分布式资源管理系统YARN

Grissom| 2025年10月

TRANSWARP
星 环 科 技

TRANSWARP
星 环 科 技

目录》
CONTENTS

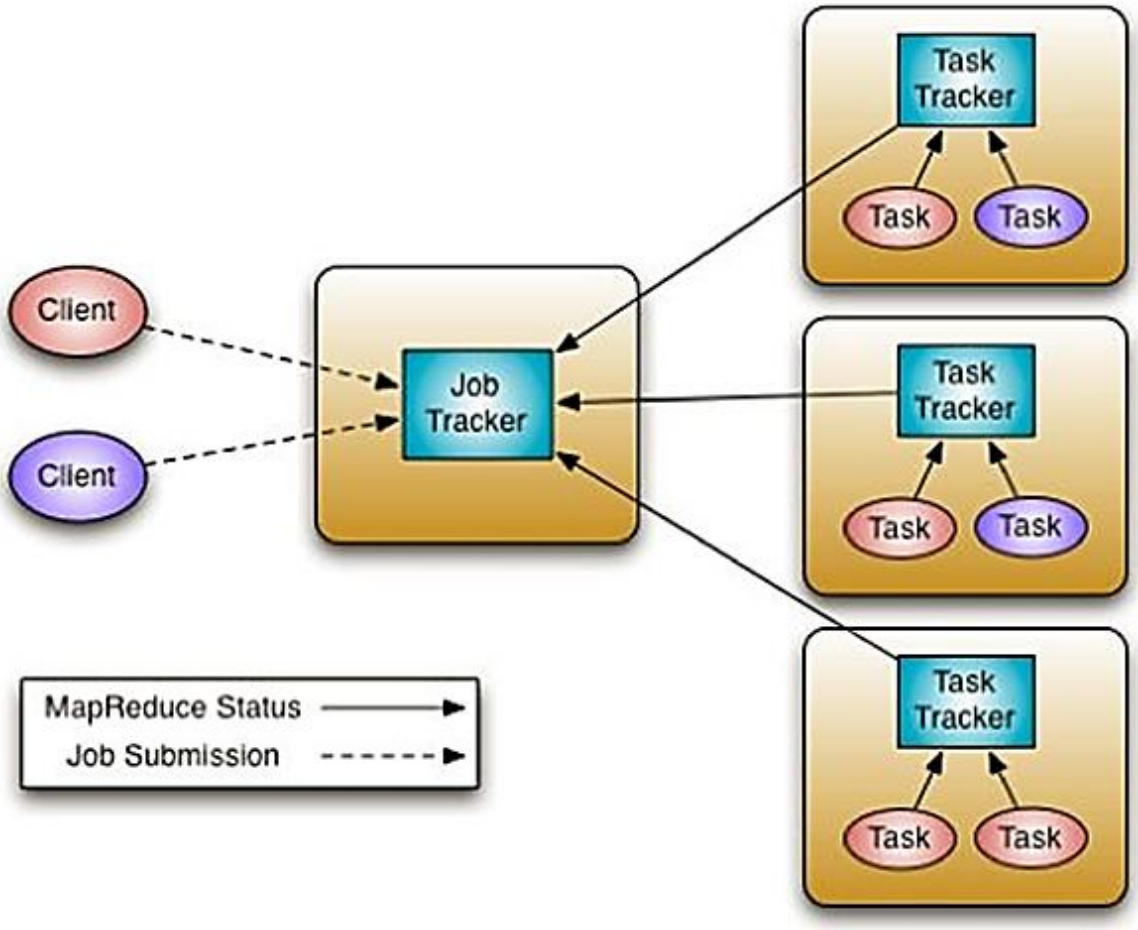
1
chapter

YARN简介

- ✓ YARN的由来
- ✓ 什么是YARN

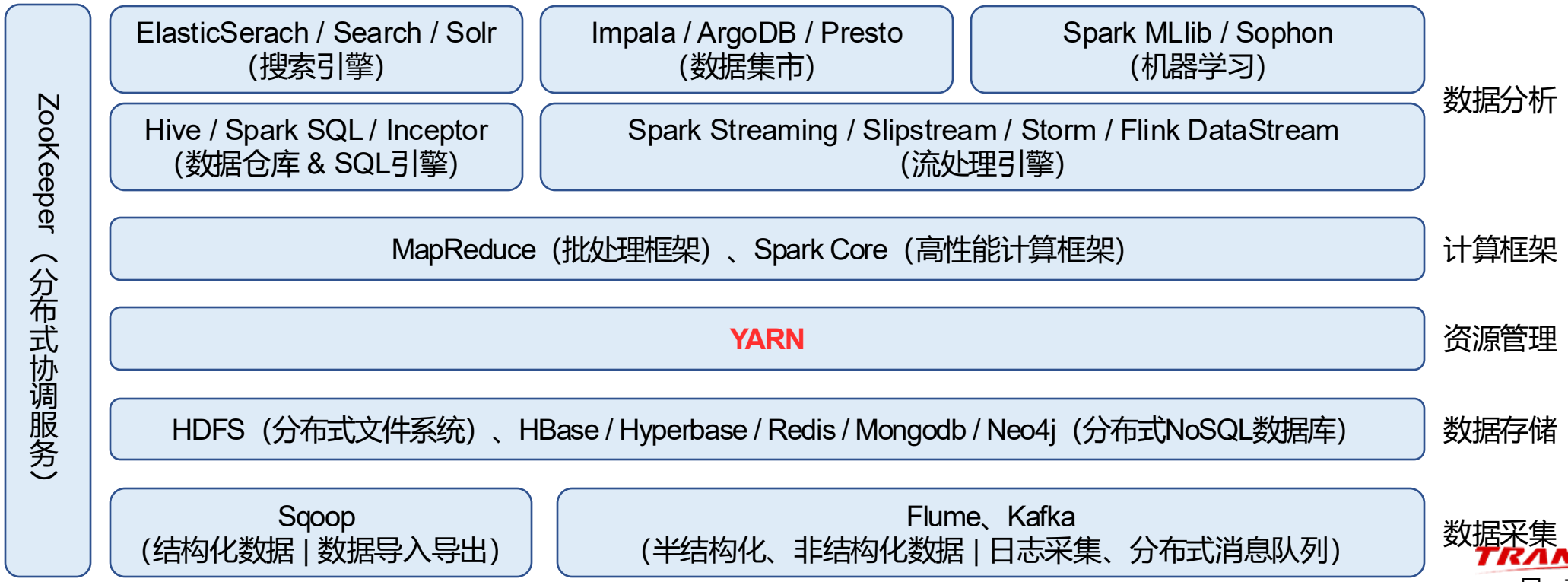
Hadoop 1.x中的MapReduce存在先天缺陷

- 身兼两职
  - 计算框架（称职）
  - 资源管理系统（不称职）
- JobTracker
  - 既管理资源，又管理作业
  - 负担太重，开销过大
  - 存在单点故障
- 资源描述模型过于简单，资源利用率较低
  - 仅把Task数量看作资源，没有考虑CPU和内存
  - 强制把资源分成Map Task Slot和Reduce Task Slot
