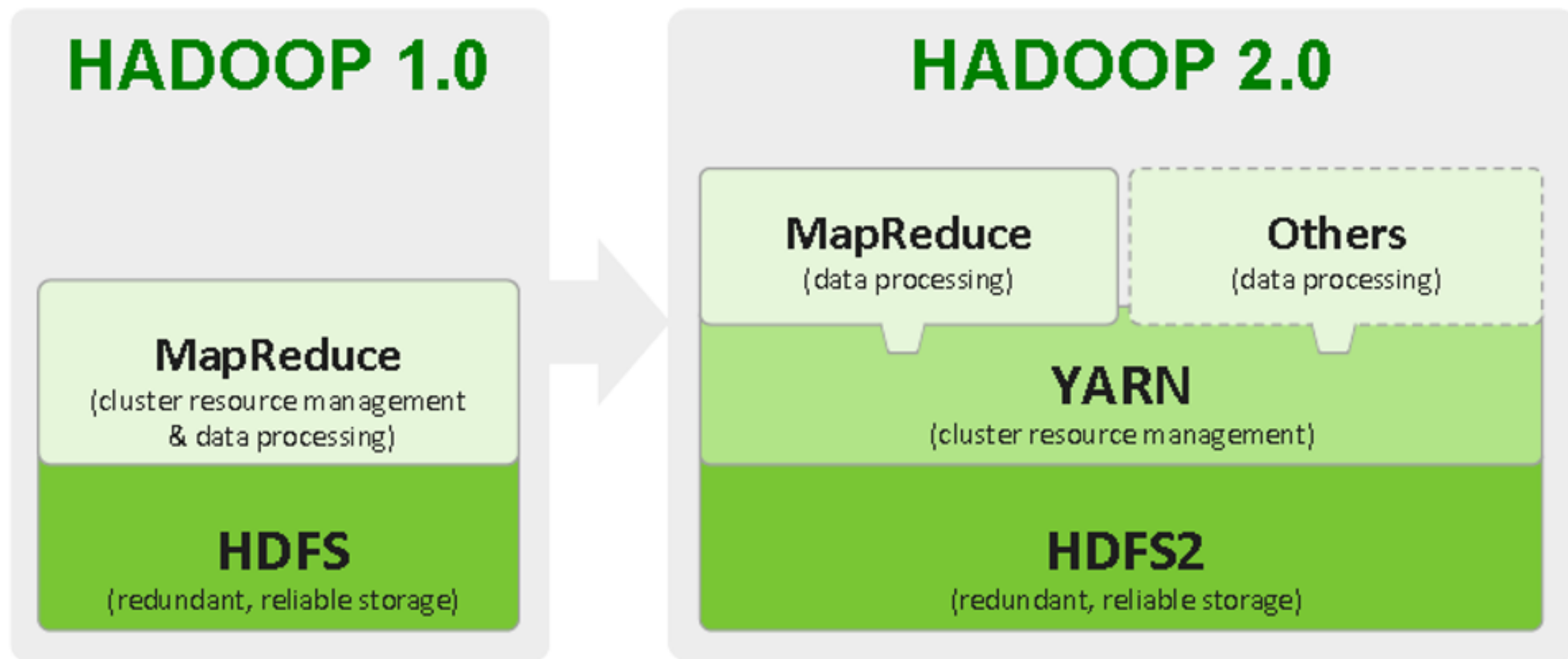

Yarn

Outline

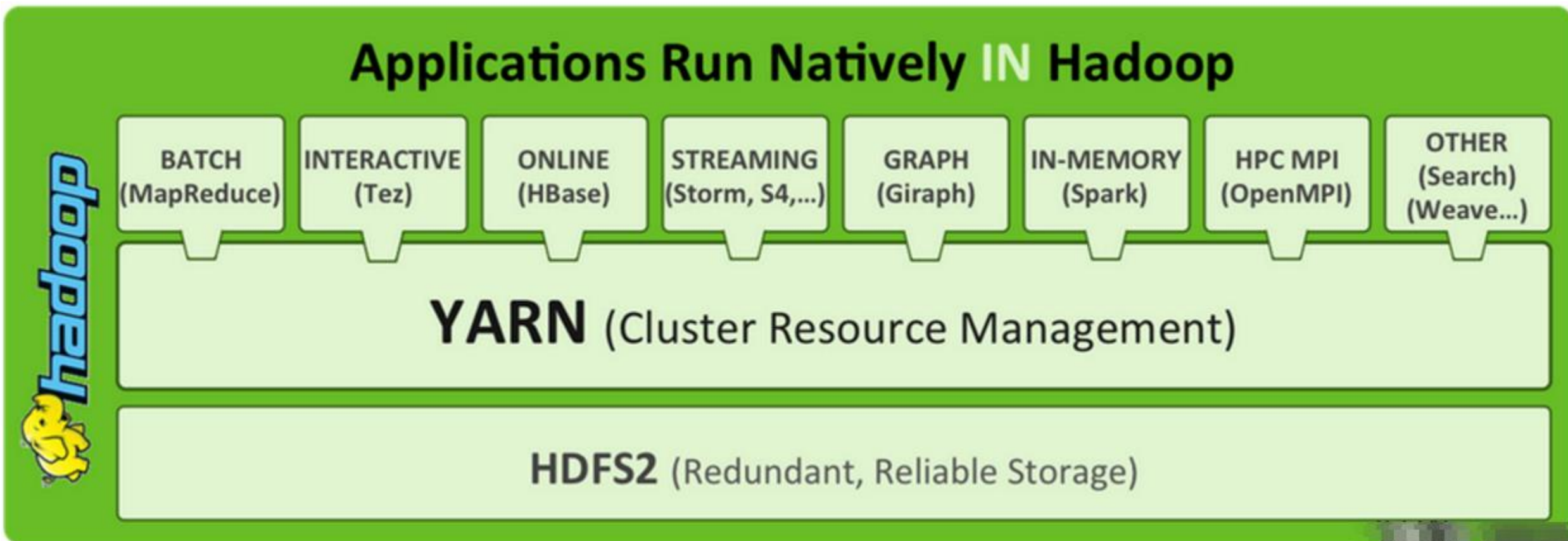
Yarn基础

【基础实践】Hadoop2.0集群搭建

Hadoop 1.0 和 Hadoop 2.0



YARN



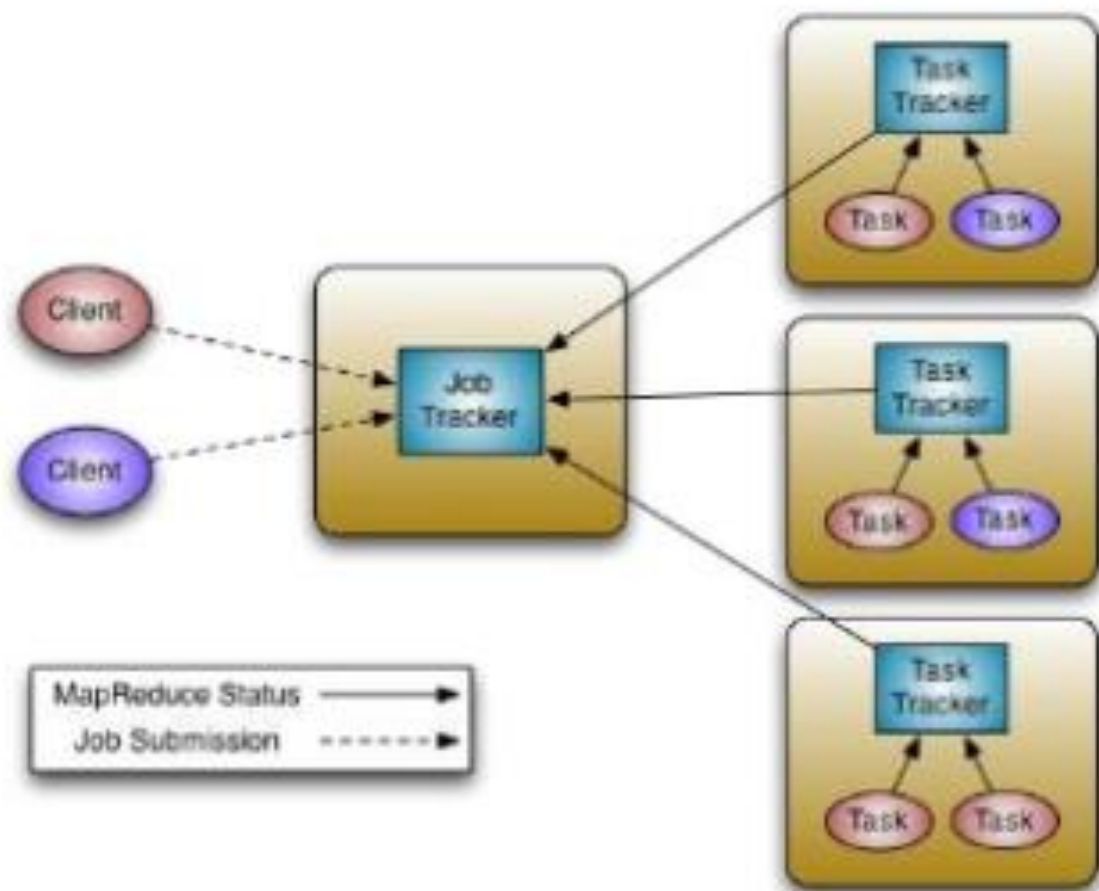
YARN (Yet Another Resource Negotiator)

- Hadoop集群的资源管理系统 (ResourceManger->RM)
- 更高级：集群操作系统
 - 为应用程序提供了基本服务来更好地利用大的、动态的、并行的基础设施资源
- Hadoop2.0对MapReduce框架做了彻底的重构
- Hadoop2.0中的MapReduce成为MRv2或者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
- 在Yarn中，Job的概念换成了application

