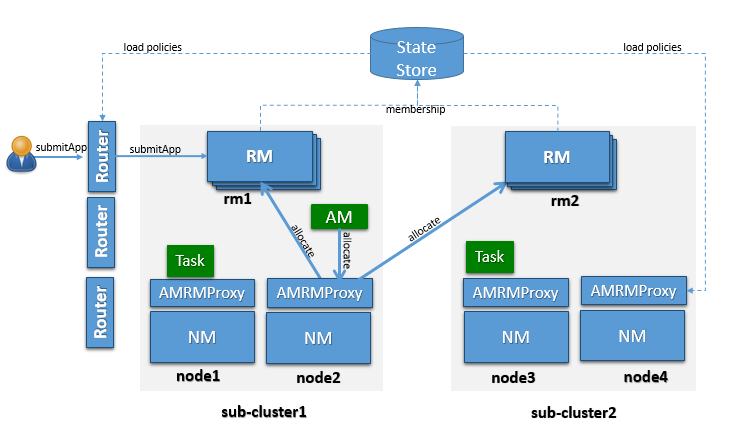
YARN Federation

Hadoop允许使用简单的变成模型跨计算机集群分布式处理大型数据集，可以从单点集群扩展到数千台集群，每台集群提供本地计算和存储，YARN是作业调度和集群资源管理框架。YARN Federation可以联合多个YARN子集群将单个YARN集群扩展到成千上万个节点，其方案是将一个大的（10-100K个节点）簇划分成子簇的较小单元，每个簇具有其自己的YARN资源管理器和计算节点，Federation将这些子簇联合在起义，作为一个大的YARN集群来运行应用程序，可以将任务分发到任意的计算节点上。

# 1.YARN Federation的系统架构

Federation System是构建在YARN基础上的上层系统，对YARN机制的改变很少，其在多个子簇RM之间进行交互，给应用提供资源，YARN Federation系统架构图如下所示：



相关概念如下：

1. **YARN Sub-cluster**

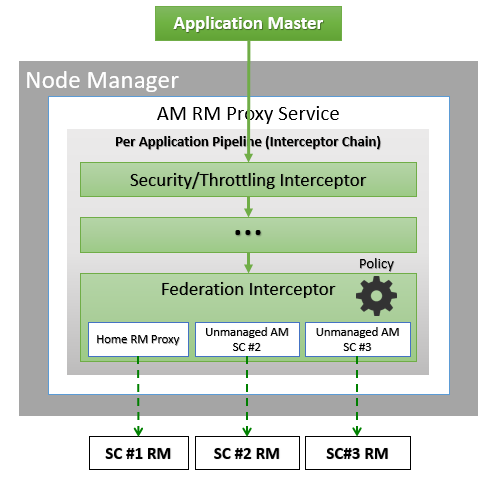
每个YARN集群的子簇可以有数千个节点，子簇的节点规模由部署及维护的工作量、网络的规模、可用性区域部署等决定的。YARN子簇是一个完整的YARN集群，在其中可以开启work-preserving和高可用，Jobs可以重新分配给其他子簇。

1. **Router**

YARN应用提交给Router，Router从Policy Store中获取对应的routing policy，并从State Store中获取子集群的URL，然后提交给正确的子集群RM。提交到的Sub-Cluster称为“Home Sub-Cluster”，其他的Sub-Cluster称为“Secondary Sub-Clusters”。Router对外提供ApplicationClientProtocol，透明的访问RM。

1. **AMRMProxy**

实现跨Sub-Cluster提交程序的关键组件。AMRMProxy运行在所有的NM机器上，作为AM与RM之间的代理，实现与RM之间的透明访问，AM不允许直接和Sub-Cluster RM之间直接交互。Job可以跨home SubCluster和多个Secondary Sub-Clusters，AMRMProxy的结构如下图所示：



其功能如下：

* 避免Misbehaving AM向YARN Sub-Cluster发送DDOS attacks，申请无用的资源。
* 实现AM透明的向多RM申请资源
* AMRMProxy提供load-balancing/overflow policies

1. **Global Policy Generator**

GPG与YARN Federation交互，保证系统正确的运行及维护。集群可用性不需要GPG持续运行，GPG与所有集群的操作是独立的。GPG维护全局的变量等信息，影响负载均衡、触发Sub-Clusters的维护等，更新用户与子集群的资源分配映射，同时修改Routers、AMRMProxy中运行的Policies。