- 扩展性较差，集群规模上限4K
- 源码难于理解，升级维护困难



1.2 什么是YARN

- YARN，Yet Another Resource Negotiator，另一种资源管理器
- 分布式通用资源管理系统
- 设计目标：聚焦资源管理、通用（适用各种计算框架）、高可用（元数据和Master高可用）、高扩展（与HDFS同步扩展）、高容错（计算容错）

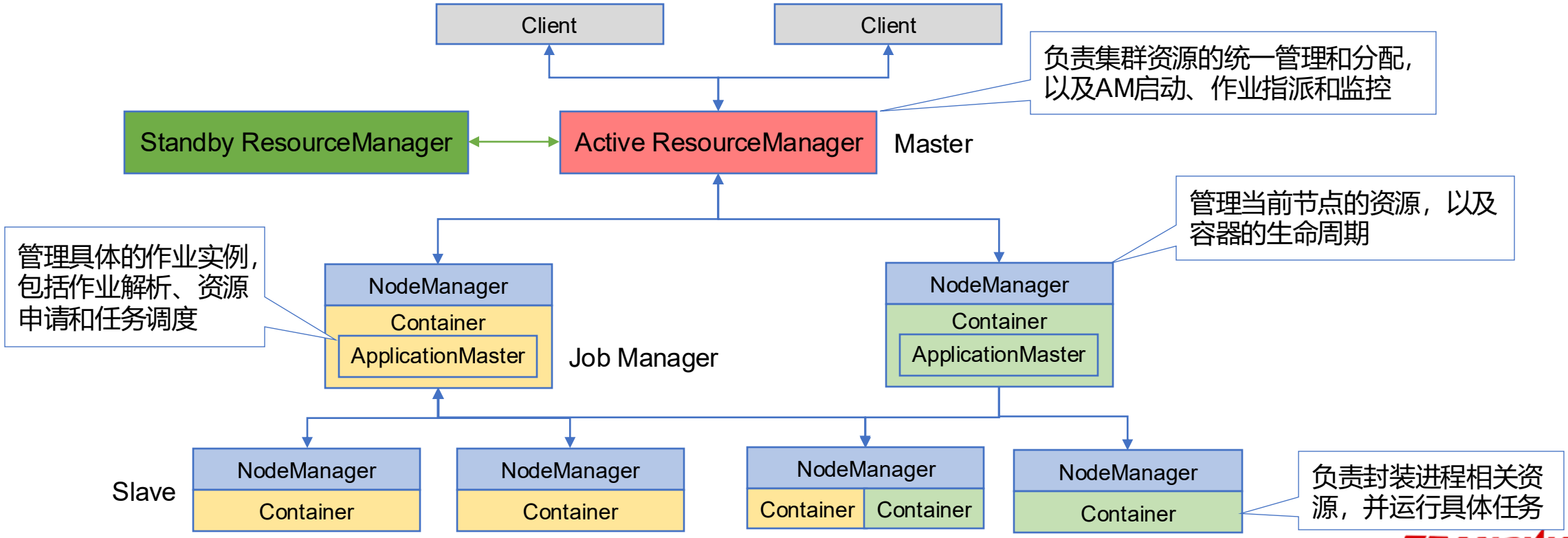


YARN原理

- ✓ 系统架构
- ✓ 高可用

➢ 系统架构图（Master/Slave + Active/Standby）

- 基本思路：将JobTracker的资源管理和作业管理职能分离开来
- 四种角色：ResourceManager、ApplicationMaster（作业管家）、NodeManager、Client



