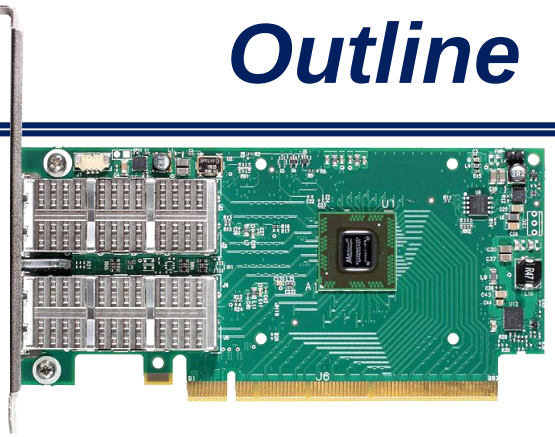
Quick Start Guide

Federico Silla and Carlos Reaño
Technical University of Valencia
Spain

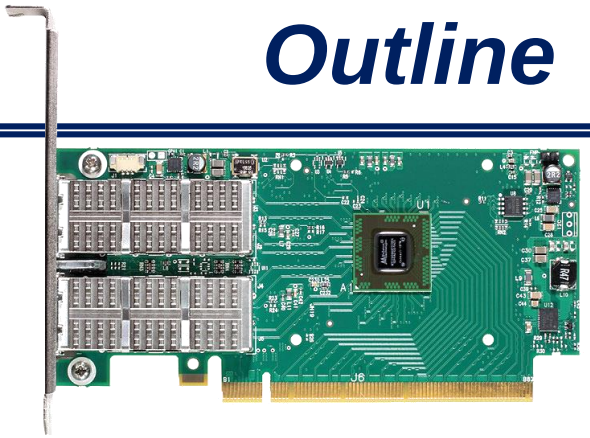
May 2017

Outline



- What is rCUDA?
- Prerequisites before installing rCUDA
- Installing and using rCUDA
- Multi-GPU scenario

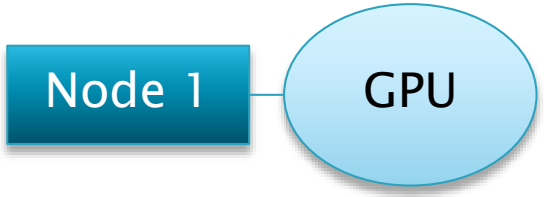
Outline



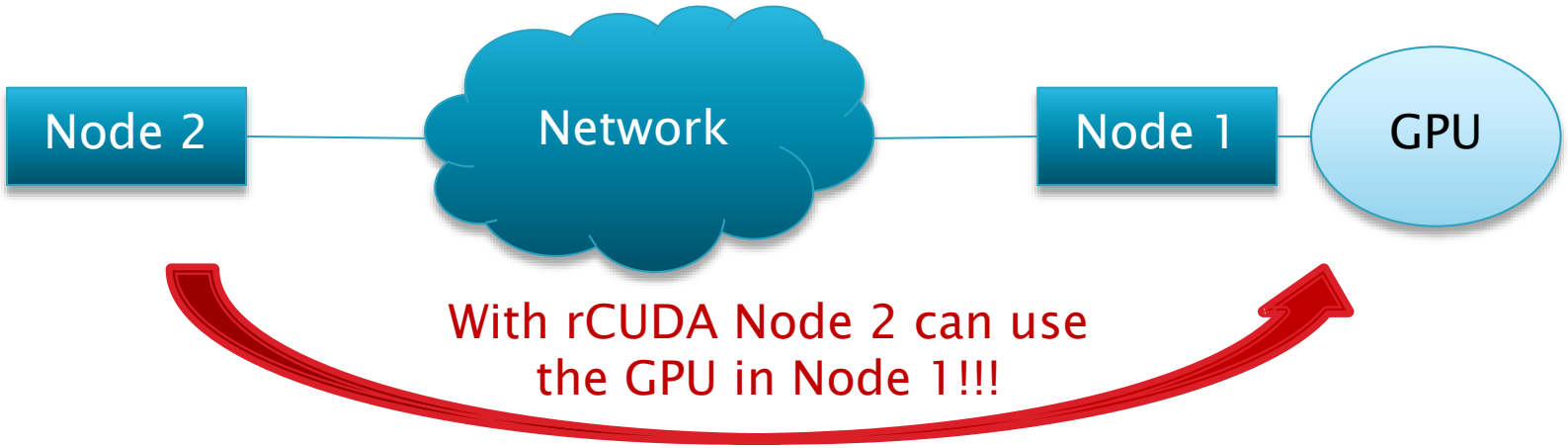
- **What is rCUDA?**
- Prerequisites before installing rCUDA
- Installing and using rCUDA
- Multi-GPU scenario

