Cloudera Runtime 7.2.8

Managing and Allocating Cluster Resources using Capacity Scheduler

Date published: 2020-02-11
Date modified: 2021-03-25

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Resource Scheduling and Management 5

YARN resource allocation of multiple resource-types 5
Hierarchical queue characteristics 5
Scheduling among queues 6
Application reservations 6
Resource distribution workflow 6
Resource allocation overview 8
Use CPU scheduling 8
- Configure CPU scheduling and isolation 9
- Use CPU scheduling with distributed shell 9

Use FPGA scheduling 9
- Configure FPGA scheduling and isolation 10
- Use FPGA with distributed shell 11

Limit CPU usage with Cgroups 11
- Use Cgroups 12
- Enable Cgroups 12

Manage Queues 14

Prerequisite 15
Add queues using YARN Queue Manager UI 16
Configure cluster capacity with queues 20
Change resource allocation mode 21
Start and stop queues 22
Delete queues 23

Configure Scheduler Properties at the Global Level 23

Set global maximum application priority 26
Configure preemption 27
Enable Intra-Queue preemption 27
Set global application limits 28
Set default Application Master resource limit 28
Enable asynchronous scheduler 29
Configure placement rules 29
- Dynamic queues 29
- Create placement rules 30
- Reorder placement rules 30
- Edit placement rules 31
- Delete placement rules 31

Configure queue mapping to use the user name from the application tag using Cloudera Manager 31
Configure NodeManager heartbeat 32
Configure data locality 32
Configure placement rules 33

Configure Per Queue Properties 33

Set user limits within a queue 35
Set Maximum Application limit for a specific queue 37

Set Application-Master resource-limit for a specific queue.... 38
Control access to queues using ACLs.... 38
Enable preemption for a specific queue....39
Enable Intra-Queue Preemption for a specific queue.... 39
Configure dynamic queue properties.... 40
Set Ordering policies within a specific queue.... 40
Configure queue ordering policies.... 41

Configure Partitions.... 41
Enabling node labels on a cluster to configure partition....43
Create partitions....44
Assign or unassign a node to a partition.... 45
View partitions....46
Associate partitions with queues....47
Disassociate partitions from queues....50
Delete partitions....51
Use partitions when submitting a job.... 52

Resource Scheduling and Management

You can manage resources for the applications running on your cluster by allocating resources through scheduling, limiting CPU usage by configuring cgroups, and partitioning the cluster into subclusters using partitions, and launching applications on Docker containers.

The *CapacityScheduler* is responsible for scheduling. The *CapacityScheduler* is used to run Hadoop applications as a shared, multi-tenant cluster in an operator-friendly manner while maximizing the throughput and the utilization of the cluster.

The *ResourceCalculator* is part of the YARN *CapacityScheduler*. If you have only one type of resource, typically a CPU virtual core (vcore), use the DefaultResourceCalculator. If you have multiple resource types, use the Dominant ResourceCalculator.

YARN resource allocation of multiple resource-types

You can manage your cluster capacity using the Capacity Scheduler in YARN. You can use the Capacity Scheduler's DefaultResourceCalculator or the DominantResourceCalculator to allocate available resources.

The fundamental unit of scheduling in YARN is the queue. The capacity of each queue specifies the percentage of cluster resources available for applications submitted to the queue. You can set up queues in a hierarchy that reflects the database structure, resource requirements, and access restrictions required by the organizations, groups, and individuals who use the cluster resources.

You can use the default resource calculator when you want the resource calculator to consider only the available memory for resource calculation. When you use the default resource calculator (DefaultResourceCalculator), resources are allocated based on the available memory.

If you have multiple resource types, use the dominant resource calculator (DominantResourceCalculator) to enable CPU and memory scheduling. The dominant resource calculator is based on the Dominant Resource Fairness (DRF) model of resource allocation. DRF is designed to fairly distribute CPU, and memory resources among different types of processes in a mixed-workload cluster.

For example, if User A runs CPU-heavy tasks and User B runs memory-heavy tasks, the DRF allocates more CPU and less memory to the tasks run by User A, and allocates less CPU and more memory to the tasks run by User B. In a single resource case, in which all jobs are requesting the same resources, the DRF reduces to max-min fairness for that resource.

Hierarchical queue characteristics

You must consider the various characteristics of the Capacity Scheduler hierarchical queues before setting them up.

There are two types of queues: parent queues and leaf queues.

- Parent queues enable the management of resources across organizations and sub- organizations. They can contain more parent queues or leaf queues. They do not themselves accept any application submissions directly.
- Leaf queues are the queues that live under a parent queue and accept applications. Leaf queues do not have any child queues, and therefore do not have any configuration property that ends with ".queues".
- There is a top-level parent root queue that does not belong to any organization, but instead represents the cluster itself.
- Using parent and leaf queues, administrators can specify capacity allocations for various organizations and sub-organizations.

Scheduling among queues

Hierarchical queues ensure that guaranteed resources are first shared among the sub-queues of an organization before any remaining free resources are shared with queues belonging to other organizations. This enables each organization to have control over the utilization of its guaranteed resources.

- At each level in the hierarchy, every parent queue keeps the list of its child queues in a sorted manner based on demand. The sorting of the queues is determined by the currently used fraction of each queue's capacity (or the full-path queue names if the reserved capacity of any two queues is equal).
- The root queue understands how the cluster capacity needs to be distributed among the first level of parent queues and invokes scheduling on each of its child queues.
- Every parent queue applies its capacity constraints to all of its child queues.
- Leaf queues hold the list of active applications (potentially from multiple users) and schedules resources in a FIFO (First In, First Out) manner, while at the same time adhering to capacity limits specified for individual users.

Application reservations

For a resource-intensive application, the Capacity Scheduler creates a reservation on a cluster node if the node's free capacity can meet the particular application's requirements. This ensures that the resources are utilized only by that particular application until the application reservation is fulfilled.

The Capacity Scheduler is responsible for matching free resources in the cluster with the resource requirements of an application. Many times, a scheduling cycle occurs such that even though there are free resources on a node, they are not sized large enough to satisfy the application waiting for a resource at the head of the queue. This typically happens with high-memory applications whose resource demand for Containers is much larger than the typical application running in the cluster. This mismatch can lead to starving these resource-intensive applications.

The Capacity Scheduler reservations feature addresses this issue as follows:

- When a node reports in with a finished Container, the Capacity Scheduler selects an appropriate queue to utilize the newly available resources based on capacity and maximum capacity settings.
- Within that selected queue, the Capacity Scheduler looks at the applications in a FIFO order along with the user limits. Once a needy application is found, the Capacity Scheduler tries to see if the requirements of that application can be met by the node's free capacity.
- If there is a size mismatch, the Capacity Scheduler immediately creates a reservation on the node for the application's required Container.
- Once a reservation is made for an application on a node, those resources are not used by the Capacity Scheduler for any other queue, application, or Container until the application reservation is fulfilled.
- The node on which a reservation is made reports back when enough Containers finish running such that the total free capacity on the node now matches the reservation size. When that happens, the Capacity Scheduler marks the reservation as fulfilled, removes it, and allocates a Container on the node.
- In some cases another node fulfills the resources required by the application, so the application no longer needs the reserved capacity on the first node. In this situation, the reservation is simply cancelled.

To prevent the number of reservations from growing in an unbounded manner, and to avoid any potential scheduling deadlocks, the Capacity Scheduler maintains only one active reservation at a time on each node.

Resource distribution workflow

During scheduling, queues at any level in the hierarchy are sorted in the order of their current used capacity, and the available resources are distributed among them starting with queues that are currently the most under-served.

With respect to capacities alone, the resource scheduling has the following workflow:

- The more under-served a queue is, the higher the priority it receives during resource allocation. The most under-served queue is the queue with the least ratio of used capacity as compared to the total cluster capacity.
  - The used capacity of any parent queue is defined as the aggregate sum of used capacity of all of its descendant queues, recursively.
  - The used capacity of a leaf queue is the amount of resources used by the allocated Containers of all of the applications running in that queue.
- Once it is decided to give a parent queue the currently available free resources, further scheduling is done recursively to determine which child queue gets to use the resources -- based on the previously described concept of used capacities.
- Further scheduling happens inside each leaf queue to allocate resources to applications in a FIFO order.
  - This is also dependent on locality, user level limits, and application limits.
  - Once an application within a leaf queue is chosen, scheduling also happens within the application. Applications may have different priorities of resource requests.
- To ensure elasticity, capacity that is configured but not utilized by any queue due to lack of demand is automatically assigned to the queues that are in need of resources.

Resource Distribution Process in Relative Mode - Example

To understand the resource distribution workflow, consider the example of a 100-node cluster, each with 10 GB of memory allocated for YARN containers, for a total cluster capacity of 1000 GB (1 TB).

For example, in the Relative allocation mode, the "engineering" organization is assigned 60% of the cluster capacity, that is, an absolute capacity of 600 GB. Similarly, the "support" organization is assigned 100 GB, and the "marketing" organization gets 300 GB.

Under the "engineering" organization, capacity is distributed between the "development" team and the "qa" team in a 1:4 ratio. So "development" gets 120 GB, and 480 GB is assigned to "qa".

Now consider the following timeline of events:

- Initially, the entire "engineering" queue is free with no applications running, while the "support" and "marketing" queues are utilizing their full capacities.
- Users Sid and Hitesh first submit applications to the "development" leaf queue. Their applications are elastic and can run with either all of the resources available in the cluster, or with a subset of cluster resources (depending upon the state of the resource-usage).
  - Even though the "development" queue is allocated 120 GB, Sid and Hitesh are each allowed to occupy 120 GB, for a total of 240 GB.
  - This can happen despite the fact that the "development" queue is configured to be run with a capacity of 120 GB. Capacity Scheduler allows elastic sharing of cluster resources for better utilization of available cluster resources. Since there are no other users in the "engineering" queue, Sid and Hitesh are allowed to use the available free resources.
- Next, users Jian, Zhijie and Xuan submit more applications to the "development" leaf queue. Even though each is restricted to 120 GB, the overall used capacity in the queue becomes 600 GB -- essentially taking over all of the resources allocated to the "qa" leaf queue.
- User Gupta now submits an application to the "qa" queue. With no free resources available in the cluster, the user's application must wait.
  - Given that the "development" queue is utilizing all of the available cluster resources, Gupta may or may not be able to immediately get back the guaranteed capacity of his "qa" queue -- depending upon whether or not preemption is enabled.
- As the applications of Sid, Hitesh, Jian, Zhijie, and Xuan finish running and resources become available, the newly available Containers will be allocated to Gupta’s application.

This will continue until the cluster stabilizes at the intended 1:4 resource usage ratio for the "development" and "qa" queues.

From this example, you can see that it is possible for abusive users to submit applications continuously, and thereby lock out other queues from resource allocation until Containers finish running or get preempted. To avoid this scenario, Capacity Scheduler supports limits on the elastic growth of any queue. For example, you can restrict the "development" queue from monopolizing the "engineering" queue capacity by setting the maximum capacity property.

To set the maximum capacity property based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the development queue and select Edit Child Queues option.
3. Enter 40 in the Configured Value field.
4. Click Save.

Once this is set, users of the "development" queue can still go beyond their capacity of 120 GB, but they will not be allocated any more than 40% of the "engineering" parent queue's capacity (that is, 40% of 600 GB = 240 GB).

The capacity and maximum-capacity properties can be used to control sharing and elasticity across the organizations and sub-organizations utilizing a YARN cluster. Administrators should balance these properties to avoid strict limits that result in a loss of utilization, and to avoid excessive cross-organization sharing.

Resource allocation overview

You can allocate resources to queues by using either the Absolute mode, in which the actual units of vCores and memory resources are allocated, or the Relative mode, in which resources are allocated as a percentage of total resources.

- Relative mode: In the YARN Queue Manager UI, the capacity value is shown as a percentage. When editing queue resources, capacities are specified in percentages. The capacity allocated to a set of sibling queues must equal 100% of the parent.
- Absolute mode: In the YARN Queue Manager UI, the capacity value is shown in units. When editing queue resources, a separate tab is displayed for each resource type. The total allocated capacity must be less than or equal to the capacity of the parent.

Note: YARN Queue Manager does not support mixed allocation of resources. That is, some queues allocated using percentages and some queues using absolute units.

Related Information

Change resource allocation mode

Add queues using YARN Queue Manager UI

Configure cluster capacity with queues

Use CPU scheduling

Cgroups with CPU scheduling helps you effectively manage mixed workloads.

Note: You should use CPU scheduling only in a Linux environment, because there is no isolation mechanism (cgroups equivalent) for Windows.

MapReduce jobs only

If you primarily run MapReduce jobs on your cluster, enabling CPU scheduling does not change performance much. The dominant resource for MapReduce is memory, so the DRF scheduler continues to balance MapReduce jobs in a manner similar to the default resource calculator. In the case of a single resource, the DRF reduces to max-min fairness for that resource.

Mixed workloads

An example of a mixed workload is a cluster that runs both MapReduce and Storm on YARN. MapReduce is not CPU-constrained, but Storm on YARN is; its containers require more CPU than memory. As you add Storm jobs along with MapReduce jobs, the DRF scheduler tries to balance memory and CPU resources, but you might see some performance degradation in as a result. As you add more CPU-intensive Storm jobs, individual jobs start to take longer to run as the cluster CPU resources are consumed.

To solve this problem, you can use cgroups along with CPU scheduling. Using cgroups provides isolation for CPU-intensive processes such as Storm on YARN, thereby enabling you to predictably plan and constrain the CPU-intensive Storm containers.

You can also use partitions in conjunction with CPU scheduling and cgroups to restrict Storm on YARN jobs to a subset of cluster nodes.

Configure CPU scheduling and isolation

You can configure CPU scheduling on your cluster to allocate the best possible nodes having the required CPU resources for application containers.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for Resource Calculator Class.
4. Select the org.apache.hadoop.yarn.util.resource.DominantResourceCalculator option.
5. Search for yarn.nodemanager.resource.cpu-vcores and set the number of vcores to match the number of physical CPU cores on the NodeManager host by providing the number of physical cores.

Related Information

Limit CPU usage with Cgroups

Using GPU on YARN

Enable Cgroups

Use CPU scheduling with distributed shell

You can run the distributed shell by specifying resources other than memory and vCores. The following is an example for distributed shell but you can use CPU scheduling with other frameworks as well.

Procedure

- Use the following command to allocate two cores for two containers:

```
yarn jar </opt/cloudera/parcels/<CDH-version>/lib/hadoop-yarn/hadoop-yarn-
applications-distributedshell.jar> \
                  -jar </opt/cloudera/parcels/<CDH-version>/lib/hadoop-ya
rn/hadoop-yarn-applications-distributedshell.jar> \
                  -shell_command "sleep 120" \
                  -container_resources memory-mb=3072,vcores=2 \
                  -num_containers 2
```

Use FPGA scheduling

You can use FPGA as a resource type.

A Field Programmable Gate Array (FPGA) is a card that contains a matrix of configurable logic blocks, mostly logic gates. FPGA can be reprogrammed, meaning that the interconnections among the gates and integrated devices can be re-wired. That is in contrast to a regular CPU, as the internal wiring and connections inside a CPU cannot be changed: only the software that runs on it can be changed, but the hardware is always the same. FPGA, on the other hand, allows changes on a circuit level.

An FPGA device does not contain only rewirable logic gates (like AND or OR gates), but also memory, DSP, chip, external connectors and more.

Programming FPGAs

FPGAs can be programmed in various ways. The most popular methods are VHDL and Verilog.

Intel FPGA cards allow the users to upload so-called OpenCL kernels to perform certain computations simultaneously and therefore rapidly. OpenCL is a framework and a set of standards that is currently developed and maintained by the Khronos Group. For more information, see Khronos' OpenCL documentation.

Configure FPGA scheduling and isolation

You can enable and configure FPGA as a resource type using Cloudera Manager.

Before you begin

Ensure that the FPGA Runtime or SDK is installed in such a way that the aocl command is executable by the NodeManager. For more information about how to install FPGA, see Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Select FPGA Management category under Filters.
4. Find the Enable FPGA Usage property and select the NodeManagers that can provide FPGA devices to YARN applications that request them.
5. Use the Path to FPGA (aocl) tool property to specify the full local path to the Intel aocl tool installed on the applicable nodes.
6. Use the Allowed FPGA devices to specify the FPGA devices which can be managed by YARN NodeManager.

   Valid values are the following:

   - Auto: Default value. All available FPGA devices are allowed.
   - Comma separated list: List the minor number of the allowed FPGAs. For example: 0,1
7. Use the List of available FPGA devices property to manually specify the available FPGA devices.

   Provide the value in the following format: deviceA/major_number:minor_number,deviceB/major_number:minor_number

   For example: acl0/238:0,acl1/238:1

   The acl numbers are displayed using the aocl diagnose command.

   The major and minor device numbers can usually be determined by listing the device files under /dev. Every card has a corresponding device file. The ls -l command shows both numbers.
8. Use the FPGA device major number property to provide the major device number of the FPGA card.

   This property is used in the container-executor.cfg file.

   Important: Only one major number can be specified for each node, meaning that a node can have only one kind of FPGA card.

9. Use the FPGA initializer script property to provide the path to the shell script that sets up the environment for the FPGA device.

   The initializer script sets up various environment variables, and sources other scripts allowing the aocl tool to run properly. The contents of this script depends heavily on the installation. For example, in case of Intel Processing Accelerator Card (PAC), it is necessary to source init_env.sh and init_opencl.sh.

   The following environment variables are possibly required to be exported:

   - AOCL_BOARD_PACKAGE_ROOT
   - CL_CONTEXT_COMPILER_MODE_INTELFPGA

   Please consult the vendor's documentation for further details.
10. Click Save Changes.
11. Restart the affected NodeManagers.

