Reducing Power Consumption of Data Centers with rCUDA

Federico Silla & Carlos Reaño
Technical University of Valencia
Spain
14.00 - 15.30 SESSION 1
- Presentation of remote GPU virtualization techniques and rCUDA features (60 minutes)
- Practical demonstration about how to install and use rCUDA (30 minutes)

15.30 - 16.00 Coffee break

16.00 - 17:30 SESSION 2
- Guided exercises so that the audience uses rCUDA in a cluster located at Technical University of Valencia, Spain
- Time for attendees to freely exercise with rCUDA in the remote cluster (a set of exercises is proposed)
Reducing Power Consumption of Data Centers with rCUDA

Federico Silla
Technical University of Valencia
Spain
What is “rCUDA”?
rCUDA is a middleware that implements the remote GPU virtualization mechanism.
What is “remote GPU virtualization”?
It has to do with GPUs, obviously!
Basic behavior of CUDA
Remote GPU virtualization

A software technology that enables a more flexible use of GPUs in computing facilities.
Basics of remote GPU virtualization
Basics of remote GPU virtualization
Remote GPU virtualization envision

- Remote GPU virtualization allows a new vision of a GPU deployment, moving from the usual cluster configuration … :

![Physical configuration diagram]

… to the following one:

![Logical connections diagram]
Remote GPU virtualization envision

Without GPU virtualization

With GPU virtualization

GPU virtualization allows all nodes to share all GPUs
2nd

Why is “remote GPU virtualization” needed?
Which is the problem with GPU-enabled clusters?
A GPU-enabled cluster is a set of independent self-contained nodes. The cluster follows the shared-nothing approach:

- Nothing is directly shared among nodes (MPI required for aggregating computing resources within the cluster, included GPUs)
- GPUs can only be used within the node they are attached to
• Non-accelerated applications keep GPUs idle in the nodes where they use all the cores.

Hybrid MPI shared-memory non-accelerated applications usually span to all the cores in a node (across $n$ nodes).

A CPU-only application spreading over these nodes will make their GPUs unavailable for accelerated applications.
Money leakage in current clusters?

For some workloads, GPUs may be idle for significant periods of time:

- Initial acquisition costs not amortized
- Space: GPUs reduce CPU density
- Energy: idle GPUs keep consuming power

![Image of GPU nodes]

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Idle Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 GPU node</td>
</tr>
<tr>
<td>7</td>
<td>4 GPUs node</td>
</tr>
</tbody>
</table>

- 1 GPU node: Two E5-2620 v2 sockets and 32GB DDR3 RAM. One Tesla K20 GPU
- 4 GPU node: Two E5-2620 v2 sockets and 128GB DDR3 RAM. Four Tesla K20 GPUs

25%
First concern with accelerated clusters (II)

- Accelerated applications keep CPUs idle in the nodes where they execute.

Hybrid MPI shared-memory non-accelerated applications usually span to all the cores in a node (across n nodes).

An accelerated application using just one CPU core may avoid other jobs to be dispatched to this node.
First concern with accelerated clusters (II)

- Accelerated applications **keep CPUs idle** in the nodes where they execute

An accelerated MPI application using just one CPU core per node may keep part of the cluster busy

Hybrid MPI shared-memory non-accelerated applications usually span to all the cores in a node (across $n$ nodes)
Second concern with accelerated clusters

- Non-MPI multi-GPU applications cannot make use of the tremendous GPU resources available across the cluster (even if those GPU resources are idle)

Non-MPI multi-GPU application cannot be used by the multi-GPU application being executed.
One more concern with accelerated clusters

- Do applications completely squeeze the GPUs available in the cluster?
  - When a GPU is assigned to an application, computational resources inside the GPU may not be fully used
    - Application presenting low level of parallelism
    - CPU code being executed (GPU assigned ≠ GPU working)
    - GPU-core stall due to lack of data
    - etc …

![Diagram of a cluster with nodes connected via an interconnection network.](image-url)
GPU usage of GPU-Blast

![Graph showing GPU utilization and power consumption over time.](image-url)

- **Core Utilization**
- **Memory Utilization ( accesses)**

**Key Observations**:
- GPUs assigned but not used at certain intervals.
- Power consumption spikes correspond to periods of GPU usage.

