Using GPU Virtualization with TensorFlow

Carlos Reaño
Universitat Politècnica de València, Spain
http://mural.uv.es/caregon

HPC Advisory Council Swiss Conference 2018
April 9-12, 2018, Lugano, Switzerland
- What is rCUDA?
- Installing and using rCUDA
- rCUDA over HPC networks
  - InfiniBand
- How taking benefit from rCUDA
  - Sample scenarios
- Questions & Answers
Outline

- What is rCUDA?
- Installing and using rCUDA
- rCUDA over HPC networks
  - InfiniBand
- How taking benefit from rCUDA
  - Sample scenarios
- Questions & Answers
What is rCUDA?

CUDA:

rCUDA (remote CUDA):

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

- What is rCUDA?
- Installing and using rCUDA
- rCUDA over HPC networks
  - InfiniBand
- How taking benefit from rCUDA
  - Sample scenarios
- Questions & Answers
Where obtain rCUDA?

- www.rCUDA.net: Software Request Form

Package contents. Important folders:

- doc: rCUDA user’s guide & quick start guide
- bin: rCUDA server daemon
- lib: rCUDA library

Installing rCUDA

- Just untar the tarball in both the server(s) 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:
  
  ```
  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
    ```
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
Running a CUDA program with rCUDA:

- 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
  ```

- Compile CUDA program using dynamic libraries:

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

- Run the CUDA program as usual:

  ```
  ./deviceQuery
  ```
Running a CUDA program with rCUDA:

- 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
  ```

- Compile CUDA program using dynamic libraries:

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

- Run the CUDA program as usual:

  ```
  ./deviceQuery
  ...
Installing and using rCUDA

- Running a CUDA program with rCUDA:
  - 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
    ```
  - Compile CUDA program using dynamic libraries:
    ```
    cd $HOME/NVIDIA_CUDA_Samples/1Utilities/deviceQuery
    make EXTRA_NVCCFLAGS=--cudart=shared
    ```
  - Run the CUDA program as usual:
    ```
    ./deviceQuery
    ...
Running a CUDA program with rCUDA:

- 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
  ```

- Compile CUDA program using dynamic libraries:
  ```
  cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
  make EXTRA_NVCCFLAGS=--cudart=shared
  ```

- Run the CUDA program as usual:
  ```
  ./deviceQuery
  ```
Running a CUDA program with rCUDA:

- 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 RCUDADEVICE_0=<server_name_or_ip_address>:0
  ```

- Compile CUDA program using dynamic libraries:
  ```
  cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
  make EXTRA_NVCCFLAGS=--cudart=shared
  ```

- Run the CUDA program as usual:
  ```
  ./deviceQuery
  ```
Installing and using rCUDA

- Running a CUDA program with rCUDA:
  - 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
  - Compile CUDA program using dynamic libraries:
    ```
    cd $HOME/NVIDIA_CUDA_Samples/1Utilities/deviceQuery
    make EXTRA_NVCCFLAGS=--cudart=shared
    ```
  - Run the CUDA program as usual:
    ```
    ./deviceQuery
    ...
    ```
Installing and using rCUDA

- Running a CUDA program with rCUDA:
  - 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 RCUDADEVICE_0=<server_name_or_ip_address>:0
    ```
  - Compile CUDA program using dynamic libraries:
    ```
    cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
    make EXTRA_NVCCFLAGS=--cudart=shared
    ```
  - Run the CUDA program as usual:
    ```
    ./deviceQuery
    ...  
    ```

Very important!!!
Running a CUDA program with rCUDA:

- Set env. vars as follows:
  ```
  export PATH=$PATH:/usr/local/cuda/bin
  export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
  export RCUDADEVICECOUNT=1
  export RCUDADEVICE0=<server_name_or_ip_address>:0
  ```

- Compile CUDA program using dynamic libraries:
  ```
  cd $HOME/NVIDIA_CUDA_Samples/1Utilities/deviceQuery
  make EXTRA_NVCCFLAGS=--cudart=shared
  ```

- Run the CUDA program as usual:
  ```
  ./.deviceQuery
  ```
Live demonstration:
- deviceQuery
- bandwidthTest
Live demonstration:
- deviceQuery
- bandwidthTest

Problem: bandwidth with rCUDA is too low!!
- Why? We are using TCP
Installing and using rCUDA

- Live demonstration:
  - deviceQuery
  - bandwidthTest

- Problem: bandwidth with rCUDA is too low!!
  - Why? We are using TCP

- Solution: HPC networks
  - InfiniBand (IB)
Outline

- What is rCUDA?
- Installing and using rCUDA
- rCUDA over HPC networks
  - InfiniBand
- How taking benefit from rCUDA
  - Sample scenarios
- Questions & Answers
rCUDA over HPC networks: InfiniBand

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

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

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

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

- Live demonstration:
  - bandwidthTest using IB
  - Bandwidth is no more a problem!!
Outline