Results

FPGA support is enabled and configured.

What to do next

Request FPGA resource by specifying yarn.io/fpga as a resource.

Use FPGA with distributed shell

Once FPGA support is enabled, an FPGA resource can be requested by specifying yarn.io/fpga as a resource.

The following is a distributed shell example:

```
yarn jar /path/to/hadoop-yarn-applications-distributedshell.jar
-jar /path/to/hadoop-yarn-applications-distributedshell.jar
-shell_command "date"
-container_resources memory-mb=2048,vcores=1,yarn.io/fpga=1
-num_containers 1
```

This command runs the date command on a node which has an FPGA card installed.

Limit CPU usage with Cgroups

You can use cgroups to limit CPU usage in a Hadoop Cluster.

You can use cgroups to isolate CPU-heavy processes in a Hadoop cluster. If you are using CPU Scheduling, you should also use cgroups to constrain and manage CPU processes. If you are not using CPU Scheduling, do not enable cgroups.

When you enable CPU Scheduling, queues are still used to allocate cluster resources, but both CPU and memory are taken into consideration using a scheduler that utilizes Dominant Resource Fairness (DRF). In the DRF model, resource allocation takes into account the dominant resource required by a process. CPU-heavy processes (such as Storm-on-YARN) receive more CPU and less memory. Memory-heavy processes (such as MapReduce) receive more memory and less CPU. The DRF scheduler is designed to fairly distribute memory and CPU resources among different types of processes in a mixed- workload cluster.

Cgroups compliments CPU Scheduling by providing CPU resource isolation. It enables you to set limits on the amount of CPU resources granted to individual YARN containers, and also lets you set a limit on the total amount of CPU resources used by YARN processes.

Cgroups represents one aspect of YARN resource management capabilities that includes CPU Scheduling, partitions, archival storage, and memory as storage. If CPU Scheduling is used, cgroups should be used along with it to constrain and manage CPU processes.

Related Information

Configure CPU scheduling and isolation

Use Cgroups

You can use strict cgroups CPU limits to constrain CPU processes in mixed workload clusters.

One example of a mixed workload is a cluster that runs both MapReduce and Storm-on-YARN. MapReduce is not CPU-constrained (MapReduce containers do not ask for much CPU). Storm-on-YARN is CPU-constrained: its containers ask for more CPU than memory. As you start adding Storm jobs along with MapReduce jobs, the DRF scheduler attempts to balance memory and CPU resources, but as more CPU-intensive Storm jobs are added, they may begin to take up the majority of the cluster CPU resources.

You can use cgroups along with CPU scheduling to help manage mixed workloads. cgroups provide isolation for CPU-heavy processes such as Storm-on-YARN, thereby enabling you to predictably plan and constrain the CPU-intensive Storm containers.

When you enable strict cgroup CPU limits, each resource gets only what it asks for, even if there is extra CPU available. This is useful for scenarios involving charge-backs or strict SLA enforcement, where you always need to know exactly what percentage or CPU is being used.

Also, enabling strict CPU limits would make job performance predictable, whereas without setting strict limits a CPU-intensive job would run faster when the cluster was not under heavy use, but slower when more jobs were running in the cluster. Strict CPU limits would therefore also be useful for benchmarking.

You can also use partitions in conjunction with cgroups and CPU scheduling to restrict Storm-on-YARN jobs to a subset of cluster nodes.

If you are using cgroups and want more information on CPU performance, you can review the statistics available in the /cgroup/cpu/yarn/cpu.stat file.

Enable Cgroups

You can enable CPU Scheduling to enable cgroups. You must configure certain properties in yarn-site.xml on the ResourceManager and NodeManager hosts to enable cgroups.

About this task

cgroups is a Linux kernel feature. cgroups is supported on the following Linux operating systems:

- CentOS 6.9, 7.3
- RHEL 6.9, 7.3
- SUSE 12
- Ubuntu 16

At this time there is no cgroups equivalent for Windows. cgroups are not enabled by default on CDP. cgroups require that the CDP cluster be Kerberos enabled.

Important:

The yarn.nodemanager.linux-container-executor.cgroups.mount property must be set to false. Setting this value to true is not currently supported.

Procedure

Enable cgroups

The following commands must be run on every reboot of the NodeManager hosts to set up the cgroup hierarchy. Note that operating systems use different mount points for the cgroup interface. Replace /sys/fs/cgroup with your operating system equivalent.

```
mkdir -p /sys/fs/cgroup/cpu/yarn
chown -R yarn /sys/fs/cgroup/cpu/yarn
```

```
mkdir -p /sys/fs/cgroup/memory/yarn
chown -R yarn /sys/fs/cgroup/memory/yarn
mkdir -p /sys/fs/cgroup/blkio/yarn
chown -R yarn /sys/fs/cgroup/blkio/yarn
mkdir -p /sys/fs/cgroup/net_cls/yarn
chown -R yarn /sys/fs/cgroup/net_cls/yarn
mkdir -p /sys/fs/cgroup/devices/yarn
chown -R yarn /sys/fs/cgroup/devices/yarn
```

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for Always Use Linux Container Executor and select YARN-1 (Service-Wide) option.
4. Search for NodeManager Advanced Configuration and set the below property in the NodeManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml field.

   ```
   Name:  yarn.nodemanager.linux-container-executor.group
   Value: hadoop
   ```

5. Search for CGroups and select YARN-1 (Service-Wide) option in the Use CGroups for Resource Management field.
6. Search for CGroups Hierarchy and set the NodeManager Default Group value to /hadoop-yarn.
7. Search for NodeManager Advanced Configuration and set the below property in the NodeManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml field.

   ```
   Name:  yarn.nodemanager.linux-container-executor.cgroups.mount
   Value: false
   ```

   ```
   Name:  yarn.nodemanager.linux-container-executor.cgroups.mount-path
   Value: /sys/fs/cgroup
   ```

8. (Optional) Set the percentage of CPU to be used by YARN. Search for Containers CPU Limit and set the value in the Containers CPU Limit Percentage field.

   Set the percentage of CPU that can be allocated for YARN containers. In most cases, the default value of 100% should be used. If you have another process that needs to run on a node that also requires CPU resources, you can lower the percentage of CPU allocated to YARN to free up resources for the other process.
9. (Optional) Set flexible or strict CPU limits. Search for Strict CGroup Resource Usage and select the NodeManager Default Group field.

   CPU jobs are constrained with CPU scheduling and cgroups enabled, but by default these are flexible limits. If spare CPU cycles are available, containers are allowed to exceed the CPU limits set for them. With flexible limits,

the amount of CPU resources available for containers to use can vary based on cluster usage -- the amount of CPU available in the cluster at any given time.

You can use cgroups to set strict limits on CPU usage. When strict limits are enabled, each process receives only the amount of CPU resources it requests. With strict limits, a CPU process will receive the same amount of cluster resources every time it runs.

Strict limits are not enabled (set to false) by default.

Note: Irrespective of whether this property is true or false, at no point will total container CPU usage exceed the limit set in yarn.nodemanager.resource.percentage-physical-cpu-limit.

Note: CPU resource isolation leverages advanced features in the Linux kernel. At this time, setting yarn.nodemanager.linux-container-executor.cgroups.strict-resource-usage to true is not recommended due to known kernel panics. In addition, with some kernels, setting yarn.nodemanager.resource.percentage-physical-cpu-limit to a value less than 100 can result in kernel panics. If you require either of these features, you must perform scale testing to determine if the in-use kernel and workloads are stable. As a starting point, Linux kernel version 4.8.1 works with these features. However, testing the features with the desired workloads is very important.

Related Information

Configure CPU scheduling and isolation

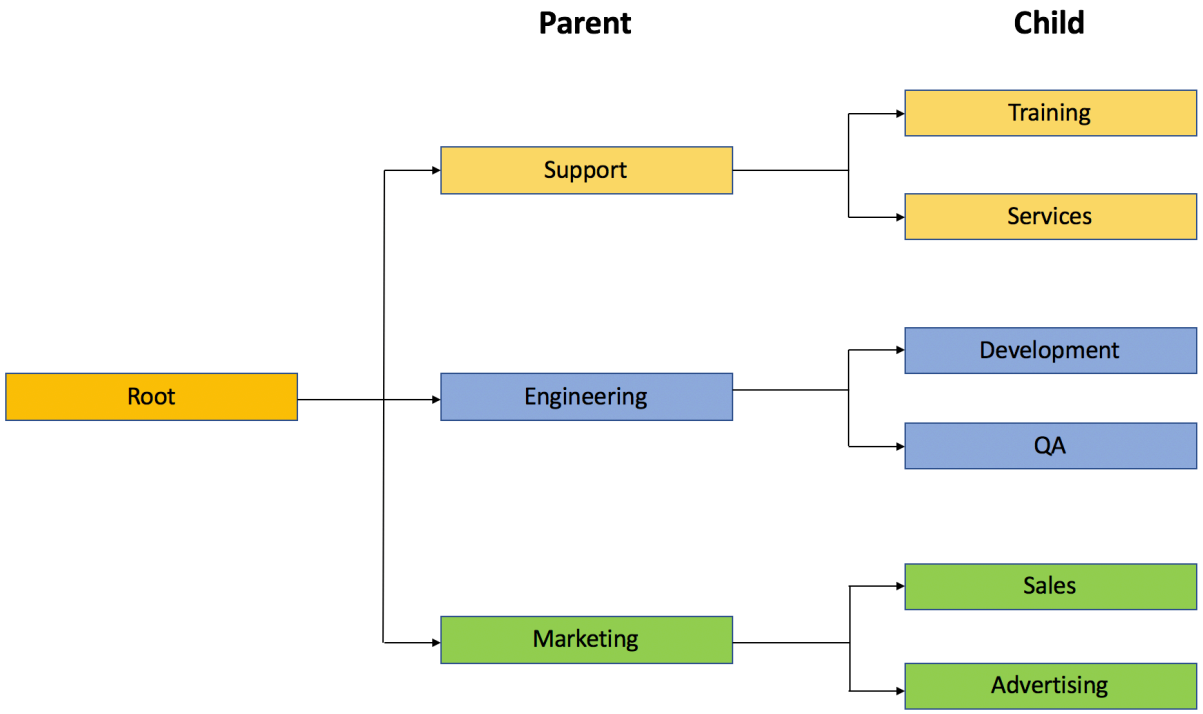
Manage Queues

YARN Queue Manager is the queue management graphical user interface for Apache Hadoop YARN Capacity Scheduler. You can use YARN Queue Manager UI to manage your cluster capacity using queues to balance resource requirements of multiple applications from various users. Using YARN Queue Manager UI, you can set scheduler level properties and queue level properties.

You can view, sort, search, and filter queues using YARN Queue Manager UI. Queue Manager stores history of previous changes and provides the ability to view the changes of each version in the Overview and Scheduler Configuration tabs. The previous versions will be in the read-only mode and you must select the latest version to make changes.

The fundamental unit of scheduling in YARN is a queue. The capacity of each queue specifies the percentage of cluster resources that are available for applications submitted to the queue. Capacity Scheduler queues can be set up in a hierarchy that reflects the database structure, resource requirements, and access restrictions required by the various organizations, groups, and users that utilize cluster resources.

For example, suppose that a company has three organizations: Engineering, Support, and Marketing. The Engineering organization has two sub-teams: Development and QA. The Support organization has two sub-teams: Training and Services. And finally, the Marketing organization is divided into Sales and Advertising. The following image shows the queue hierarchy for this example:



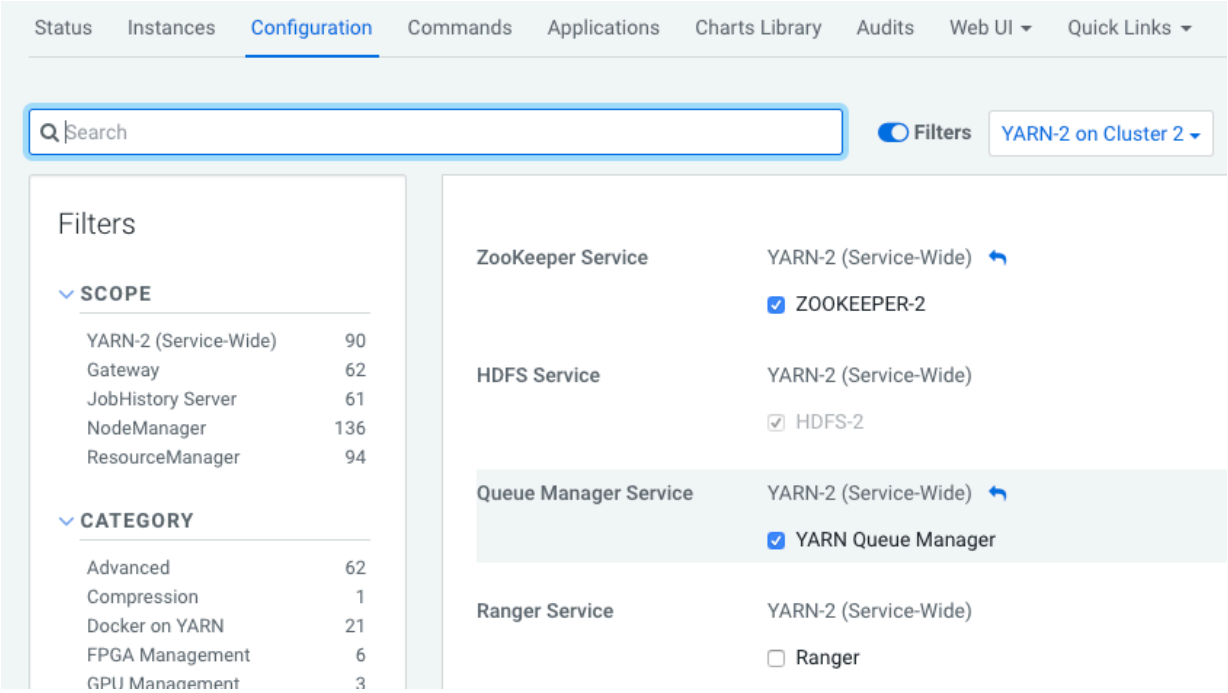
Each child queue is tied to its parent queue and the top-level "support", "engineering", and "marketing" queues would be tied to the "root" queue.

Prerequisite

If you add the YARN Queue Manager service to the cluster after installing the cluster, you must configure the YARN Queue Manager dependency in the Yarn Configuration tab.

1. In Cloudera Manager, click the Cluster > YARN service.
2. Click the Configuration tab.
3. Search for Queue Manager service.

4. Select YARN Queue Manager checkbox.



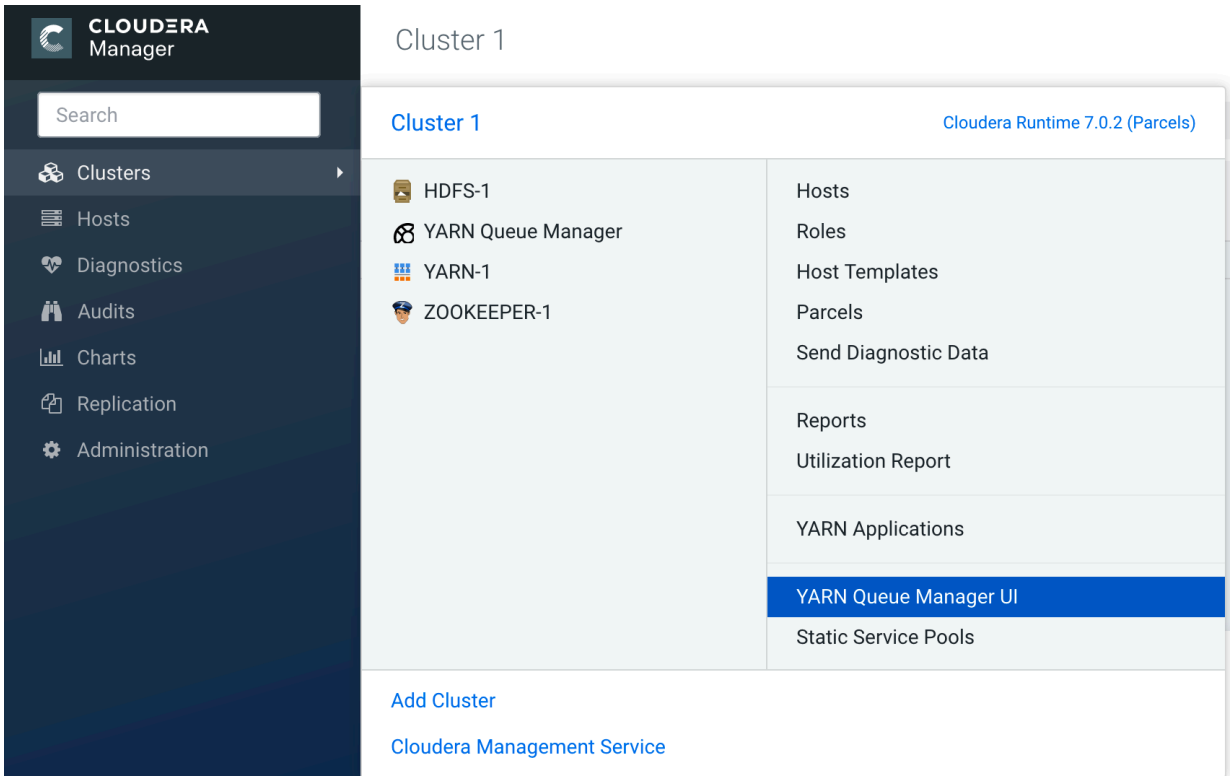
5. Click Save Changes.
6. Restart YARN and YARN Queue Manager services.