This chart illustrates the dynamic nature of GPU utilization and power consumption, highlighting periods where GPUs are assigned but not actively used, which can be critical for optimizing GPU usage in data center environments.
GPU usage of CUDA-MEME

- GPU utilization is far away from maximum

Graph showing memory utilization and core utilization over time.
GPU usage of LAMMPS

- Core Utilization
- Memory Utilization (accesses)

GPU assigned but not used
GPU allocation vs GPU utilization

GPUs assigned but not used
In summary …

- There are scenarios where GPUs are available but cannot be used
- Also, CPU cores are available but cannot be used
- Accelerated applications do not make use of 100% of GPU resources

In conclusion …

- GPU and CPU cycles are lost, thus reducing cluster performance
We need something more in the cluster

The current model for using GPUs is too rigid

What is missing is ...

... some flexibility for using the GPUs in the cluster
We need something more in the cluster

The current model for using GPUs is too rigid

What is missing is ...

... some flexibility for using the GPUs in the cluster

A way of seamlessly sharing GPUs across nodes in the cluster

(remote GPU virtualization)
Sharing GPUs among applications

Without GPU virtualization

With GPU virtualization

GPU virtualization allows all nodes to share all GPUs
Sharing a GPU among jobs: GPU-Blast

One instance required about 51 seconds

Two concurrent instances of GPU-Blast
Sharing a GPU among jobs: GPU-Blast

First instance

Two concurrent instances of GPU-Blast
Sharing a GPU among jobs: GPU-Blast

Two concurrent instances of GPU-Blast

First instance

Second instance

Core Utilization
Memory Utilization (accesses)

Power (W)

Time

Utilization

Time
Sharing a GPU among jobs

K20 GPU

- LAMMPS: 876 MB
- mCUDA-MEME: 151 MB
- BarraCUDA: 3319 MB
- MUMmerGPU: 2104 MB
- GPU-LIBSVM: 145 MB
Increasing flexibility: one step further

- Once GPUs are shared, their amount can be reduced to match the actual workload.
- This would increase GPU utilization, also lowering power consumption, at the same time that initial acquisition costs are reduced.
Sharing GPUs among applications

Without GPU virtualization

With GPU virtualization

GPU virtualization allows all nodes to share all GPUs
Several efforts have been made to implement remote GPU virtualization during the last years:

<table>
<thead>
<tr>
<th>Framework</th>
<th>CUDA Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>rCUDA</td>
<td>CUDA 7.0</td>
</tr>
<tr>
<td>GVirtuS</td>
<td>CUDA 3.2</td>
</tr>
<tr>
<td>DS-CUDA</td>
<td>CUDA 4.1</td>
</tr>
<tr>
<td>vCUDA</td>
<td>CUDA 1.1</td>
</tr>
<tr>
<td>GViM</td>
<td>CUDA 1.1</td>
</tr>
<tr>
<td>GridCUDA</td>
<td>CUDA 2.3</td>
</tr>
<tr>
<td>V-GPU</td>
<td>CUDA 4.0</td>
</tr>
</tbody>
</table>

rCUDA is a development by Technical University of Valencia
Remote GPU virtualization frameworks

FDR InfiniBand + K20 !!

