Quick Start Guide

rCUDA Team

Universitat Politècnica de València
Spain

October 2018
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):**

With rCUDA Node 2 can use the GPU in Node 1!!!
What is rCUDA?

rCUDA features a client–server distributed architecture

- rCUDA (remote CUDA):
  - This is the client node
    - Node 2
  - Network
  - This is the server node
    - Node 1
  - GPU

With rCUDA Node 2 can use the GPU in Node 1!!!
Outline

- What is rCUDA?
- Prerequisites before installing rCUDA
- Installing and using rCUDA
- Multi-GPU scenario
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
Where to obtain rCUDA?
- www.rCUDA.net: Software Request Form

Package contents. Important folders:
- doc: rCUDA user guide
  - rCUDA Quick Start Guide
- bin: rCUDA server daemon
  - rCUDA license server daemon
- config: rCUDA license configuration file
- lib: rCUDA library

Installing rCUDA
- Just untar the tarball in both the server and the client(s) node(s)
Installing the rCUDA license server

- Include in the config/rCUDA.conf file two new lines:
  - The license key
  - The IP address of the node running the rCUDA license server

```bash
cd $HOME/rCUDA/config

Add the following two lines to the config/rCUDA.conf file
RCUDA_LICENSE_KEY=3232145765453730306042551...
RCUDA_LICENSE_SERVER=192.168.0.34
```

- Execute the rCUDA license server in the bin directory with `-iv` options:

```bash
cd $HOME/rCUDA/bin
./rCUDAls -iv
```
You should get an output similar to:

```
rcUDAlics v1.0
Copyright 2009-2018 UNIVERSITAT POLITECNICA DE VALENCIA. All rights reserved.
rcUDAlics[16812]: License Features:
- Expiration date: 31 Dec 2019
- Max rcUDA server(s): 10
- Max remote GPU(s): 50
- Max virtual GPU(s): 1 per remote GPU
- Allow Migration: no
rcUDAlics[16812]: License server daemon successfully started.
```

Once you have checked that the rcUDA license server is properly running, you can stop using the –iv options
Include in the config/rCUDA.conf file two new lines:

- The license key
- The IP address of the node running the rCUDA license server

```
cd $HOME/rCUDA/config

Add the following two lines to the config/rCUDA.conf file
RCUDA_LICENSE_KEY=3232145765453730306042551...
RCUDA_LICENSE_SERVER=192.168.0.34
```
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 with `–iv` options:

  ```
  cd $HOME/rCUDA/bin
  ./rCUDAd –iv
  ```
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 with `-iv` options:
    ```
    cd $HOME/rCUDA/bin
    ./rCUDAd -iv
    ```
Starting rCUDA server:

- Set env. vars as if you were going to run a CUDA program:
  
  ```bash
  export PATH=$PATH:/usr/local/cuda/bin
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
  ```

- Start rCUDA server with `-iv` options:
  
  ```bash
  cd $HOME/rCUDA/bin
  ./rCUDAd -iv
  ```
Installing and using rCUDA

- Starting rCUDA server:
  - Set env. vars as if you were going to run a CUDA program:
    ```bash
    export PATH=$PATH:/usr/local/cuda/bin
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
    ```
  - Start rCUDA server with -iv options:
    ```bash
    cd $HOME/rCUDA/bin
    ./rCUDA -iv
    ```
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 with `-iv` options:

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

Start rCUDA server in verbose and interactive mode

(once you have checked that rCUDA is working in your cluster, you can stop using the `-iv` options)
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
  ```
Running a CUDA program with rCUDA (client node):

- In the client node, set env. vars as follows:
  
  ```bash
  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:
  
  ```bash
  cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
  make EXTRA_NVCCFLAGS=--cudart=shared
  ```

- In the client node, run the CUDA program as usual:
  