Add queues using YARN Queue Manager UI

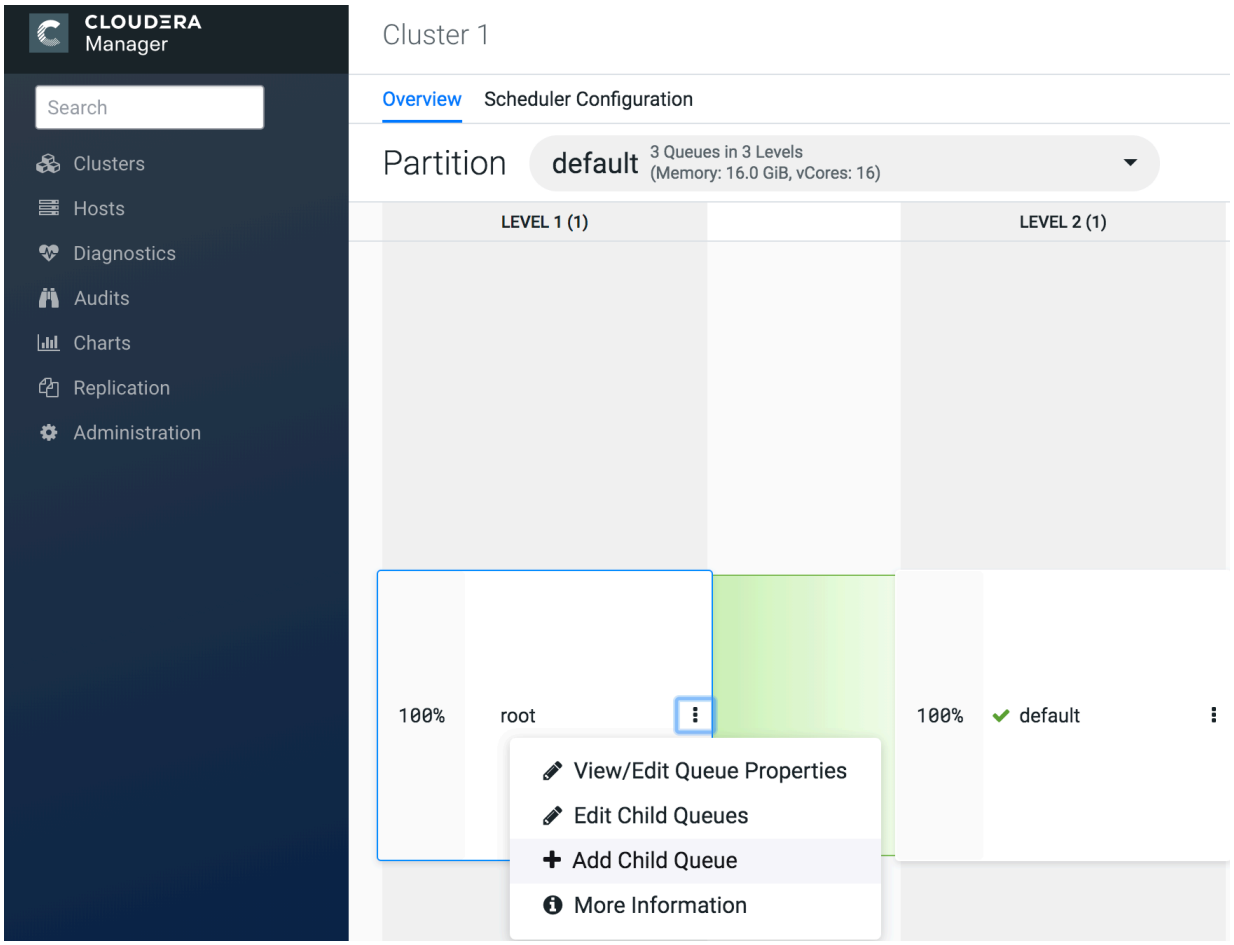
You can add queues to the predefined queue called root from the Yarn Queue Manager UI. The Capacity Scheduler has a predefined queue called root. All queues in the system are children of the root queue. Each child queue is tied to its parent queue, but children queues do not inherit properties directly from the parent queue unless otherwise specified.

Procedure

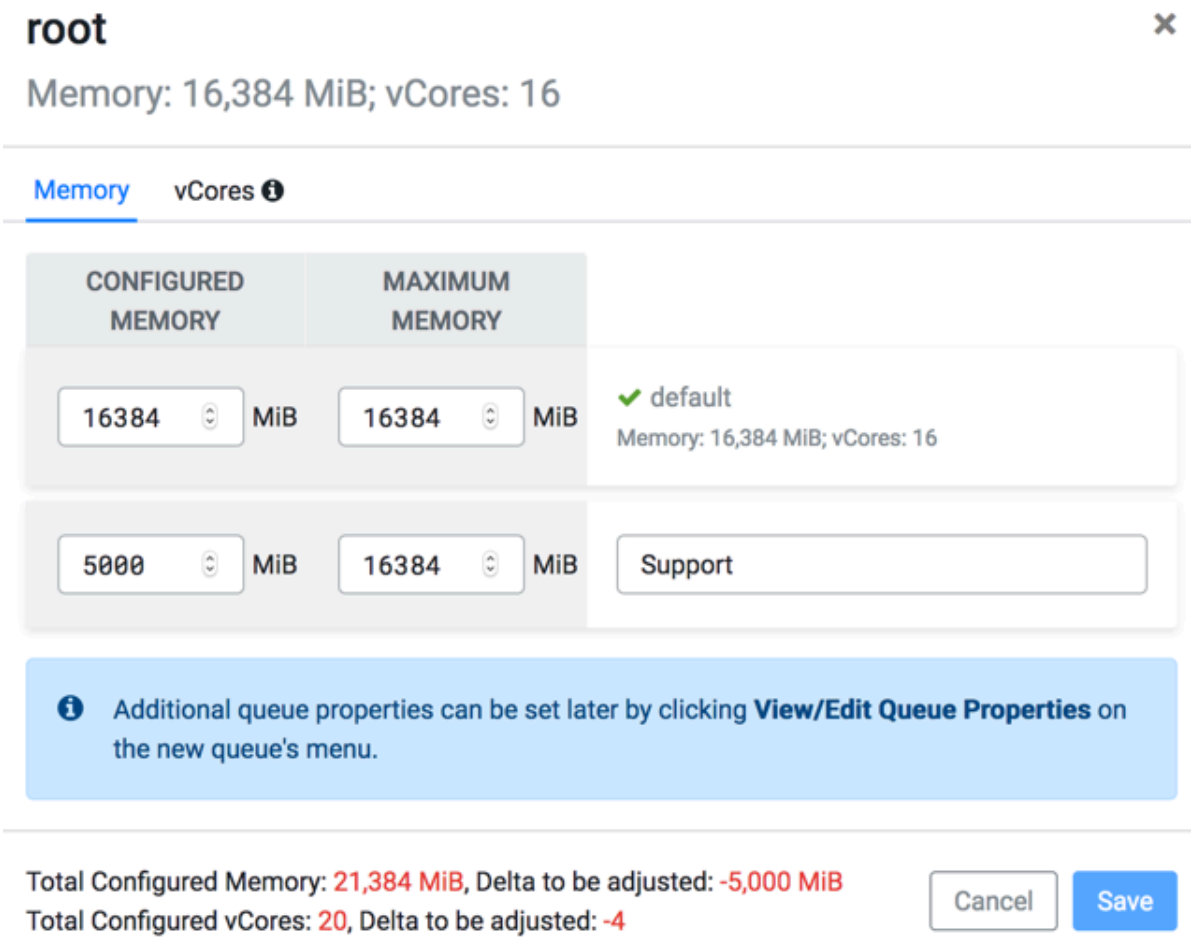
1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service.



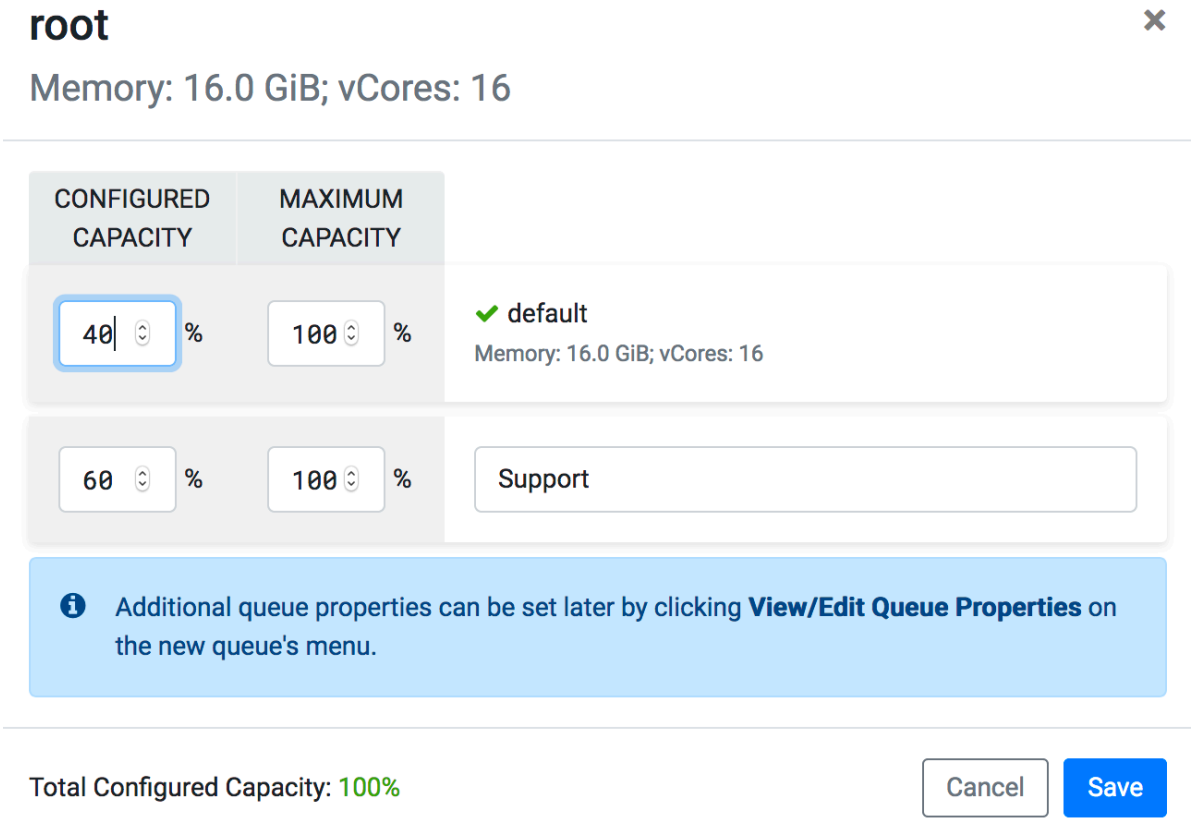
2. Click on the three vertical dots on the root and select the Add Child Queue option.



3. Enter the information based on the Relative or Absolute allocation mode.
   - Absolute allocation mode: Enter the name of the queue and units of memory in MiB in the Memory tab. Enter the number of cores in the vCores tab.



   Note: When decreasing a queue's configured resource capacity to allocate resources for the new sibling queue you must update resource capacities starting from the lowest queue level. This is because a parent queue's capacity cannot be decreased until the total resource capacity of all of its child queues is reduced. You might have to change both the minimum and maximum values.
   - Relative allocation mode: Enter the name of the queue, Configured Capacity, and Maximum Capacity values for the queue.



4. Click Save.

   You can continue to add more parent and child queues by following the same steps.

Related Information

Resource allocation overview

Change resource allocation mode

Configure cluster capacity with queues

You can manage your cluster capacity using queues to balance resource requirements of multiple applications from various users.

You can use the Capacity Scheduler to share cluster resources using FIFO (first in, first out) queues.

You can configure queues either to own a fraction of the capacity using the Relative mode, or the actual units of vCores and memory using the Absolute mode. The Relative allocation mode is the default. You can change the allocation mode to the Absolute mode.

You can submit applications to different queues at multiple levels in the queue hierarchy if the capacity is available on the nodes in the cluster. Because total cluster capacity can vary, capacity configuration values are expressed using percentages or units.

Example – Configuring capacity using the Relative mode

You can specify the capacity property to allocate a floating-point percentage value of cluster capacity to a queue. The following configurations divide the cluster resources between the "engineering", "support", and "marketing" organizations in a 6:1:3 ratio (60%, 10%, and 30%).

To specify the capacity property based on the above example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the root and select Edit Child Queues option.
3. Enter the Configured Capacity of "engineering" to 60, "support" to 10, and "marketing" to 30.
4. Click Save.

Example – Configuring capacity using the Absolute mode

You can specify the capacity in units of memory in MiB and the number of cores to a queue. If the total units of memory is 16384 MiB and 16 cores, and the the cluster resources between the "engineering", "support", and "marketing" organizations are allocated as follows:

Organization	Memory	vCores
engineering	9830	10
support	1638	2
marketing	4916	4

To specify the capacity property based on the above example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the root and select Edit Child Queues option.
3. Enter the Configured Memory of "engineering" to 9830, "support" to 1638, and "marketing" to 4916.
4. Enter the Configured vCores of "engineering" to 10, "support" to 2, and "marketing" to 4.
5. Click Save.

Related Information

Resource allocation overview

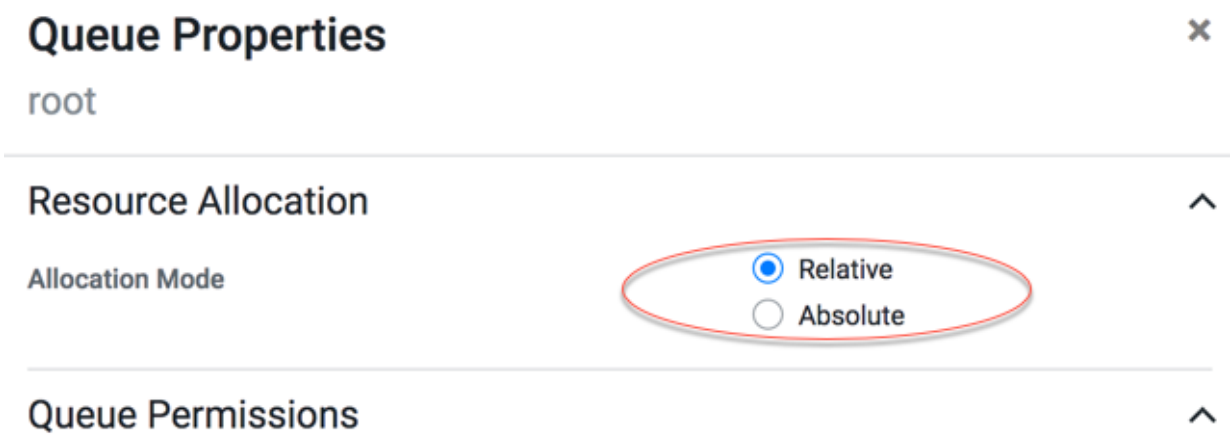
Change resource allocation mode

You can change the resource allocation mode from the root queue by editing the queue properties from the Yarn Queue Manager UI. The Relative resource allocation mode is the default allocation mode. You can specify the actual units of vCores and memory resources by using the Absolute allocation mode or specify a percentage of total resources using the Relative allocation mode.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the root and select View/Edit Queue Properties option.

3. In the Queue Properties dialog-box, select the resource allocation mode.



4. Click Save.
5. Enter yes in the Switch Allocation Mode dialog box and click OK. Queue Manager calculates and updates the resource allocations for all the existing queues. If required, you can further modify the resource allocation.
6. Click Save.

Related Information

Resource allocation overview

Add queues using YARN Queue Manager UI

Start and stop queues

Queues in YARN can be in two states: RUNNING or STOPPED. A RUNNING state indicates that a queue can accept application submissions, and a STOPPED queue does not accept application submissions. The default state of any configured queue is RUNNING.

In Capacity Scheduler, parent queues and leaf queues can be stopped. For an application to be accepted at any leaf queue, all the queues in the hierarchy all the way up to the root queue must be running. This means that if a parent queue is stopped, all of the descendant queues in that hierarchy are inactive, even if their own state is RUNNING.

To stop a queue:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select Stop Queue .
3. You will be prompted for a confirmation. Click OK to stop the queue.

To start a queue:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select Start Queue .
3. You will be prompted for a confirmation. Click OK to start the queue.

Administrators can use the ability to stop and drain applications in a queue for a number of reasons, such as when decommissioning a queue and migrating its users to other queues. Administrators can stop queues at run-time, so that while current applications run to completion, no new applications are accepted. Existing applications can continue until they finish running, and thus the queue can be drained gracefully without any end-user impact.

Delete queues

You must first stop the queue before deleting the queue. You can delete a single queue and also parent queue and its children if all the queues in the hierarchy are stopped.

In Capacity Scheduler, parent queues, child queues, and the root queue can all be stopped. For an application to be accepted at any child queue, all the queues in the hierarchy all the way up to the root queue must be running. This means that if a parent queue is stopped, all of the descendant queues in that hierarchy are inactive, even if their own state is RUNNING.

Note: If the queue is associated with one or more partitions, you must first set the partition capacity to zero using Edit Child Queue for that queue for all the partitions before you delete the queue.

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select Delete Queue .

   You can use the Delete Queues and its Children option to delete both parent queue and its children queues.
3. You will be prompted for a confirmation. Click OK to stop the queue.

Note: Queue associated with a placement rule cannot be deleted until its associated placement rule is deleted. For more information about deleting placement rules, see Delete placement rules on page 31.