What is rCUDA?

- CUDA:

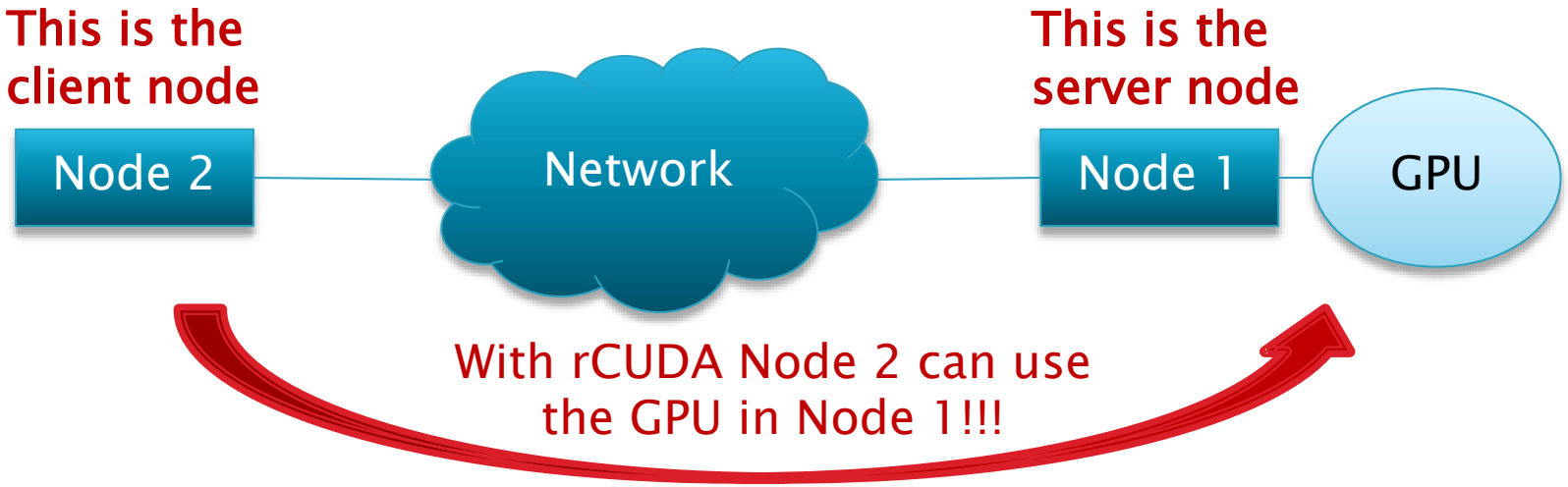


- rCUDA (remote CUDA):

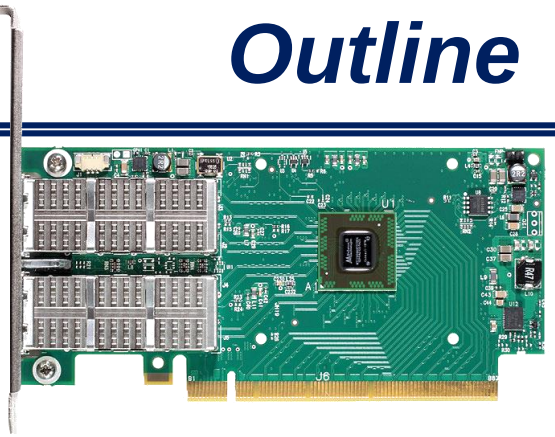


rCUDA features a client-server distributed architecture

- rCUDA (remote CUDA):



