YARN Placement Constrains

在集群规模日益增大的背景下，集群内运行的主机类型会越来越多，主机磁盘读写性能、网络、CPU计算资源在不同类型的主机上有差异。目前在YARN中，仅支持Task数据本地性（计算运行在数据所在的节点或者机架）上，可以根据node-label将分配Container到相应主机上，为了支持更灵活的计算放置上，YARN-5692引入了富放置约束，可以指定应用使用这些放置约束提高性能和扩展性，有以下几种场景：

1. Affinity(亲和性)，将同Job的Tasks放置在同一个机架上（亲和性）减少网络带宽
2. Anti-affinity（反亲和性），将同Job的Tasks分布到不同的主机上（反亲和性）来减少资源影响
3. Cardinality（基数），在一个Node Group（Cardinality）中允许Job启动的特定Task数目，来实现Group的均衡

选择恰当的放置约束策略提高Job的扩展性，例如在同一集群升级域的Task可以同时下线根据以上的场景，YARN为了支持放置约束支持一下需求：

1. 支持Job及跨Job的多种放置约束，包括Affinity、Anti-affinity及Cardinality等，在调度时对底层网络拓扑及其他部署的应用是不感知的
2. 支持不同的调度器
3. 支持集群的调整，可以对跨节点负载均衡及资源碎片
4. 要充分考虑多种放置需求及约束

# 相关概念及实现

1. **Nodelabel**

NodeLabel是将相似节点进行分组的一种途径，将整个集群在逻辑上划分成多个分段。目前在YARN中支持基于NodeLabel的调度策略，用户可以将具有相同特点的NM标注标签，然后指定队列使用哪个标签，提交到该队列的应用只能使用这些节点上的资源，YARN的这种NodeLabel是Partition Label，使用配置参数为：

*<property>*

*<name>yarn.node-labels.enabled</name>*

*<value>true</value>*

*</property>*

*<property>*

*<name>[yarn.node-labels.manager-](http://223.105.0.132:8090/display/BCH/yarn.node-labels.manager-)class</name>*

*<value>\*\*\*.[RMNodeLabelsManager</value](http://223.105.0.132:8090/display/BCH/org.apache.hadoop.yarn.server.resourcemanager.nodelabels.RMNodeLabelsManager</value)></property>*

YARN还支持以下几种设置Container的运行标签：

* ApplicationSubmissionContext.setNodeLabelExpression
* ResourceRequest.setNodelabelExpression

在MapReduce中可以通过以下参数配置NodeLabel

*mapreduce.job.node-label-expression*

*mapreduce.job.am.node-label-expression*

*mapreduce.map.node-label-expression*

*mapreduce.reduce.node-label-expression*

1. **Allocation Tags**

NodeLabel用于标记节点，但是Allocation Tags用于标记allocations。Scheduler将allocation分配到一个节点上时，自动将Allocation Tag添加到Node上，同时Rack也继承节点的Tag。

Allocation属于一个HBase Job，那么该Allocation具有Tag: hbase，用于表述该应用的类型。在使用过程中，可以使用Tags来标识应用的组件：

*HBase Master => hbase-m*

*HBase RegionServers => hbase-rs*

1. **Node attributes**

NM节点有多种不同的属性，例如hostname、lib版本号等，可以将这些NM属性保存到NM State Store中，在启动时汇报给RM。Node Attributed的表示如下：

*key=value*

默认情况下每个Node具有的属性为host和rack，分别表示主机和机架。YARN调度器可以使用节点属性来进行分配：

*{target: node-attribute:host IN ANY, scope: host}*

# **Placement Constraints**

向YARN申请资源通过AllocateRequest来封装，其类如下：

*public class AllocateRequestPBImpl extends AllocateRequest {*

*private List<ResourceRequest> ask = null;*

*private List<ContainerId> release = null;*

*private List<UpdateContainerRequest> updateRequests = null;*

*private List<SchedulingRequest> schedulingRequests = null; //new*

*private ResourceBlacklistRequest blacklistRequest = null;*

*}*

ResourceRequest用于allocation的数目、资源大小、execution type等，为了支持更丰富的放置，在AllocateRequest中增加SchedulingRequest，其包括的数据：

* requestId
* Priority
* Allocation tags
* ResourceSizing，分配的数目及每个分配的资源带下
* PlacementConstraint Expression ，富分配约束的表达式，也是整个核心功能的初始数据，

使用户可以描述affinity、anti-affinity及cardinality restrictions的表达式，下面介绍相关的概念，下面介绍PlacementConstaint基本概念及使用

1. Hierarchical Placement Constraints，根据应用定义范围有不同层次的放置约束

* SchedulingRequest-level，在Allocations Request中使用
* Application-Level，当AM注册时定义在RegisterApplicationMasterRequest中
* Cluster admin-level，通过API来定义，可以被所有应用使用

1. Placement Constraint Expression definition，约束表达式由简单的约束通过运算符组合而成

* ConstraintExpressionOperation

1. AND，满足所有约束条件

*on this rack AND do not more than 2 on each machine*

1. OR，至少满足一个约束

*“host n1 OR host n2”*

1. DELAYED\_OR，类似于OR，但是约束的顺序和delays

*Try for 2 mins to place me to host n1；then try for 3 mins to place me on rack r1;if everything else fails, place me on any node.*