1. **Federation State-Store，**

将多个Sub Clusters构建成单个YARN Cluster需要的状态信息。包括：

* Sub-cluster Membership，YARM RM与State Store发送心跳信息，并写入当前RM的资源和负载信息。GPG用这些信息来做正确的Policy，选择最佳的sub-cluster。
* Application’s Home Sub-Cluster，写入AM的Home Sub-Cluster信息

1. **Federation Policy Store**

逻辑上独立实现的Store，用于保持将应用和资源请求路由到其他Sub-Clusters的信息，当前实现的Policy，包括random/hashing/roundrobin/priority及考虑子集群的负载、资源请求本地性需求的Policy。

# 2. YARN Federation的使用

启用Federation功能进行的配置如下：

1. **配置yarn-site.xml文件**

* 通用参数

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.federation.enable | true | 开启federation功能 |
| yarn.resourcemanager.cluster-id | <unique-subcluster-id> | RM子集群的唯一标识 |

* State-Store的配置

目前支持两种State Store，基于Zookeeper和SQL，使用基于ZK的State Store配置

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.federation.state-store.class | org.apache.hadoop.yarn.server.federation.store.impl.ZookeeperFederationStateStore | 使用StateStore的类型 |
| hadoop.zk.address | host:port | ZK集群的地址 |
| yarn.federation.state-store.heartbeat-interval-sec | 60 | RM向State Store汇报membership的周期 |

* 配置RM相关的配置

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.resourcemanager.epoch | <unique-epoch> | Epoch的seed value，确保不同的RM生成的Container ID号唯一。Epoch值以1000为增量 |

* Router参数

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.router.bind-host | 0.0.0.0 | Router的主机IP |
| yarn.router.clientrm.interceptor-class.pipeline | FederationClientIntercepor | 可以配置多个，逗号隔开，客户端与Router交互时的拦截器 |

* NodeManager参数

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.nodemanager.amrmproxy.enabled | true | 是否启用AMRMProxy |
| yarn.nodemanager.amrmproxy.interceptor-class.pipeline | amrmproxy.FederationInterprector | 运行在AMRMProxy上的拦截器列表 |
| yarn.client.failover-proxy-provider | FederationRMFailoverProxyProvider | 与RM交互，从StateStore查找memberhip信息 |

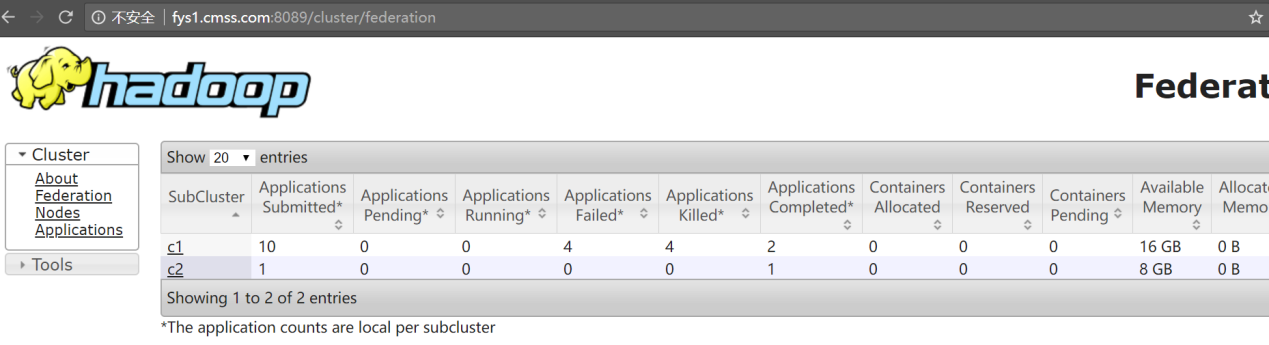
1. 启动RM及NM

*yarn-daemon.sh start resourcemanager*

*yarn-daemon.sh start nodemanager*

*yarn-daemon.sh start router*

1. 启动后Router页面如下



1. 配置客户端参数，与服务端参数不同

为了向Federation Cluster提交Job，需要配置与服务端不同的参数，配置如下

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.resourcemanager.address | <router\_host>:8050 | 将Job提交到Router的客户端RM port |
| yarn.resourcemanager.scheduler.address | localhost:8049 | 将Job提交到Federation AMRMProxy端口 |