Outline



- What is rCUDA?
- **Prerequisites before installing rCUDA**
- Installing and using rCUDA
- Multi-GPU scenario

Prerequisites before installing rCUDA

- In order to use rCUDA, you have to make sure that:
  1. CUDA is running in the server node
  2. Communications are properly working between client and server nodes

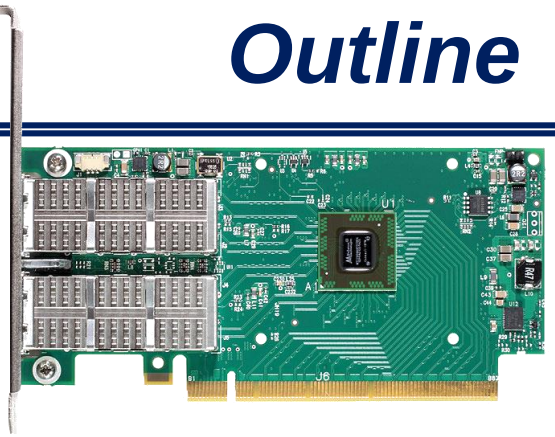
1. In order to have CUDA working in the server node, please consult the instructions and directions by NVIDIA
2. In order to properly set communication between client and server nodes:
   a) You can use TCP/IP based communications (Ethernet, for instance)
   b) You can use RDMA-based communications (InfiniBand or RoCE). In this case please refer to the directions provided by Mellanox

Prerequisites before installing rCUDA

- How to check that everything is OK:
  1. CUDA is running in the server node
  2. Communications are properly working between client and server nodes

1. **CUDA:** In the server node, use CUDA to execute the **deviceQuery** and **bandwidthTest** samples included in the CUDA distribution
2. **RDMA:** With InfiniBand (IB) or RoCE use the **ib_write_bw** and **ib_read_bw** tests included in the Mellanox OFED

Outline



- What is rCUDA?
- Prerequisites before installing rCUDA
- **Installing and using rCUDA**
- Multi-GPU scenario

Installing and using rCUDA

- Where to obtain rCUDA?
  - www.rCUDA.net: Software Request Form
- Package contents. Important folders:
  - doc: rCUDA user guide
  - bin: rCUDA server daemon
  - lib: rCUDA library
- Installing rCUDA
  - Just untar the tarball in both the server and the client(s) node(s)

Installing and using rCUDA

- Starting rCUDA server:
  - Set env. vars as if you were going to run a CUDA program:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

  - Start rCUDA server:

```
cd $HOME/rCUDA/bin
./rCUDAd
```

Installing and using rCUDA

- Starting rCUDA server:
  - Set env. vars as if you were going to run a CUDA program:


```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

  - Start rCUDA server:

```
cd $HOME/rCUDA/bin
./rCUDAd
```

Installing and using rCUDA

- Starting rCUDA server:
  - Set env. vars as if you were going to run a CUDA program:

Path to CUDA libraries

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

  - Start rCUDA server:

```
cd $HOME/rCUDA/bin
./rCUDAd
```

Installing and using rCUDA

- Starting rCUDA server:
  - Set env. vars as if you were going to run a CUDA program:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

  - Start rCUDA server:


```
cd $HOME/rCUDA/bin
./rCUDAd
```

Installing and using rCUDA

- Starting rCUDA server:
  - Set env. vars as if you were going to run a CUDA program:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

  - Start rCUDA server:

```
cd $HOME/rCUDA/bin
./rCUDAd
```

Start rCUDA server in background

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:


```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:


```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:

```
export PATH=$PATH:/usr/local/cuda/bi
export LD_LIBRARY_PATH=$HOME/rC      /lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

Number of remote GPUs: 1, 2, 3...

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows: Name/IP of rCUDA server

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

GPU of remote server to use

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

Very important!!!