2.1 系统架构：角色分工

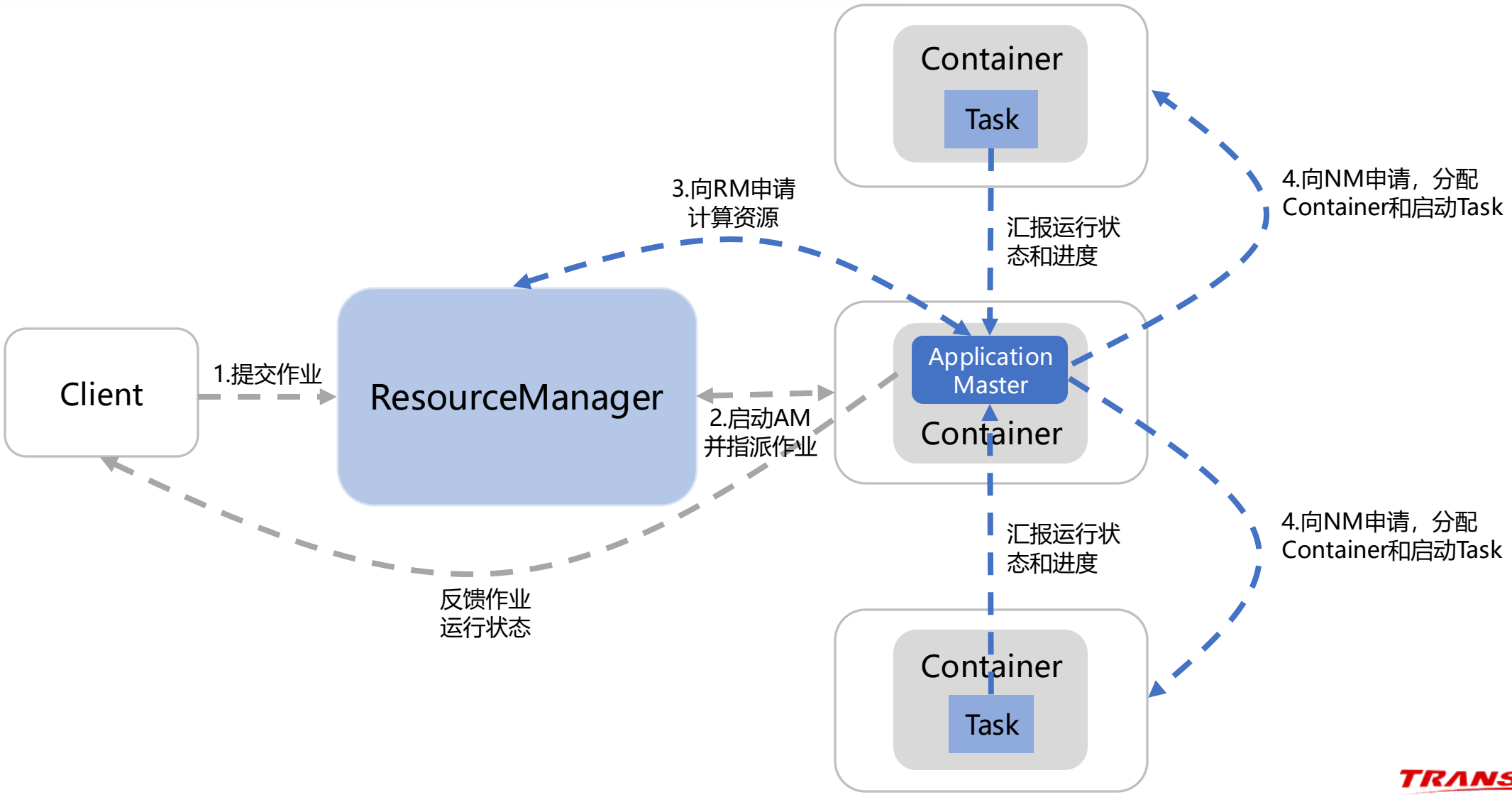
- Active ResourceManager（ARM）
  - 活动资源管理节点（Master / 集群唯一）
  - 统一管理集群计算资源
  - 负责启动ApplicationMaster、作业指派和监控
  - 将资源按照一定的调度策略分配给作业
  - 接收NodeManager的运行状况和资源上报信息
- Standby ResourceManager（SRM）
  - 热备资源管理节点（允许多个）
  - 主备切换
    - AR宕机后，经过Master选举和状态信息恢复，SRM升级为ARM
    - 重启AM，杀死所有运行中的Container

- NodeManager（NM）
  - 计算节点（Slave / 高扩展）
  - 管理单个节点的资源
  - 管理Container的生命周期（从创建到销毁的全过程）
  - 向ResourceManager汇报运行状况和资源使用情况
- Container
  - 容器：对进程相关资源的封装，对资源的抽象，分配资源即分配Container
  - 分为两类：运行AM的Container 、运行Task的Container

➢ ApplicationMaster（AM）

- 作业管家
  - 一对一管理：每个作业实例都由一个专职的AM来管理
  - 作业解析：将Job解析为由若干Task组成的有向无环图
  - 申请资源：向RM申请Job运行所需的计算资源
  - 任务调度和监管：向NM申请分配Container和启动Task，同时监测Task的运行状态和进度
  - 反馈：向Client反馈Job的运行状态和结果
- 实现方式
  - YARN缺省提供MapReduce的AM实现，但其他计算框架需自备作业管理组件（如Spark Driver）
  - 采用基于事件驱动的异步编程模型，由中央事件调度器统一管理所有事件
  - AM是一种事件处理器，在中央事件调度器中注册，这样可实现解耦，以确保YARN的通用性

