Capacity Scheduler support of “vector of resources percentage”

TOC

Motivation

Currently, the Capacity Scheduler queue configuration supports two ways to set queue **capacity**.

a) In percentage of all available resources as a float (eg. 25.0) means 25% of the resources of its parent queue for all resource types equally (eg. 25% of all memory, 25% of all CPU cores, and 25% of all available GPU in the cluster). The percentages of all queues has to add up to 100%.
b) In an absolute amount of resources (eg. memory=4GB,vcores=20,yarn.io/gpu=4). The amount of all resources in the queues has to be less than or equal to all resources in the cluster.

Apart from these two already existing ways, there is a demand to add capacity percentage of each available resource type separately. (eg. memory=20%,vcores=40%,yarn.io/gpu=100%). At the same time, a similar concept should be included with queues **maximum-capacity** as well.

Proposed configuration

```
<property>
  <name>yarn.scheduler.capacity.root.a.capacity</name>
  <value>memory-mb=25%,vcores=50%,yarn.io/gpu=0%</value>
  <description>
    The minimum capacity of queue “a“.
  </description>
</property>

<property>
  <name>yarn.scheduler.capacity.root.b.capacity</name>
  <value>memory-mb=75%,vcores=50%,yarn.io/gpu=100%</value>
  <description>
    The minimum capacity of queue “b“.
  </description>
</property>

<property>
  <name>yarn.scheduler.capacity.root.a.maximum-capacity</name>
  <value>memory-mb=100%,vcores=100%,yarn.io/gpu=0%</value>
  <description>
    The maximum capacity of queue “a“.
  </description>
</property>

<property>
  <name>yarn.scheduler.capacity.root.b.maximum-capacity</name>
  <value>memory-mb=100%,vcores=100%,yarn.io/gpu=100%</value>
  <description>
    The maximum capacity of queue “b“.
  </description>
</property>
```

Design thoughts

The following distinct parts could be separated for the implementation:

a) Create unit tests for absolute resource capacity
b) Create an EffectiveCapacityCalculator interface + implementations, and extract functionality from ParentQueue#calculateEffectiveResourcesAndCapacity
c) Support percentage vector of resources in Capacity Scheduler configuration
d) Update Capacity Scheduler queue to support vector of resources percentage
e) Update Capacity scheduler config to handle absolute resources and resource types percentages for `leaf-queue-template.capacity` and `max-capacity`.
f) Update UI and metrics to support vector of resources for Capacity Scheduler
    i) Update UIv1 to support vector of resources percentage
    ii) Update UIv2 to support vector of resources percentage
g) Update inter-queue preemption to support vector of resource percentages
h) Documentation for the vector of resources percentage

Create unit tests for absolute resource capacity

To avoid breaking anything we need to create some test cases that cover current functionality eg. around regex parsing with `RESOURCE_PATTERN`.

Create an EffectiveCapacityCalculator interface + implementations

Implementation

Create `EffectiveCapacityCalculator` interface and implementations:
`NoneStrategyEffectiveCapacityCalculator`
`AbsoluteResourceEffectiveCapacityCalculator`
`GlobalPercentageEffectiveCapacityCalculator`
`ResourcePercentageEffectiveCapacityCalculator`

These would get a `CSQueue` object, a `label` string, and a `ResourceCalculator` object (and whatever else it may need), and update `effectiveMinResource` and `effectiveMaxResource` for the given `CSQueue`.

Functionality from `ParentQueue#calculateEffectiveResourcesAndCapacity` should be extracted to these implementations.

The `effectiveMinResource` sets the minimum amount of resources a queue should get if it has running applications, but it may use more than this limit up to `effectiveMaxResource` if there are free resources in the cluster providing elasticity.

`AbsoluteResourceEffectiveCapacityCalculator`
This implementation would implement current parsing logic to digest absolute resource values, eg. “`memory=4Gb, vcores=2`”.