H2D pageable

D2H pageable

H2D pinned

D2H pinned

Bandwidth (MB/s) vs Transfer Size (MB)

CUDA  rCUDA  GVirtuS  DS-CUDA

rCUDA Tutorial. HiPEAC 2016 Conference. Prague
Cons of “remote GPU virtualization”?
The main drawback of GPU virtualization is the reduced bandwidth to the remote GPU
Using InfiniBand networks

Client side  

Server side

Application

CUDA Runtime API

rCUDA client engine

common communication API
TCP/IP module
InfiniBand Network "X" module

rCUDA server engine

common communication API
TCP/IP module
InfiniBand Network "X" module

CUDA Runtime library
CUDA Driver library

GPU

Network

Software

Hardware
Performance of rCUDA

- **CUDASW++**

  Bioinformatics software for Smith-Waterman protein database searches

---

Dual socket E5-2620 v2 Intel Xeon node with NVIDIA K20 GPU
- **GPU-Blast**

  Accelerated version of the NCBI-BLAST (Basic Local Alignment Search Tool), a widely used bioinformatics tool

  ![Graph](image)

  *Dual socket E5-2620 v2 Intel Xeon node with NVIDIA K20 GPU*
Initial transfers within rCUDA

H2D pageable

D2H pageable

H2D pinned

D2H pinned

CUDA K20  rCUDA FDR Orig  rCUDA FDR Opt
CUDA K40  rCUDA EDR Orig  rCUDA EDR Opt
Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

D2H pinned
rCUDA optimizations on applications

- Several applications executed with CUDA and rCUDA
  - K20 GPU and FDR InfiniBand
  - K40 GPU and EDR InfiniBand

Lower is better
Rodinia performance with rCUDA

InfiniBand EDR + K40!!

Short execution time

Medium execution time

Long execution time
LAMMPS performance with rCUDA

InfiniBand EDR + K40

InfiniBand FDR + K20
Pros of "remote GPU virtualization"?
1: more GPUs for a single application

- GPU virtualization is useful for multi-GPU applications

**Without** GPU virtualization

**With** GPU virtualization

- Many GPUs in the cluster can be provided to the application
1: more GPUs for a single application

Detected 64 CUDA Capable device(s)

<table>
<thead>
<tr>
<th>Device 0: &quot;Tesla M2090&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA Driver Version / Runtime Version</td>
</tr>
<tr>
<td>CUDA Capability Major/Minor version number</td>
</tr>
<tr>
<td>Total amount of global memory</td>
</tr>
<tr>
<td>(16) Multiprocessors x (32) CUDA Cores/MP</td>
</tr>
<tr>
<td>GPU Clock rate</td>
</tr>
<tr>
<td>Memory Clock rate</td>
</tr>
<tr>
<td>Memory Bus Width</td>
</tr>
<tr>
<td>L2 Cache Size</td>
</tr>
<tr>
<td>Max Texture Dimension Size (x,y,z)</td>
</tr>
<tr>
<td>Max Layered Texture Size (dim) x layers</td>
</tr>
<tr>
<td>Total amount of constant memory</td>
</tr>
<tr>
<td>Total amount of shared memory per block</td>
</tr>
<tr>
<td>Total number of registers available per block</td>
</tr>
<tr>
<td>Warp size</td>
</tr>
<tr>
<td>Maximum number of threads per multiprocessor</td>
</tr>
<tr>
<td>Maximum number of threads per block</td>
</tr>
<tr>
<td>Maximum sizes of each dimension of a block</td>
</tr>
<tr>
<td>Maximum sizes of each dimension of a grid</td>
</tr>
<tr>
<td>Maximum memory pitch</td>
</tr>
<tr>
<td>Texture alignment</td>
</tr>
<tr>
<td>Concurrent copy and kernel execution</td>
</tr>
<tr>
<td>Run time limit on kernels</td>
</tr>
<tr>
<td>Integrated GPU sharing Host Memory</td>
</tr>
<tr>
<td>Support host page-locked memory mapping</td>
</tr>
<tr>
<td>Alignment requirement for Surfaces</td>
</tr>
<tr>
<td>Device has ECC support</td>
</tr>
<tr>
<td>Device supports Unified Addressing (UVA)</td>
</tr>
<tr>
<td>Device PCI Bus ID / PCI location ID</td>
</tr>
</tbody>
</table>

Device 1: "Tesla M2090"

CUDA Driver Version / Runtime Version | 5.0 / 5.0

64 GPUs!
1: more GPUs for a single application

- Monte Carlo Multi-GPU (from NVIDIA samples)

FDR InfiniBand + NVIDIA Tesla K20

Higher is better

Lower is better
2: busy CPU cores do not block GPUs

Physical configuration

Logical configuration
3: easier cluster upgrade

• Let’s suppose that a cluster without GPUs needs to be upgraded to use GPUs

No GPU

• GPUs require large power supplies
  • Are power supplies already installed in the nodes large enough?