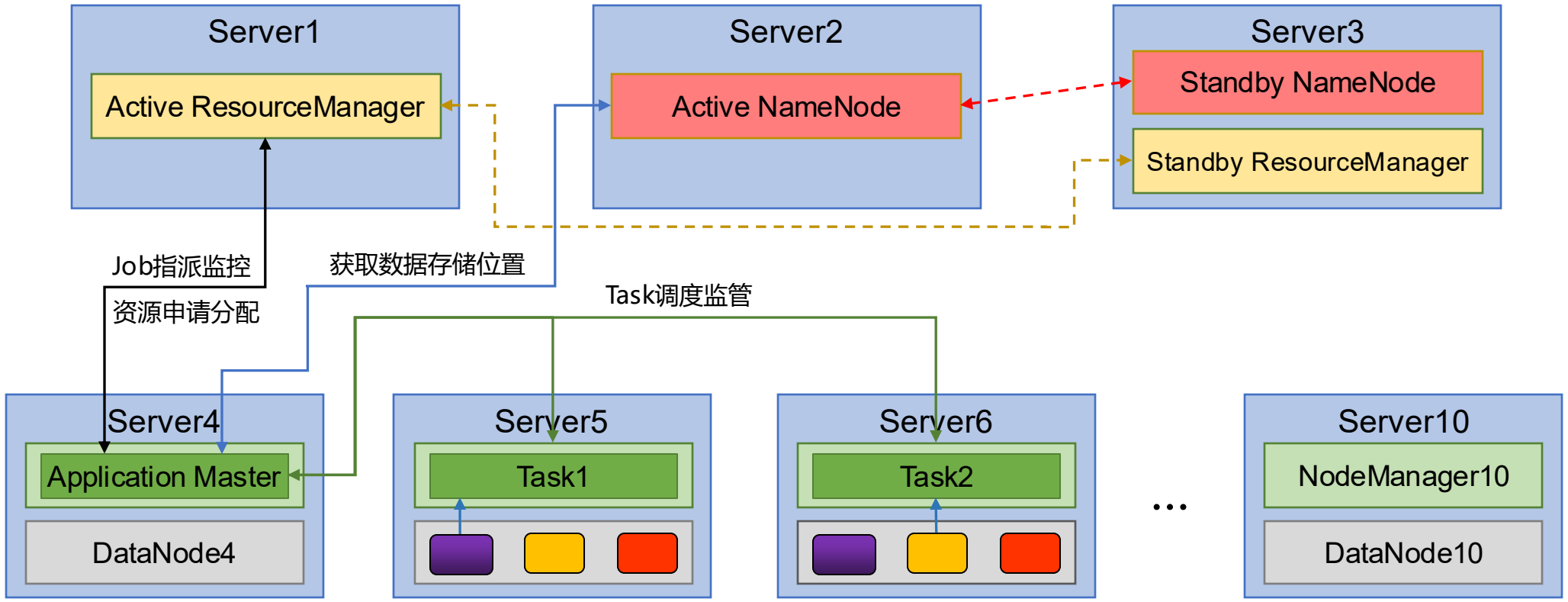
2.1 系统架构：工作机制


Container
Task
3.向RM申请
计算资源
汇报运行状
态和进度
4.向NM申请，分配
Container和启动Task
Client
1.提交作业
ResourceManager
2.启动AM
并指派作业
Application
Master
Container
汇报运行状
态和进度
4.向NM申请，分配
Container和启动Task
反馈作业
运行状态
Container
Task

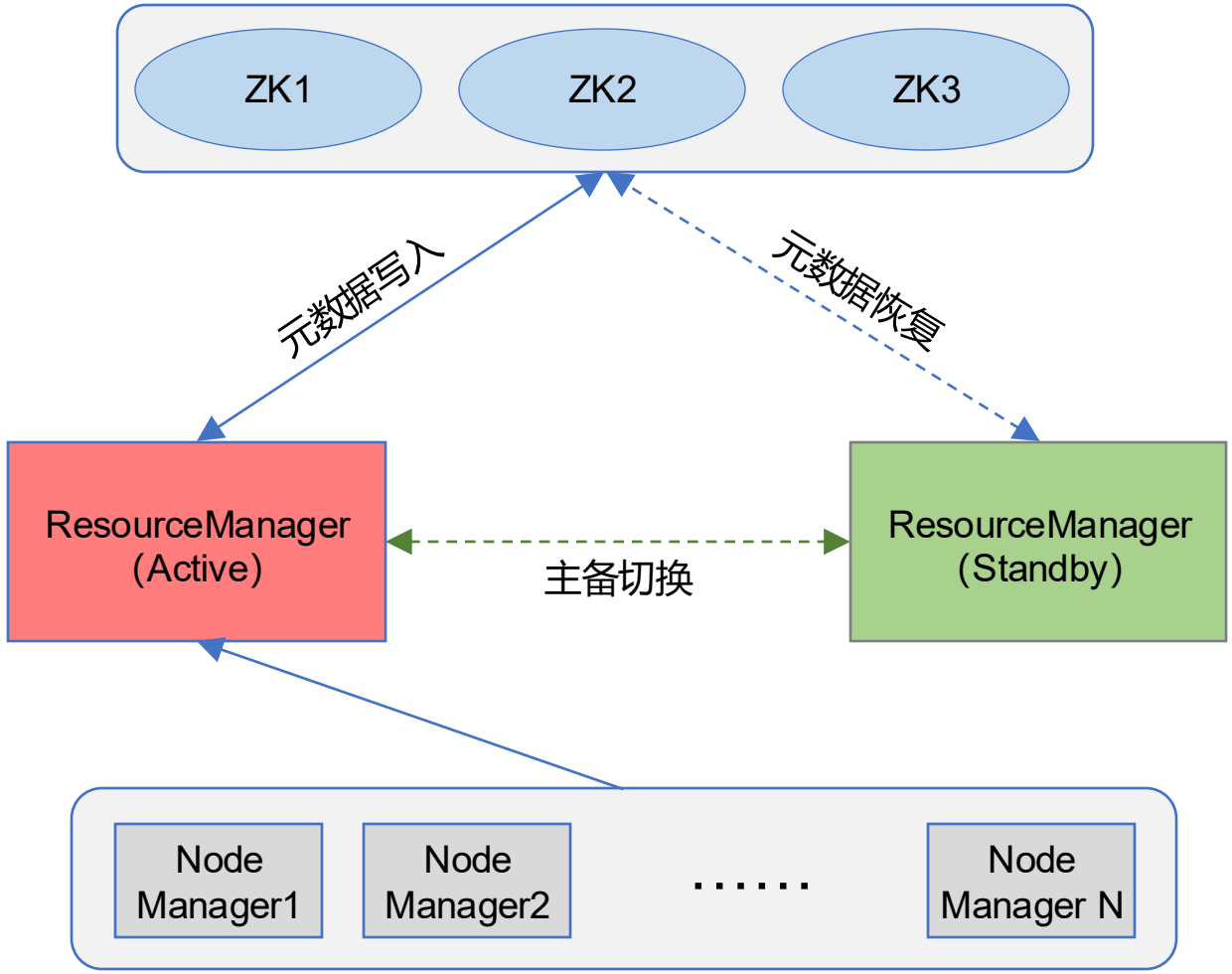
➢ 基本流程

1. Client向RM提交编译好的分布式程序（Job）
2. RM接收Job后，分配一个NM来启动AM，并将Job指派给AM，由它来一对一管理
3. AM将Job解析为一个由若干Task组成的有向无环图DAG，并从NameNode获取Task输入数据的存储位置（即Block存储位置），然后向RM申请计算资源
4. 根据AM提交的Task Set及其对应的Block存储位置，RM为Job分配计算资源，即为每个Task分配一个NM List，并返回给AM
   - 计算跟着数据走：NM所在Server的DataNode上存储了Task的输入Block
5. 根据Task DAG和NM List，AM按照并行/串行次序将Task提交给NM
6. NM接收Task，验证身份后，启动Container，运行Task，并向AM汇报运行状态和进度
7. 在Job运行期间，AM向Client反馈Job运行进度和状态，并返回最终结果

- 计算跟着数据走


Server1
Active ResourceManager
Server2
Active NameNode
Server3
Standby NameNode
Standby ResourceManager
Job指派监控
资源申请分配
获取数据存储位置
Task调度监管
Server4
Application Master
DataNode4
Server5
Task1
Server6
Task2
...
Server10
NodeManager10
DataNode10

- 基于ZK的元数据高可用
  - RM状态
  - Job状态和Token（访问身份验证）
- 基于ZK的RM高可用（主备切换）
  1. Master选举
  2. 恢复RM的原有状态信息
  3. 重启AM，并杀死所有运行中的Container
- 计算高可用
  - Task失败后，AM会把其调度到其他NM上重新执行（默认4次）
  - Job失败后，RM会在其他NM上重启AM（默认2次）



RM管理命令

```
# yarn  rmadmin  [command_options]
```

Command Options	Description
-refreshQueues	Reload the queues' acls, states and scheduler specific properties.
-refreshNodes	Refresh the hosts information at the ResourceManager.
-failover [-forceactive] <serviceId1> <serviceId2>	Initiate a failover from serviceId1 to serviceId2.
-getServiceState <serviceId>	Returns the state of the service.