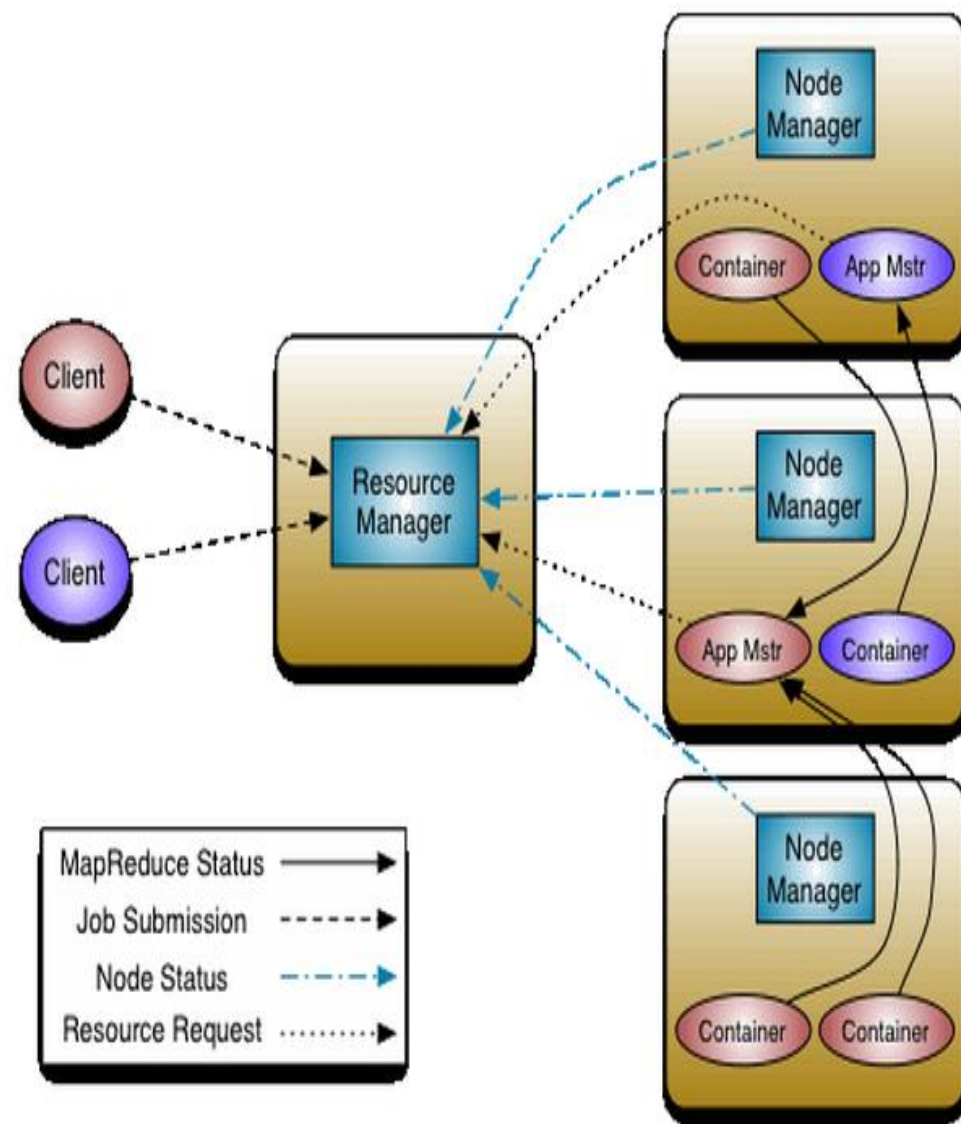
YARN

- 特点：
 - 良好的扩展性、高可用
 - 对多种类型应用进行统一管理和调度
 - 自带了多种用户调度器，适合共享集群环境
 - 相比传统模式，提高了资源利用率、降低运维成本和数据共享成本