`GlobalPercentageEffectiveCapacityCalculator`
This implementation would use current parsing logic to parse global percentage across all resources eg. “`25%`”

`NoneStrategyEffectiveCapacityCalculator`
The purpose of NoneStrategy implementation is that the default strategy is none as the default, and the proper strategy is selected after matching some regular expressions. In case someone accidentally breaks the code, or probably an incorrect setting is given in one of the properties, the strategy may remain “none”. NoneStrategy should throw an `IllegalStateException` stating that an unreachable state has been reached so these states are easier to debug.

`ResourcePercentageEffectiveCapacityCalculator`
The last implementation could be created in a separate Jira from the refactoring part. This would digest the new style of configuration eg.
“`memory-mb=25%,vcores=50%,yarn.io/gpu=0%`”

Testing

Unit tests need to be created for all implementations including None implementation that should throw an exception. Regex parsers and validation should be tested thoroughly.

Support percentage vector of resources in CS configuration

Implementation

`CapacitySchedulerConfiguration` already supports a float or a string as `queue.capacity` and `queue.maximum-capacity` used for percentage and absolute declaration of resources accordingly.

As the new way of declaration will also be string-based, that part of the code can be reused, with some renaming across method parameter names, eg.
`public void setCapacity(String queue, String absoluteResourceCapacity)`

A separate regex pattern should be introduced for the new percentage vector declaration. This should be added to all methods that use `RESOURCE_PATTERN`.

As from now on Capacity Scheduler will use a similar pattern to Fair Scheduler, it would be a neat thing to extract resource type parsing to a separate class, and use this parser in both schedulers, but current implementations differ so much, that it seems to be a better idea to do this in a separate Jira.

The rest of the processing is done in `AbstractCSQueue`, thus it will be broken down in the "Update Capacity Scheduler queue to support vector of resources percentage" chapter.

Testing

There are a limited number of tests touching the `CapacitySchedulerConfiguration` class min/max resources setting in `TestCapacityScheduler`. Some new test cases could be introduced that test the areas that touch `RESOURCE_PATTERN` and the newly introduced pattern.

Update CS queue to support vector of resources percentage

Implementation

In `AbstractCSQueue`, absolute config is already supported as of YARN-5881, so:

a) In `AbstractCSQueue CapacityConfigType.PERCENTAGE_RESOURCE` should be introduced and used accordingly when needed.
b) Sum of 100% per each resource type for every queues children should be done here, as done for global percentages in `org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.ParentQueue#setChildQueues`
c) In `ParentQueue` effective queue capacity calculation should be updated accordingly. `EffectiveCapacityCalculator` should be used in `calculateEffectiveResourcesAndCapacity`
d) Update `CSQueueUtils` to be aware of resource percentage calculation

Testing

Create characterization tests for `ParentQueue` to assure, that the current functionality works as expected after the refactor.
Add a new test case after `ResourcePercentageEffectiveCapacityCalculator` has been added, to assure that new resource type calculation works as expected.
Check validation of 100% sum for child queues per resource type.

Update CS config to handle resource types for leaf-queue-template.

The properties `leaf-queue-template.capacity` and `leaf-queue-template.max-capacity` determine the capacity and max-capacity values of new dynamically created queues. According to YARN-9893, updating them is also necessary.

Implementation

In `CapacitySchedulerConfiguration`, add necessary methods that call respective `setCapacity` and `setMaximumCapacity` implementations.

Testing

Add respective test cases to `TestCapacitySchedulerAutoQueueCreation`.

Update UI and metrics to support vector of resources for CS

Both UIs display absolute resources, so probably no update is necessary for the UI-s. This needs further investigation as soon as implementation for the feature is ready. If any change is necessary, that should be done in a separate jira.

Update preemption to support vector of resource percentages

Preemption seems to use capacity, not resource limits. Capacity is set by the use of `ResourceCalculator` and if `DominantResourceCalculator` is used, that should include resource types for preemption as well, so hopefully, no change is needed here. Further investigation is necessary.