Configure Scheduler Properties at the Global Level

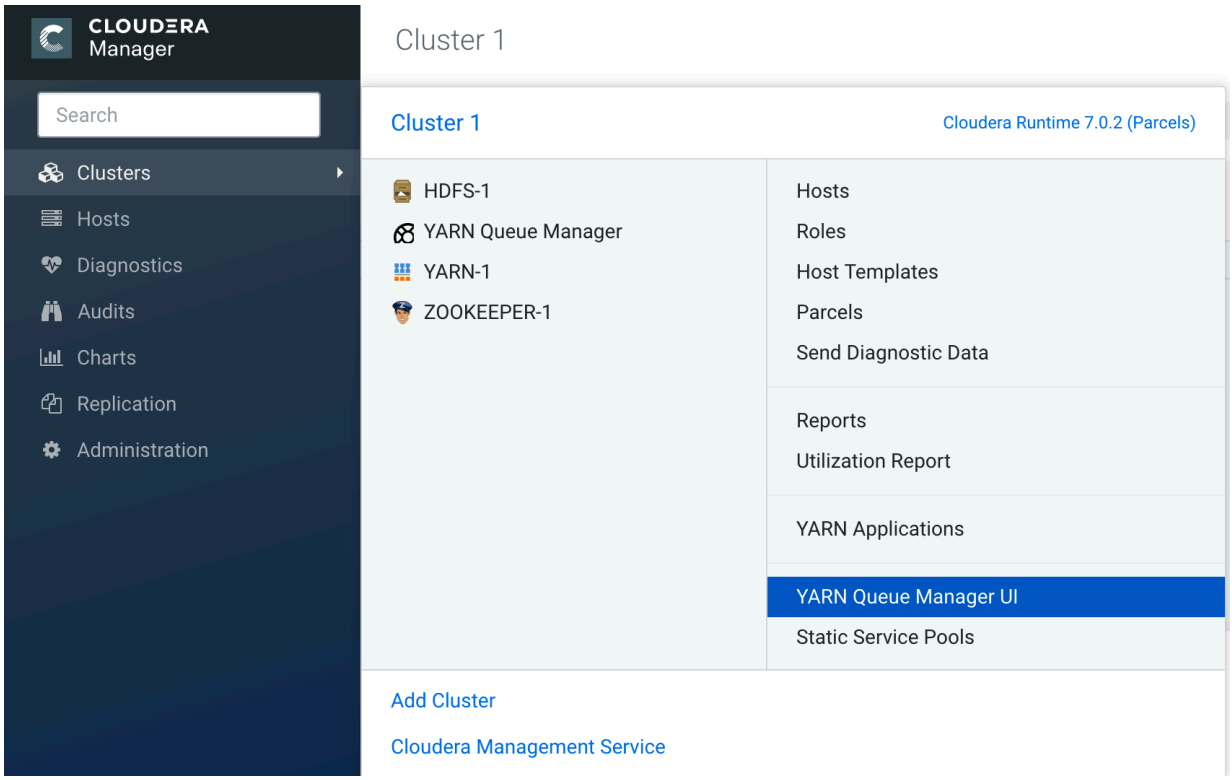
You can configure the scheduler properties to define the behaviour of all the queues. All parent and children queues inherit the properties set using the Scheduler Properties.

About this task

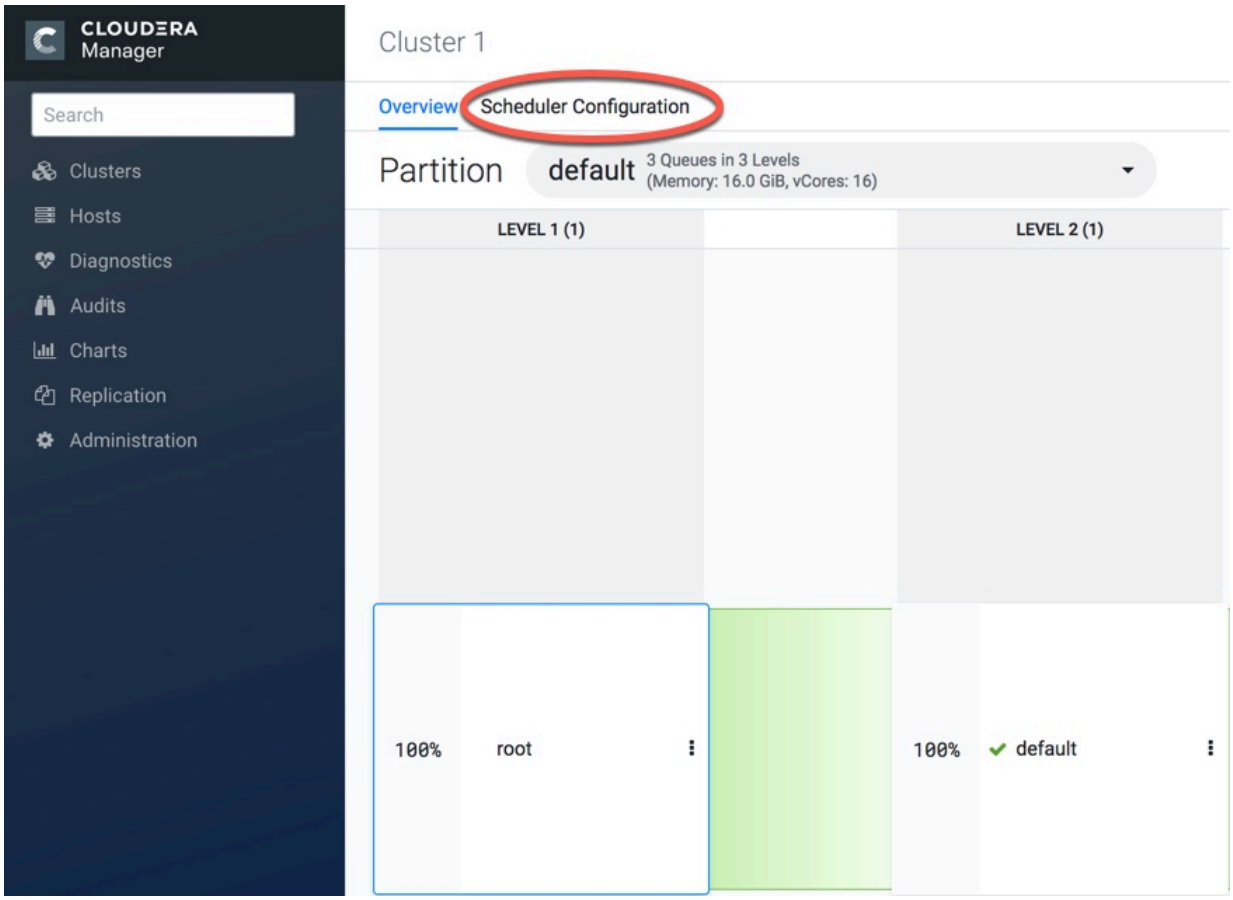
In Cloudera Manager, you can use the Scheduler Configuration tab to configure the scheduler properties.

Procedure

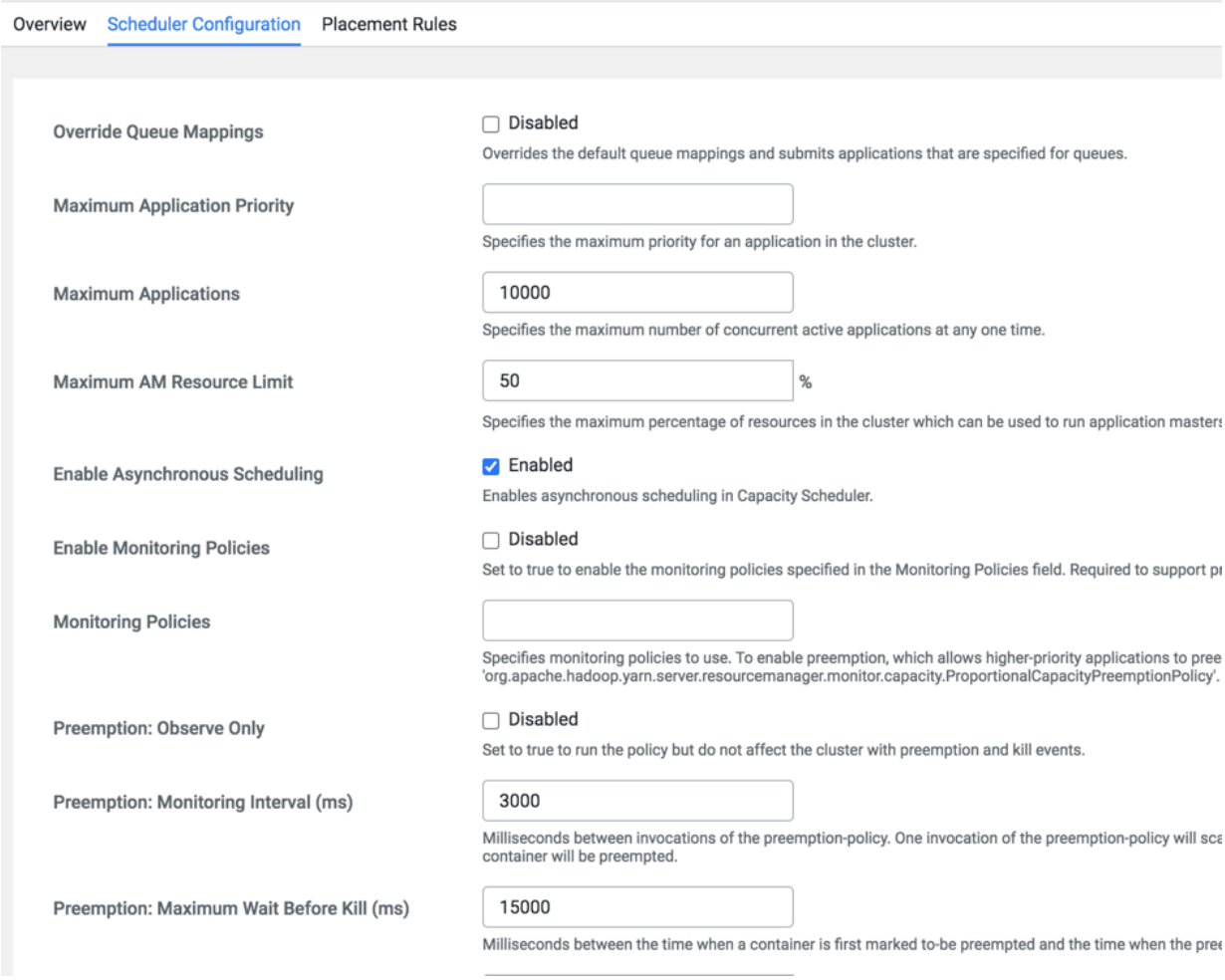
1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service.



2. In the YARN Queue Manager window, click on the Scheduler Configuration tab.



3. In the Scheduler Configuration window, enter the value of the property and click Save.



Set global maximum application priority

You can use Priority Scheduling to run YARN applications at higher priority, regardless of other applications that are already running in the cluster. YARN allocates more resources to applications running at a higher priority over those running at a lower priority. Priority Scheduling enables you to set an application's priority both at the time of submission and dynamically at run time.

About this task

Priority Scheduling works only with the FIFO (First In, First Out) ordering policy. FIFO is the default Capacity Scheduler ordering policy.

To set appliation priority (yarn.cluster.max-application-priority), perform the following:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the Scheduler Configuration tab.
3. Enter the priority in the Maximum Application Priority text box.
4. Click Save.

Configure preemption

Preemption allows applications of higher priority to preempt applications of lower priority.

About this task

A scenario can occur in which a queue has a guaranteed level of cluster resources, but must wait to run applications because other queues are utilizing all of the available resources. If preemption is enabled, applications of higher priority do not have to wait because applications of lower priority have taken up the available capacity. When you enable Preemption (yarn.resourcemanager.scheduler.monitor.enable) underserved queues can begin to claim their allocated cluster resources almost immediately, without having to wait for other queues' applications to finish running.

You can use Priority Scheduling to run YARN applications at a higher priority, regardless of other applications that are already running in the cluster . For more information, see Set global maximum application priority on page 26.

To configure the Queue Preemption options on all queues, perform the following:

Procedure

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Scheduler Configuration tab.
3. Select the Enable Monitoring Policies checkbox.
4. Specify org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCapacityPreemptionPolicy as the monitoring policy to be used in the Monitoring Policies textbox.
5. Configure the required preemption properties:
   - Preemption: Observe Only - Select the checkbox to run the policy, but not affect the cluster with preemption and kill events.
   - Preemption: Monitoring Interval (ms) - The time in milliseconds between invocations of this policy. Setting this value to a longer time interval causes the Capacity Monitor to run less frequently.
   - Preemption: Maximum Wait Before Kill (ms) - The time in milliseconds between requesting a preemption from an application and killing the container. Setting this to a higher value gives applications more time to respond to preemption requests and gracefully release containers.
   - Preemption: Total Resources Per Round - The maximum percentage of resources preempted in a single round. You can use this value to restrict the pace at which containers are reclaimed from the cluster. After computing the total desired preemption, the policy scales it back to this limit.
   - Preemption: Over Capacity Tolerance - The maximum amount of resources above the target capacity ignored for preemption. The default value is 0.1, which means that the Resource Manager starts preemption for a queue only when it goes 10% above its guaranteed capacity. This avoids resource-rotation and aggressive preemption.
   - Preemption: Maximum Termination Factor - The maximum percentage of each queue's preemption target capacity that is preempted per cycle. You can increase this value to speed up resource reclamation.
6. Click Save.

   For information about configuring preemption for a specific queue, see Configure preemption for a specific queue.

Enable Intra-Queue preemption

Intra-queue preemption prevents resource imbalances in a queue.

About this task

Intra-queue preemption helps in effective distribution of resources within a queue based on configured user limits or application priorities.

To configure Intra-Queue Preemption (yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled) on all queues, perform the following:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the Scheduler Configuration tab.
3. Select the Enable Intra Queue Preemption checkbox.
4. Click Save.

   For information about configuring intra-queue preemption for a specific queue, see Configure Intra-Queue Preemption for a specific queue.

Set global application limits

To avoid system-thrash due to an unmanageable load -- caused either by malicious users, or by accident -- the Capacity Scheduler enables you to place a static, configurable limit on the total number of concurrently active (both running and pending) applications at any one time.

You can set the maximum applications limit (yarn.scheduler.capacity.maximum-applications) using this configuration property. The default value is 10,000.

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the Scheduler Configuration tab.
3. Enter the maximum application limit in the Maximum Applications text box.
4. Click Save.

For information about overriding this limit on a per-queue basis, see Set Application limit for a specific queue.

Set default Application Master resource limit

The Application Master (AM) resource limit that can be used to set a maximum percentage of cluster resources allocated specifically to Application Masters. This property has a default value of 10%, and exists to avoid cross-application deadlocks where significant resources in the cluster are occupied entirely by the Containers running ApplicationMasters.

About this task

The Maximum AM Resource Limit (yarn.scheduler.capacity.maximum-am-resource-percent) property also indirectly controls the number of concurrent running applications in the cluster, with each queue limited to a number of running applications proportional to its capacity.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the Scheduler Configuration tab.
3. Enter the maximum AM resource limit in the Maximum AM Resource Limit text box.

4. Click Save.

   For information about overriding this limit on a per-queue basis, see Configure Application Master resource limit for a specific queue.

Enable asynchronous scheduler

Asynchronous scheduler decouples the CapacityScheduler scheduling from Node Heartbeats. This improves the latency significantly.

About this task

To enables asynchronous scheduling (yarn.scheduler.capacity.schedule-asynchronously.enable), perform the following:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the Scheduler Configuration tab.
3. Select the Enable Asynchronous Scheduler check-box.
4. Click Save.

Configure placement rules

Placement Rules allow you to specify a set of rules for assigning jobs and applications to queues. You can define placement rules to dynamically create queues based on the rules and the applications would be submitted to those queues. These predefined rules enable you to submit jobs without specifying the queue name at the time of job submission.

Types of placement rules:

- Static Rules: These rules use static queue mapping expressions for which you must create target queues and specify those queues based on the mapping expressions. You must provide input for the parameters such as userName, queueName and groupName based on the mapping expression.
- Dynamic Rules: These rules allow the creation of queues dynamically based on predefined expressions. You must create a leaf queue first and then provide that queueName, which will be converted to a managed parent queue while executing the rule.

If no rule is satisfied when the application or job runs, the YARN application is rejected.

Dynamic queues

Based on the predefined expressions of the dynamic placement rule, dynamic queues are created automatically during runtime .

YARN supports only one level of dynamic leaf queues. You can view the queue properties of the dynamically created leaf queues in the *Dynamic Auto-Creation of Queue* section of the Queue Properties. You can configure the dynamic leaf-queue properties by clicking on the Edit Child Queues option of its managed parent queue. The queue properties set at the managed parent queue level is applied to all of its leaf queues. For more information about setting properties for dynamic leaf queues, see Configure dynamic queue properties on page 40.

A leaf icon is displayed next to the queue name for dynamically created leaf-queues.