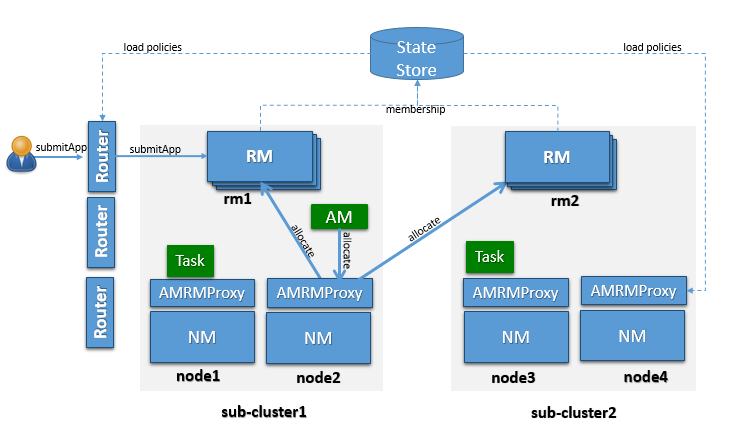
在Router所在主机上提交作业

*$HADOOP\_HOME/bin/yarn --config conf/client jar hadoop-mapreduce-examples-3.0.0.jar pi 16 1000*

5）在Global Policy Gnerator中定义Policy来选择Job提交的RM

# 3. Yarn Federation源码分析

Federation System是构建在YARN基础上的上层系统，对YARN机制的改变很少，其在多个子簇RM之间进行交互，给应用提供资源，YARN Federation系统架构图如下所示：



下面分别针对Router、StateStore、AMRMProxy及相关的模块进行源码分析

## 3.1 Router源码分析

### 3.1.1 Router启动及参数

Router的启动命令为：

*yarn-daemon.sh start router*

启动后Router进程如下：

*$ jps*

*12331 Router*

*$ netstat -anp|grep 12331*

*tcp 0 0 10.139.4.98:8050 0.0.0.0:\* LISTEN 12331/java*

*tcp 0 0 10.139.4.98:8052 0.0.0.0:\* LISTEN 12331/java*

*tcp 0 0 10.139.4.98:8089 0.0.0.0:\* LISTEN 12331/java*

*tcp 0 0 10.139.4.98:42472 10.139.4.107:2181 ESTABLISHED 12331/java*

其中Router使用的端口:

* 8089，Router的Web UI，用于展示信息
* 8050，客户端连接端口
* 8052，Router管理端口

相关参数配置如下，参数前缀ROUTER\_PREFIX <= yarn.router ：

|  |  |  |
| --- | --- | --- |
| 参数 | 描述 | 默认值 |
| bind-host | Router绑定的主机 |  |
| clientrm.address | 客户端服务绑定address | 0.0.0.0:8050 |
| clientrm.interceptor-class.pipeline | 客户端请求拦截器 | DefaultClientRequestInterceptor |
| pipeline.cache-max-size | 拦截器缓存大小 | 25 |
| rmadmin.address | 管理服务的address | 0.0.0.0:8052 |
| rmadmin.interceptor-class.pipleline | 管理请求拦截器 | DefaultRMAdminRequestInterceptor |
| submit.retry | 提交或者查询请求的retry次数 | 3 |
| webapp.address | Web | 0.0.0.0:8090 |
| webapp.interceptor-class.pipeline | Web请求拦截器 | DefaultRequestInterceptorREST |
| partial-result.enabled | Web请求返回部分结果 | false |

### 3.1.2 Router系统架构

根据分析，Router由Web、ClientRM和RMAdmin三部分及对应的Interceptor构成



1. RouterClientRMService，拦截并检查客户端到RM的请求，并根据拦截器的功能对请求和响应进行修改，其请求定义在ApplicationClientProtocol中，实现的功能包括：获取应用信息、提交应用、获取集群、队列等信息
2. RouterRMAdminService，与RouterClientRMService功能相同，不同的是实现的协议不同，其实现ResourceManagerAdministrationProtocol，主要对RM进行管理，例如刷新队列等
3. WebApp，启动Web UI
4. Interprector，在Router中实现了三种拦截器，分别对应了提供的三个服务，相关类图如下：





Interceptor的实现中，Default Interceptor的作用是基于Conf中配置的RM Address初始化clientRMProxy：

*return ClientRMProxy.createRMProxy(conf,ApplicationClientProtocol.class)*

为了支持Federation，在FederationClientInterceptor中支持多个RM：

*private Map<SubClusterId, ApplicationClientProtocol> clientRMProxies;*

不同的子集群都会有对应的ClientRMProxy，其根据SubClusterId进行初始化：

*clientRMProxy = FederationProxyProviderUtil.createRMProxy(getConf(),*

*ApplicationClientProtocol.class, subClusterId, user);*

## 3.2 StateStore

YARN Federation以StateStore为核心，保存将多个Sub Clusters构建成单个YARN Cluster需要的状态信息。StateStore相关类如下所示：



