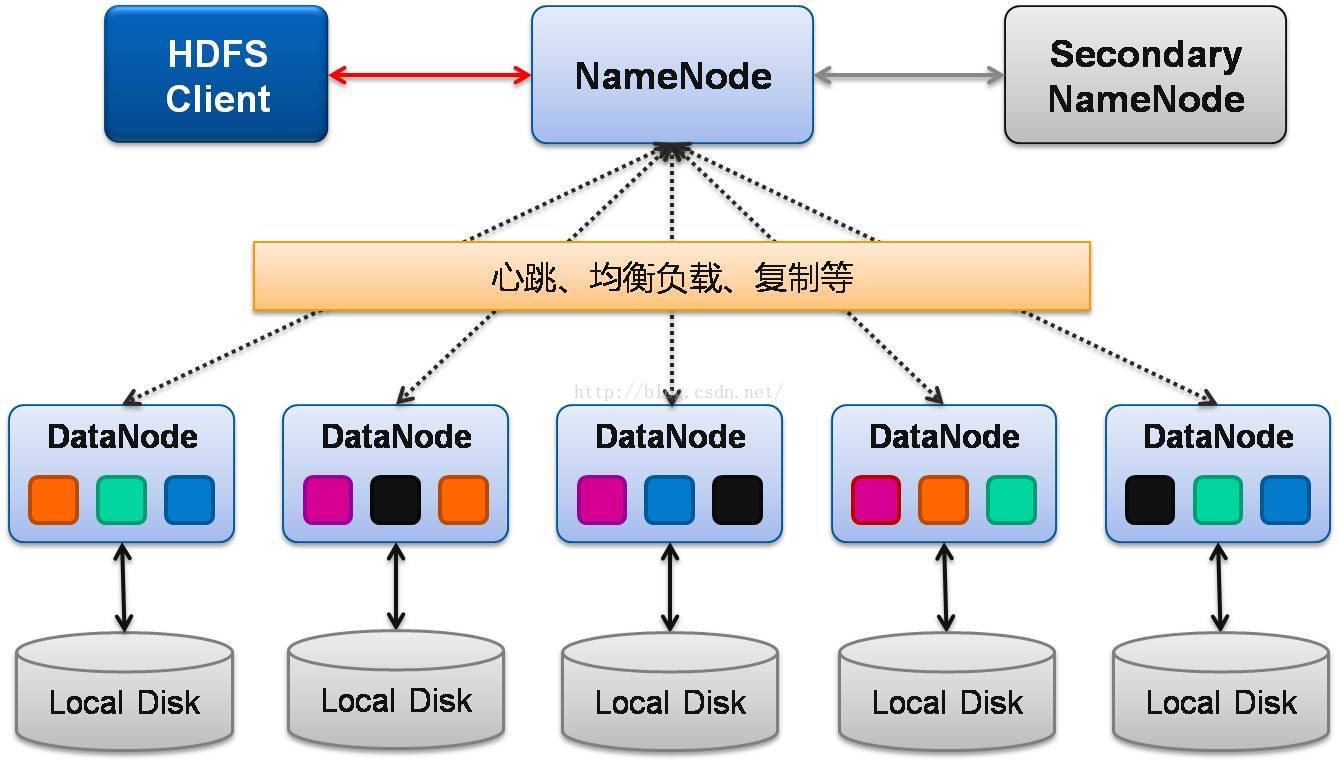
**Hadoop 架构**

**Hadoop关键组成：**

1. **分布式存储系统HDFS**：提供了高可靠性、高拓展性和高吞吐率的数据存储服务
2. **分布式计算框架MapReduce**：具有易于编程、高容错性和高拓展性等优点
3. **资源管理系统 Yarn**：负责集群资源的统一管理和调度
4. 底层支持 Common：为其他Hadoop模块提供基础设施

Hadoop的框架最核心的设计就是：HDFS和MapReduce。HDFS为海量的数据提供了存储，而MapReduce则为海量的数据提供了计算。

**Hadoop 架构——HDFS**



* HDFS分布式文件系统采用**主从结构**，一个NameNode作为主节点，多个DataNode作为从节点，NameNode存放文件的元数据，DataNode存放文件的数据块。

为了保证文件系统的可靠性，HDFS在一些做了**冗余**：

1. **备用NameNode**：通常有一台NameNode作为备用，定期与NameNode（active）进行数据同步，且实时监控NameNode（active）的状态，当发现NameNode（active）宕机，可立即顶替它的工作，防止NameNode单点故障导致的文件系统不可用。
2. **数据块有多个副本**：数据块通常副本数量为3，也就是一个数据块在HDFS中至少存在3个副本，以保证DataNode单点故障导致的数据丢失。

