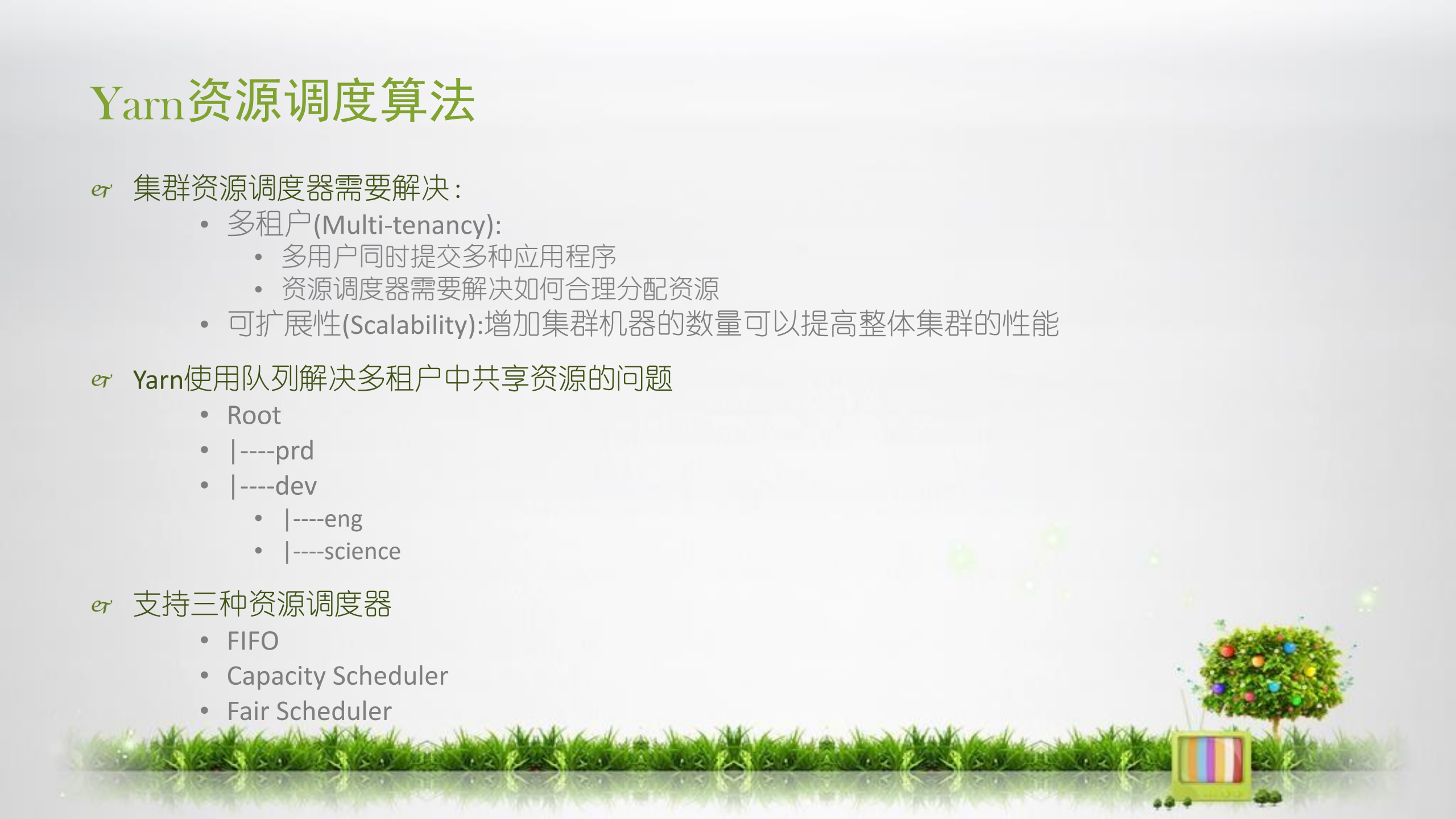
Yarn资源调度算法

- 集群资源调度器需要解决：
  - 多租户(Multi-tenancy):
    - 多用户同时提交多种应用程序
    - 资源调度器需要解决如何合理分配资源
  - 可扩展性(Scalability):增加集群机器的数量可以提高整体集群的性能
- Yarn使用队列解决多租户中共享资源的问题
  - Root
  - |----prd
  - |----dev
    - |----eng
    - |----science
- 支持三种资源调度器
  - FIFO
  - Capacity Scheduler
  - Fair Scheduler