DELAY使用标准如下：

表达式中每个约束的delay units数目

Delay unit可以不调度或者wall clock time

等获取的Allocate后可以重置Delay Unit

* SimplePlacementConstraint

1. List<TargetExpression>，TargetExpression的格式:

*<target-type>:<target-key> <op> <target-values>*

* Target-type:{node-attribute,allocation-tag}
* Target-key: node attribute的key值或者empty(allocation-tag)
* Op: {IN, NOT\_IN}
* Target-values: 逗号分离的值列表

*node-attribute:host IN {n1,n2}*

*allocation-tag NOT\_IN spark*

当配置多个Targets，需要满足所有的条件

*<node-attribute> IN “node-partition=x”,*

*<node-attribute> IN “host=n1,n2,n3”,*

*<node-attribute> NOT\_IN "os=window"*

1. Min/Max-cardinality: int值
2. Scope，定义node attributed key(例如，host,rack)

在Application-level和SchedulingRquest-Level的约束，尽量让约束简单，两种配置格式：

* TargetConstraint{List<TargetExpression>,scope}，允许Container分配到一个Scope中（例如host或者rack）
* CardinalityConstraint{min/max-cardinality,scope}，限制Container在一个Scope中的数量

在Cluster Admin-level，允许使用所有的字段，例如集合中有N个节点属于一个约束的范围，如果TargetExpression在[min-cardinality~max-cardinality]范围内，则满足约束条件。

1. Node Attribute Kyes

每个集群节点有多个属性，格式为<key>=<value>，默认情况下内嵌host和rack属性，例如在约束中可以配置：

*node1: “host=n1”, “rack=r1”, “fault\_domain=fd1”*

*node2: “host=n2”, “rack=r1”, “fault\_domain=fd2"*

# **约束使用**

## 3.1 约束的定义

下面是通过PlacementConstraintExpression API的约束使用示例：

1. Affinity between allocations

{

*target: allocation-tag IN “hbase-m”,scope:host*

}

在*HBase Master*所在节点上启动Container

1. Affinity between allocations and max cardinality of containers placed

*AND: [*

*{ Target: allocation-tag IN “zookeeper”, scope:rack},*

*{max-cardinality:3, scope: host}*

*]*

在Zookeeper所在的rack上启动，但是每个节点上最多启动3个Container

1. Affinity to node attributes

{

*target: node-attribute: node-partition IN “GPU”,*

*scope:host*

}

在GPU主机上分配Container

1. Anti-affinity to allocation tag

*{*

*target: allocation-tag NOT\_IN “spark”, scope: fault\_domain*

*}*

启动在非Spark所在的fault\_domain上启动Container

1. Affinity and delayed scheduling

*DELAYED\_OR: [*

*AND: [*

*{target: node-attribute: host IN “n1,n2,n3”, scope: host},*

*{max-cardinality:2, scope: host}*

*],*

*OR: [*

*AND: [*

*{target: node-attribute: rack IN ‘r1’, scope: host},*

*{max-cardinality:4, scope: rack}*

*],*

*AND:*

*{target: node-attribute: rack IN ‘r2’, scope: host},*

*{max-cardinality:2, scope: rack}*

*]*

*],*

*{target: node-attribute: host IN ANY, scope: host}*

*]*

机架R1: 节点N1和N2

机架R2: 节点N3

* 每个节点至多分配2个Container
* Rack 1上至多分配4个Container, Rack2上至多2个Container
* 集群上的任意节点都可以分配Container

## **3.2 调度上使用约束**

约束通过SchedulingRequest传递给YARN来使用，在SchedulingRequest中定义使用哪种约束：

*sourceTags: List<allocation-tag>*

*constraintList: List<PlacementConstraintExpression>*

分配Container必须要满足Source Tag定义的约束。下面是一个例子：

*{*

*sourceTags: {storm, appId\_0045},*

*constraintList:{*

*{target: allocation-tag IN “hbase-rs”},*

*{target: allocation-tag IN “appID\_0023”}*

*},*

*Scope: host*

*}*

appId\_0045的应用申请Storm Container，启动在appId\_0023的HBase的某个RegionServer所在的节点上。

第二个示例：

*{*

*sourceTags: {storm},*

*constraintList:{*

*target: allocation-tag IN “hbase-rs”,*

*scope:”rack”,*

*min-cardinality:3*

*}*

*}*

在至少启动3个RegionServer上的Rack上申请Storm的Container。

Allow For Labels on nodes and resource-requests:

https://issues.apache.org/jira/browse/YARN-2492

Rich Placement Constaints in YARN:

https://issues.apache.org/jira/browse/YARN-6592

Allocation Tags: https://issues.apache.org/jira/browse/YARN-7779

Node Attribute: https://issues.apache.org/jira/browse/YARN-3409

https://blog.csdn.net/androidlushangderen/article/details/52495850

https://dataworkssummit.com/san-jose-2018/session/rich-placement-constraints-who-said-yarn-cannot-schedule-services/

https://zh.hortonworks.com/blog/explore-latest-apache-hadoop-yarn-dataworks-summit-san-jose-2018/