* NameNode通过**心跳检测**机制来确定DataNode是否可用。

心跳检测：每个Datanode节点**周期性**地向Namenode**发送心跳信号**。网络割裂可能导致一部分Datanode跟Namenode失去联系。Namenode通过心跳信号的缺失来检测这一情况，并将这些近期不再发送心跳信号Datanode标记为宕机，不会再将新的IO请求发给它们。任何存储在宕机Datanode上的数据将不再有效。

* 文件的写操作与副本的备份



1. 客户端在本机设立一个缓存对文件进行分块，向缓存输入文件数据流，直到缓存满停止。
2. 客户端向NameNode提交写文件请求。
3. NaneNode挑选最佳DataNode节点，然后将节点地址及写入映射位置返回给客户端。
4. 客户端与目标节点建立连接，传输数据块。
5. 于此同时DataNode寻找一个合适的DataNode建立隧道连接，转发接收到的数据，以建立副本。
6. 与5相同， 直到这个数据块传送完毕，此时已产生3个副本。
7. 一个数据块写入完成后，客户端向NameNode请求写入下一数据块，循环步骤3-6，直到文件写入完毕。
8. 当文件写入完毕，NameNode才更新元数据信息，此时文件才真正完成写入

* 文件的读操作



1. 客户端向NameNode提交读文件请求。
2. NameNode返回文件元数据和映射表。
3. 客户端去最近存放数据块的DataNode节点取数据块。
4. 拼接所有数据块，还原文件，完成文件的读取。

* 副本的存放机制

Hadoop认为，同一机架上的带宽远大于不同机架的带宽。HDFS的存放策略是将一个副本存放在本地机架的节点上，一个副本放在同一机架的另一个节点上，最后一个副本放在不同机架的节点上。这种策略减少了机架间的数据传输，这就提高了写操作的效率。机架的错误远远比节点的错误少，所以这个策略不会影响到数据的可靠性和可用性。于此同时，因为数据块只放在两个（不是三个）不同的机架上，所以此策略减少了读取数据时需要的网络传输总带宽。在这种策略下，副本并不是均匀分布在不同的机架上。三分之一的副本在一个节点上，三分之二的副本在一个机架上，其他副本均匀分布在剩下的机架中，这一策略在不损害数据可靠性和读取性能的情况下改进了写的性能。

* 安全模式

当NameNode启动后，会进入一个叫“安全模式”的状态中，NameNode会停止一切数据块复制的行为，然后通过统计从DataNode收到的心跳信息和块状态报告，计算“副本安全”的副本在所有副本中的占比（副本安全：实际副本数>=设定副本数? True：False），当占比大于设定的阈值，NameNode会退出“安全模式”，并对非“副本安全”的副本进行副本增殖。

* 集群均衡

当某DataNode的剩余空间比例小于设定的阈值，集群均衡系统将会转移该DataNode的部分数据至其他合适的DataNode节点，以缓解此DataNode 的存储压力。当对某个文件的请求突然增加，那么也可能启动一个计划创建该文件新的副本，并且同时重新平衡集群中的其他数据。

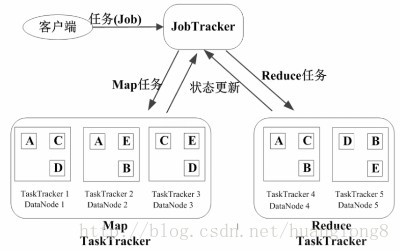
**Hadoop 架构——Map/Reduce**

MapReduce将应用划分为**Map**和**Reduce**两个步骤，其中Map对数据集上的独立元素进行指定的操作，生成**键-值对**形式的中间结果。Reduce则**对中间结果中相同“键”的所有“值”进行规约**，以得到最终结果。

通常，**Map/Reduce框架**和**分布式文件系统**是运行在一组**相同的节点上**的，也就是说，计算节点和存储节点通常在一起。这种配置允许框架在那些已经存好数据的节点上**高效地调度任务**，这可以使整个集群的**网络带宽**被非常**高效地利用**。

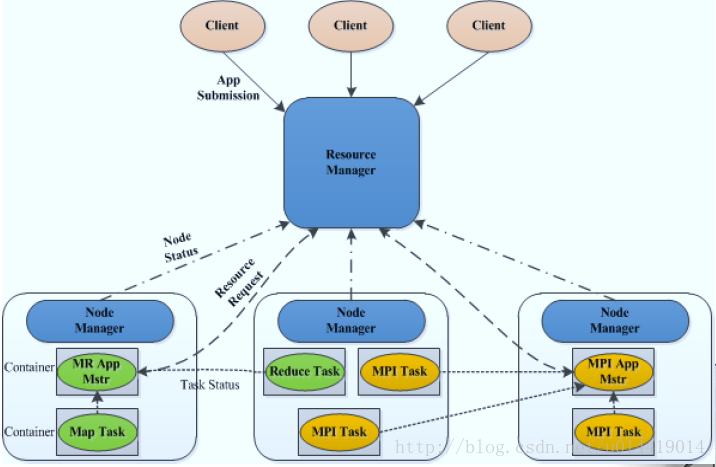
Map/Reduce框架由一个**单独的master JobTracker** 和**每**个集群**节点一个slave TaskTracker**共同组成。master负责**调度**构成一个作业的所有任务，这些任务**分布**在不同的slave上，master**监控**它们的执行，**重新执行**已经失败的任务。而**slave仅负责执行**由master指派的任务。