Hadoop 1.0 和 Hadoop 2.0



——八斗大数据内部资料



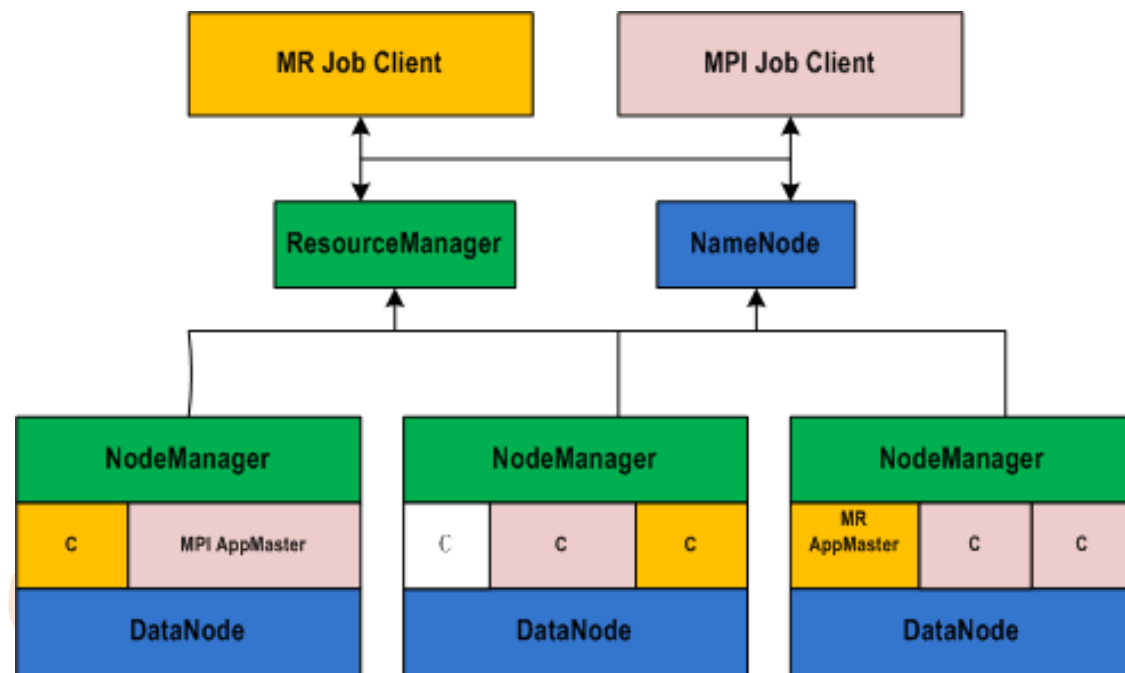
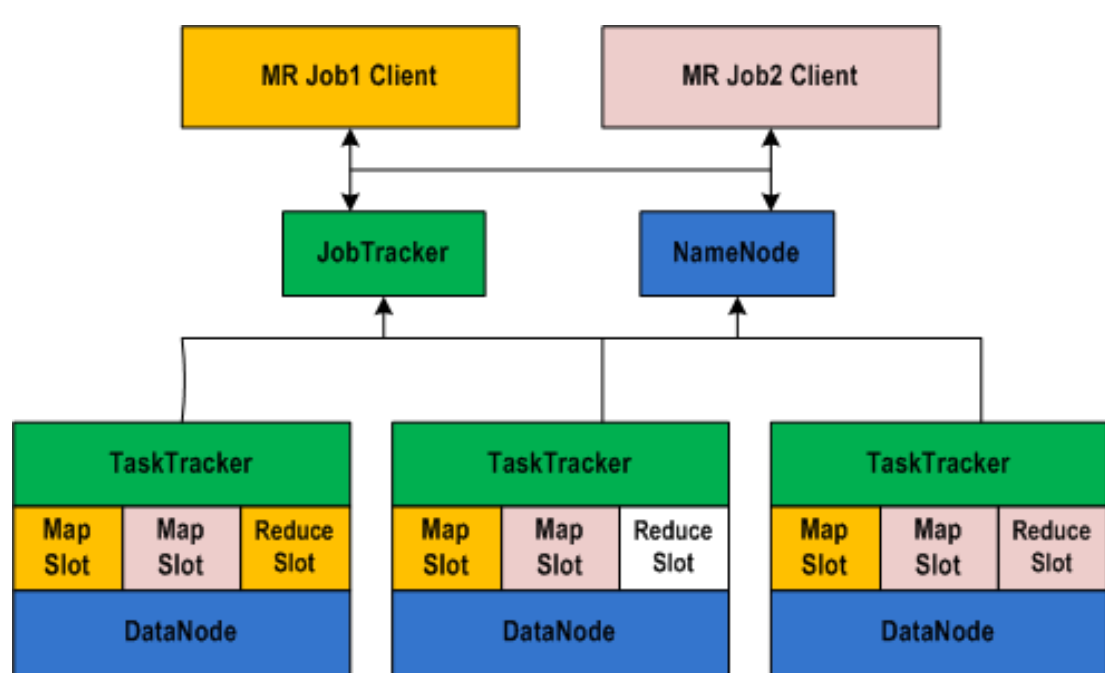
Hadoop 1.0 中的状况

- JobTracker必须不断跟踪所有TaskTracker和所有map、reduce任务，TaskTracker上的任务都是JobTracker来分配的
- 优化方向：
 - 我们减少了单个 JobTracker 的职责，将部分职责委派给 TaskTracker，因为集群中有许多 TaskTracker。在新设计中，这个概念通过将 JobTracker 的双重职责（**集群资源管理**和**任务协调**）分开为两种不同类型的进程来反映
 - 不再拥有JobTracker，引入集群管理器，负责跟踪集群中的活动节点和可用资源，并将它们分配给任务
 - 对于提交给集群的每个作业，会启动一个专用的、短暂的 JobTracker 来控制该作业中的任务的执行，短暂的 JobTracker 由在从属节点上运行的 TaskTracker 启动

Hadoop 2.0 中的设计

- ResourceManager (RM) 代替集群管理器
- ApplicationMaster (AM) 代替一个专用且短暂的 JobTracker
- NodeManager (NM) 代替 TaskTracker
- 一个分布式应用程序代替一个 MapReduce 作业
- 重构的根本思想：**将 JobTracker 两个主要的功能分离成单独的组件，这两个功能分别是资源管理和任务调度 / 监控**

