DECOUPLE INIT & DESTROY OF CONTAINERS FROM START& STOP

Arun Suresh, Konstantinos Karanasos, Sarvesh Sakalanaga

MOTIVATION

Introduce **initialize** and **destroy** container API into the **ContainerManagementProtocol** and decouple the actual start of a container from the initialization. This will allow AMs to re-start a container without having to lose the allocation.

Additionally, if the localization of the container is associated to the initialize (and the cleanup with the destroy), This can also be used by applications to upgrade a Container by **re-initializing** with a new **ContainerLaunchContext**

PROPOSAL

- Add 'initializeContainers' new API in ContainerManagementProtocol:
  - initializeContainers(InitContainersRequest) -> InitContainersResponse
    - InitContainersRequest = List<InitContainerRequest>
    - InitContainerRequest= {
      - ContainerLaunchContext containerLaunchContext
      - Token containerToken

      }
    - InitContainersResponse = {
      - Map<ContainerId, SerializedException> failedContainers
      - List<ContainerId> successfullContainers

      }
  - destroyContainers(List<ContainerId>) -> DestroyContainersResponse
    - DestroyContainersResponse = {
      - Map<ContainerId, SerializedException> failedContainers
      - List<ContainerId> successfullContainers

      }

- Modify 'StartContainerRequest' Record:
  - Need not have the ContainerLaunchContext.
  - If ContainerLaunchContext present, then treat request as Composite Command [initializeContiner + startContainer]
  - Add int field 'destroyDelay' to each 'StartContainerRequest':
    - Value = -1, if NM will wait for explicit 'destroyContainer' from the AM for the container to be deemed destroyed, and notified to RM (either Killed or Completed). Container will move to LOCALIZED state.
    - Value = 0, NM will not wait for explicit 'destroyContainer' from the AM. Default value to preserve backward compatibility

        - Value = <any int>, time in secs for NM to wait after Container Killed/Completed for the NM to notify the RM. The Container will move to LOCALIZED stated once completed/killed. The AM can then either call 'destroyContainer', or may call 'startContainer' again to restart the container during that time. AM may also call 'initializeContainer' with new ContainerLaunchContext to re-localize / upgrade the container bits prior to 'startContainer'

- Modify 'StopContainerRequest' Record:
    - Add boolean 'destroyContainer':
        - Value=True(default), destroys the container
        - Value=False, wait for either
            - explicit 'destroyContainer' call from AM
            - 'initializeContainer' from AM to re-initialize
            - 'startContainer' to restart the containers
            - Until 'destroyContainer' time has elapsed, and send Completed/Killed status to RM

IMPLEMENTATION DETAILS

- Introduce a new 'ContainerEventType.START_CONTAINER' event type.
- Introduce a new 'ContainerEventType.DESTROY_CONTAINER' event type.
- The Container remains in the LOCALIZED state until it receives the 'START_CONTAINER' event.
- For Backward compatibility, if the 'initializeContainer' API has not been called AND the 'startContainer' API includes a 'ContainerLaunchContext' then we still allow this event to be fired.
- The Container returns to 'LOCALIZED' state from 'EXITED_WITH_SUCCESS/FAILURE'.
- The Container waits in the 'LOCALIZED' state until 'DESTROY_CONTAINER' event is received. This happens either
    - When 'destroyDelay' timer expires (can be immediately)
    - Explicit 'destroy Container' is called by the AM
- Container cleanup is triggered only on DESTROY_CONTAINER.
- If 'initializeContainer' with a new ContainerLaunchContext is called by the AM while the Container is RUNNING, It is treated as a KILL_CONTAINER event followed by a CONTAINER_RESOURCE_CLEANUP and an INIT_CONTAINER event to kick of re-localization after which the Container will return to LOCALIZED state.
- If 'startContainer' with a new ContainerLaunchContext is called by the AM while the Container is RUNNING, it is treated exactly as the previous case, but is also followed by a START_CONTAINER event.
- If 'intializeContainer' is called WITHOUT a new ContainerLaunchContext by the AM, it is considered a restart, and will follow the same code path as 'initializeContainer' with new ContainerLaunchContext, but will not perform a CONTAINER_RESOURCE_CLEANUP and INIT_CONTAINER. The Container process will be killed and the container will be returned to LOCALIZED state.
- If 'startContainer' is called WITHOUT a new ContainerLaunchContext by the AM, it treated exactly as the above case, but it will also trigger a START_CONTAINER event.
- The NMStateStore has to store the InitContainerRequest and well as the 'destroyDelay' for a container.

ADDITIONAL CONSIDERATIONS

- It maybe cleaner to add an explicit READY_TO_START state for the Container, (The state at which the Container will wait at for a 'START_CONTAINER' event) but it looks like the LOCALIZED state serves the purpose.
- If, resource localization fails when an AM calls 'initializeContainer' or 'startContainer', then 'destroyDelay' is ignored and the Containers is immediately destroyed and notified to the RM.
- If NodeManager recovery is enabled, after NodeManager has restarted and if a Container which was earlier running is now killed, the NM consults the 'destroyDelay' value of the Container.
    - If Value = -1, It will wait for the AM to explicitly take action (getContainerStatus / startContainer / initializeContainer etc.)
    - Else, the value is ignored and RM is notified of completed container.