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.

Design

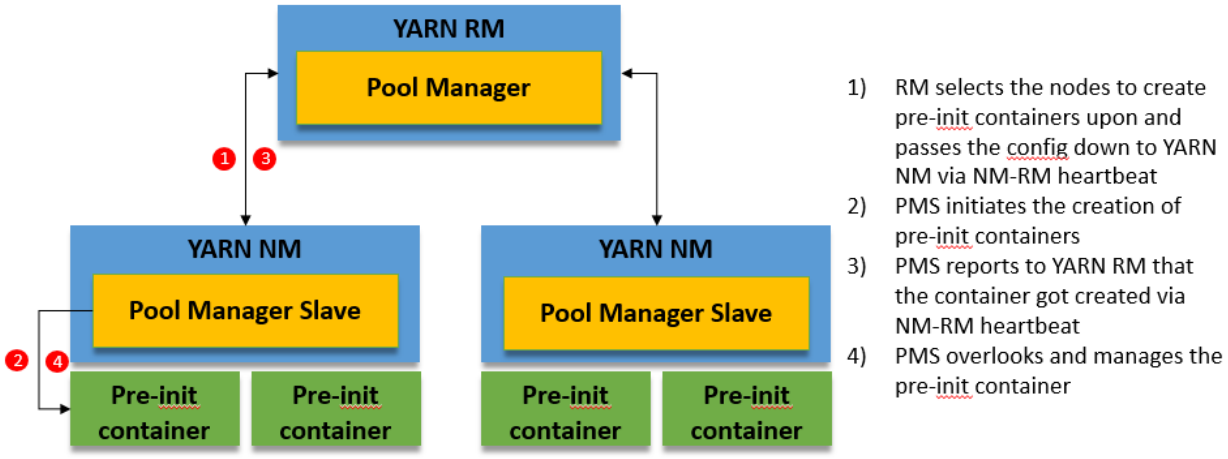
Creation Phase

We propose to add "Pool Manager Master" in YARN RM (PM) which receives the configuration to create a pool of pre-initialized containers in the cluster. To begin with this config will be provided by yarn-site.xml in a static manner. PM selects the nodes where container pooling is enabled and passes the config to start the the pre-initialized containers in the RM-NM heartbeat.

The config to start the pre-initialized container contains all the details to launch the container like environment, user, commands, resource constraints, resources to localize, and how many pre-initialized containers to create. In the future this config can be extended to include more details regarding placement of the pre-initialized container.

YARN NM will have an abstract service named "Pool Manager Slave", which will be responsible for managing the lifecycle of the pre-initialized container on the NM. Once NM gets the config to start the pre-initialized container then PMS will call into the ContainerManager to create and start the container. When the pre-initialized container is started then it moves into the PREINITIALIZED state (instead of RUNNING) and is reported to YARN RM in the NM-RM heartbeat. As PMS calls into the ContainerManager to start the pre-initialized container we follow the regular code paths for localizing, launching the container, and monitor it. PMS is responsible for managing the pre-initialized container through its lifecycle like creating a new one as a pre-initialized container gets used up or exits out due to an error.

The diagram below captures this interaction.



Running Phase

As YARN RM will be aware of where the pre-initialized containers are running it can schedule the applications that request for the pre-initialized container on the corresponding node. How this needs to be done is something to figure out and one approach is to use resource profiles (YARN-3926). This approach will allow us to advertise pre-initialized containers as resource types which makes it easier for ApplicationMaster to request for the pre-initialized containers, and for them to be scheduled.

When NM needs to use pre-initialized containers to satisfy a resource request then it “detaches” them from the pool and “attaches” them to the requesting YARN application. The detach is carried out by raising a CONTAINER_DETACH event, which is propagated down to the ContainerExecutor to do any platform specific operations, and it cleans up the YARN NM data structures to indicate that the pre-initialized container is now part of a new YARN application. A CONTAINER_ATTACH event associates the existing pre-initialized container to the requesting YARN AM. Specifically, each pre-initialized container has a container ID given to it and during the attach event we update the PID to container mapping in the ContainerExecutor to transparently associate the container ID of the existing pre-initialized container with that of the new container which was to be launched by YARN NM. Note that this container ID is obtained from launch request from the requesting AM, so the it would be the same as if a fresh container is launched for that request

As mentioned earlier, the container attach and detach events are propagated down to the ContainerExecutor, which allows us to customize what happens in these events. For example, as part of attach container we can launch some new process within the pre-initialized container.

Example use case

It might be helpful to describe how we used some of the above ideas in our internal PoC of Container Pooling.

One of our internal YARN application took a lot of time to launch the containers because it had to localize resources, run some process, wait for it to initialize, and then do job specific work. In our PoC with Container Pooling we pre-initialized the container with resources that were common across all jobs and launched the process that was required to be running a priori. Our YARN app was aware of container pooling and requested for a pre-initialized container by setting a specific environment variable. Our container launch commands/script checked whether it got a pre-initialized container or not in which case it would skip some of the steps and use the processes started a priori to carry out job specific tasks. Further in our scenario we adjusted the job object on the process to match with the resources that were allocated to container. This allowed us to launch the pre-initialized container with minimal resources while it was able to use more resources as it was attached to an application for doing some real work.