Hadoop 1.0 和 Hadoop 2.0



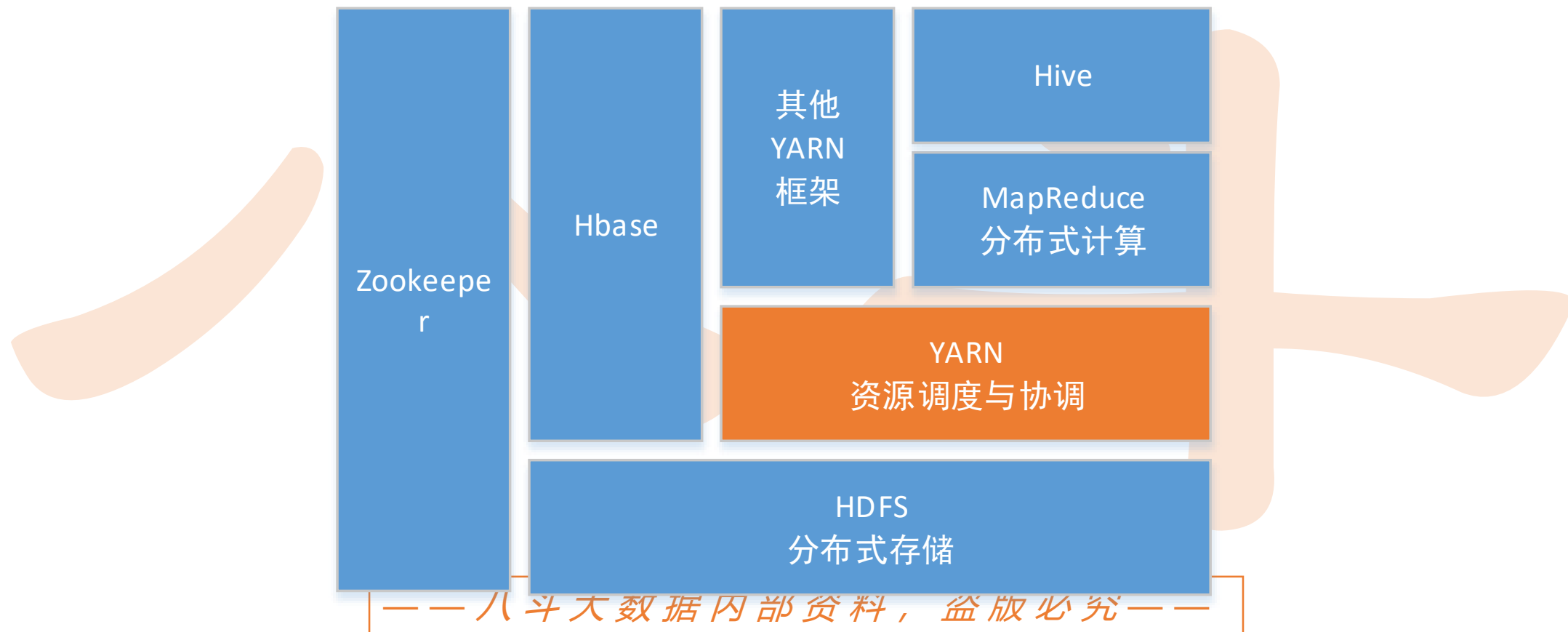
框架前后对比

- 原框架中核心的 JobTracker 和 TaskTracker 不见了，取而代之的是 ResourceManager, ApplicationMaster 与 NodeManager 三个部分