  ```bash
  ./deviceQuery
  ```
- Running a CUDA program with rCUDA (client node):
  - In the client node, set env. vars as follows:
    ```
    export 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
    ```
  - Path to rCUDA library
  - In the client node, compile CUDA program using dynamic libraries:
    ```
    cd $HOME/NVIDIA_CUDA_Samples/1Utilities/deviceQuery
    make EXTRA_NVCCFLAGS=--cudart=shared
    ```
  - In the client node, run the CUDA program as usual:
    ```
    ./deviceQuery
    ...
    ```
Running a CUDA program with rCUDA (client node):

- In the client node, set env. vars as follows:
  
  ```bash
  export PATH=$PATH:/usr/local/cuda/bin
  export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
  export RCUDA_DEVICE_COUNT=1
  export RCUDADEVICE_0=<server_name_or_ip_address>:0
  ```

- In the client node, compile CUDA program using dynamic libraries:
  
  ```bash
  cd $HOME/NVIDIA_CUDA_Samples/1Utilities/deviceQuery
  make EXTRA_NVCCFLAGS=--cudart=shared
  ```

- In the client node, run the CUDA program as usual:
  
  ```bash
  ./deviceQuery
  ```
Installing and using rCUDA

- Running a CUDA program with rCUDA (client node):
  - In the client node, compile CUDA program using dynamic libraries:
    ```
    cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
    make EXTRA_NVCCFLAGS=-cudart=shared
    ```
  - In the client node, set env. vars as follows:
    ```
    export PATH=$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, run the CUDA program as usual:
    ```
    ./deviceQuery
    ...
    ```
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
  ```
Running a CUDA program with rCUDA (client node):

- In the client node, set env. vars as follows:

  ```bash
  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:

  ```bash
  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:

  ```bash
  ./deviceQuery
  ...
  ```
Running a CUDA program with rCUDA (client node):

- In the client node, set env. vars as follows:

```bash
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:

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

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

```
./deviceQuery
...```
Testing the installation of rCUDA:

- `deviceQuery`
- `bandwidthTest`

Set the env. variables as in the previous slides. These samples should work.
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 RCUDA_NETWORK=IB
  cd $HOME/rCUDA/bin
  ./rCUDA
  ```

- Run CUDA program using rCUDA over IB:
  ```
  export RCUDA_NETWORK=IB
  cd $HOME/NVIDIA_CUDA_Samples/1Utilities/bandwidthTest
  ./bandwidthTest
  ```
rCUDA over HPC networks: InfiniBand

- Starting rCUDA server using IB:
  ```
  export RCUDA_NETWORK=IB
  cd $HOME/rCUDA/bin
  ./rCUDAd
  ```
  **Tell rCUDA we want to use IB**

- Run CUDA program using rCUDA over IB:
  ```
  export RCUDA_NETWORK=IB
  cd $HOME/NVIDIA_CUDA_Samples/1_Utils/bandwidthTest
  ./bandwidthTest
  ```
  **Also in the client!!**

Do not forget to set the other env. variables required by rCUDA!!!
rCUDA over HPC networks: InfiniBand

- Starting rCUDA server using IB:
  ```
  export RCUDA_NETWORK=IB
  cd $HOME/rCUDA/bin
  ./rCUDA
  ```

- Run CUDA program using rCUDA over IB:
  ```
  export RCUDA_NETWORK=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
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
CUDA:

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

rCUDA (remote CUDA):

Multi-GPU running in Node 0 using all GPUs in the cluster
In the client node, configure rCUDA for Multi-GPU:

- `export PATH=$PATH:/usr/local/cuda/bin`
- `export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDA1:$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
In the client node, configure rCUDA for Multi-GPU:

```bash
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDA1:$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
In the client node, configure rCUDA for Multi-GPU:

- `export PATH=$PATH:/usr/local/cuda/bin`
- `export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDA1:$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
Get a free copy of rCUDA at
http://www.rcuda.net

info@rcuda.net
@rcuda_