FIFO调度器

- 所有向集群提交的作业使用一个队列
- 根据提交作业的顺序来运行
- 优点：
  - 简单易懂
  - 可以按照作业优先级调度
- 缺点：
  - 资源利用率不高
  - 不允许抢占

Capacity Scheduler资源调度器

- 设计思想：
  - 资源按照比例分配给各个队列
- 特点：
  - 计算能力保证
    - 以队列为单位划分资源，每个队列最低资源保证
  - 灵活性
    - 当某个队列空闲时，其资源可以分配给其他的队列使用
  - 支持优先级
    - 单个队列内部使用FIFO，支持作业优先级调度
  - 多租户
    - 综合考虑多种因素防止单个作业、用户或者队列独占资源
    - 每个队列可以配置一定比例的最低资源配置和使用上限
    - 每个队列有严格的访问限制，只能向自己的队列提交任务
  - 基于资源的调度
    - 支持内存资源度和CPU资源的调度
  - 支持抢占(Hadoop2.8.0+)

Capacity Scheduler配置

配置capacity-scheduler.xml

```xml
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>prd,dev</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.dev.queues</name>
  <value>eng,science</value>
</property>
<property>
  <name>yarn.scheduler.capacity.prd.capacity</name>
  <value>70</value>
</property>
<property>
  <name>yarn.scheduler.capacity.prd.user-limit-factor</name>
  <value>30</value>
</property>
<property>
  <name>yarn.scheduler.capacity.dev.maximum-capacity</name>
  <value>50</value>
</property>
<property>
  <name>yarn.scheduler.capacity.dev.capacity</name>
  <value>30</value>
</property>
<property>
  <name>yarn.scheduler.capacity.dev.eng.capacity</name>
  <value>50</value>
</property>
<property>
  <name>yarn.scheduler.capacity.dev.science.capacity</name>
  <value>50</value>
</property>
```

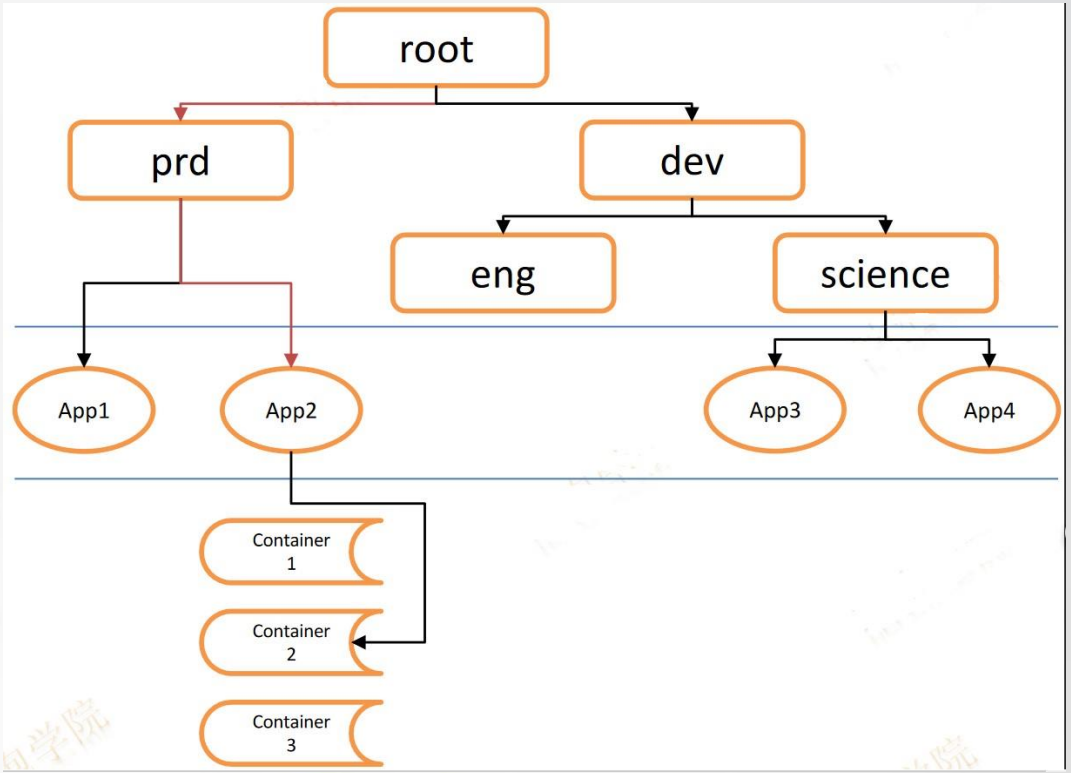
配置yarn-site.xml

```xml
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
</property>
```