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```

Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:

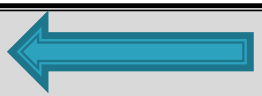
```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=1
export RCUDA_DEVICE_0=<server_name_or_ip_address>:0
```

  - In the client node, compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

  - In the client node, run the CUDA program as usual:

```
./deviceQuery
...
```



Installing and using rCUDA

- Testing the installation of rCUDA:
  - deviceQuery
  - bandwidthTest

Set the env. variables as in the previous slides. These samples should work

Installing and using rCUDA

- Testing the installation of rCUDA:
  - deviceQuery
  - bandwidthTest

- Problem: bandwidth with rCUDA is too low!!
  - Why? We are using TCP. rCUDA by default uses TCP/IP. If your network supports RDMA (InfiniBand or RoCE), then you have to use an additional env. variable **both in the client and in the server nodes**

rCUDA over HPC networks: InfiniBand

- Starting rCUDA server using IB:

```
export RCUDAPROTO=IB
cd $HOME/rCUDA/bin
./rCUDAd
```

- Run CUDA program using rCUDA over IB:

```
export RCUDAPROTO=IB
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/bandwidthTest
./bandwidthTest
```

rCUDA over HPC networks: InfiniBand

- Starting rCUDA server using IB:

Tell rCUDA we want to use IB

```
export RCUDAPROTO=IB
cd $HOME/rCUDA/bin
./rCUDAd
```

- Run CUDA program using rCUDA over IB:

Also in the client!!

```
export RCUDAPROTO=IB
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/bandwidthTest
./bandwidthTest
```

Do not forget to set the other env. variables required by rCUDA!!!

rCUDA over HPC networks: InfiniBand

- Starting rCUDA server using IB:

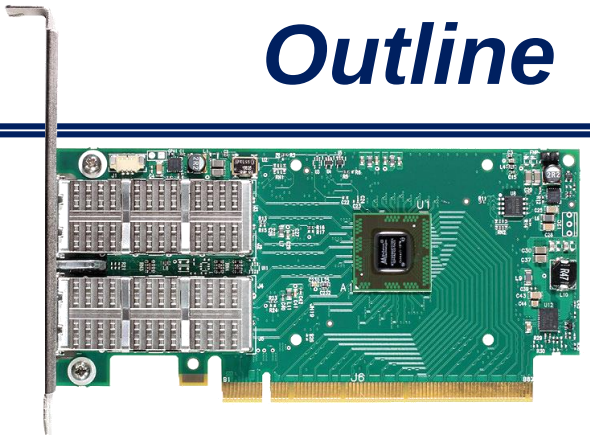
```
export RCUDAPROTO=IB
cd $HOME/rCUDA/bin
./rCUDAd
```

- Run CUDA program using rCUDA over IB:

```
export RCUDAPROTO=IB
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/bandwidthTest
./bandwidthTest
```

- Testing bandwidth:
  - bandwidthTest using IB
  - Has bandwidth improved?

Outline



- What is rCUDA?
- Prerequisites before installing rCUDA
- Installing and using rCUDA
- **Multi-GPU scenario**

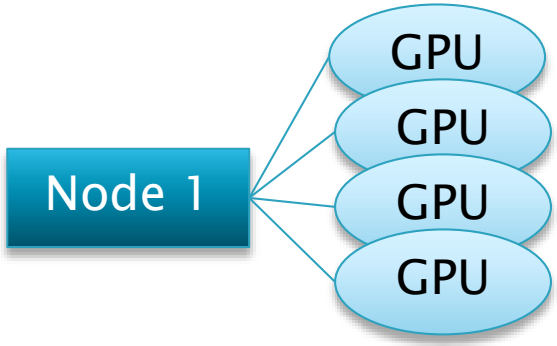
Multi-GPU scenario

- **Scalable applications**: more GPUs, less execution time
  - rCUDA can use all the GPUs in the cluster, whereas CUDA only can use the ones installed inside one node: for some applications, rCUDA can get better results than CUDA