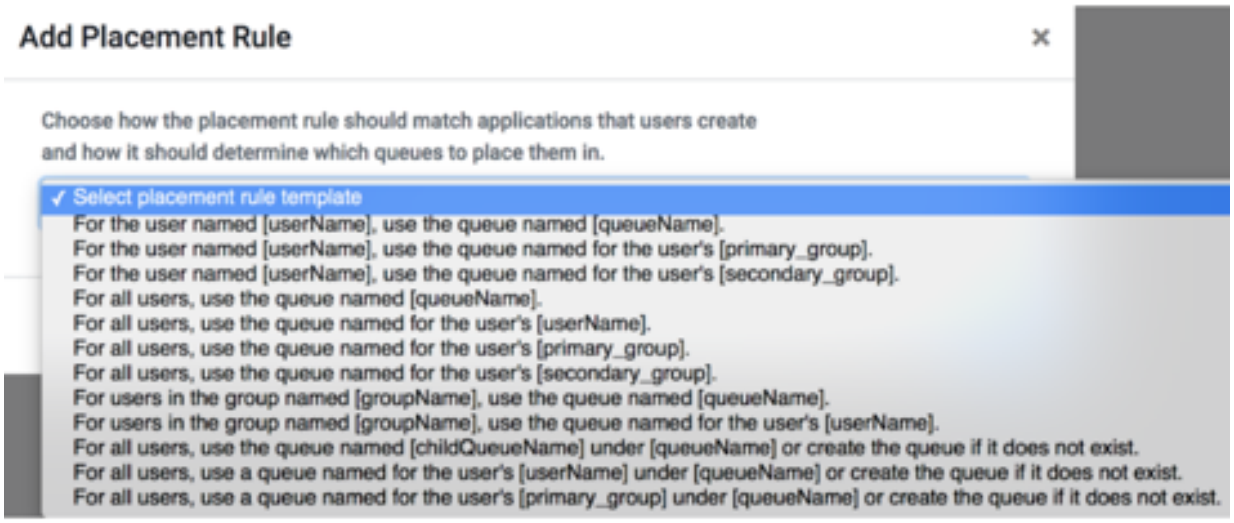
Note: The managed parent queue does not support other pre-configured queues to co-exist along with dynamically-created queues.

Create placement rules

Placement rules determine the queues to which applications and jobs are assigned.

You must first create the target leaf queues before creating the placement rules. For static rules, the UI will display all the existing leaf queues. Hence, you must create leaf queues before creating rules. In a dynamic placement rule, once the leaf-queue is converted to a managed parent queue, you cannot add new child queues to it or edit the child queue capacity.

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Placement Rules tab.
3. Click +Add. The Add Placement Rule dialog box is displayed.
4. Select a placement rule template from the drop-down list.



5. Add the rule to match applications with the queues using the Add Placement Rule dialog box.

   The static rules use the existing queues and dynamic rules create a queue if it does not exist. The dynamic rule templates end with the phrase or create the queue if it does not exist.
6. Click Save.

   The rule is added to the bottom of the placement rule list and becomes the last rule to be evaluated.

Reorder placement rules

Placement rules are evaluated in the order in which they appear in the placement rule list. When a job is submitted, the rules are evaluated, and the first matching rule is used to determine the queue in which the job is run.

If a rule is always satisfied, subsequent rules are not evaluated. By default, placement rules are ordered in the order they were added; the first rule added appears first. You can easily reorder rules.

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Placement Rules tab. A list of placement rules is displayed.
3. Click Reorder.
4. Click Move Up or Move Down arrow buttons in a rule row.
5. Click Save Reorder.

Edit placement rules

You can modify the existing placement rules.

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Placement Rules tab. A list of placement rules is displayed.
3. Click the Edit Rule at the right of a rule.
4. Edit the rule in the Edit Placement Rule dialog box.
   - For static rules, edit the target queue name associated with the rule queue mapping expression.
   - For dynamic rules, edit the configured capacity and maximum capacity for dynamically created leaf-queues.
5. Click Save.

Delete placement rules

You can delete the placement rules from the UI. You can delete the managed parent queue created using the dynamic rule after deleting that placement rule.

1. In Cloudera Manager, select theClusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Placement Rules tab. A list of placement rules is displayed.
3. Click Delete Rule.
4. Click Save.

Note: Queue associated with a placement rule cannot be deleted until its associated placement rule is deleted.

Configure queue mapping to use the user name from the application tag using Cloudera Manager

You can configure queue mapping to use the user name from the application tag instead of the proxy user who submitted the job. You can add only service users like *hive* using the yarn.resourcemanager.application-tag-based-placement.username.whitelist property and not normal users.

When a user runs Hive queries, HiveServer2 submits the query in the queue mapped from an end user instead of a hive user. For example, when user *alice* submits a Hive query with doAs=false mode, job will run in YARN as hive user. If application-tag based scheduling is enabled, then the job will be placed to a target queue based on the queue mapping configuration of user *alice*.

For more information about queue mapping configurations, see Configure placement rules. For information about Hive access, see Apache Hive documentation.

How to configure queue mapping

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for ResourceManager. In the Filters pane, under Scope, select ResourceManager.
4. In ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml add the following:
   a. Enable the application-tag-based-placement property to enable application placement based on the user ID passed using the application tags.

```
Name: yarn.resourcemanager.application-tag-based-placement.enable
Value: true
Description: Set to "true" to enable application placement based on the
 user ID passed using the application tags. When it is enabled, it chec
ks for the userid=<userId> pattern and if found, the application will be
```

```
 placed onto the found user's queue, if the original user has the requir
ed rights on the passed user's queue.
```

b. Add the list of users in the allowlist who can use application tag based placement. The applications when the submitting user is included in the allowlist, will be placed onto the queue defined in the yarn.scheduler.capacity.queue-mappings property defined for the user from the application tag. If there is no user defined, the submitting user will be used.

```
Name: yarn.resourcemanager.application-tag-based-placement.username.whit
elist
Value: <Hive process user(s)>
Description: Comma separated list of users who can use the application t
ag based placement, if "yarn.resourcemanager.application-tag-based-place
ment.enable" is enabled.
```

Note: Check the Hive system user value(s) and add the value(s) to the allowlist:

1. In Cloudera Manager, navigate to Hive Configuration .
2. Search for System User.
3. Note down the value(s) set as process username(s), and add the value(s) to the allowlist.

5. Restart the ResourceManager service for the changes to apply.

Configure NodeManager heartbeat

You can control how many containers can be allocated in each NodeManager heartbeat.

To configure NodeManager heartbeat for all queues, perform the following:

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Scheduler Configuration tab.
3. Select the Enable Multiple Assignments Per Heartbeat checkbox to allow multiple container assignments in one NodeManager heartbeat.
4. Configure the following NodeManager heartbeat properties:
   - Maximum Container Assignments Per Heartbeat - The maximum number of containers that can be assigned in one NodeManager heartbeat. Setting this value to -1 disables this limit.
   - Maximum Off-Switch Assignments Per Heartbeat - The maximum number of off-switch containers that can be assigned in one NodeManager heartbeat.
5. Click Save.

Configure data locality

Capacity Scheduler leverages Delay Scheduling to honor task locality constraints. There are three levels of locality constraint: node-local, rack-local, and off-switch. The scheduler counts the number of missed opportunities when the locality cannot be satisfied and waits for this count to reach a threshold before relaxing the locality constraint to the next level. You can configure this threshold using the Node Locality Delay (yarn.scheduler.capacity.node-locality-delay) and Rack Locality Additional Delay (yarn.scheduler.capacity.rack-locality-additional-delay) fields.

To configure data locality, perform the following:

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Scheduler Configuration tab.
3. In the Node Locality Delay textbox, enter the number of scheduling opportunities that can be missed.

   Capacity Scheduler attempts to schedule rack-local containers only after missing this number of opportunities. You must ensure this number is same as the number of nodes in the cluster.

4. In the Rack Locality Additional Delay textbox, enter the number of missed scheduling opportunities, over the Node Locality Delay ones, after which Capacity Scheduler should attempt to schedule off-switch containers.

   A value of -1 means that the value is calculated based on the formula L * C / N, where L is the number of locations (nodes or racks) specified in the resource request, C is the number of requested containers, and N is the size of the cluster.
5. Click Save.

Configure placement rules

Placement Rules allow you to specify a set of rules for assigning jobs and applications to queues. You can define placement rules to dynamically create queues based on the rules and the applications would be submitted to those queues. These predefined rules enable you to submit jobs without specifying the queue name at the time of job submission.

Types of placement rules:

- Static Rules: These rules use static queue mapping expressions for which you must create target queues and specify those queues based on the mapping expressions. You must provide input for the parameters such as userName, queueName and groupName based on the mapping expression.
- Dynamic Rules: These rules allow the creation of queues dynamically based on predefined expressions. You must create a leaf queue first and then provide that queueName, which will be converted to a managed parent queue while executing the rule.

If no rule is satisfied when the application or job runs, the YARN application is rejected.

Configure Per Queue Properties

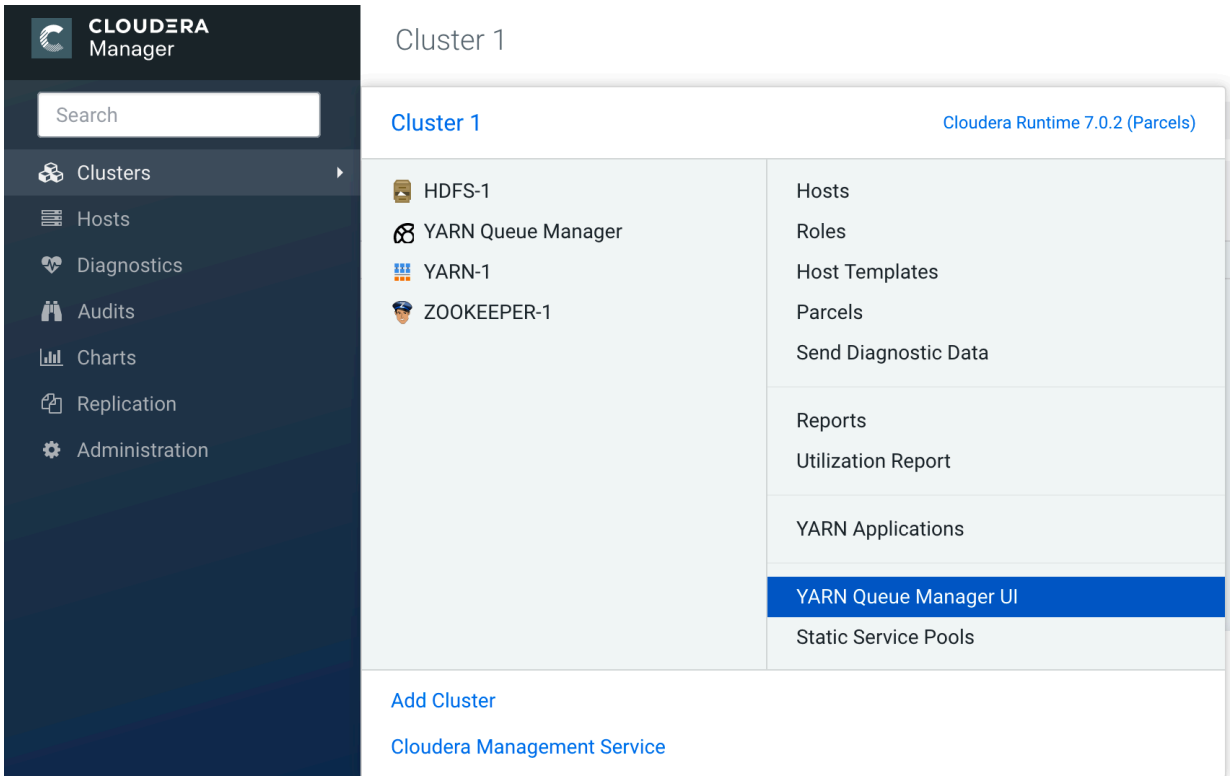
Queue properties contain the settings that define the behavior of the queue. Using queue properties, you can define the setting that need not inherit properties directly from the parent queue and define the setting specific to the queue.

About this task

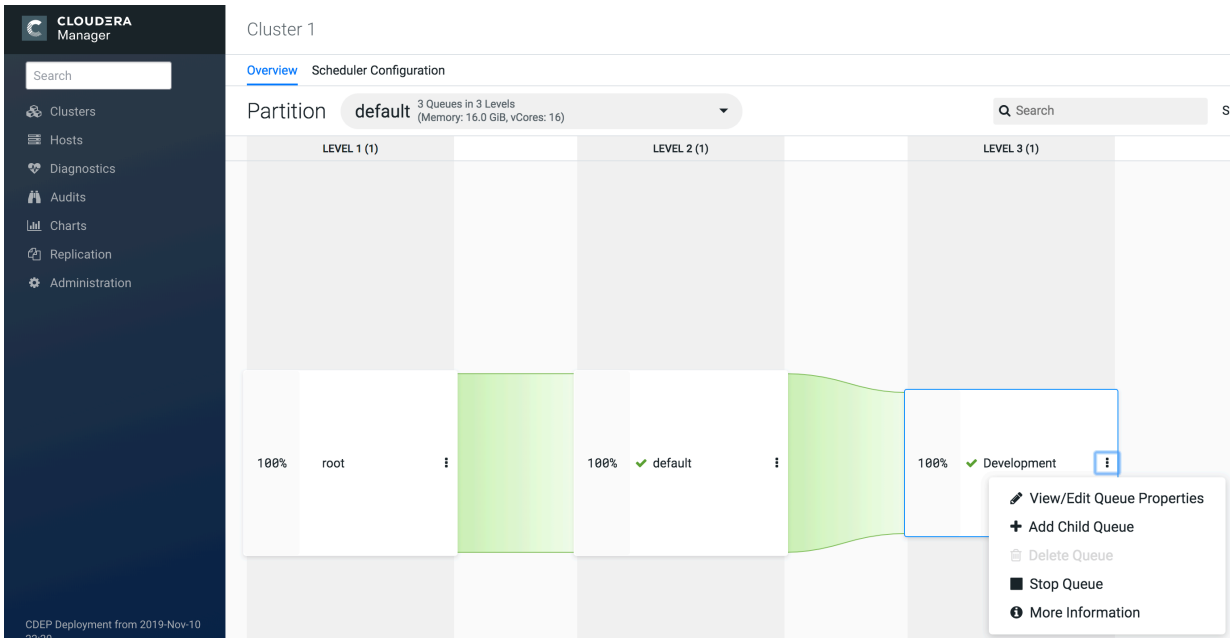
In Cloudera Manager, you can use the Queue Properties to view and configure the queue properties.

Procedure

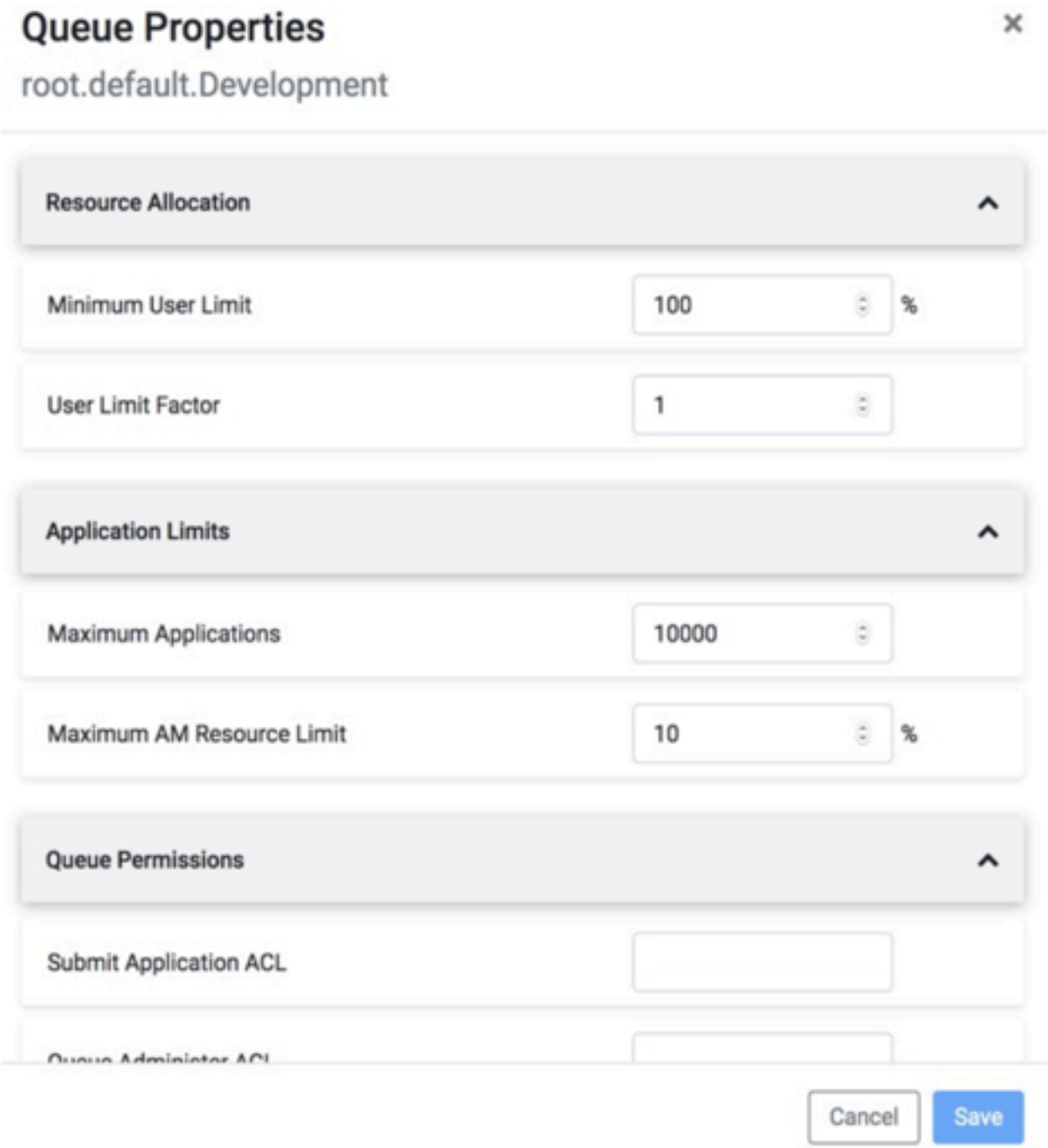
1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service.