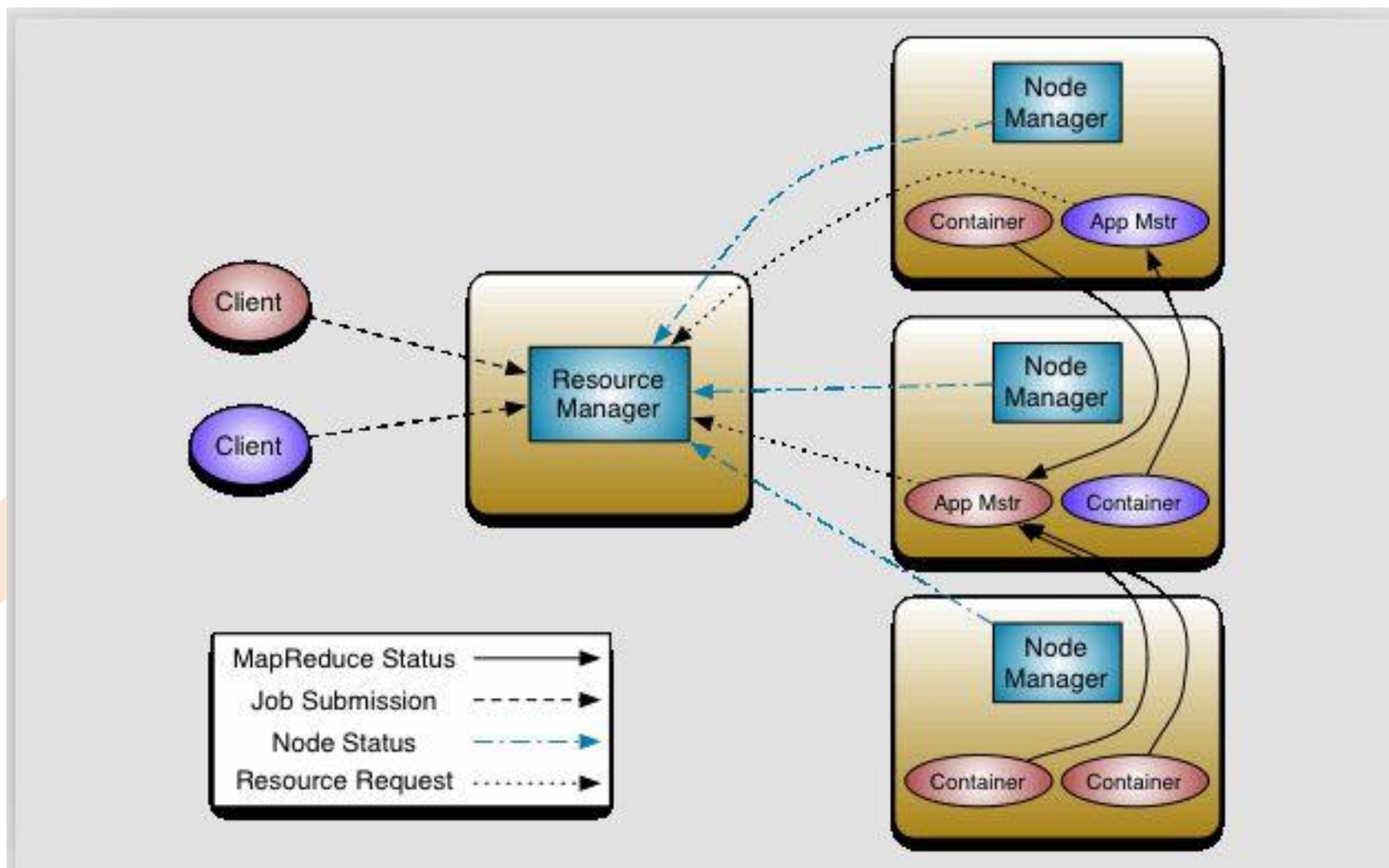


Apache Hadoop Yarn 核心概念

- MapReduce经历了完全重构，不再是Hadoop的核心组件，而成为Yarn上的一种应用框架，称为MRv2（MapReduce的第二版）



Apache Hadoop Yarn 核心概念



R M

- RM处理客户端请求，接收JobSubmitter提交的作业，按照作业的上下文(Context) 信息，以及从 NodeManager (NM) 收集来的状态信息，启动调度过程，分配一个 Container 作为 App Mstr
- RM拥有为系统中所有应用资源分配的决定权，是中心服务，做的事情就是**调度**、启动每一个Job所属的Application、另外监控Application的存在情况
- 与运行在每个节点上的NM进程交互，通过心跳通信，达到监控NM的目的
- RM有一个可插拔的调度器组件Scheduler
 - Scheduler是一个纯粹的调度器：
 - 不负责应用程序的监控和状态跟踪
 - 不保证应用程序失败或者硬件失败的情况下对Task的重启

N M

- 是slave进程，类似TaskTracker的角色，是每个机器框架代理
- 处理来自RM的任务请求
- 接收并处理来自ApplicationMaster的Container启动、停止等各种请求
- 负责启动应用程序的Container（执行应用程序的容器），并监控他们的资源使用情况（CPU、内存、磁盘和网络），并报告给RM
- 总的来说，在单节点上进行资源管理和任务管理

A M

- 应用程序的Master，每一个应用对应一个AM，在用户提交一个应用程序时，一个AM的轻量型进程实例会启动，AM协调应用程序内的所有任务的执行
- 负责一个Job生命周期内的所有工作，类似旧的JobTracker
- 每一个Job都有一个AM，运行在RM以外的机器上
- 与RM协商资源
 - 与Scheduler协商合适的Container
- 与NM协同工作与Scheduler协商合适的Container进行Container的监控
- 是一个普通Container的身份运行

Container

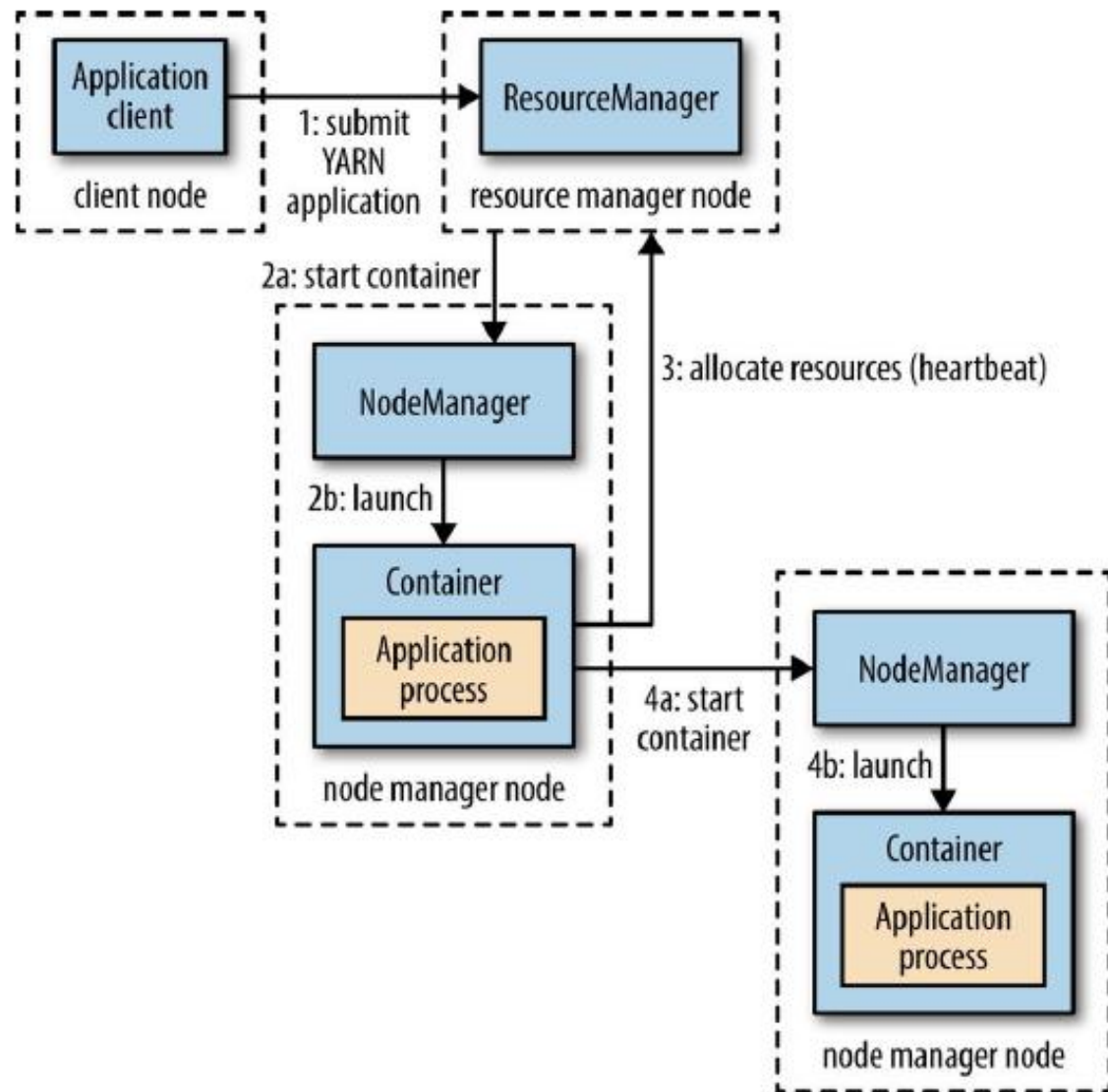
- 是任务运行环境的抽象封装
- Container只是使用NM上指定资源的权利
- AM必须向NM提供更多的信息来启动Container
- 描述任务的运行资源（节点、内存、cpu）、启动命令和运行环境