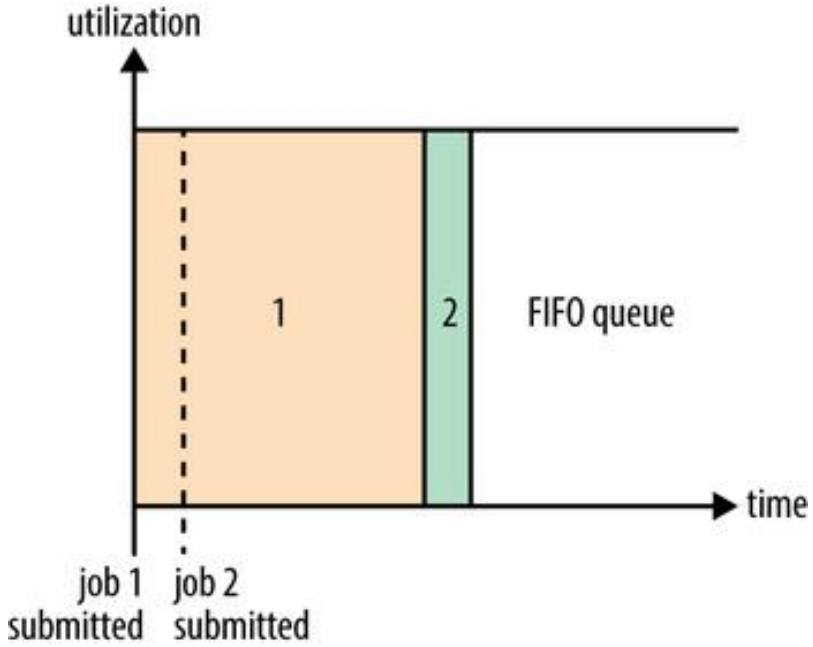
3
chapter

YARN资源调度策略

- ✓ FIFO调度器
- ✓ 容量调度器
- ✓ 公平调度器

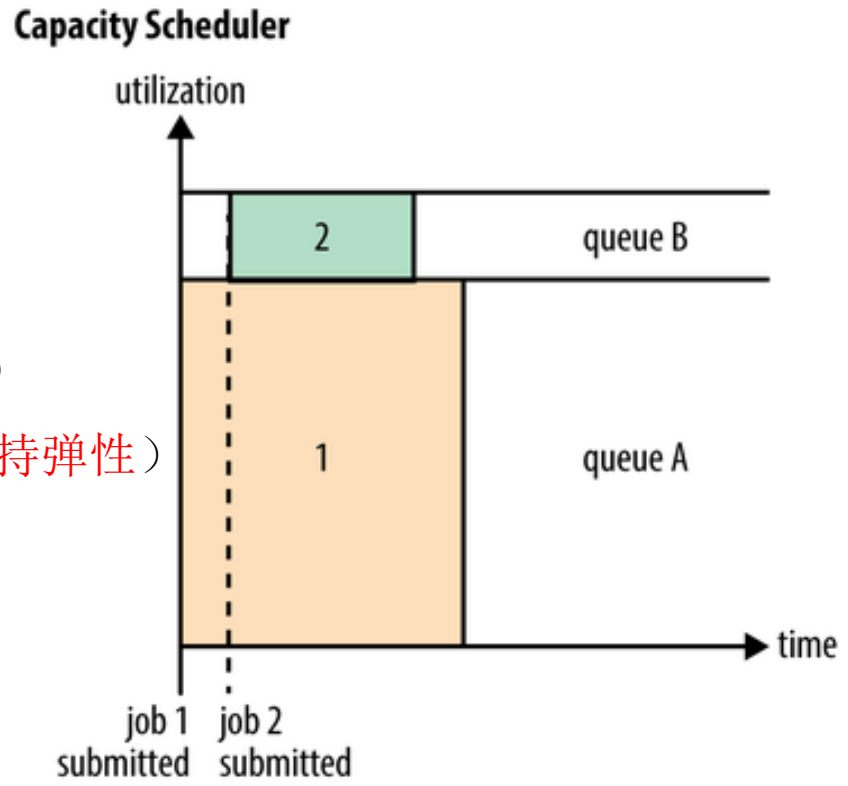
➢ FIFO Scheduler（先进先出调度器）

- 调度策略
  - 将所有作业放入一个队列，先进队列的先获得资源，排在后面的作业只能等待
- 缺点
  - 资源利用率低，无法交叉运行作业
  - 灵活性差，如紧急作业无法插队，耗时长的作业拖慢整个队列



➢ Capacity Scheduler（容量调度器）

- 核心思想：提前做预算，在预算指导下分享集群资源
- 调度策略
  - 集群资源由多个队列分享
  - 每个队列都要预设资源分配比例（提前做预算，预算是指导原则）
  - 空闲资源优先分配给“实际资源/预算资源”比值最低的队列（保持弹性）
  - 队列内部采用FIFO调度策略
- 特点
  - 层次化的队列设计：子队列可使用父队列资源
  - 容量保证：每个队列都要预设资源占比，防止资源独占
  - 弹性分配：空闲资源可以分配给任何队列，但当多个队列争用时，会按比例进行平衡
  - 支持动态管理：既可以动态调整队列的容量、权限等参数，也可以动态增加、暂停队列
  - 访问控制：用户只能向自己的队列中提交作业，不能访问其他队列
  - 多租户：多用户共享集群资源



- 全局配置（yarn-site.xml）

Property	Value
yarn.resourcemanager.scheduler.class	org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler

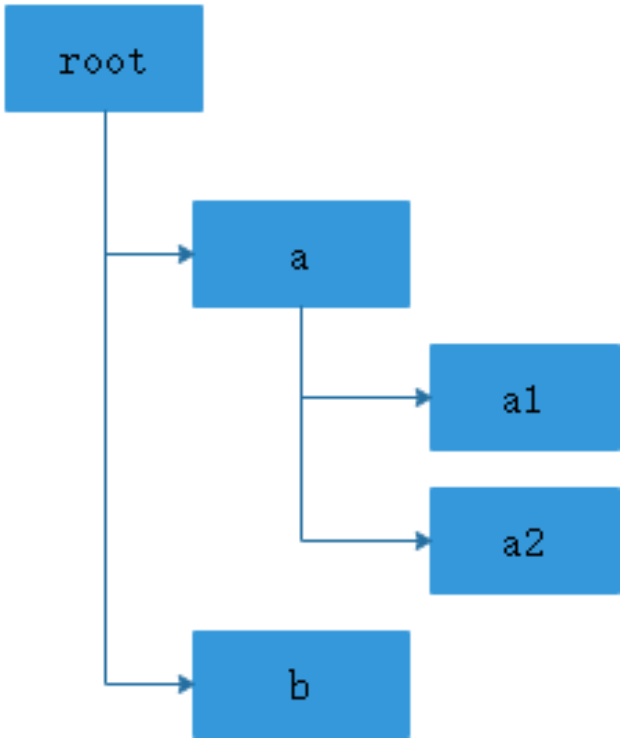
- 自定义配置（capacity-scheduler.xml）

Property	Value
yarn.scheduler.capacity.<queue-path>.capacity	Queue capacity in percentage (%) as a float (e.g. 12.5). The sum of capacities for all queues, at each level, must be equal to 100.
yarn.scheduler.capacity.<queue-path>.maximum-capacity	Maximum queue capacity in percentage (%) as a float. Defaults to -1 which disables it.
yarn.scheduler.capacity.<queue-path>.minimum-user-limit-percent	The user limit can vary between a minimum and maximum value. The the former (the minimum value) is set to this property value and the latter (the maximum value) depends on the number of users who have submitted applications. A value of 100 implies no user limits are imposed. （用户最低资源限制）

➢ 队列设计（capacity-scheduler.xml）

```
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>a,b</value>
  <description>The queues at the this level (root is the root queue).</description>
</property>

<property>
  <name>yarn.scheduler.capacity.root.a.queues</name>
  <value>a1,a2</value>
  <description>The queues at the this level (root is the root queue).</description>
</property>
```



根队列为root，其他队列必须定义为root的子队列，队列的继承关系以“.”号分隔

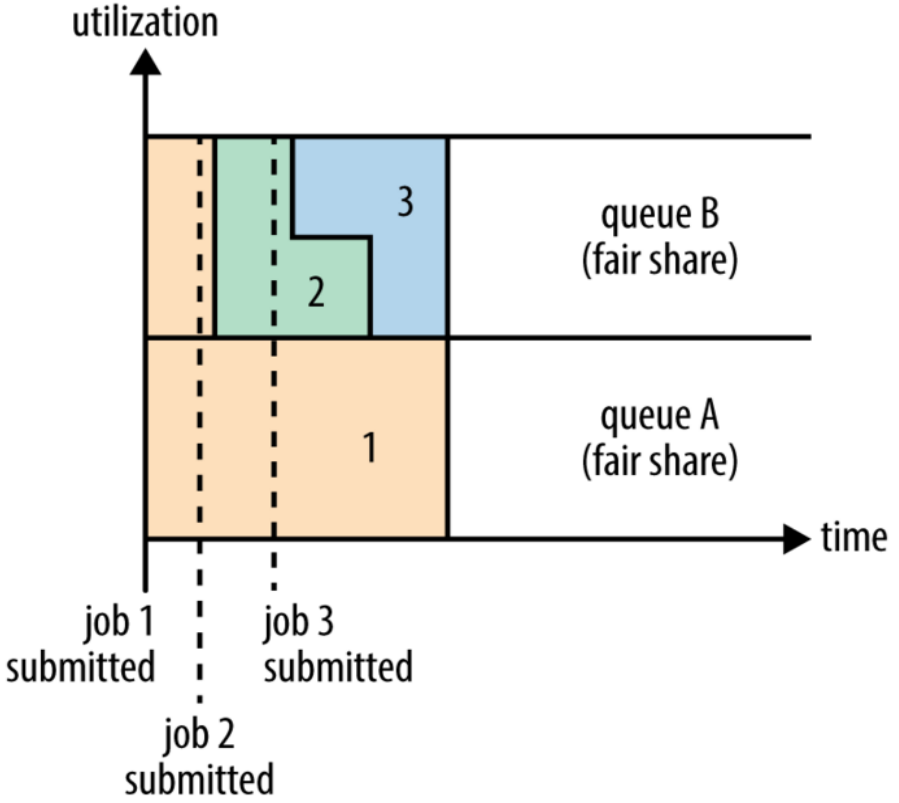
- 全局配置和队列设计（Web）

- 自定义配置（Web）


Queue配置
Queue 名称:
test
最大容量:
-1
用户限制因子:
1
最大Applications Master资源:
0.1
提交Application访问控制列表(ACL):
*
Queue容量:
40
最小用户限制比:
100
最大Application数量:
10000
状态:
Running
管理Application访问控制列表(ACL):
*

➢ Fair Scheduler（公平调度器）

- 调度策略
  - 多队列公平共享集群资源
  - 通过平分的方式，动态分配资源，无需预先设定资源分配比例，即“不提前做预算、见面分一半、实现绝对公平”
  - 队列内部可配置调度策略：FIFO、Fair（默认）
- 资源抢占
  - 终止其他队列的作业，使其让出所占资源，然后将资源分配给占用资源量少于最小资源量限制的队列（通过杀富济贫保持弹性）
- 队列权重
  - 当队列中有作业等待，并且集群中有空闲资源时，每个队列可以根据权重获得不同比例的空闲资源（通过政策倾斜保持弹性）