* MapReduce的运行流程图



1. Job & Task：1个Job分成多个Task
2. JobTracker：作业调度，分配任务、监控任务执行进度，监控TaskTracker的状态。
3. TaskTracker：执行任务，汇报任务状态

**Hadoop 架构——Yarn**

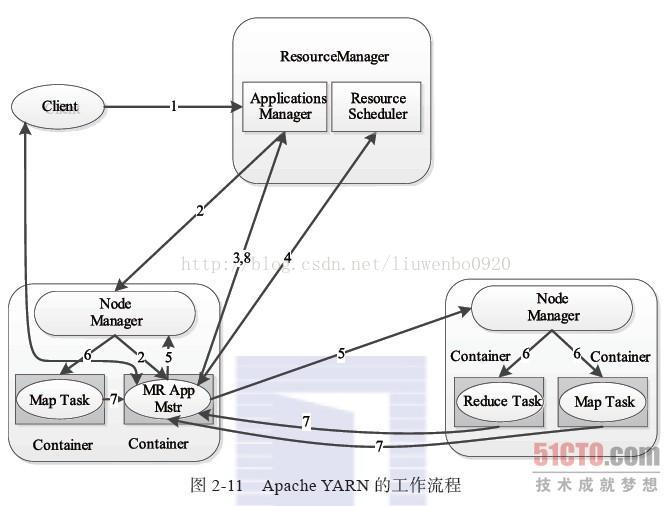


Yarn**负责集群的资源管理和调度**，使得**多种计算框架**可以运行在**一个集群**中。它有良好的**扩展性、高可用性**，对多种类型的应用程序进行统一管理和调度，自带多种多用户调度器，适合共享集群环境。

Yarn也是**主从结构**，ResourceManager整个集群中只有一个，负责**集群资源的统一管理和调度**。NodeManager整个集群有多个，负责**单节点**资源管理和使用。

当用户提交一个应用程序时，需要提供一个用以**跟踪和管理**这个程序的ApplicationMaster（每个应用只有一个，负责对应用程序的管理），它负责向ResourceManager申请资源，并要求NodeManger启动任务。

* **YARN工作流程**



1. 用户向YARN中提交应用程序，其中包括ApplicationMaster程序、启动ApplicationMaster的命令、用户程序等。
2. ResourceManager为该应用程序分配第一个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），并与对应的Node-Manager通信，要求它在这个Container中启动应用程序的ApplicationMaster。
3. ApplicationMaster首先向ResourceManager注册，这样用户可以直接通过ResourceManager查看应用程序的运行状态，然后它将为各个任务申请资源，并监控它的运行状态，直到运行结束。
4. ApplicationMaster采用轮询的方式通过RPC协议向ResourceManager申请和领取资源。
5. 一旦ApplicationMaster申请到资源后，便与对应的NodeManager通信，要求它启动任务。
6. NodeManager为任务设置好运行环境（包括环境变量、JAR包等）后，将任务启动命令写到一个脚本中，并通过运行该脚本启动任务。
7. 各个任务通过某个RPC协议向ApplicationMaster汇报自己的状态和进度，以让ApplicationMaster随时掌握各个任务的运行状态，从而可以在任务失败时重新启动任务。
8. 应用程序运行完成后，ApplicationMaster向ResourceManager注销并关闭自己。

* 可将YARN看做一个云操作系统，它负责为应用程序启动ApplicationMaster（相当于主线程），然后再由ApplicationMaster负责数据切分、任务分配、启动和监控等工作，而由ApplicationMaster启动的各个Task（相当于子线程）仅负责自己的计算任务。当所有任务计算完成后，ApplicationMaster认为应用程序运行完成，然后退出。

ps：蓝字部分为引用，HDFS学习通过官方说明文档，更新日期为2019/8/23，可能有些老了，有些它说没实现的功能，貌似已经在实现并运用了，如副本存放的决策机制。Map/Reduce的描述参照网络，我感觉没什么好说的，就是分而治之的策略，重点在于程序的编写方面。Yarn部分是我通过一篇大佬的文章学的，他讲述的已经很通俗且精简了，我觉得他写的很好，我再去重述一遍，感觉反而会有负面效果。

<https://blog.csdn.net/weixin_41988628/article/details/81083376?utm_medium=distribute.pc_relevant.none-task-blog-baidujs-1>

我开始是尝试录音来做这个作业的，结果录了很久，发现效果很差，想到什么讲什么，没系统化的流程，然后打算写下来汇总再录，结果写完发现，我把想说的都写了，录的话也就是读一遍，太尬了，所以作业就剩这篇word 了。