- What is rCUDA?
- Installing and using rCUDA
- rCUDA over HPC networks
  - InfiniBand
- How taking benefit from rCUDA
  - Sample scenarios
- Questions & Answers
Sample scenarios:

- **Typical behavior of CUDA applications**: moving data to the GPU and performing a lot of computations there to compensate the overhead of having moved the data
  - This benefits rCUDA: more computations, less rCUDA overhead

- **Scalable applications**: more GPUs, less execution time
  - rCUDA can use all the GPUs of the cluster, while CUDA only can use the ones directly connected to one node: for some applications, rCUDA can get better results than with CUDA

- **Heterogeneous clusters**: access to GPU servers from ATOM, ARM...
  - rCUDA can be used to access GPU servers in x86 or Power8 machines, from different systems and architectures (ATOM, ARM, Intel–D...)
Three main types of applications:

- Bandwidth bounded: more transfers, more rCUDA overhead
- Computations bounded: more computations, less rCUDA overhead
- Intermediate
GPU vs. remote GPU
  - Overhead of remote GPUs?

Live demonstration:
  - Tensorflow with CUDA
  - Tensorflow with rCUDA
Tensorflow

- CPU vs. remote GPU
  - What is better: a local CPU or a remote GPU?
- Live demonstration:
  - Tensorflow on CPU (without CUDA)
Sample scenarios:

- **Typical behavior of CUDA applications:** moving data to the GPU and performing a lot of computations there to compensate the overhead of having moved the data
  - This benefits rCUDA: more computations, less rCUDA overhead

- **Scalable applications:** more GPUs, less execution time
  - rCUDA can use all the GPUs of the cluster, while CUDA only can use the ones directly connected to one node: for some applications, rCUDA can get better results than with CUDA

- **Heterogeneous clusters:** access to GPU servers from ATOM, ARM...
  - rCUDA can be used to access GPU servers in x86 or Power8 machines, from different systems and architectures (ATOM, ARM, Intel-D...)

---

**How taking benefit from rCUDA**

- **Sample scenarios:**
  - Typical behavior of CUDA applications: moving data to the GPU and performing a lot of computations there to compensate the overhead of having moved the data
    - This benefits rCUDA: more computations, less rCUDA overhead
  - Scalable applications: more GPUs, less execution time
    - rCUDA can use all the GPUs of the cluster, while CUDA only can use the ones directly connected to one node: for some applications, rCUDA can get better results than with CUDA
  - Heterogeneous clusters: access to GPU servers from ATOM, ARM...
    - rCUDA can be used to access GPU servers in x86 or Power8 machines, from different systems and architectures (ATOM, ARM, Intel-D...)
CUDA:

Multi-GPU App running in Node 1 using all the GPUs in the node

rCUDA (remote CUDA):

Multi-GPU running in Node 0 using all GPUs in the cluster
Configure rCUDA for Multi-GPU:

```bash
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDA:$LD_LIBRARY_PATH
export RCUDA_DEVICE_COUNT=5
export RCUDA_DEVICE_0=node1:0
export RCUDADEVICE_1=node1:1
export RCUDA_DEVICE_2=node2:0
export RCUDADEVICE_3=node3:0
export RCUDA_DEVICE_4=node4:0
```

- Check configuration by running deviceQuery sample
Configure rCUDA for Multi-GPU:

- Export PATH:
  ```bash
  export PATH=$PATH:/usr/local/cuda/bin
  ```
- Export LD_LIBRARY_PATH:
  ```bash
  export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDA1:$LD_LIBRARY_PATH
  ```
- Export RCUDA_DEVICE_COUNT:
  ```bash
  export RCUDADEVICE_COUNT=5
  ```
- Export RCUDA_DEVICE:
  ```bash
  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
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
Multi-GPU Tensorflow

- Live demonstration:
  - deviceQuery sample with multiple GPUs
  - Multi-GPU Tensorflow
How taking benefit from rCUDA

Sample scenarios:

- Typical behavior of CUDA applications: moving data to the GPU and performing a lot of computations there to compensate the overhead of having moved the data
  - This benefits rCUDA: more computations, less rCUDA overhead

- Scalable applications: more GPUs, less execution time
  - rCUDA can use all the GPUs of the cluster, while CUDA only can use the ones directly connected to one node: for some applications, rCUDA can get better results than with CUDA

- Heterogeneous clusters: access to GPU servers from ATOM, ARM...
  - rCUDA can be used to access GPU servers in x86 or Power8 machines, from different systems and architectures (ATOM, ARM, Intel-D...)

HPCAC Swiss Conference 2018
How taking benefit from rCUDA

- Heterogeneous clusters:
  - Access from low power nodes (Atom, ARM, Intel D...) to x86 GPU accelerated nodes
  - Access from no-Power8 nodes to Power8 GPU accelerated nodes
● What is rCUDA?
● Installing and using rCUDA
● rCUDA over HPC networks
  ● InfiniBand
● How taking benefit from rCUDA
  ● Sample scenarios
● Questions & Answers