## Add support for dynamic resource updates with multiple resource types (YARN - 5592)

**Phase 1: Refreshing resource types configured in RM dynamically**

We can introduce a file named dymanic-resource-types.xml to refresh the new resource types dynamically.

**RMAdmin CLI:**

We can introduce an option something like “-refreshResourceTypes” to achieve this client side functionality similar to RMAdminCLI#refreshNodesResources->AdminService#refreshNodesResource->AdminService#updateNodeResource. On running this command, RM will load dymanic-resource-types.xml like any other config file and proceed accordingly.

While loading the file, we will need to handle below cases:

Lets say, RM has below 2 resource types currently:

resource1 and resource2

and dymanic-resource-types.xml has 2 resource types - resource1 and resource3

On comparing these new resource types with existing resource types, we will end up in having below cases:

1. Add new resource types (resource3)

It would as simple as adding those new resource types to the existing resource types array (with preserving order) available in ResourceUtils and ensure it is being displayed properly in RM Web UI with minimum allocation and maximum allocations.

2. Update existing resource types in RM (resource1)

On comparison, if there is any difference in Resource information (for ex, Units for resource ‘resource1’ is Gi at server side and new units from file is Mi), values at server side should be overridden with new values. Otherwise, no issues. In addition to updating resource type ‘resource1’ in existing resource types array available in ResourceUtils, We will need to fetch list of NM’s for which ‘resource1’ has been configured.

Approach A:

Traverse each node and do the following:

* Remove running containers only if ‘resource1’ has been used. These containers can be killed as “ABORTED” with proper diagnostic messages. Users will need to re-submit the jobs based on new resource requirements.
* Remove reservations, if any, only if ‘resource1’ has been used.
* Remove using NodeTracker#removeNode API
* Add node with new resource info using NodeTracker#addNode API
* Update cluster resource
* Update ResourceUtils#readOnlyNodeResources to ensure ‘resource1’ has been updated with new info (Yes, as of now, it is unmodifiable map).

Above update process will require new event types in SchedulerEventType etc.

Approach B:

Traverse each node and do the following:

* Remove using NodeTracker#removeNode API
* Add node with new resource info using NodeTracker#addNode API. New resource info is converted value based on new unit.
* Update cluster resource
* All running containers and reservations using ‘resource1’ can be converted to new units updated recently. Similar to UpdateContainerRequest flow, we can kill the current container as “ABORTED” with proper diagnostic message and release a new container based on new resource info and attach it to the task.
* Update ResourceUtils#readOnlyNodeResources to ensure ‘resource1’ has been updated with new info (Yes, as of now, it is unmodifiable map).

Though cluster resource gets updated as and when each node process over in both the approaches, it will be in inconsistent state till all nodes completes the above steps. Is it ok?

Overall, Approach A looks simpler to implement when compared to Approach B but users (App Submitters) don’t even realise the change.

3. Delete existing resource types in RM (resource2)

In addition to removing resource type ‘resource2’ from existing resource types array available in ResourceUtils, We will need to handle lot of cases as described below:

First, we will need to fetch list of NM’s for which ‘resource2’ has been configured. For each node, Similar to CapacityScheduler#removeNode and as described in above Approach A for Update operation, We can remove running containers, reservation and update node resources etc

* Remove running containers only if ‘resource2’ has been used. These containers can be marked with “ABORTED” state with proper diagnostic messages. Users will need to re-submit the jobs based on new resource requirements.
* Remove reservations, if any, only if ‘resource2’ has been used.
* Remove using NodeTracker#removeNode API
* Add node with new resource info using NodeTracker#addNode API
* Update cluster resource
* Update ResourceUtils#readOnlyNodeResources to ensure ‘resource2’ has been removed (Yes, as of now, it is unmodifiable map).

Both UPDATE and DELETE will require new event types in SchedulerEventType etc

**RM’s restarts:**

During RM restarts or even when starting RM for first time, this file can be ignored. In this case, it should not be picked up for processing. In both the cases, like current flow, it continues to look for resource-types.xml only for further process.

**Open Questions:**

1. How are we going to handle multiple RM’s?

As new option in RMAdminCLI would be executed in RM, we can ensure that is allowed to run only on Active RM. Or Do we need to understand the effect of running this command in StandBy RM?

2. When Standby become Active, What will happen to these resource types configuration?

**Phase 2: Refreshing resource types configured in NM dynamically**

To do later. Need to think about it.