2. Click on the three vertical dots on the queue and select the View/Edit Queue Properties option.



3. In the Queue Properties window, enter the value of the property and click Save.



Set user limits within a queue

Set a minimum percentage of resources allocated to each leaf queue user.

Minimum User Limit

The Minimum User Limit (minimum-user-limit-percent) property can be used to set the minimum percentage of resources allocated to each leaf queue user. The Minimum User Percentage is a soft limit on the smallest amount of resources a single user should get access to if they are requesting it. For example, if you set the Minimum User

Percentage property to 10%, 10 users get 10% each, assuming they are all requesting it; this value is a soft limit because if one of the users asks for less capacity, you can place more users in the queue.

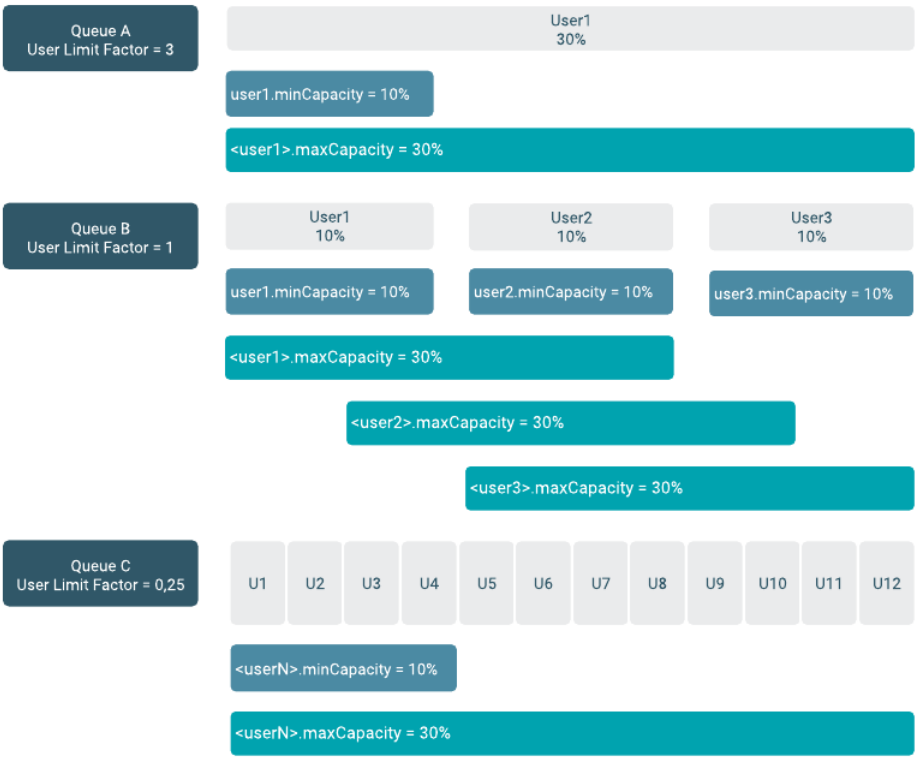
For example, to enable equal sharing of the *services* leaf queue capacity among five users, you must set the Minimum User Limit value to 20%.

To configure user limits based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the Services queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter 20 in the Minimum User Limit text box.
4. Click Save.

User Limit Factor

You can use the User Limit Factor (user-limit-factor) to control the maximum amount of resources that a single user can consume. User Limit Factor is set as a multiple of the queues minimum capacity where a user limit factor of one means the user can consume the entire minimum capacity of the queue. If the user limit factor is greater than 1, it is possible for the user consumption to grow into maximum capacity and if the value is set to less than 1, such as 0.5, a user can only obtain half of the minimum capacity of the queue. If you want a user to obtain the maximum capacity of a queue, set a value greater than 1 to allow the minimum capacity to be overtaken by a user that many times.



The following table shows how the queue resources are adjusted as users submit jobs to a queue with a minimum user limit percentage is set to 20%:

Table 1: Minimum User Limit

Minimum user limit percent = 20	
1st user submits job	Sole user gets 100% of the queue capacity
2nd user submits job	Each user equally shares 50% of the queue capacity
3rd user submits job	Each user equally shares 30.33% of the queue capacity

Minimum user limit percent = 20	
4th user submits job	Each user equally shares 25% of the queue capacity
5th user submits job	Each user equally shares 20% of the queue capacity
6th user submits job	6th user must wait for queue capacity to free up

- Queue resources are adjusted in the same manner for a single user submitting multiple jobs in succession. If no other users are requesting queue resources, the first job receives 100% of the queue capacity. When the user submits a second job, each job receives 50% of queue capacity. When the user submits a third job, each job receives 33% of queue capacity. If a fourth user then submits a job, each job would receive 25% of queue capacity. When the number of jobs submitted by all users reaches a total of five, each job will receive 20% of queue capacity, and subsequent users must wait for queue capacity to free up (assuming preemption is not enabled).
- The Capacity Scheduler also manages resources for decreasing numbers of users. As users' applications finish running, other existing users with outstanding requirements begin to reclaim that share.
- Note that despite this sharing among users, the FIFO application scheduling order of Capacity Scheduler does not change. This guarantees that users cannot monopolize queues by submitting new applications continuously. Applications (and thus the corresponding users) that are submitted first always get a higher priority than applications that are submitted later.

Capacity Scheduler's leaf queues can also use the user-limit-factor property to control user resource allocations. This property denotes the fraction of queue capacity that any single user can consume up to a maximum value, regardless of whether or not there are idle resources in the cluster.

To configure the maximum limit (user-limit-factor) based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue you want to set limit and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter 1 in the User Limit Factor text box.
4. Click Save.

The default value of 1 means that any single user in the queue can, at the most, only occupy only the queue's configured capacity. This prevents users in a single queue from monopolizing resources across all queues in a cluster. Setting the value to 2 restricts the queue's users to twice the queue's configured capacity. Setting it to a value of 0.5 restricts any user from using resources beyond half of the queue capacity.

Set Maximum Application limit for a specific queue

To avoid system-thrash due to an unmanageable load -- caused either by malicious users, or by accident -- the Capacity Scheduler enables you to place a static, configurable limit on the total number of concurrently active (both running and pending) applications at any one time.

About this task

You can set the maximum-application-limit property using the Maximum Application queue property. The limit for running applications in any specific queue is a fraction of this total limit, proportional to its capacity. This is a hard limit, which means that once this limit is reached for a queue, any new applications to that queue will be rejected, and clients will have to wait and retry later.

To set the application limit (yarn.scheduler.capacity.<queue-path>.maximum-applications) on a specific queue, perform the following:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.

2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter the maximum application limit in the Maximum Application text box.
4. Click Save.

   For information about setting application limits on all the queues, see Set Application Limits.

Set Application-Master resource-limit for a specific queue

The Application Master (AM) resource limit can be used to set a maximum percentage of cluster resources allocated specifically to Application Masters. The default value is 10% and exists to avoid cross-application deadlocks where significant resources in the cluster are occupied entirely by the Containers running Application Masters.

About this task

This property also indirectly controls the number of concurrent running applications in the cluster, with each queue limited to a number of running applications proportional to its capacity.

To set the maximum Application Masters resource limit (yarn.scheduler.capacity.maximum-am-resource-percent) for a specific queue:

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, enter the limit in the Maximum AM Resource Limit text box.
4. Click Save.

   For information about setting Application Master resource limits on all the queues, see Set Application Master Resource Limit.

Control access to queues using ACLs

Use Access-control lists (ACLs) to control access rights to users and administrators to the Capacity Scheduler queues.

Application submission can really only happen at the leaf queue level, but an ACL restriction set on a parent queue will be applied to all of its descendant queues.

In the Capacity Scheduler, ACLs are configured by granting queue access to a list of users and groups with the Submit Application ACL parameter. The format of the list is "user1,user2 group1,group2" -- a comma-separated list of users, followed by a space, followed by a comma-separated list of groups.

Note: The default value of Submit Application ACL for a root queue is yarn, which means that only the default yarn user can submit applications to that queue. Therefore, to provide specific users and groups with access to the queue, you must explicitly set the value of Submit Application ACL to those users and groups.

The value of Submit Application ACL (acl_submit_applications) can also be set to "*" (asterisk) to allow access to all users and groups, or can be set to " " (space character) to block access to all users and groups.

As mentioned previously, ACL settings on a parent queue are applied to all of its descendant queues. Therefore, if the parent queue uses the "*" (asterisk) value (or is not specified) to allow access to all users and groups, its child queues cannot restrict access. Similarly, before you can restrict access to a child queue, you must first set the parent queue to " " (space character) to block access to all users and groups.

For example, the following properties would set the root Submit Application ACL value to " " (space character) to block access to all users and groups, and also restrict access to its child "support" queue to the users "sherlock" and "john" and the members of the "cfo-group" group:

Each child queue is tied to its parent queue with the configuration property. The top-level "support", "engineering", and "marketing" queues would be tied to the "root" queue.

To set the ACLs based on this example, perform the following:

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue you want to set ACL and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, add sherlock,john cfo-group in the Submit Application ACL text box.
4. Click Save.

A separate ACL can be used to control the administration of queues at various levels. Queue administrators can submit applications to the queue, kill applications in the queue, and obtain information about any application in the queue (whereas normal users are restricted from viewing all of the details of other users' applications).

If the Queue Administer ACL value is set to " " (space character), it blocks access to all users and groups. If the ACL is set to sherlock,john cfo-group , it allows access to the users "sherlock" and "john" and the members of the "cfo-group" group.

Enable preemption for a specific queue

Capacity Scheduler Preemption allows higher-priority applications to preempt lower-priority applications.

About this task

A scenario can occur in which a queue has a guaranteed level of cluster resources, but must wait to run applications because other queues are utilizing all of the available resources. If Preemption is enabled, higher-priority applications do not have to wait because lower priority applications have taken up the available capacity. With Preemption enabled, under-served queues can begin to claim their allocated cluster resources almost immediately, without having to wait for other queues' applications to finish running.

Note: If the preemption policy is disabled in the scheduler configurations, you cannot enable preemption for a specific queue. For information about setting scheduler level preemption, see Configure preemption on page 27.

You can disable the queue preemption (yarn.resourcemanager.scheduler.monitor.enable) for a specific queue.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, uncheck the Enable Preemption checkbox.
4. Click Save.

Enable Intra-Queue Preemption for a specific queue

Intra-queue preemption prevents resource imbalances in a queue.

About this task

Intra-queue preemption helps in effective distribution of resources within a queue based on configured user limits or application priorities.

Note: If the intra-queue preemption policy is disabled in the scheduler configurations, you cannot enable intra-queue preemption for a specific queue. For information about setting scheduler level intra-queue preemption, see Configure Intra-Queue Preemption.

To disable Intra-Queue Preemption (yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled) for a specific queue

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue and select View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, uncheck the Enable Intra Queue Preemption checkbox.
4. Click Save.

Configure dynamic queue properties

Dynamic queues are auto created based on the predefined expressions of the dynamic placement rule.

A leaf icon will be displayed next to the queue name for dynamically created leaf-queues. You can view the queue properties of the dynamically created leaf queues in the *Dynamic Auto-Creation of Queue* section of the Queue Properties. You can configure the dynamic leaf-queue properties like setting user limits, ACLs, ordering policies by clicking on the Edit Child Queues option of its managed parent queue. The queue properties set at the managed parent queue level will be applied to all of its leaf queues. For information about the queue properties, see Configure Per Queue Properties on page 33.

Set Ordering policies within a specific queue

Set FIFO (First In, First Out) or Fair scheduling policies in Capacity Scheduler depending on your requirements.

The default ordering policy in Capacity Scheduler is FIFO (First In, First Out). FIFO generally works well for predictable, recurring batch jobs, but sometimes not as well for on-demand or exploratory workloads. For these types of jobs, Fair Scheduling is often a better choice. Flexible scheduling policies enable you to assign FIFO or Fair ordering policies for different types of workloads on a per-queue basis.

Examples FIFO and Fair Sharing Policies

Both FIFO (First In, First Out) and Fair scheduling policies work differently in batch jobs and ad hoc jobs.

Batch Example

In the below example, two queues have the same resources available. One uses the FIFO ordering policy, and the other uses the Fair Sharing policy. A user submits three jobs to each queue one after another, waiting just long enough for each job to start. The first job uses 6x the resource limit in the queue, the second 4x, and last 2x.

- In the FIFO queue, the 6x job would start and run to completion, then the 4x job would start and run to completion, and then the 2x job. They would start and finish in the order 6x, 4x, 2x.
- In the Fair queue, the 6x job would start, then the 4x job, and then the 2x job. All three would run concurrently, with each using 1/3 of the available application resources. They would typically finish in the following order: 2x, 4x, 6x.

Ad Hoc Plus Batch Example

In this example, a job using 10x the queue resources is running. After the job is halfway complete, the same user starts a second job needing 1x the queue resources.

- In the FIFO queue, the 10x job will run until it no longer uses all queue resources (map phase complete, for example), and then the 1x job will start.

- In the Fair queue, the 1x job will start, run, and complete as soon as possible – picking up resources from the 10x job by attrition.

Best Practices for Ordering Policies

- Ordering policies are configured on a per-queue basis, with the default ordering policy set to FIFO. Fairness is usually best for on-demand, interactive, or exploratory workloads, while FIFO can be more efficient for predictable, recurring batch processing. You should segregate these different types of workloads into queues configured with the appropriate ordering policy.
- · In queues supporting both large and small applications, large applications can potentially "starve" (not receive sufficient resources). To avoid this scenario, use different queues for large and small jobs, or use size-based weighting to reduce the natural tendency of the ordering logic to favor smaller applications.
- · Use the Maximum AM Resource Limit scheduler property to restrict the number of concurrent applications running in the queue to avoid a scenario in which too many applications are running simultaneously. Limits on each queue are directly proportional to their queue capacities and user limits. This property is specified as a float, for example: 0.5 = 50%. The default setting is 0.1=10%. This property can be set for all queues by setting the Maximum AM Resource Limit property at the root level, and can also be overridden on a per-queue basis by setting the Maximum AM Resource LimitSet default Application Master resource limit on page 28property at the queue level.

Configure queue ordering policies

You can configure the property for queue ordering policies (yarn.scheduler.capacity.<queue-path>.ordering-policy) to FIFO or Fair. The default setting is FIFO.

Procedure

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on the queue you want to configure queue ordering policies and select the View/Edit Queue Properties option.
3. In the Queue Properties dialog-box, select the ordering policy as FIFO or Fair using the Ordering Policy drop-down box .
4. Click Save.

Configure Partitions

You can partition a cluster into sub-clusters so that jobs run on nodes with specific characteristics. You can configure these partitions, so that you run YARN applications on cluster nodes of the specified partition.

Note: The term *Partitions* is used instead of *Node Labels* to be consistent with the YARN terminology.

Partition can be specified as exclusive or non-exclusive/shareable:

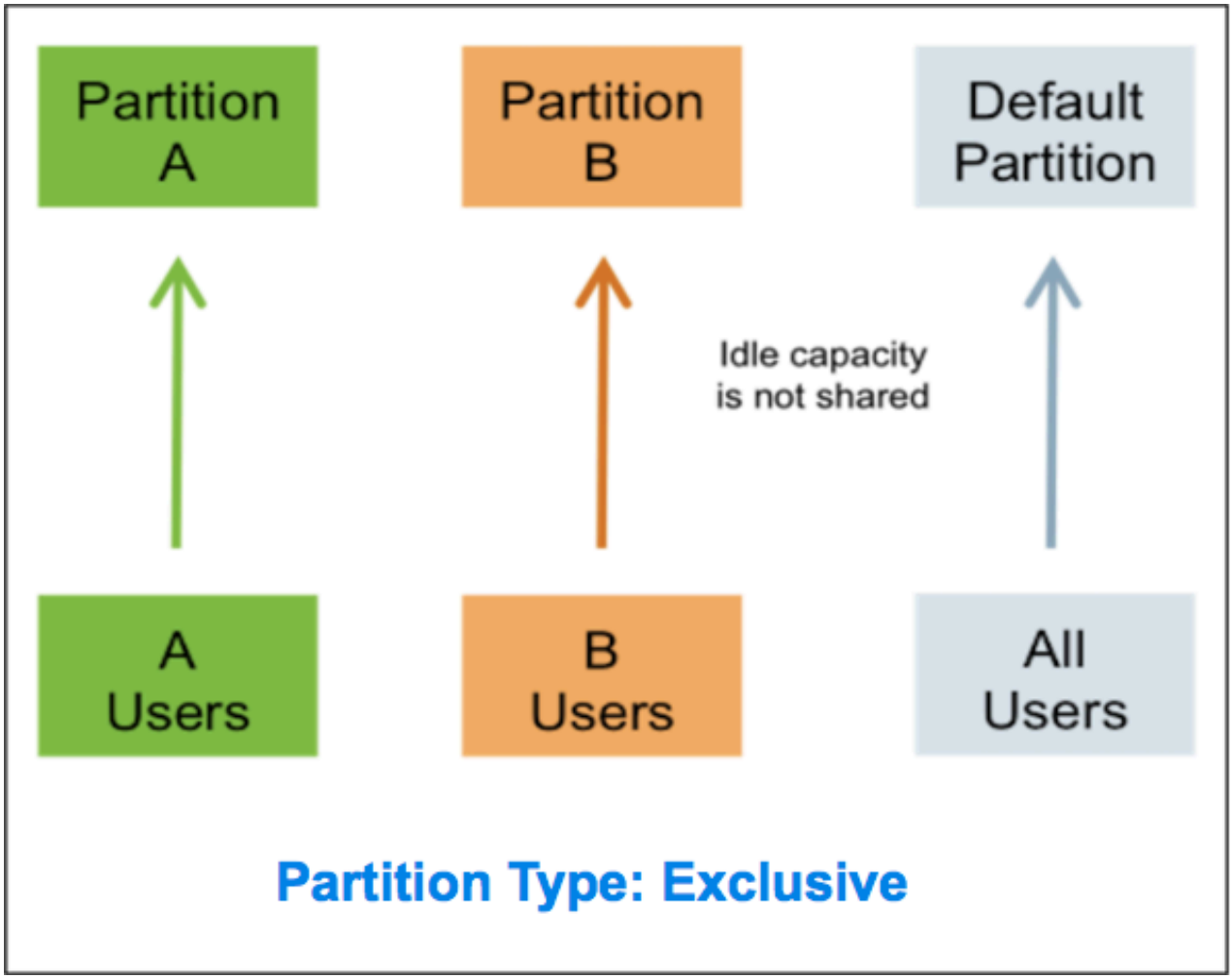
- exclusive - Access is restricted to applications running in queues associated with the partition.
- non-exclusive - If idle capacity is available on the partition, resources are shared with all applications in the cluster.

