# RDMA Controller

## 1. Overview

### 1-1. What is RDMA controller?

RDMA controller allows user to limit RDMA/IB specific resources that a given set of processes can use. These processes are grouped using RDMA controller.

RDMA controller defines two resources which can be limited for processes of a cgroup.

### 1-2. Why RDMA controller needed?

Currently user space applications can easily take away all the rdma verb specific resources such as AH, CQ, QP, MR etc. Due to which other applications in other cgroup or kernel space ULPs may not even get chance to allocate any rdma resources. This can lead to service unavailability.

Therefore RDMA controller is needed through which resource consumption of processes can be limited. Through this controller different rdma resources can be accounted.

### 1-3. How is RDMA controller implemented?

RDMA cgroup allows limit configuration of resources. Rdma cgroup maintains resource accounting per cgroup, per device using resource pool structure. Each such resource pool is limited up to 64 resources in given resource pool by rdma cgroup, which can be extended later if required.

This resource pool object is linked to the cgroup css. Typically there are 0 to 4 resource pool instances per cgroup, per device in most use cases. But nothing limits to have it more. At present hundreds of RDMA devices per single cgroup may not be handled optimally, however there is no known use case or requirement for such configuration either.

Since RDMA resources can be allocated from any process and can be freed by any of the child processes which shares the address space, rdma resources are always owned by the creator cgroup css. This allows process migration from one to other cgroup without major complexity of transferring resource ownership; because such ownership is not really present due to shared nature of rdma resources. Linking resources around css also ensures that cgroups can be deleted after processes migrated. This allow progress migration as well with active resources, even though that is not a primary use case.

Whenever RDMA resource charging occurs, owner rdma cgroup is returned to the caller. Same rdma cgroup should be passed while uncharging the resource. This also allows process migrated with active RDMA resource to charge to new owner cgroup for new resource. It also allows to uncharge resource of a process from previously charged cgroup which is migrated to new cgroup, even though that is not a primary use case.

Resource pool object is created in following situations. (a) User sets the limit and no previous resource pool exist for the device of interest for the cgroup. (b) No resource limits were configured, but IB/RDMA stack tries to charge the resource. So that it correctly uncharge them when applications are running without limits and later on when limits are enforced during uncharging, otherwise usage count will drop to negative.

Resource pool is destroyed if all the resource limits are set to max and it is the last resource getting deallocated.

User should set all the limit to max value if it intents to remove/unconfigure the resource pool for a particular device.

IB stack honors limits enforced by the rdma controller. When application query about maximum resource limits of IB device, it returns minimum of what is configured by user for a given cgroup and what is supported by IB device.

Following resources can be accounted by rdma controller.

| | |
|---|---|
| hca_handle | Maximum number of HCA Handles |
| hca_object | Maximum number of HCA Objects |

## 2. Usage Examples

a. Configure resource limit:

```
echo mlx4_0 hca_handle=2 hca_object=2000 > /sys/fs/cgroup/rdma/1/rdma.max
echo ocrdma1 hca_handle=3 > /sys/fs/cgroup/rdma/2/rdma.max
```

b. Query resource limit:

```
cat /sys/fs/cgroup/rdma/2/rdma.max
#Output:
mlx4_0 hca_handle=2 hca_object=2000
ocrdma1 hca_handle=3 hca_object=max
```

c. Query current usage:

```
cat /sys/fs/cgroup/rdma/2/rdma.current
#Output:
mlx4_0 hca_handle=1 hca_object=20
ocrdma1 hca_handle=1 hca_object=23
```

d. Delete resource limit:

```
echo mlx4_0 hca_handle=max hca_object=max > /sys/fs/cgroup/rdma/1/rdma.max
```