在StateStore中保存三类信息：

1. MemberShip，<SubCluterId,SubClusterInfo>，保存YARN子集群信息，包括各服务的Address、状态及资源情况



1. Applications，<ApplicationId,SubClusterId>，保存应用id运行在哪个YARN子集群上
2. Policies，<String,SubClusterPolicyConfiguration>，保存队列及对应的Policy



包括队列，policy类型及policyParams等。其中policyParams为SubClusterPolicyConfiguration

## 3.3 SubClusterResolver

用于维护节点/rack与Sub-Cluster的映射关系，其定义在文件中，配置参数为：

*<property>*

*<name>yarn.federation.machine-list</name>*

*<value>.. /etc/hadoop/fed-node</value>*

*</property>*

实现类如下：



## 3.4 RouterPolicy及PolicyManager

提交应用时，根据RouterPolicy决定提交到哪个Sub-Cluster，在FederationClientInterceptor中拦截提交请求并基于Policy选择子集群，调用RouterPolicyFacade.getHomeSubCluster，其相关类如下所示：



RouterPolicyFacade#getHomeSubcluster执行流程如下：

1. 在policyFacade中维护队列与Policy的对应关系
2. 从appSubmissionContext中获取队列信息，如果不设置则使用DERAULT\_QUEUE
3. 根据队列获取SubClusterPolicyConfiguration，如果队列没有配置Policy，则使用默认的FEDERATION\_POLICY\_KEY => “\*”，根据Conf对Policy进行重新初始化
4. 使用FederationPolicyManager对PolicyMap进行重新赋值
5. 从policyMap中获取队列对应的FederationRouterPolicy
6. 调用Policy.getHomeSubcluster

其中核心的是FederationPolicyManager，其配置参数如下：

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.federation.policy-manager | WeightedLocalityPolicyManager | 决定应用和请求如何在系统中进行路由 |
| yarn.federation.policy-manager-params | <binary> | Policy的PayLoad，例如Router和amrmproxy中的Weightes列表，通过policyManager进行.json文件的反序列化 |

在步骤4中，会调用FederationPolicyManager对Policy的信息进行更新，其相关类如下：



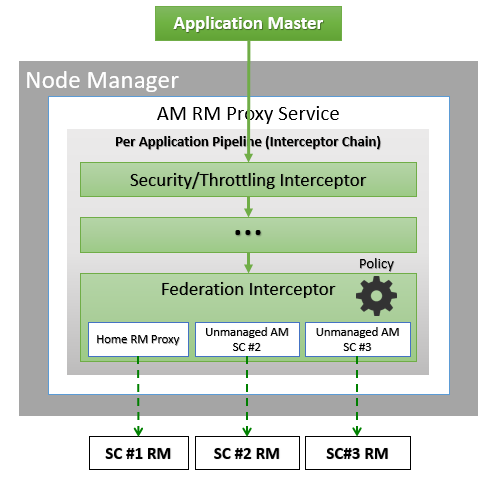
SubClusterPolicyConfiguration用于配置Policy，类图如下所示：



配置中包含了队列、Policy的类型和PolicyParams，其中policyParams是二进制字符串。对于PolicyParams进行反序列化的类目前有WeightedPolicyInfo，定义SubCluster及其权重，如上类图。

## 3.5 AMRMProxy

实现跨Sub-Cluster提交程序的关键组件。AMRMProxy运行在所有的NM机器上，作为AM与RM之间的代理，实现与RM之间的透明访问，AM不允许直接和Sub-Cluster RM之间直接交互。Job可以跨home SubCluster和多个Secondary Sub-Clusters，AMRMProxy的结构如下图所示：



相关参数如下表所示：

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 描述 |
| yarn.nodemanager.amrmproxy.enabled | true | 是否启用AMRMProxy |
| yarn.nodemanager.amrmproxy.interceptor-class.pipeline | amrmproxy.FederationInterprector | 运行在AMRMProxy上的拦截器列表 |
| yarn.client.failover-proxy-provider | FederationRMFailoverProxyProvider | 与RM交互，从StateStore查找memberhip信息 |