root
|----prd 70%
|----dev 30%
　　|----eng 50%
　　|----science 50%

Capacity Scheduler资源分配算法

- 1.选择队列
  - 从跟队列开始，使用深度优先算法找出资源占用率最低的叶子节点
- 2.选择作业
  - 默认按照作业优先级和提交时间顺序选择
- 3.选择Container
  - 取该作业中最高优先级的Container，如果优先级相同会选择满足本地性的Container： Node Local > Rack Local > Different Rack

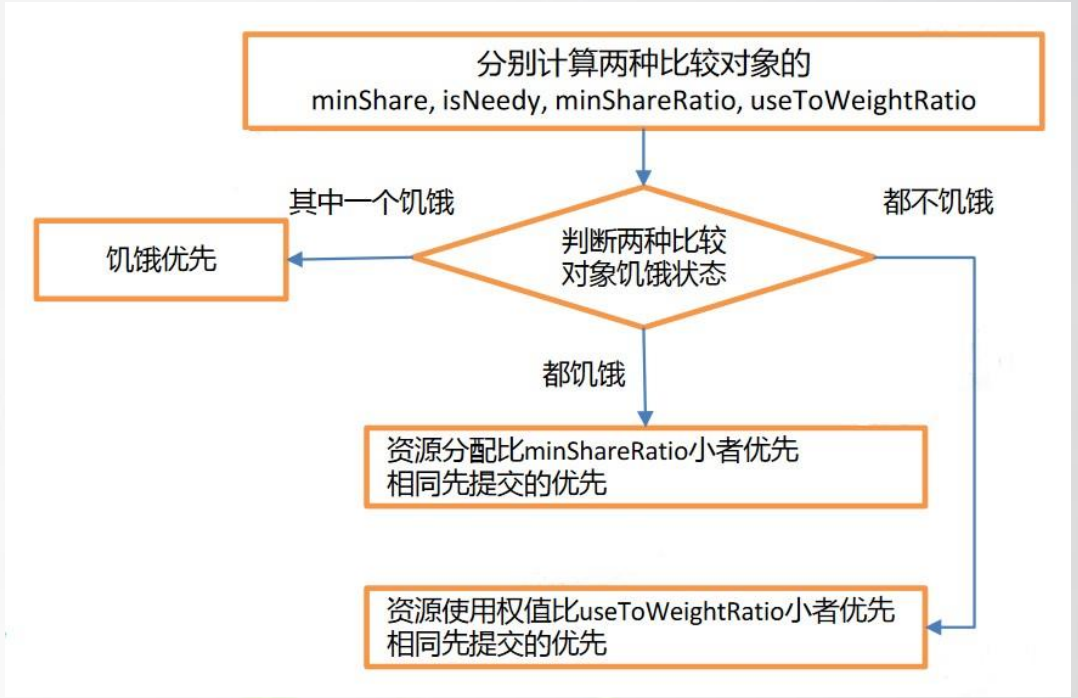


Fair Scheduler资源调度器

- 设计思想
  - 资源公平分配
- 具有与Capacity Scheduler相似的特点
  - 树状队列
  - 每个队列有独立的自小资源保证
  - 空闲时可以分配资源给其他队列使用
  - 支持内存资源调度和CPU资源调度
- 不同点
  - 核心策略不同
    - Capacity Scheduler优先选择资源利用率低的队列
    - Fair Scheduler考虑的是公平，公平体现在作业对资源的
  - 单独设置队列间资源分配方式
    - FAIR (默认used memory/min share)
    - DRF (主资源公平调度)
  - 单独设置队列内部资源分配方式
    - FAIR DRF FIFO

Fair Scheduler - FAIR资源分配算法

- 总体流程与Capacity Scheduler一致
  - 1.选择队列
  - 2.选择作业
  - 3.选择Container
- 选择队列和作业使用公平排序算法
  - 实际最小份额
    - mindshare = min(资源需求量，配置minShare)
  - 是否饥饿
    - isNeedy = 资源使用量 < minShare
  - 资源分配比
    - minShareRatio = 资源使用量/max(mindshare,1)
  - 资源使用权重比
    - useToWeightRatio = 资源使用量/权重
    - 权重在配置文件中配置



Capacity Scheduler和Fair Scheduler对比