The fundamental unit of scheduling in YARN is the queue. The capacity of each queue specifies the percentage of cluster resources that are available for applications submitted to the queue. Queues can be set up in a hierarchy that reflects the resource requirements and access restrictions required by the various organizations, groups, and users that utilize cluster resources.

Using partition, you can divide a cluster into sub-clusters so that jobs can be run on partitions with specific characteristics. For example, you can use a partition to run memory-intensive jobs only on nodes with a larger amount of RAM. Partitions can be assigned to cluster nodes, and specified as exclusive or non-exclusive. You can then associate partition with capacity scheduler queues. Each node can be associated with only one partition.

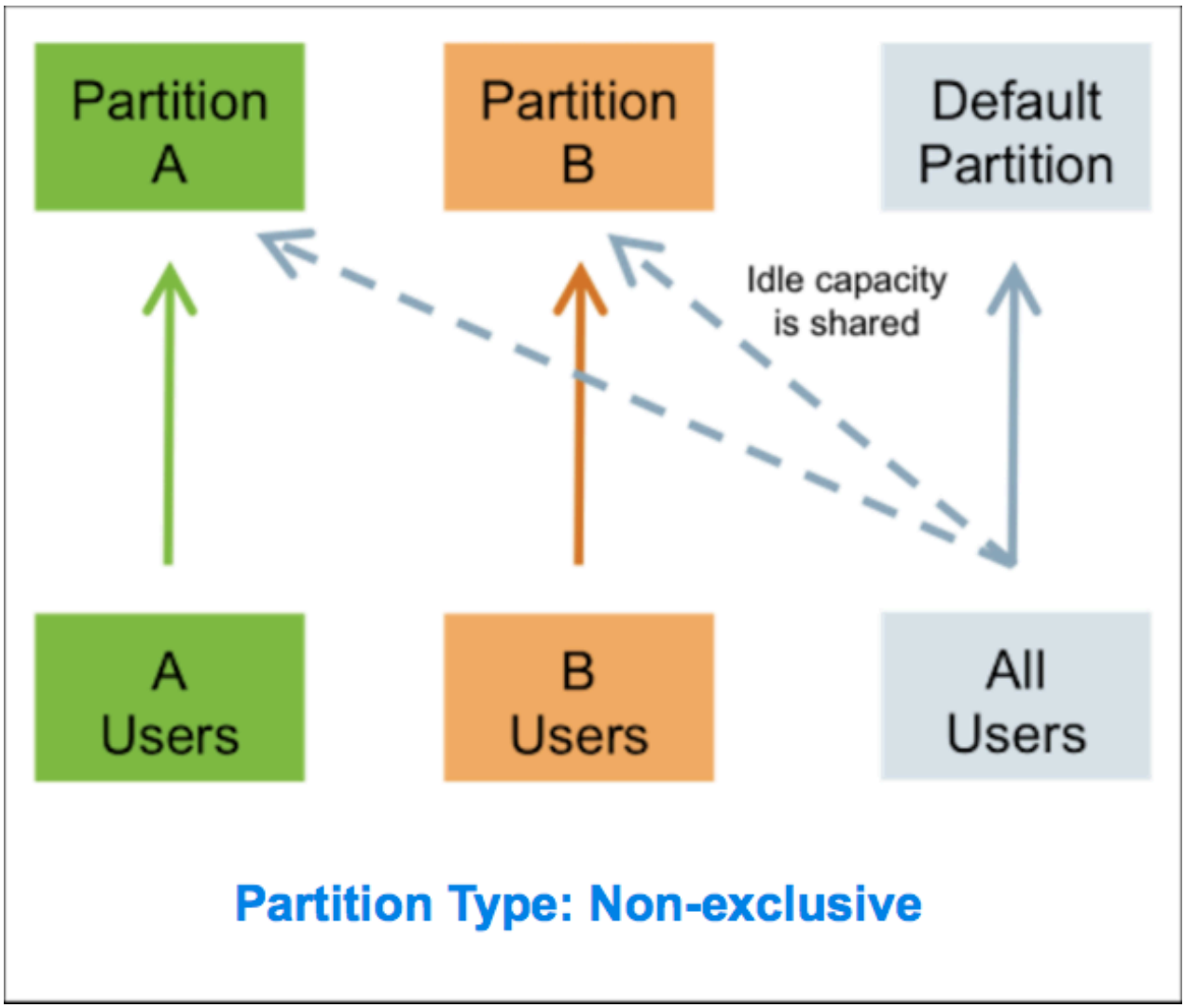
Partition Type: Exclusive

When a queue is associated with one or more exclusive partitions, all applications submitted by the queue will have exclusive access to the nodes in those partitions.



Partition Type: Non-exclusive

When a queue is associated with one or more non-exclusive partitions, all applications submitted by the queue get first priority on nodes in those partitions. If idle capacity is available on those partition nodes, resources are shared with other applications in the cluster. Non-labeled applications are preempted if labeled applications request new resources on the labeled nodes.



Queues without associate partition

If no partition is assigned to a queue, the applications submitted by the queue can run on any node without a partition, and on nodes with non-exclusive partitions if idle resources are available.

Preemption

Labeled applications that request labeled resources preempt non-labeled applications on labeled nodes. If a labeled resource is not explicitly requested, the normal rules of preemption apply. Non-labeled applications cannot preempt labeled applications running on labeled nodes.

Enabling node labels on a cluster to configure partition

You can configure partitions on a cluster by making configuration changes on the YARN ResourceManager host.

Procedure

1. Create a Label Directory in HDFS.
   a) Use the following commands to create a node-labels directory in which to store the Node Labels in HDFS:

```
sudo su hdfs
```

```
hadoop fs -mkdir -p /yarn/node-labels
hadoop fs -chown -R yarn:yarn /yarn
hadoop fs -chmod -R 700 /yarn
```

Where -chmod -R 700 specifies that only the yarn user can access the node-labels directory.

b) Use the following command to confirm that the directory was created in HDFS:

```
hadoop fs -ls /yarn
```

The new node-labels directory should appear. The owner should be yarn, and the permission should be drwx:

```
Found 1 items
                                drwx------ - yarn yarn 0 2014-11-24 13:09 /
yarn/node-labels
```

2. In Cloudera Manager, navigate to YARN Configuration .
3. Search for node label.
4. Find the Node Labels property and ensure that it is selected (enabled).
5. Find the Node Labels Directory property and set its value to reference the HDFS node label directory.

   For example: hdfs://node-1.example.com:8020/yarn/node-labels/.

   HDFS paths are automatically translated, however, any other file system requires a full path to the Node Labels Directory.

   The default value of the Node Labels Directory property is /yarn/node-labels.
6. Start or Restart the YARN ResourceManager.

Create partitions

You must first create partitions to assign them to nodes and associate it with queues.

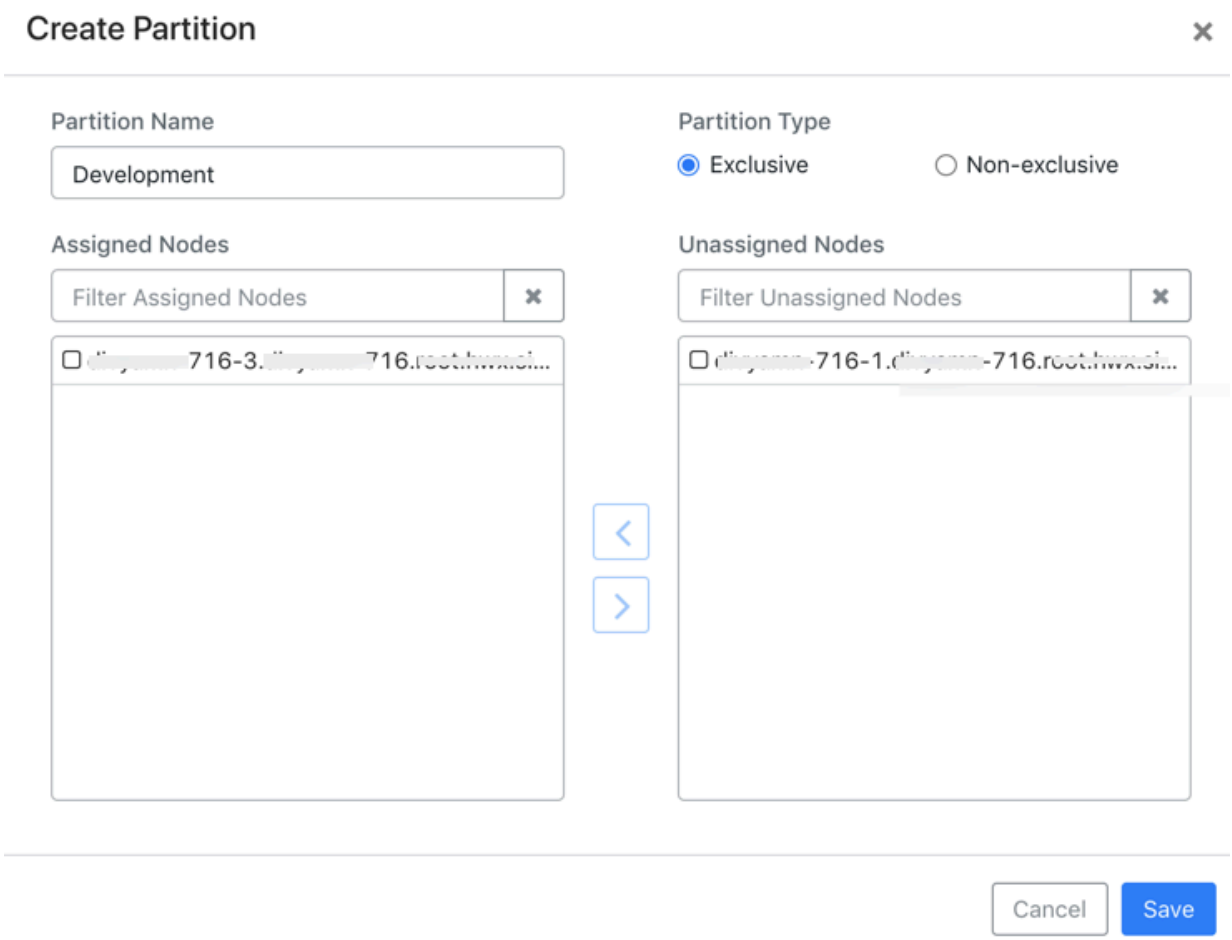
Before you begin

You must enable node labels on a cluster before you create a partition. For more information, see Enable node labels on a cluster.

Procedure

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Partitions tab.

3. Click +Create. The Create Partition dialog box is displayed.



4. Add a name for the partition in Partition Name.
5. Select Exclusive or Non-Exclusive node label type under Partition Type. For information about Exclusive or Non-Exclusive partition type, see Configure Partitions.
6. Select one or more unassigned nodes listed under Unassigned Nodes and click the < arrow button to move it under Assigned Nodes to assign it to the partition. You can also search or filter the nodes using the regular expression.
7. Click Save.

Assign or unassign a node to a partition

You can assign or unassign a node to an existing partition.

Procedure

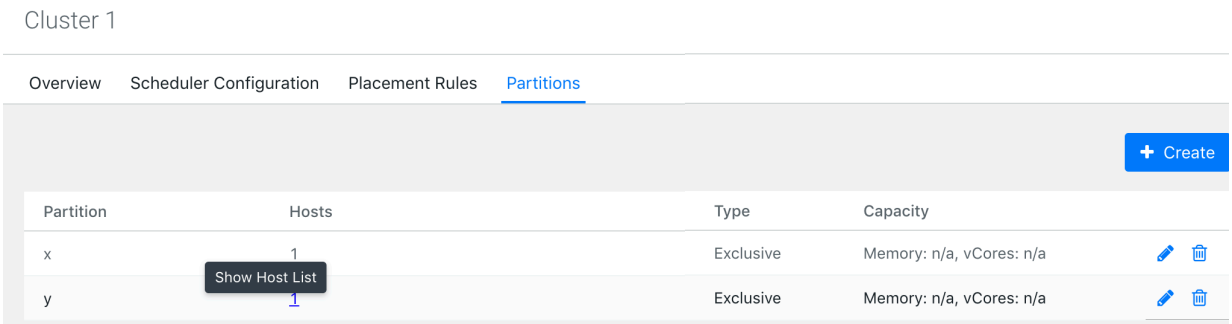
1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Partition tab. A list of partitions is displayed.
3. Click the Edit icon at the right of a partition.

4. Assign or unassign the nodes in the Edit Partition dialog box.
   a. Assign Nodes: Select an unassigned node listed under Unassigned Nodes and click the < arrow button to move it to under Assigned Nodes.
   b. Unassign Nodes: Select an assigned node listed under Assigned Nodes and click the > arrow button to move it to under Unassigned Nodes.
5. Click Save.

View partitions

You can view the list of available partitions in the cluster. For each partition, it lists the number of associated nodes under the Hosts column along with the partition type and capacity.

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Partitions tab. A list of existing partitions is displayed.
3. Optionally, you can click on the number listed in the Host column to view the associated nodes.



View Node Label Assignments

You can use the following commands to view information about partitions.

- List all running nodes in the cluster: yarn node -list

  Example:

```
[root@node-1 /]# yarn node -list
14/11/21 12:14:06 INFO impl.TimelineClientImpl: Timeline service address:
 http://node-1.example.com:8188/ws/v1/timeline/
14/11/21 12:14:07 INFO client.RMProxy: Connecting to ResourceManager at no
de-1.example.com/240.0.0.10:8032
Total Nodes:3
 Node-Id Node-State Node-Http-Address Number-of-Running-Containers
node-3.example.com:45454 RUNNING node-3.example.com:50060 0
node-1.example.com:45454 RUNNING node-1.example.com:50060 0
node-2.example.com:45454 RUNNING node-2.example.com:50060 0
```

- List the status of a node (includes partition): yarn node -status <Node_ID>

  Example:

```
[root@node-1 /]# yarn node -status node-1.example.com:45454
14/11/21 06:32:35 INFO impl.TimelineClientImpl: Timeline service address:
 http://node-1.example.com:8188/ws/v1/timeline/
14/11/21 06:32:35 INFO client.RMProxy: Connecting to ResourceManager at
 node-1.example.com/240.0.0.10:8032
Node Report :
 Node-Id : node-1.example.com:45454
 Rack : /default-rack
```

```
Node-State : RUNNING
Node-Http-Address : node-1.example.com:50060
Last-Health-Update : Fri 21/Nov/14 06:32:09:473PST
Health-Report :
Containers : 0
Memory-Used : 0MB
Memory-Capacity : 1408MB
CPU-Used : 0 vcores
CPU-Capacity : 8 vcores
Node-Labels : x
```

Partitions are also displayed in the ResourceManager UI on the Nodes and Scheduler pages.

Associate partitions with queues

You can use partitions to run YARN applications on cluster nodes that have a specified partition.

Before you begin

Before associating partitions, you must create partitions and assign partitions to cluster nodes. For more information about creating partitions see, Create partitions on page 44.

Note: After you associate a partition with one or more queues, in the YARN Queue Manager UI, click Overview > <*Partition name*> from the dropdown list and distribute capacity to the queues before switching allocation mode or creating placement rules.

Note: The term *Partitions* is used instead of *Node Labels* to be consistent with the YARN terminology.

