YARN-1011. Schedule containers based on utilization of currently allocated containers.

Karthik Kambatla

December 24, 2015

1 Motivation

Yarn allocates resources based on the resource availability across nodes in the cluster. Today, a node's resource availability is calculated as its "capacity" (determined by *yarn.nodemanager.resource.**) less any resources allocated to containers on the node. A container's allocation is the upper limit of resources the container could use, and is based on the application's request.

In practice, applications are very conservative with resource requests leading to (potentially severe) cluster under-utilization. The estimates are conservative because it is hard to predict the exact amount of resources required for an application (job): (1) A job could have multiple tasks; each task's resource needs depend on the input it processes. (2) A job's/task's input (size, skew etc.) could change from run to run, and production jobs need to provision for peak load. (3) A task's usage varies over time.

2 Proposal

Like traditional operating systems, we could address this under-utilization by having the scheduler consider the actual utilizaiton on each of the nodes. One could over-subscribe the nodes, since most containers under-utilize the allocated resources. However, once the node is fully utilized, a container might need more resources upto its allocation adversely affecting other containers or its own execution.

In YARN-1011, we seek to implement basic over-subscription of nodes with proactive, graceful handling of variations in resource usage. At a high-level, we propose:

1. The notion of OPPORTUNISTIC containers that can soak up the un-utilized resources on a fully-allocated node. (YARN-2882)
2. Monitor container- and container-aggregate- utilization on the node. (YARN-3534 and YARN-1012)
3. Plumb individual nodes' aggregate-container utilization through to the scheduler. (YARN-3980)
4. Provide a way to turn on over-subsciption on a per-node basis. When turned on,
   (a) When the utilization goes over a configurable threshold, the NM should preempt enough OPPORTUNISTIC containers to bring the utilization under the threshold.
   (b) Launch opportunistic containers at a lower priority to address cases where monitor-and-preempt doesn't kick in soon enough. More details in Section **?**.
   (c) When a REGULAR container finishes, attempt/request to "promote" one or more of the OPPORTUNISTIC containers.
5. For heartbeats from nodes with over-subscription turned on, the scheduler takes the node utilization into consideration when allocating resources: fill the nodes upto their individual over-subscription limits with containers that are next in line to be scheduled.

3 Details

Think of this section as a constant work-in-progress. Will update this based on the discussions on the JIRA.

3.1 Pro-actively prioritizating REGULAR containers over OPPORTUNISTIC containers

Even though we monitor the node's usage and preempt OPPORTUNISTIC containers reactively, it is still possible REGULAR containers suddenly spike their usage potentially saturating the node and adversely affecting execution of some containers. When this happens, we need to ensure the REGULAR containers aren't the ones affected.

In the first version (YARN-1011), we assume CPU, disk, and network are malleable resources that can tolerate a minor delay in corrective actions. Memory, on the other hand, is not malleable and needs to be strictly prioritized. There are a couple of options here:

1. Required: Configure the OOM Killer priorities so the OPPORTUNISTIC containers are killed first when we run out of memory.
2. Potential optimization: YARN-1856 adds support for enforcing memory through cgroups. OPPORTUNISTIC containers could have memory.oom_control enabled (disallow the client/AM enabling) so they never go over their allocated memory limit and lead to killing other containers.

4 Out of scope (a.k.a Future work)

In the interesting of putting together a usable version at the earliest, this JIRA aims to implement the simplest version of over-subscription. The following items are out of scope and reserved for future work (in no order of priority):

1. Support for non-Linux operating systems.
2. Pro-active prioritization of REGULAR containers for CPU, disk and network.
3. Sophisticated policies to (1) pick applications more conducive to be run using OPPORTUNISTIC containers, (2) dynamic thresholds for the extent of over-subscription.