• GPUs require large amounts of space
  • Does current form factor of the nodes allow to install GPUs?

The answer to both questions is usually “NO”
Approach 1: augment the cluster with some CUDA GPU-enabled nodes → only those GPU-enabled nodes can execute accelerated applications
Approach 2: augment the cluster with some rCUDA servers → all nodes can execute accelerated applications

3: easier cluster upgrade

GPU-enabled
3: easier cluster upgrade

- Dual socket E5-2620v2 Intel Xeon + 32GB RAM + K20 GPU
- FDR InfiniBand based cluster

15 nodes without GPU + 1 node with 4 GPUs
Applications used for tests:

- GPU-Blast (21 seconds; 1 GPU; 1599 MB)
- LAMMPS short (90 seconds; 1 GPU; 2633 MB)
- LAMMPS long 2p (149 seconds; 2 GPUs; 3950 MB)
- LAMMPS long 4p (71 seconds; 4 GPUs; 2385 MB)
- mCUDA-MEME short (510 seconds; 1 GPU; 151 MB)
- mCUDA-MEME long 2p (1182 seconds; 2 GPUs; 152 MB)
- mCUDA-MEME long 4p (631 seconds; 4 GPUs; 152 MB)
- BarraCUDA (10 minutes; 1 GPU; 3319 MB)
- GPU-LIBSVM (5 minutes; 1 GPU; 145 MB)
- MUMmerGPU (5 minutes; 1 GPU; 2804 MB)
- GROMACS (167 seconds)
- NAMD (11 minutes)
### 3: easier cluster upgrade

<table>
<thead>
<tr>
<th>Application</th>
<th>Workload</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WL 1</td>
<td>WL 2</td>
<td></td>
</tr>
<tr>
<td>GPU-Blast</td>
<td>41</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>LAMMPS short</td>
<td>39</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>LAMMPS long 2p</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>LAMMPS long 4p</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>mCUDA-MEME short</td>
<td>39</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>mCUDA-MEME long 2p</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>mCUDA-MEME long 4p</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>GROMACS</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>BarraCUDA</td>
<td>40</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>MUMmerGPU</td>
<td>41</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>GPU-LIBSVM</td>
<td>40</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>NAMD</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>400</strong></td>
<td><strong>400</strong></td>
<td></td>
</tr>
</tbody>
</table>
3: easier cluster upgrade

![Graph showing execution time comparison between CUDA and rCUDA for workload WL 1 and WL 2.](image)

-68% - 60%

![Graph showing GPU utilization comparison between CUDA and rCUDA for workload WL 1 and WL 2.](image)

+131% +119%

![Graph showing energy consumption comparison between CUDA and rCUDA for workload WL 1 and WL 2.](image)

-63% -56%
4: GPU consolidation

- Concentrate GPUs into **dedicated GPU boxes** (with a low-power CPU?)
- Allow GPU task migration
Box A has 4 GPUs but only one is busy

Box B has 8 GPUs but only two are busy

1. Move jobs from Box B to Box A and switch off Box B
2. Migration should be transparent to applications (decided by the global scheduler)
4: GPU consolidation: one step beyond

Job granularity instead of GPU granularity
5: virtual machines can easily access GPUs

- How to access the GPU in the native domain from the inside of the virtual machines?
5: virtual machines can easily access GPUs

- The GPU is assigned by using PCI passthrough exclusively to a single virtual machine
- Concurrent usage of the GPU is not possible
5: virtual machines can easily access GPUs

Computer hosting several KVM virtual machines

KVM Host Linux

SW BRIDGE

Gb ETH

IB PF

IB VF

InfiniBand Fabric

KVM Guest Linux 1

rCUDA client

vGPU

vETH

IB

PCI PT

KVM Guest Linux n

rCUDA client

vGPU

vETH

IB

PCI PT

High performance network available

Low performance network available
Application performance with KVM

FDR InfiniBand + K20 !!

