

Pluggable sharing policy for Partition Node Label resources

Author : Naganarasimha G R & XinXianYin

Inputs from : Wanda Tan, Sun Mark & Rohith Sharma K S

Last Modified : 06-Dec-2015

Existing Sharing policy of Partition Node Label

1. As part of **YARN-3214**, concept of non exclusive Partitions was introduced and these partitions were sharing the resources to **Default Partition only**
2. While scheduling nodes resources of a specific non exclusive partition, Resource requests(resource request) for the same partition are given first preference and later if resources are available then tries to allocate for resource request with Default Partition.
3. AM containers of Default partition will not be launched in Non Exclusive Partition thus avoids risk of preemption.
4. Application's resource request's with Default Partition will be allocated with containers in non exclusive partitions only after Application misses scheduling opportunities (in any non exclusive partition) equivalent to the size of cluster.
5. In non exclusive mode, there is no max capacity on any queue, queue's can make use total non exclusive partition's resource

Problem Statement

1. Existing sharing is only to resource request of Default Partition, so its not suitable when we want fixed set of nodes for particular queue(/tenant) and on demand use resources from other partitions who have free resources. With existing implementation only way to get this functionality is to make AM request for resources in other partitions which requires changes in each application hence not suitable.
2. As per existing solution if apps in Default Partition have to use resources of any non exclusive partition then it has to miss scheduling opportunity of non exclusive partition nodes equivalent to the size of cluster. This will delay launching as after each allocation(to any partition) counter is reset. So when the partition resources are filled then App's resource requests will have to wait for longer duration.
3. No app level interface (or @ priority level in resource requests), to disable using shared partitions though applications are running in non exclusive Partitions.

Use cases for better Partition sharing

1. **Multi - tenant Scenario :** In a multi tenant cluster we would like to have each tenant to have a private partition which he **doesnt want to share resources but occasionally would require additional resources for faster processing.**
2. **Hierarchical** In most of the cases queues/partitions are structured based on workload or by departments(/tenants) and some cases there can be hybrid i.e. first level division based on tenant and second level based on hardware classification(High /Low). There will be

necessity to share the resources of these sub types only to the resources under its parent. In this scenario they want the resources to be shared from high to Low under one parent. Basically **sharing of resources for other specific partitions only**.

3. **Effective Resource Utilization,**

- Consider clusters are partitioned based on workload, i.e. High configurations machines for spark which usually run some analytical queries which require high RAM. During **off peak hours they want to share the partition** to batch kind of jobs instead of sharing resources when intermittently resources are free to be used as it would need to unsafe preemption to get the resources back.
 - **Share based on the resource utilization** of the partition. Ex: Share Non-Exclusive partition when the resource usage is less than 50% (including resources shared for other partitions)
 - **Share a quantum of resource** like share only 25% of partition resources to other partitions.
4. **Flexible mechanism for partitions to use resources of non Exclusive partitions** : In certain multi tenant scenarios, some of the customers would reserve initially low number of nodes but as the demand for resources increases then immediately they want to use the resources present in the non exclusive partitions.
For example if the partition's used capacity for the queue has reached guranteed capacity (or configured capacity)

Solution Proposal

- Support **PartitionSharingPolicy**, which provides information for a given non exclusive partition how it wants share its resources.
- Support **PartitionElasticityPolicy** which provides information as to when to use resources of other non exclusive partition.
- Support application specific option to disable usage of non exclusive partitions though the application is run in a partition which has **PartitionElasticityPolicy** set to use other.

Note:

- As per current behavior CS doesnt allow to share the resources of Exclusive partitions to others, actually we do not require explicit types of partitions if the above policies are defined, but in terms of compatability (wrt existing installations already using Exclusive and non exclusive partitions) we propose the **PartitionSharingPolicy** only to NonExclusive Partitions only.
- **PartitionElasticityPolicy**, we think is required for Exclusive partitions too as in cases where configured Exclusive partition nodes are not sufficient then they can make use of it. But by default we propose **PartitionElasticityPolicy** for Exclusive partition as to not use resources from other nonExclusive partitions.