	Capacity Scheduler	Fair Scheduler
设计思想	资源按照比例分配，并添加各种严格限制，防止个别用户或者队列单独占资源。	基于公平算法分配资源
队列是树状结构	是	是
多用户	支持	支持
最小资源保证	支持	支持
最大资源限制	支持	支持
资源共享	均支持。当一个队列有资源剩余时，可暂时将剩余资源共享给其他队列。	
限制集群或队列并发作业数	支持	支持
支持抢占	支持	支持
支持批量调度	均支持。当一个节点有大量资源可用时，可一次性分配完成，不用多次分配每次分配一份资源。	
为每个队列单独设置调度策略	不支持	支持
资源分配策略	FIFO或DRF	Fair、FIFO或DRF
动态加载配置文件	支持	支持

Yarn常用命令

- Yarn Application
  - 列出所有Application

    yarn application -list
  - 根据Application状态过滤

    yarn application –list –appStatus ACCEPTED
  - Kill掉Application

    yarn application –kill <ApplicationId>

- Yarn logs
  - 查询Application日志

    yarn logs –applicationId <ApplicationId>
  - 查询Container日志

    yarn logs -application <ApplicationId>
    -containerId <ContainerId>
    -nodeAddress <NodeAddress>

    端口在配置文件中yarn.nodemanager.webapp.address参数指定

常用Yarn 命令

- Yarn applicationattempt
  - 列出所有Application尝试的列表

    yarn applicationattempt –list <ApplicationId>
  - 打印ApplicationAttempt状态

    yarn applicationattempt –status <ApplicationAttemptId>
- Yarn Container
  - 列出所有Container