➢ 全局配置（yarn-site.xml）

Property	Value
yarn.resourcemanager.scheduler.class	org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler（配置调度器类型）
yarn.scheduler.fair.allocation.file	公平调度器自定义配置文件路径，该文件每10秒加载一次。
yarn.scheduler.fair.user-as-default-queue	当应用程序未指定队列名时，是否指定用户名作为应用程序所在队列的名字。如果设置为false或未设置，所有未知队列的应用程序将被提交到default队列中，默认值为true。
yarn.scheduler.fair.preemption	如果一个队列占用的资源量少于最小资源量，是否启用资源抢占机制，默认值是false。

➢ 全局配置（Web）

配置 YARN service Scheduler

Use Scheduler: ○ Capacity Scheduler ◉ Fair Scheduler

全局配置

用户名作为Queue:	True	允许抢占资源:	False
基于规模的权重:	False	分配多个Containers:	False
忽略其他Node调度因子:	-1	忽略其他Rack调度因子:	-1

Queue配置

Queue是对集群中资源的抽象,Fair Scheduler维护一组Queue,每个Queue拥有一定的资源.使用Fair Scheduler调度策略,调度器会监控每个Queue使用的资源,并且会尽量保证每个队列占用相同的资源..

共享资源抢占时间: 9223372036854774

+ 添加子队列 编辑 × 删除

Queue	最少资源	最大资源	最大Application数量	权重	调度策略
root	1024mb,1vcores	2048mb,8vcores	50	1	FAIR
default	1024mb,1vcores	2048mb,8vcores	50	1	FAIR

➢ 自定义配置（fair-scheduler.xml）

```
<allocations>
  <queue name="sample_queue">
    <minResources>10000mb,5vcores</minResources>
    <maxResources>90000mb,45vcores</maxResources>
    <maxRunningApps>50</maxRunningApps>
    <maxAMShare>0.1</maxAMShare>
    <weight>2.0</weight>
    <schedulingPolicy>fair</schedulingPolicy>
    <queue name="sample_sub_queue">
      <aclSubmitApps>charlie</aclSubmitApps>
      <minResources>5000 mb,3vcores</minResources>
    </queue>
  </queue>
</allocations>
```

minResources：分配给队列的最小资源量

maxResources：分配给队列的最大资源量

maxRunningApps：队列内同时运行的最大作业数

maxAMShare：AM使用资源占队列资源的最大比例

weight：队列的权重，默认值是1

schedulingPolicy：队列内部的调度策略，fifo或fair

aclSubmitApps：有权提交作业的用户列表

- 自定义配置（Web）

Queue配置

Queue 名称:	default	最小内存:	1024
最小VCores:	1	最大内存:	2048
最大VCores:	8	最大Application数量:	50
权重:	1	最小资源抢占时间:	92233720368547
调度策略:	FAIR	提交Application访问控制列表(ACL):	*
管理Application访问控制列表(ACL):	*		

确定

4
chapter

YARN运维与监控

- ✓ YARN运维
- ✓ YARN监控

Shell命令

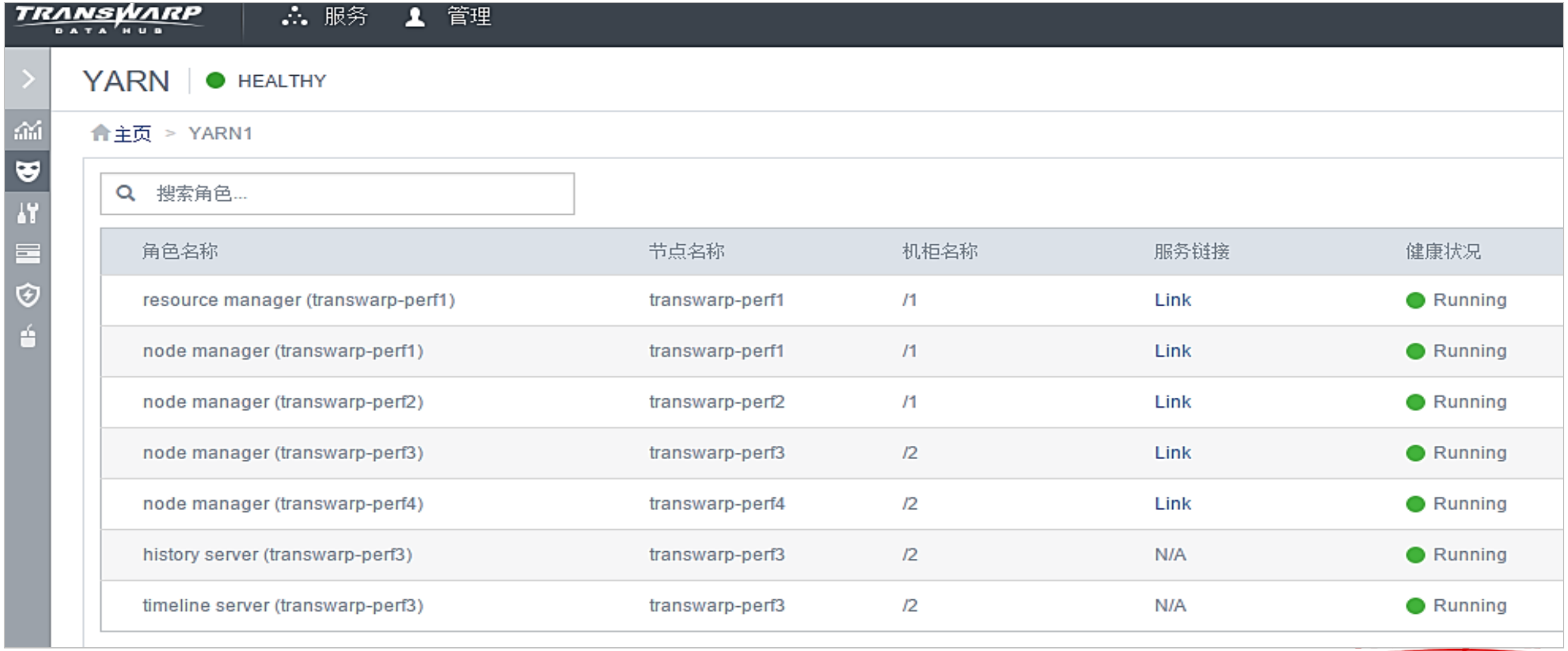
```
# yarn  application  [command_options]
```

Command Options	Description
-list	Lists applications from the RM
-kill <ApplicationId>	Kills the application
-status <ApplicationId>	Prints the status of the application