![Graph showing execution time for different models and DNA sequence lengths.]

- **LAMMPS**
- **CUDA-MEME**
- **CUDASW++**
- **GPU-BLAST**

<table>
<thead>
<tr>
<th>Model Type</th>
<th>CUDA</th>
<th>CUDA VM-PT</th>
<th>rCUDA non-VM</th>
<th>rCUDA VM IB</th>
<th>rCUDA VM Local</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LAMMPS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CUDA-MEME</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CUDASW++</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GPU-BLAST</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Query length**
  - 144
  - 1000
  - 2005
  - 3005
  - 4061
  - 5147

- **Sequence length**
  - 800
  - 1600
  - 2000
  - 2404
  - 2800
Application performance with Xen

FDR InfiniBand + K20 !!

LAMMPS

CUDA-MEME

CUDA SW++

GPU-BLAST
5th

What happens at the cluster level?
• **GPUs can be shared** among jobs running in remote clients
  • Job scheduler required for coordination
  • **Slurm** was selected
Ongoing work: studying rCUDA+Slurm

- Applications used for tests:
  - GPU-Blast (21 seconds; 1 GPU; 1599 MB)
  - LAMMPS (15 seconds; 4 GPUs; 876 MB)
  - MCUDA-MEME (165 seconds; 4 GPUs; 151 MB)
  - GROMACS (167 seconds)
  - NAMD (11 minutes)
  - BarraCUDA (10 minutes; 1 GPU; 3319 MB)
  - GPU-LIBSVM (5 minutes; 1 GPU; 145 MB)
  - MUMmerGPU (5 minutes; 1 GPU; 2804 MB)

- Short execution time
  - Set 1
  - Set 2

- Long execution time

- Three workloads:
  - Set 1
  - Set 2
  - Set 1 + Set 2

- Three workload sizes:
  - Small (100 jobs)
  - Medium (200 jobs)
  - Large (400 jobs)
### Ongoing work: studying rCUDA+Slurm

<table>
<thead>
<tr>
<th>Application</th>
<th>Workload</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set 1</td>
<td>Set 2</td>
<td>Set 1+2</td>
</tr>
<tr>
<td>GPU-Blast</td>
<td>112</td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>LAMMPS</td>
<td>88</td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>mCUDA-MEME</td>
<td>99</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>GROMACS</td>
<td>101</td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td></td>
<td>112</td>
<td>51</td>
</tr>
<tr>
<td>MUMmerGPU</td>
<td></td>
<td>88</td>
<td>52</td>
</tr>
<tr>
<td>GPU-LIBSVM</td>
<td></td>
<td>99</td>
<td>37</td>
</tr>
<tr>
<td>NAMD</td>
<td></td>
<td>101</td>
<td>49</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
</tbody>
</table>
Test bench for studying rCUDA+Slurm

- Dual socket E5-2620v2 Intel Xeon + 32GB RAM + K20 GPU
- FDR InfiniBand based cluster

4 GPU nodes

node with the Slurm scheduler

8 GPU nodes

node with the Slurm scheduler

16 GPU nodes

node with the Slurm scheduler
Results for 16-node cluster

Workload size is 400 jobs

Lower is better

Higher is better
<table>
<thead>
<tr>
<th>Application</th>
<th>Workload</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set 2</td>
<td>Set 1+2</td>
<td>New Set 2</td>
<td>New Set 1+2</td>
</tr>
<tr>
<td>GPU-Blast</td>
<td></td>
<td></td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>LAMMPS</td>
<td></td>
<td></td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>mCUDA-MEME</td>
<td></td>
<td></td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>GROMACS</td>
<td></td>
<td></td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>BarraCUDA</td>
<td>112</td>
<td>51</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>MUMmerGPU</td>
<td>88</td>
<td>52</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>GPU-LIBSVM</td>
<td>99</td>
<td>37</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>NAMD</td>
<td>101</td>
<td>49</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td></td>
</tr>
</tbody>
</table>
Results for 16-node cluster