    yarn container –list <ApplicationAttemptId>
  - 打印Container状态

    yarn container –status <ContainerId>
- Yarn node
  - 列出所有节点

    yarn node –list -all

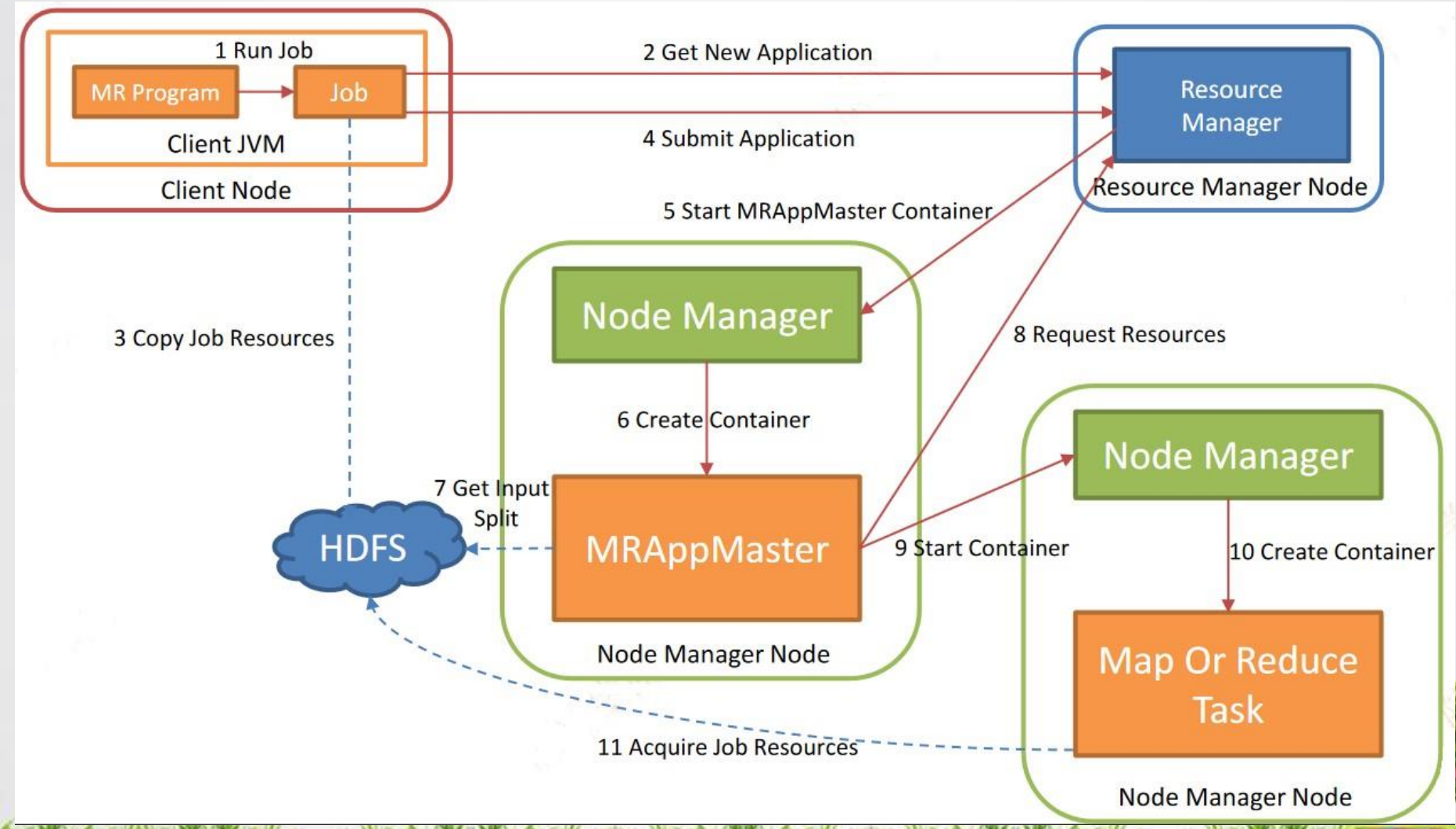
常用Yarn 命令

- Yarn rmadmin
  - 加载队列配置

    yarn rmadmin -refreshQueues
- Yarn queue
  - 打印队列信息

    yarn queue –status <QueueName>

Mapreduce On Yarn



Spark

- 基于内存的大数据计算引擎
- MapReduce不适合的计算场景
  - 迭代式作业
    - 机器学习、图计算需要迭代运行Mapper和Reducer多次
  - 交互式作业
    - 交互式数据分析，尤其是涉及到机器学习算法
  - 流式作业
    - 无穷无尽的流式数据，需要不断的对数据进行聚合计算
- Spark提供了一种新的数据抽象RDD弹性分布式数据集，可以将数据存储到内存中，而不是存储到HDFS上，是的快速迭代计算成为可能

Spark On Yarn