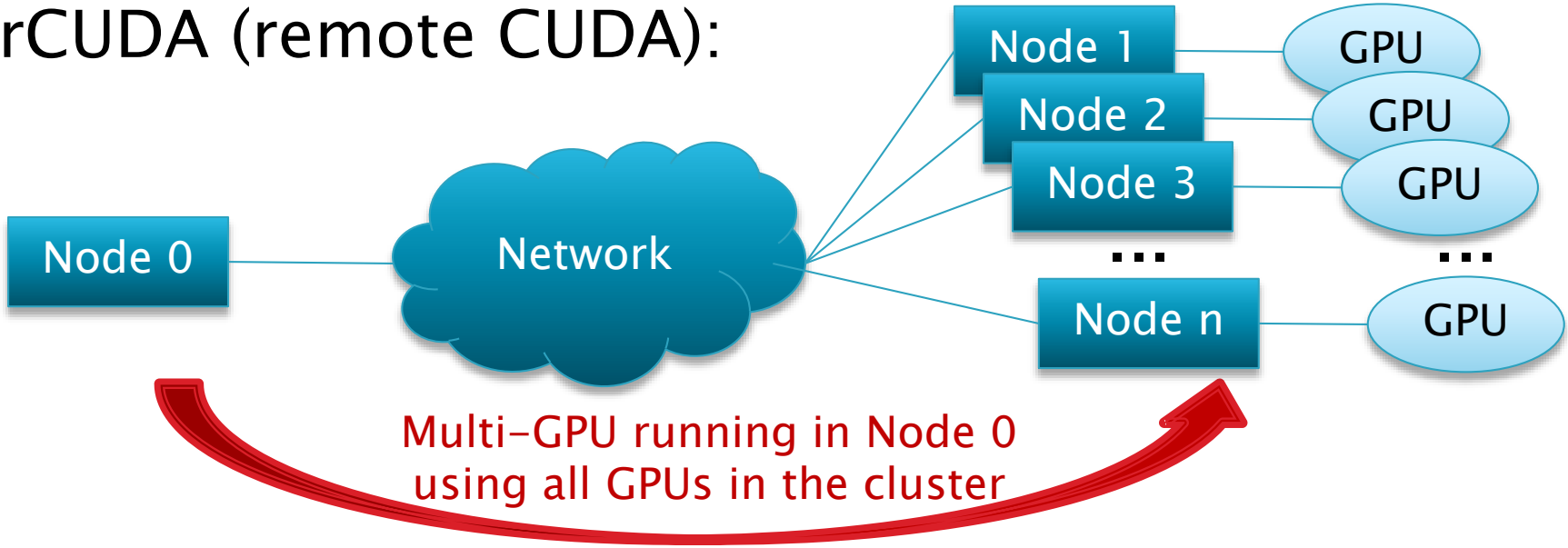
Multi-GPU scenario

- CUDA:

Multi-GPU App running in Node 1 using its 4 GPUs



- rCUDA (remote CUDA):



Multi-GPU Configuration

- Configure rCUDA for Multi-GPU:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDAl:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=5
export RCUDA_DEVICE_0=node1:0
export RCUDA_DEVICE_1=node1:1
export RCUDA_DEVICE_2=node2:0
export RCUDA_DEVICE_3=node3:0
export RCUDA_DEVICE_4=node4:0
```

  - Check configuration by running deviceQuery sample

Multi-GPU Configuration

- Configure rCUDA for Multi-GPU:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDAl:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=5
export RCUDA_DEVICE_0=node1:0
export RCUDA_DEVICE_1=node1:1
export RCUDA_DEVICE_2=node2:0
export RCUDA_DEVICE_3=node3:0
export RCUDA_DEVICE_4=node4:0
```

Number of remote GPUs

  - Check configuration by running deviceQuery sample

Multi-GPU Configuration

- Configure rCUDA for Multi-GPU:

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDAl:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=5
export RCUDA_DEVICE_0=node1:0
export RCUDA_DEVICE_1=node1:1
export RCUDA_DEVICE_2=node2:0
export RCUDA_DEVICE_3=node3:0
export RCUDA_DEVICE_4=node4:0
```

Location of each GPU

  - Check configuration by running deviceQuery sample

Get a free copy of rCUDA at

http://www.rcuda.net

info@rcuda.net

@rcuda_r