AMRMProxy相关类图如下所示：



在AMRMProxyService中维护应用与拦截器的映射关系：

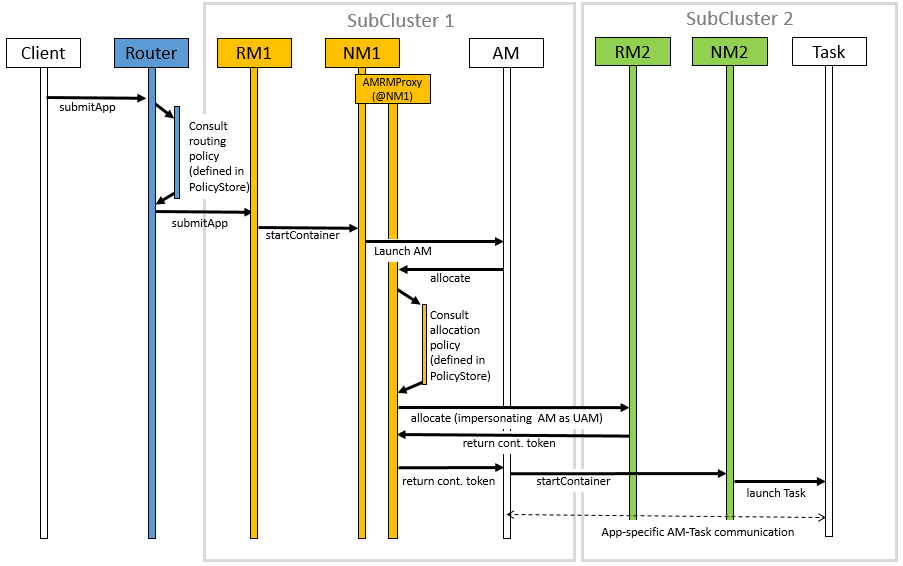
*Private Map<ApplicationId,RequestInterceptorChainWrapper> applPipelineMap;*

其核心方法是RequestInterceptor#allocate(request:AllocateRequest):AllocateResponse，向Home RM和sub-Cluster RM发送心跳：

1. 将AllocateRequest切分成多个SubCluster RM对应的AllocateRequest
2. 将请求发送到secondary sub-cluster的RM，在Sub-Cluster中会创建Unmanaged AM，如果是新创建的Unmanaged AM，则注册到Interceptor中。将返回的分配响应信息与Home SubCluster的响应信息进行合并
3. 将请求发送到Home SubCluster，并获取响应信息

# 4.应用提交及启动流程分析

客户端向Router提交应用创建请求，Router根据定义的Policy决定提交到哪个Sub Cluster的RM上，该Sub Cluster为Home Cluster。AM启动后，通过本地运行的AMRMProxy与RM交互，AMRMProxy实现ApplicationMasterProtocol协议。AMRMProxy所有的Sub Cluster进行资源的请求，因此对AM是透明的。客户端向Federation YARN提交程序的流程图如下所示：



1）Router实现YARN Client Protocol，接受Application Submission Request请求，其对应类为RouterClientRMService，配置参数为：

*<property>*

*<name>yarn.resourcemanager.address</name>*

*<value>fys1.cmss.com:8050</value>*

*</property>*

2） Router使用Routing Table/Policy选择Job运行的Home RM，Policy的配置信息从StateStore中获取，实现方式是使用ClientInterceptor，执行如下所示：

*SubClusterId subClusterId = policyFacade.getHomeSubcluster(*

*request.getApplicationSubmissionContext(), blacklist);*

*ApplicationClientProtocol clientRMProxy =*

*getClientRMProxyForSubCluster(subClusterId);*

3) Router将Application Submission Request发送到Home RM中，并使用Home SubCluster的标识来更新application State

1. 提交到Home RM后，开始YARN 程序的启动流程，例如application添加到scheduler queue中，并且在Home SubCluster中启动AM。
2. AM启动后，向AMRMProxy请求资源，相关配置：

*<property>*

*<name>yarn.resourcemanager.scheduler.address</name>*

*<value>localhost:8049</value>*

*</property>*

该端口为NM上的AMRMProxy。

1. AMRMProxy，通过在其他SubCluster上提交一个Unmanaged AM，当AMRMProxy启动HA后，UAM Token注册得到YARN Registry中。AM会attach这些UAMS

*token = uamPool.launchUAM(subClusterId, config,*

*appContext.getApplicationAttemptId().getApplicationId(),*

*amRegistrationResponse.getQueue(), appContext.getUser(),*

*homeSubClusterId.toString(), registryClient != null);*

1. AMRMProxy使用State Store中的policy来决定向Home RM还是Secondary RM申请资源。Secondary RM会注册tokens，AMRMProxy使用这些token与Secondary RM交互
2. AMRMProxy向AM返回Allocation response

10) AM向NM上发送Container启动请求

参考文献：

https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/Federation.html