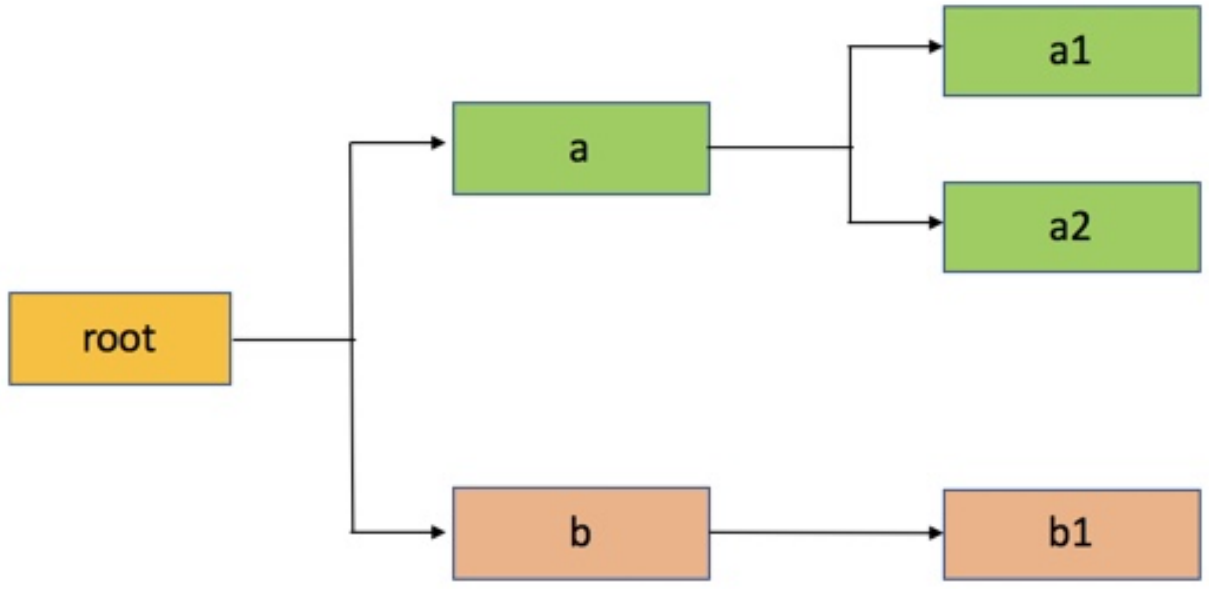
About this task

Use Queue Manager to create and assign partitions to cluster nodes, associate the partitions (yarn.scheduler.capacity.<queue-path>.accessible-node-labels) to queues and configure capacity to that queue for the specified partition. Queue Manager evenly distributes the available capacity among all the queues in the partition. You can manually modify the capacity on each partition of each queue, and also ensure that the sum of capacities of each partition of direct children of a parent queue at every level is equal to 100%. partitions that a queue can access (accessible partitions of a queue) must be the same as, or a subset of, the accessible partitions of its parent queue.

Example

Assume that a cluster has a total of 8 nodes. The first 3 nodes (n1-n3) have partition = x, the next 3 nodes (n4-n6) have partition = y, and the final 2 nodes (n7, n8) do not have any partitions. Each node can run 10 containers.

The queue hierarchy is as follows:



Assume that queue “a” can access partitions “x” and “y”, and queue “b” can only access partition “y”. By definition, nodes without labels can be accessed by all queues.

Consider the following example label configuration for the queues, in the Relative resource allocation mode:

capacity(a) = 40, capacity(a, label=x) = 100, capacity(a, label=y) = 50; capacity(b) = 60, capacity(b, label=y) = 50

This means that:

- Queue “a” can access 40% of the resources on nodes without any labels, 100% of the resources on nodes with label=x, and 50% of the resources on nodes with label=y.
- Queue “b” can access 60% of the resources on nodes without any labels, and 50% of the resources on nodes with label=y.

You can also see that for this configuration:

capacity(a) + capacity(b) = 100

capacity(a, label=x) + capacity(b, label=x) (b cannot access label=x, it is 0) = 100

capacity(a, label=y) + capacity(b, label=y) = 100

For child queues under the same parent queue, the sum of the capacity for each label should equal 100%.

Similarly, you can set the capacities of the child queues a1, a2, and b1:

a1 and a2: capacity(a.a1) = 40, capacity(a.a1, label=x) =30, capacity(a.a1, label=y) =50 capacity(a.a2) = 60, capacity(a.a2, label=x) =70, capacity(a.a2, label=y) =50;

b1: capacity(b.b1) = 100, capacity(b.b1, label=y) = 100

You can see that for the a1 and a2 configuration:

capacity(a.a1) + capacity(a.a2) = 100

capacity(a.a1, label=x) + capacity(a.a2, label=x) = 100

capacity(a.a1, label=y) + capacity(a.a2, label=y) = 100

How many resources can queue a1 access?

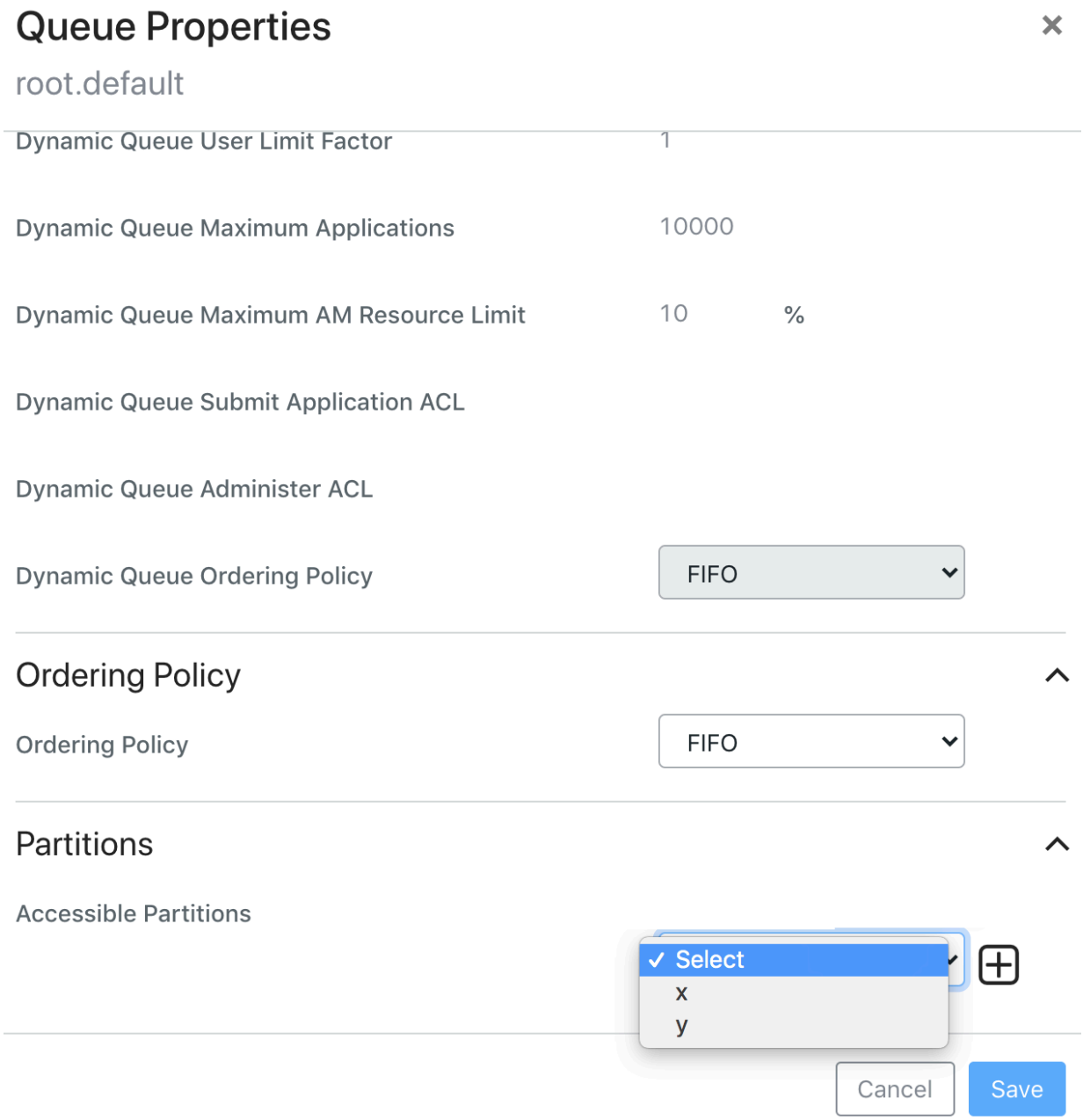
Resources on nodes without any labels: Resource = 20 (total containers that can be allocated on nodes without a label, in this case n7, n8) * 40% (a.capacity) * 40% (a.a1.capacity) = 3.2 (containers)

Resources on nodes with label=x

Resource = 30 (total containers that can be allocated on nodes with label=x, in this case n1-n3) * 100% (a.labelx.capacity) * 30% = 9 (containers)

To implement this example configuration, perform the following

1. In Cloudera Manager, select Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on a queue and select the View/Edit Queue Properties option.
3. In the Queue Properties dialog-box , select the x label from the Accessible Node Labels drop-down box, click +, again select the y label from the Accessible Node Labels drop-down boxand click Save.



4. Repeat the above steps to assign x label for a1 and a2 queues.
5. Click on the three vertical dots on b queue and select the View/Edit Queue Properties option.
6. In the Queue Properties dialog-box, select the y label from the Accessible Node Labels drop-down box, click +, and click Save.

7. Repeat the above steps to assign y label for b1, a, a1, and a2 queue.

   Queue Manager automatically distributes the available capacity among all the queues in the partition. If you want to modify the capacity of the queues, click on the Partition drop-down box in the Overview tab, select the label and modify the queue capacity.
8. In the Overview tab, click on the Partition drop-down box and select label y.
9. Click on the three vertical dots on the a queue and select the Edit Child Queues option.
10. Enter the Configured Capacity of a1 to 50 and a2 to 50 and Click Save.
11. Click on the three vertical dots on the b queue and select the Edit Child Queues option.
12. Enter the Configured Capacity of b1 to 100 and click Save.
13. Click on the three vertical dots on the root queue and select the Edit Child Queues option.
14. Enter the Configured Capacity of a to 50 and b to 50 and click Save.

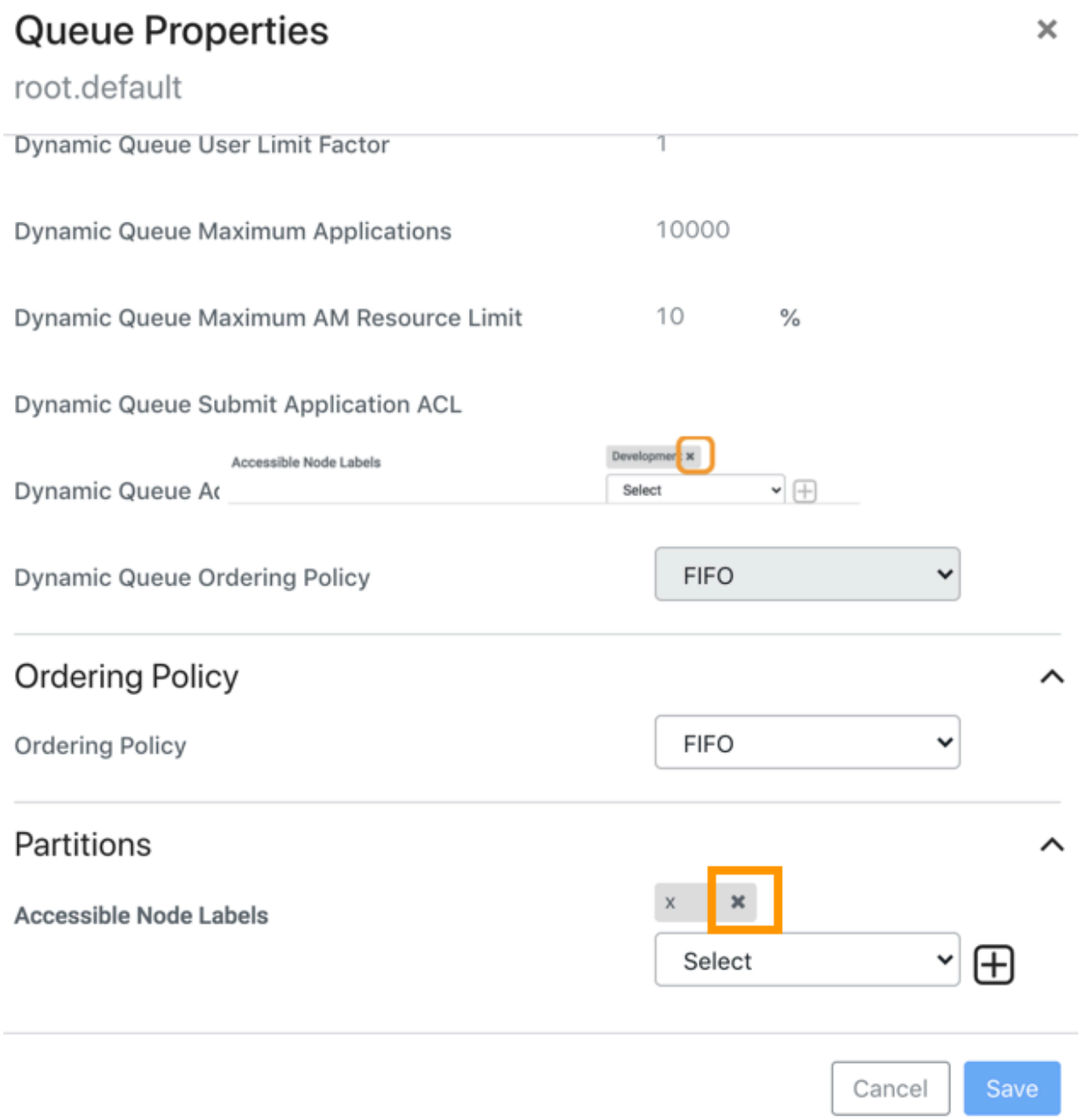
Disassociate partitions from queues

You can disassociate a partition from the queue. You should disassociate a partition before you delete the queue. Before disassociating a partition from a queue, you should remove the partition capacity for that queue by setting it to zero.

Procedure

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click on the three vertical dots on a queue and select the View/Edit Queue Properties option.

3. In the Queue Properties dialog box, from the Accessible Partitions, click X next to the name of the partition.



4. Click Save.

Delete partitions

In this release, due to a known issue, it is not recommended to delete a partition if it is associated with queues and the queues have capacities configured for that partition.

Before you begin

- Disassociate queues from partition (node label): You must disassociate all the queues associated with the partition (node label). For information on disassociating a partition (node label), see Disassociate partitions from queues on page 50.

- Configure queue capacity to zero: You must change the Configured Capacity to 0. For information about configuring queue capacity see, Configure cluster capacity with queues on page 20.

Procedure

1. In Cloudera Manager, select the Clusters > YARN Queue Manager UI service. A graphical queue hierarchy is displayed in the Overview tab.
2. Click the Partitions tab. A list of existing partitions is displayed.
3. Click the delete icon at the right of a partition name.
4. Click Delete.

Use partitions when submitting a job

You can use various methods to specify partitions when submitting jobs.

Procedure

- Set partitions when Submitting Jobs

  You can use the following methods to specify partitions when submitting jobs:

  - ApplicationSubmissionContext.setNodeLabelExpression(<node_label_expression>) - sets the partition expression for all containers of the application.
  - ResourceRequest.setNodeLabelExpression(<node_label_expression>) - sets the partition expression for individual resource requests. This overrides the partitions expression set in ApplicationSubmissionContext.set NodeLabelExpression(<node_label_expression>).
  - Specify setAMContainerResourceRequest.setNodeLabelExpression in ApplicationSubmissionContext to indicate the expected partition for the ApplicationMaster container.

  You can use one of these methods to specify a partition expression, and -queue to specify a queue, when you submit YARN jobs using the distributed shell client. If the queue has a label that satisfies the label expression, it will run the job on the partition(s). If the label expression does not reference a label associated with the specified queue, the job does not run and an error is returned. If no partitionn is specified, the job runs only on nodes without a partition, and on nodes with non-exclusive partitions if idle resources are available.

  Note:

  You can only specify one partition in the .setNodeLabelExpression methods.

  For example, the following commands run a simple YARN distributed shell "sleep for a long time" job. In this example you are asking for more containers than the cluster can run so you can see which node the job runs on. We are specifying that the job should run on queue "a1", which our user has permission to run jobs on. We are also using the -node_label_expression parameter to specify that the job will run on all nodes with label "x".

  ```
  sudo su yarn
  hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-yarn/hadoop-yarn-applic
  ations-distributedshell.jar
   -shell_command "sleep 100" -jar /opt/cloudera/parcels/CDH/lib/hadoop-ya
  rn/hadoop-yarn-applications-distributedshell.jar
   -num_containers 30 -queue a1 -node_label_expression x
  ```

  If you run this job on the example cluster we configured previously, containers are allocated on node-1, as this node has been assigned partition "x", and queue "a1" also has partition "x":

  The following commands run the same job that you specified for partition "x", but this time you will specify queue "b1" rather than queue "a1".

  ```
  sudo su yarn
  ```

```
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-yarn/hadoop-yarn-applic
ations-distributedshell.jar
 -shell_command "sleep 100000" -jar /opt/cloudera/parcels/CDH/lib/hadoop-
yarn/hadoop-yarn-applications-distributedshell.jar
 -num_containers 30 -queue b1 -node_label_expression x
```

When you attempt to run this job on our example cluster, the job will fail with the following error message because label "x" is not associated with queue "b1".

```
14/11/24 13:42:21 INFO distributedshell.Client: Submitting application to
 ASM
14/11/24 13:42:21 FATAL distributedshell.Client: Error running Client
org.apache.hadoop.yarn.exceptions.InvalidResourceRequestException: Invalid
 resource request, queue=b1 doesn't
have permission to access all labels in resource request. labelExpression
 of resource request=x. Queue labels=y
```

- MapReduce Jobs and Partitions

  Currently you cannot specify a partition when submitting a MapReduce job. However, if you submit a MapReduce job to a queue that has a default partition expression, the default partition is applied to the MapReduce job.

  Using default partition expressions tends to constrain larger portions of the cluster, which at some point starts to become counter-productive for jobs - such as MapReduce jobs - that benefit from the advantages offered by distributed parallel processing.