Yarn 框架对于旧的 MapReduce 框架的优势

- 减小了 JobTracker (也就是现在的 RM) 的资源消耗, 并且让监测每一个 Job 子任务 (tasks) 状态的程序分布式化了, 更安全、更优美
- AM是一个可变更的部分, 用户可以对不同的编程模型写自己的 AM, 让更多类型的编程模型能够跑在 Hadoop 集群中
- 对于资源的表示以内存为单位, 比之前以剩余 slot 数目更合理
- 老的框架中, JobTracker 一个很大的负担就是监控 job 下的 tasks 的运行状况, 现在, 这个部分就扔给 ApplicationMaster 做了
- 资源表示成内存量, 那就没有了之前的 map slot/reduce slot 分开造成集群资源闲置的尴尬情况

Yarn 框架的运行过程

- Client请求Resource Manager运行一个Application Master实例 (step 1) ;
- Resource Manager选择一个Node Manager, 启动一个Container并运行Application Master实例 (step 2a、step 2b) ;
- Application Master根据实际需要向Resource Manager请求更多的Container资源 (step 3) ;
- Application Master通过获取到的Container资源执行分布式计算 (step 4a、step 4b)



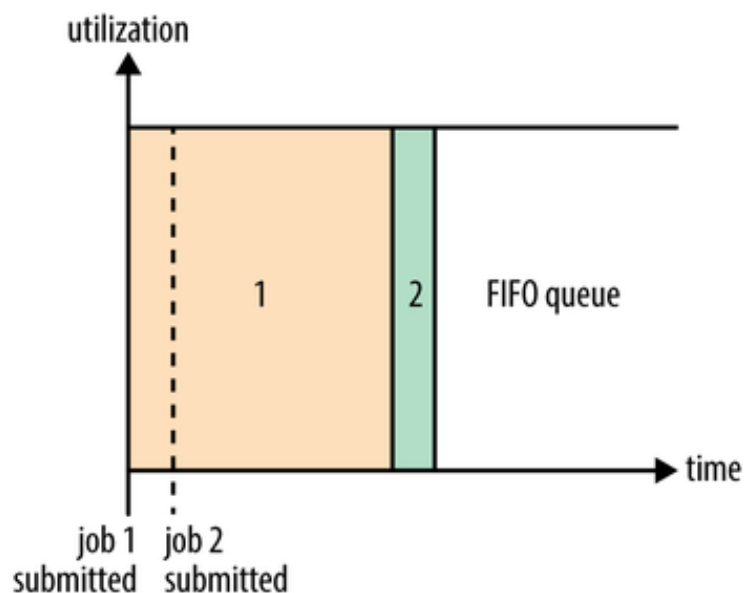
Yarn 的容错能力

- RM挂掉：单点故障，新版本可以基于Zookeeper实现HA高可用集群，可通过配置进行设置准备RM，主提供服务，备同步主的信息，一旦主挂掉，备立即做切换接替进行服务
- NM挂掉：不止一个，当一个挂了，会通过心跳方式通知RM，RM将情况通知对应AM，AM作进一步处理
- AM挂掉：若挂掉，RM负责重启，其实RM上有一个RMApplicationMaster，是AM的AM，上面保存已经完成的task，若重启AM，无需重新运行已经完成的task

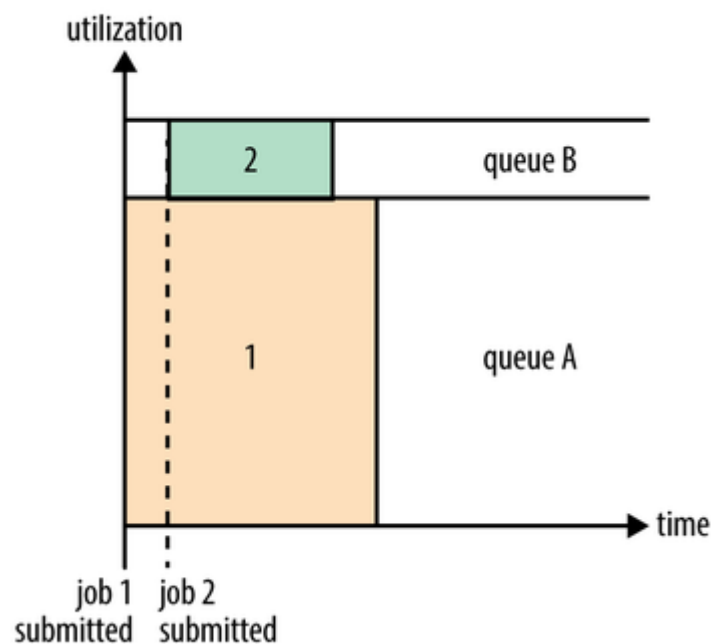
Yarn 的调度器

- FIFO Scheduler: 按提交顺序, 最简单, 大应用占用所有集群资源, 不适合共享集群
- Capacity Scheduler: 专有队列运转小任务, 预先占一定集群资源, 导致大任务执行时间落后于FIFO
- Fair Scheduler: 不需要预占, 动态调整, 公平共享

