Container Pooling in YARN

This document proposes a method for reducing the container launch latency in YARN. It introduces a notion of pre-initialized containers that can be used to serve container requests for specific applications. The pre-initialized containers are application specific and can have some processes running and resources localized within them. YARN RM instructs the NMs to create pre-initialized containers and provides them the details for doing so. The advantage of having YARN RM manage the pre-initialized pool of containers for different applications is that we can make smarter choices about container placements and dynamically adjust the pool types and sizes based on the cluster utilization. Container pooling helps with interactive workloads where launch latencies matter a lot.

Lifecycle for a pre-initialized container

Creation Phase

YARN RM will have the configuration on how many pre-initialized containers to create and will pick nodes where these containers should be started. RM will provide the configuration (commands to run, resources to copy, resource constraints, etc) on how to pre-initialize the containers as part of YARN RM-NM heartbeat. YARN node manager will have a "Pool Manager" service to create these pre-initialized containers and once they are created then NM would advertise them as resource types.

Running Phase

As NM advertises the pre-initialized containers to the RM it will be assigned some of the container requests. When the container requests come then the pre-initialized container could be resized to match the resource allocation request and get attached to the job. What does it mean to attach a container would depend upon the container executor and we will be adding some APIs there to define that. As an e.g. if the pre-initialized container had created some process during container initialization then attaching the container might mean launching the container inside the Linux CGroup or Windows job object.

Release Phase

Once the container finishes execution then it would be released back to the pool and it will be available to execute some other job. The definition of release would depend upon the container executor and might mean to kill the container and have it recreated.

Design notes

- We could also use constraints instead of resource types to select nodes for allocation. It is unclear if the selection could be done by some pluggable component.
- The application master would need to ask for pooled containers by specifying the resource type in the allocate request.
- NM config will have the info to create the pooled containers, but in the future NM could receive it from other channels.