Solution Details

1. **PartitionSharingPolicy**

package *org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity;*

```

public interface PartitionSharingPolicy {
    /**
     * @param conf
     * @param nonExecPartiton :Partition for which this Policy applies
     */
    void initialize(CapacitySchedulerConfiguration conf, String
nonExecPartiton);

    /**
     * @return List of the partitions which to which the resources to be
shared
     *         and the order in which to share
     */
    List<String> getPartitionsToShareResources(CSQueue queue);

    /**
     * Amount of Resource of a given non Exclusive Partition to Share with
the
     * target partition.
     *
     * @param queue, Queue to which the resources needs to be shared
     * @param clusterResource , required by resource calculator
     * @param partitionLabel, name of the label to which the resource needs
to be
     *         shared
     * @return
     */
    Resource getPartitionResourceToShare(CSQueue queue, Resource
clusterResource,
        String partitionLabel);
}

```

- **initialize** : During the initialization of the CapacitySch for each of the non exclusive partition label from the NodeLabelsManager check for the configuration "yarn.scheduler.capacity.<label>.partition-sharing-policy" in capacity scheduler xml file. If none is specified then default Policy similar to the existing one will be applied.
- **getPartitionsToShareResources**: In the existing approach, a given non exclusive node is shared only to Default Partition, but here we provide based on the policy. Admin can have the option to limit partitions only accessible to a queue or all partitions. (through this interface we can get partition hierarchy approach). Also through this interface particular partition can decide when to share the resource based on the resource usage of the partition in the queue. In "Ignore Partition exclusivity" mode for scheduling a non exclusive Partition Node we currently consider the target partition to be "Default Partition" only, but based on this interface we can loop for each targetted partition in the predefined order. **Default behavior is to support only for "Default Partition"**
- **getPartitionResourceToShare** : In the existing approach we allow any queue's app to use 100% of the nonExclusive partition's resource. But through this interface we will be able limit the usage of each partition's resource to a configurable size based on the target queue. This can be utilized in `LeafQueue.computeUserLimit()`

2. PartitionElasticityPolicy

```

public interface PartitionElasticityPolicy {
    /**
     * @param conf
     * @param nonExecPartiton :Partition for which this Policy applies
     */
    void initialize(CapacitySchedulerConfiguration conf, String
nonExecPartiton);

    /**
     * Decides whether to schedule the app in the non exclusive Partition
     *
     * @param queue
     * @return true when to use the
     */
    boolean canScheduleOnNonExclusivePartition(CSQueue queue,
FiCaSchedulerApp app);
}

```

- **initialize** : During the initialization of the CapacitySch for each of the label from the NodeLabelsManager check for the configuration *"yarn.scheduler.capacity.<label>.partition-elasticity-policy"* in capacity scheduler xml file. If none is specified then default Policy similar to the existing one will be applied.
 - **canScheduleOnNonExclusivePartition**: In the existing approach, Default Partition can be used only when the resource is not available to be allocated from Default Partition. But through this interface, partition can check if its usage Capacities exceeds and only then schedule in other partitions. Like user can specify if the given partition capacity usage for this queue exceeds 90% then assign and need not wait for **missed scheduling opportunities** to reach the cluster size.
Default behavior is to use Non Exclusive Partition for an App only when App misses scheduling opportunities equivalent to size of a cluster.
 - **Some additional methods like** UpdateAllocationForAnApplication, applicationCreated & applicationRemoved: Through these interface methods we can push the existing logic for app to be assigned only after app misses scheduling opportunities equivalent to size of a cluster as part of elasticity policy.
3. **Application specific disable of Partition elasticity** : Since containers of apps allocated on partitioned nodes will have propability to be preempted, we should have a way to disable such allocation. Similar to the approach mentioned in **YARN-3214** we can add a "preemptible" flag to **resource request** or **application submission context**. RM will not allocate resource such that it can preempted in such case.

Topics for Discussions

1. **Head Room calculation** : Currently we are informing the head room for the app based on default Partition's resource only, i think we need to set for each partition available in the queue. So that faster/more allocation can happen from the apps.
2. **PreEmption of resources used by Other Partitions** : More analysis needs to be done. In order to preempt resources for a given partition we can have a policy for given Non-Exclusive partition how it wants to preempt containers used by other Partitions. But priority must be first given to preemption of the resources across the partition before considering across the

queues within the same partition.

3. **PartitionElasticityPolicy for Apps**, Should the scope of app should be limited only to specify whether to follow the extension policy of the requested partition or to have its own policy when to use other Non Exclusive partitions ? For the sake of simplicity have not extended this concept to apps but can be **further** discussed.
4. **DataStructures in the exposed Interfaces** : IMO interfaces should not expose the internal data structures so may be we need to fine tune the arguments to be wrapper of the original objects and only expose the required methods only. For example instead of exposing *CSQueue* in *PartitionSharingPolicy.getPartitionsToShareResources*, we can wrap it up with some other name only exposing desired getters. For the sake of simplicity have used the internal data structures.
5. **Performance Impact on CS's scheduling** : IMO if **PartitionSharingPolicy** doesn't specify the partition to be shared with large number of partitions then there will be less or no impact. Also we will be using the default policy of the partition to be shared with only Default Partition.