CPU as a YARN Resource: A Hybrid Approach

Intro

Applications that run on YARN have different characteristics in the amount of CPU they need and can take advantage of. Exposing the right knobs for configuring CPU, both from a node-capacity perspective and application-need perspective will allow YARN schedulers to achieve better cluster resource utilization and minimize contention. It will also permit enhanced job run time predictability both within and between clusters.

This document will outline how CPU configuration should work both from the cluster administrator and application writer/submitter perspectives.

CPU Resource Attributes

CPU will be configured and requested with two complementary units.

- Virtual cores will represent parallelism. A container request with three virtual cores means that the container will generally be running three threads at the same time. Nodes should be configured with a number of virtual cores equal to the number of physical cores they contain.
- YARN compute units (YCUs) are a standardized unit for processing power. A container request is submitted with a number of YCUs per core. A YCUs per core value of -1 will mean that the task wishes to get a full core. Different processors will be rated with different amounts of YCU capacity. The typical core will have a rating somewhere near 1000 YCUs, which will allow placing multiple CPU-light tasks on a core without the need to floating point values.

How CPUs Work From YARN's Perspective

CPU as a resource functions differently from memory. Giving a task less processing power than it requests will not cause it to fail in the same way that doing so for memory can. A CPU can in some ways be looked at as the aggregate of the processing power of all its cores, but this falls apart if we allow a single-threaded task to take up more processing power than a single core.

Here are assumptions more explicitly about how YARN views CPUs.

- A CPU is essentially a bathtub full of processing power that can be doled out to threads, with a limit per thread based on the power of each core within it.
- To give X processing power to a thread means that within a standard unit of time, roughly some number of instructions proportional to X can be executed for that thread.
- No more than a certain amount of processing power (the amount of processing power per core) can be given to each thread.
- We can use CGroups to say that a task gets some fraction of the system's processing power.

- This means that if we have 5 cores with Y processing power each, we can give 5 threads Y processing power each, or 6 threads 5Y/6 processing power each, but it makes no sense to give 4 threads 5Y/4 processing power each.
- It never makes sense to use CGroups assign a higher fraction of the system's processing power than (numthreads the task can take advantage of / number of cores) to a task.
- Equivalently, if my CPU has X processing power per core, it never makes sense to assign more than (numthreads the task can take advantage of) * X processing power to a task.

From a scheduling perspective, YARN needs to keep track of a node's capacity and how much of that capacity has been allocated to containers. We will keep track of capacity and usage in terms of YCUs. **When a container is placed on a node, the amount of YCUs it takes up will be (# of virtual cores in container request) * min(# of YCUs per core in container request, # of YCUs per core on the node).**

Requesting Processing Power for Your Job

The typical MapReduce task, which runs in a single thread, should request a single virtual core. Deciding how many YCUs to request is a little trickier. Tasks that are primarily performing computation should request a full core, i.e. -1 YCUs. Tasks that are heavily I/O bound, like from a distcp job, should request less than a full core, perhaps somewhere in the 500 YCUs range. Using a utility like `top` while running the task on an uncontended system can allow for more precise values. For example, if I run my task on a system where each core is rated at 1500 YCUs and see that its `top` CPU % is 66, I would request it with 1000 YCUs. If my task has two threads and I see that its `top` CPU % is 150, I would request it with 2 virtual cores and (1500 * 1.5 / 2 = 1125) YCUs per core.

How Do We Come Up With The Actual YCU Ratings?

A number of organizations publish processor benchmarks. Here are a couple that I found with a quick google search http://www.cpubenchmark.net/ http://browser.primatelabs.com/processor-benchmarks.