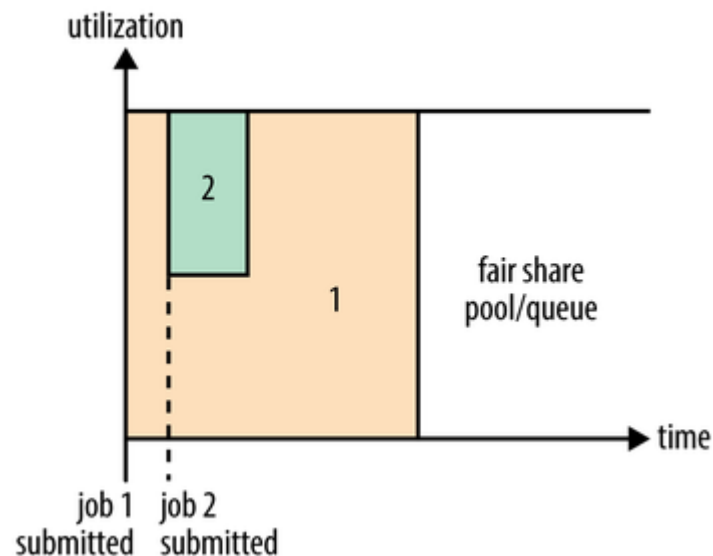
i. FIFO Scheduler



ii. Capacity Scheduler



iii. Fair Scheduler



Outline

Yarn基础

【基础实践】Hadoop2.0集群搭建

安装 Hadoop 2.0

- 先配ssh免密，再装java，接下来
-]# tar xzf hadoop-2.6.1.tar.gz
- 进入： /usr/local/src/hadoop-2.6.1/etc/hadoop目录
- hadoop-env.sh配置JAVA_HOME
- yarn-env.sh配置JAVA_HOME
- slaves配置从节点
- core-site.xml配置下面配置

```
18
19 <configuration>
20   <property>
21     <name>fs.defaultFS</name>
22     <value>hdfs://master:9000/</value>
23   </property>
24   <property>
25     <name>hadoop.tmp.dir</name>
26     <value>file:/usr/local/src/hadoop-2.6.1/tmp</value>
27   </property>
28 </configuration>
```

安装 Hadoop 2.0

- 在HADOOP_HOME创建目录:
-]# mkdir tmp
-]# mkdir -p dfs/name
-]# mkdir -p dfs/data
- hdfs-site.xml配置下面配置

```
18
19 <configuration>
20   <property>
21     <name>dfs.namenode.secondary.http-address</name>
22     <value>master:9001</value>
23   </property>
24   <property>
25     <name>dfs.namenode.name.dir</name>
26     <value>file:/usr/local/src/hadoop-2.6.1/dfs/name</value>
27   </property>
28   <property>
29     <name>dfs.datanode.data.dir</name>
30     <value>file:/usr/local/src/hadoop-2.6.1/dfs/data</value>
31   </property>
32   <property>
33     <name>dfs.replication</name>
34     <value>3</value>
35   </property>
36 </configuration>
```


安装 Hadoop 2.0

-]# cp mapred-site.xml.template mapred-site.xml
- mapred-site.xml配置

```
19 <configuration>
20     <property>
21         <name>mapreduce.framework.name</name>
22         <value>yarn</value>
23     </property>
24 </configuration>
```

安装 Hadoop 2.0

- yarn-site.xml配置

```
15 <configuration>
16   <property>
17     <name>yarn.nodemanager.aux-services</name>
18     <value>mapreduce_shuffle</value>
19   </property>
20   <property>
21     <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
22     <value>org.apache.hadoop.mapred.ShuffleHandler</value>
23   </property>
24   <property>
25     <name>yarn.resourcemanager.address</name>
26     <value>master:8032</value>
27   </property>
28   <property>
29     <name>yarn.resourcemanager.scheduler.address</name>
30     <value>master:8030</value>
31   </property>
32   <property>
33     <name>yarn.resourcemanager.resource-tracker.address</name>
34     <value>master:8035</value>
35   </property>
36   <property>
37     <name>yarn.resourcemanager.admin.address</name>
38     <value>master:8033</value>
39   </property>
40   <property>
41     <name>yarn.resourcemanager.webapp.address</name>
42     <value>master:8088</value>
43   </property>
44 </configuration>
```

安装 Hadoop 2.0

- 在HADOOP_HOME下
-]# ./bin/hadoop namenode -format
-]# ./sbin/start-dfs.sh
-]# ./sbin/start-yarn.sh

```
[root@master src]# jps
43510 SecondaryNameNode
44292 Jps
43040 NameNode
44024 ResourceManager
[root@master src]#
```

```
[root@slave1 src]# jps
31140 NodeManager
31262 Jps
31033 DataNode
```

Q & A

@八斗学院
