YARN-9879: Allow multiple leaf queues with the same name in CS

Requirements and Design doc - V1

Authors: Gergely Pollák, Szilárd Németh, Rudolf Réti

What we want to achieve

Currently in Capacity Scheduler a leaf queue name must be unique, which is a huge constraint, considering all queues have hierarchical path assigned to them, there is no real reason, besides historical ones, to identify queues only by their names.
We propose to make it possible in Capacity Scheduler to support multiple leaf queues with the same name if their parent is different.
This also implies we need to make it possible to reference queues by their full name instead of just a leaf queue name, since it will stop being unique.

Use Cases

- **Migrating users from Fair Scheduler:** We are expecting some users to migrate from Fair Scheduler to capacity, and to make the transition as seamless as possible we need to ensure they can migrate their queue hierarchy to Capacity Scheduler. Currently leaf queue names are unique in Capacity Scheduler this means we can easily migrate or introduce a queue reference by path approach. Because migrating users want to use scripts and job starters which reference the queues by their path instead of the queue name, so we need to make sure, they won't need to change anything with their start scripts. Migration to Capacity Scheduler should be transparent from the application's point of view, but it's currently not, because of the different queue naming methodology.
- **Making Capacity Scheduler truly hierarchical:** Currently in Capacity Scheduler we can define queue hierarchies but due to historical reasons these queues are stored and referenced by their leaf queue names, which makes it more like a list of queues instead of a tree. If we can drop this last constraint the queues would become actually hierarchical.
- **Different user queues for different resource types**: There are several users who define multiple queues with the leaf queue as the user name, where the difference between the queues is the available custom resources, like GPU or FPGA. These users want to keep the multiple queues named after users because of the different pricing of the resources in certain queues.

Requirements

- Make sure we don't break anything, and keep backwards compatibility
- We shouldn't force the new behaviour on old users, make it optional to use this feature
- Queue names must be unique only under their direct parent queue, or in other words paths to the queues must be unique
- Queues should be able to be referenced by their full path since the name of the leaf queue is no longer unique

Proposed solutions

Multiple phases

It is not entirely possible to solve this issue and keep backwards compatibility without separating the two CS behaviours. However if we separate them entirely then we split our user base, instead of encouraging them to move towards the new CS queue hierarchy.
We suggest a multi-phase approach to cover the most important parts as soon as we can, then help the migration of the old CS users as well.
In this version of the documentation we suggest the following three phases:

1. We need to make sure FS users can use CS without modifying their applications or start scripts. This means we need to add support for multiple leaf queues with the same name and the queue reference by path as well.
2. We should investigate if it's possible to combine the two solutions, where the queues can be referenced by leaf and full name as well. Obviously there is no way to solve it in all cases since leaf queue will not be necessarily unique, but old CS users meet this constraint which means, we can change the internal queue naming and referencing to use full queue names with backwards compatible "leaf queue reference" support. This means as long as the users won't create multiple leaf queues they still can reference them by their leaf names. This is a larger refactor since we are using the queue's name over 190 times in the CS code.
3. I'm not entirely sure how far we want to push this modification, but in the far future we might remove the support for simple leaf queue name references, and push users towards the full queue naming, but even if we want so, this is a long term goal. And it would be phase 3.

Phase 1 - Separate the two modes

We propose to introduce a configuration property which controls which mode we want to use, default mode should behave exactly like the current CS, while the new mode should force CS to use full path names for all queues and leaf queues instead of just the name of the current element of the queue. This makes sure we keep the backwards compatibility while we can test out if the new method works properly.
We need to investigate if this change in the queue naming will affect anything besides commiting jobs.
We also need to remove the leaf queue name unique constraint.

Phase 1 - Allow queues to be referenced by their path

To make sure FS users won't need to alter their applications and starting scripts, we recommend to introduce a new configuration property which controls if the queue names should

be full names or only leaf queue name. If the previous option is turned only the fully queue names should be supported, since leaf queue name can be ambiguous.

Topics for discussion

Phase 2 - Hybrid mode

In hybrid mode the queue names are referenced by the full path, but we allow application submission using only leaf queue name. This mode can help the migration effort to old CS queue naming to new.
Caveats:

- How should we handle cases when there are two leaf queues with the same name
    - We could disable queue reference by leaf name if it's ambiguous
    - We can submit the application to the first/last queue registered with the provided name
    - We can implement multiple policies and let the user choose, we think this is the most flexible and preferred solution
- Will old yarn clients be compatible if we are using full queue path instead of names internally? This will affect the behaviour of the CSQueue's getQueueName method which might cause issues. We need to check if this approach is plausible.

Phase 1 / 2 - YARN client option to force queue selection mode

We could extend the protobuf protocol to tell which queue naming mode the client wants to use, and the backend could process the request accordingly. This could even bypass the default settings on the server. So if it is running in "old mode" where queues are referenced by their leaf names, but the client provides a flag that it is using full queue name, then we could still identify the queue. This could help the migration effort by allowing all the configs to remain the same on the backend so it would obviously compatible with all the old applications, but new applications could use the new queue reference model.

Phase 3 - Do we need phase 3?

For the sake of consistency it would be nice, but if we can provide stable and user friendly solution during phase 1 and 2, we might skip phase three, and keep the queue reference by leaf name option forever, this might increase the maintenance cost a bit, but if the hybrid mode solutions are sound enough we might not want to push extra work on our clients.

Caveats

Legacy code

Both Fair Scheduler and Capacity Scheduler users have a huge legacy code base, which they don't want to change just because of a scheduler change. Unfortunately queue naming is not an internal concept but part of our external interface is we change it it can affect the applications using YARN. So whatever solution we come up with, it cannot affect these codes. This is why Phase 3 probably won't ever go live, since we just cannot break backwards compatibility this hard.

Core change

The queue name is deeply intertwined with the rest of the Capacity Scheduler so changing the queue naming and keeping backwards compatibility can be a really though job. We have around 200 references to the different queue name getter methods, so if anything changes in these methods seriously, we need to track down all their usage, and what might it break.

No real common code base

Fair Scheduler and Capacity Scheduler are two almost entirely different components with only some minimal shared code base, which makes it much harder to make their behaviour similar or at least somewhat compatible. Determining and extracting the parts which can be shared between the two schedulers might help.