Kill任务

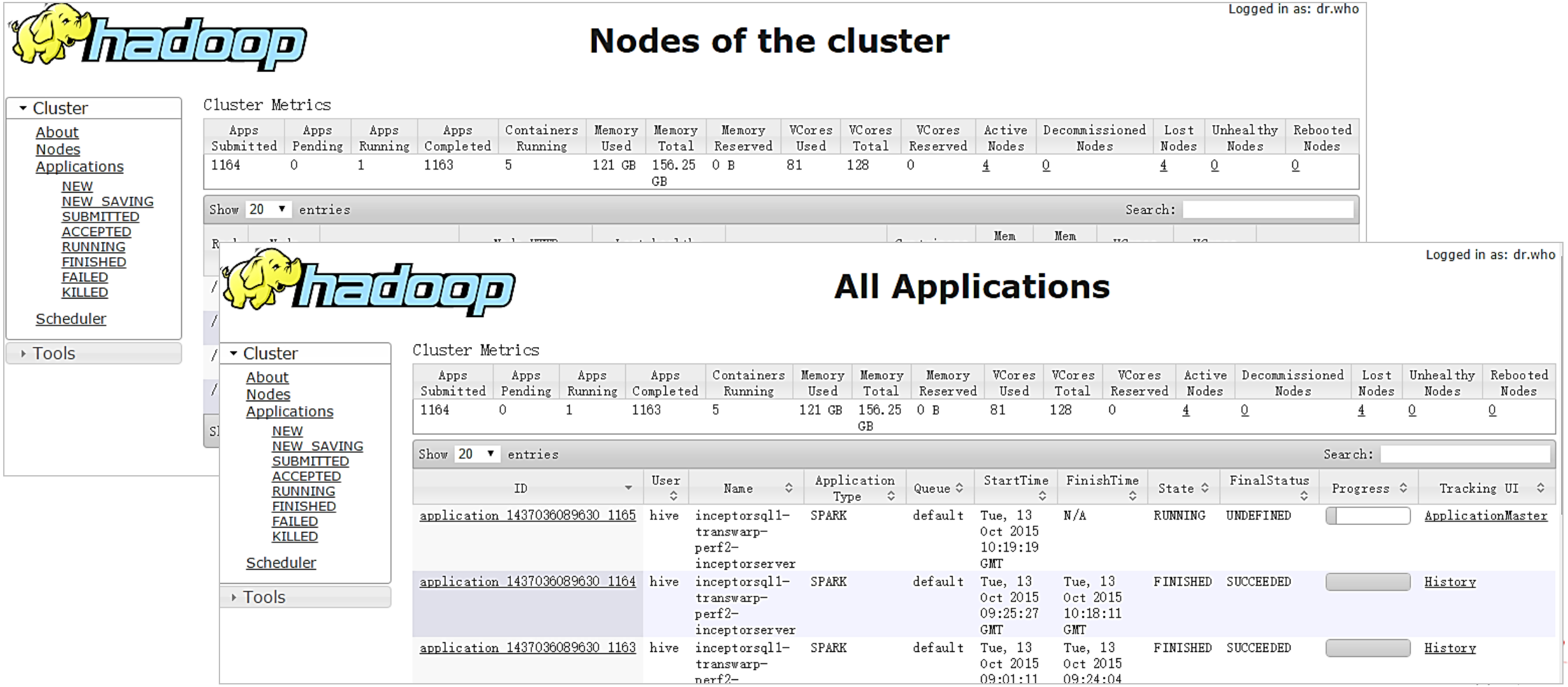
- CTRL^C不能终止作业，只是停止其在控制台的信息输出，作业仍在集群中运行
- 正确方法：先使用yarn application -list获取进程号，再使用-kill终止作业

➢ 访问Transwarp Manager → YARN → 角色 → Link，进入Resource Manager监控页面



角色名称	节点名称	机柜名称	服务链接	健康状况
resource manager (transwarp-perf1)	transwarp-perf1	/1	Link	Running
node manager (transwarp-perf1)	transwarp-perf1	/1	Link	Running
node manager (transwarp-perf2)	transwarp-perf2	/1	Link	Running
node manager (transwarp-perf3)	transwarp-perf3	/2	Link	Running
node manager (transwarp-perf4)	transwarp-perf4	/2	Link	Running
history server (transwarp-perf3)	transwarp-perf3	/2	N/A	Running
timeline server (transwarp-perf3)	transwarp-perf3	/2	N/A	Running

- 访问Resource Manager的8088端口，进入监控页面，如：http://172.16.140.204:8088



Logged in as: dr.who

Nodes of the cluster

Cluster

About
Nodes
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1164	0	1	1163	5	121 GB	156.25 GB	0 B	81	128	0	4	0	4	0	0

Show 20 entries Search:

Logged in as: dr.who

All Applications

Cluster

About
Nodes
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1164	0	1	1163	5	121 GB	156.25 GB	0 B	81	128	0	4	0	4	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1437036089630_1165	hive	inceptorsql1-transwarp-perf2-inceptorserver	SPARK	default	Tue, 13 Oct 2015 10:19:19 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaster
application_1437036089630_1164	hive	inceptorsql1-transwarp-perf2-inceptorserver	SPARK	default	Tue, 13 Oct 2015 09:25:27 GMT	Tue, 13 Oct 2015 10:18:11 GMT	FINISHED	SUCCEEDED		History
application_1437036089630_1163	hive	inceptorsql1-transwarp-perf2-	SPARK	default	Tue, 13 Oct 2015 09:01:11	Tue, 13 Oct 2015 09:24:04	FINISHED	SUCCEEDED		History

Q&A

TRANSWARP
星 环 科 技

温故知新

- 简述YARN与MapReduce的关系。
- 为什么要设计ApplicationMaster这一角色?
- Zookeeper在YARN中承担了哪些功能?
- 简述YARN的工作机制。
- 在项目实践中，如何部署YARN的ResourceManager、NodeManager和HDFS的NameNode、DataNode?

温故知新

- 队列在资源调度中起什么作用?
- 容量调度器与公平调度器的区别是什么?
- 容量调度器会严格按预设比例分配资源吗?
- 请问公平调度器如何在绝对公平的前提下适当保持弹性的?