- **Execution Time (s):**
  - CUDA, rCUDA ex, rCUDA sh
  - Lower is better

- **Workload size is 400 jobs**

- **Energy (kWh):**
  - CUDA, rCUDA ex, rCUDA sh
  - Lower is better

- **GPU Utilization:**
  - CUDA, rCUDA ex, rCUDA sh
  - Higher is better
• Dual socket E5-2620v2 Intel Xeon + 32GB RAM + K20 GPU
• FDR InfiniBand based cluster

15 nodes with 1 GPU + 1 node with 4 GPUs

node with the Slurm scheduler
<table>
<thead>
<tr>
<th>Application</th>
<th>Workload</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WL 1</td>
<td>WL 2</td>
</tr>
<tr>
<td>GPU-Blast</td>
<td>41</td>
<td>48</td>
</tr>
<tr>
<td>LAMMPS short</td>
<td>39</td>
<td>46</td>
</tr>
<tr>
<td>LAMMPS long 2p</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>LAMMPS long 4p</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>mCUDA-MEME short</td>
<td>39</td>
<td>46</td>
</tr>
<tr>
<td>mCUDA-MEME long 2p</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>mCUDA-MEME long 4p</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>GROMACS</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td>40</td>
<td>47</td>
</tr>
<tr>
<td>MUMmerGPU</td>
<td>41</td>
<td>47</td>
</tr>
<tr>
<td>GPU-LIBSVM</td>
<td>40</td>
<td>46</td>
</tr>
<tr>
<td>NAMD</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>400</strong></td>
<td><strong>400</strong></td>
</tr>
</tbody>
</table>
Workload size is 400 jobs

Lower is better

Higher is better

CUDA Tutorial. HiPEAC 2016 Conference. Prague
... in summary ...
Pros and cons of rCUDA

• Cons:
  1. Reduced bandwidth to remote GPU (really a concern??)

• Pros:
  1. Many GPUs for a single application
  2. Concurrent GPU access to virtual machines
  3. Increased cluster throughput
  4. Similar performance with smaller investment
  5. Easier (cheaper) cluster upgrade
  6. Migration of GPU jobs
  7. Reduced energy consumption
  8. Increased GPU utilization
Get a free copy of rCUDA at
http://www.rcuda.net
More than 650 requests world wide

@rcuda_

rCUDA is a development by Technical University of Valencia
Thanks!

Questions?
Hands-on Session
(Part I)

Carlos Reaño
Technical University of Valencia
Spain
• 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!!!
What is rCUDA?

Installing and using rCUDA

rCUDA over HPC networks
  - InfiniBand

How taking benefit from rCUDA
  - Sample scenarios

Questions & Answers
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)
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
  ```
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
    ./rCUDAad
    ```
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:

- Set env. vars as follows:

  ```
  export PATH=$PATH:/usr/local/cuda/bin
  export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
  export RCUDADEVICE_COUNT=1
  export RCUDADEVICE_0=<server_name_or_ip_address>:0
  ```

- Compile CUDA program using dynamic libraries:

  ```
  cd $HOME/NVIDIA_CUDA_Samples/1_Utilsitlies/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:
  ```bash
echo "export PATH=$PATH:/usr/local/cuda/bin"
echo "export LD_LIBRARY_PATH="$HOME/rCUDA/lib":$LD_LIBRARY_PATH"
echo "export RCUDA_DEVICE_COUNT=1"
echo "export RCUDA_DEVICE_0=<server_name_or_ip_address>:0"
```

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

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

- Running a CUDA program with rCUDA:
  - 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
    ```
  - Number of remote GPUs: 1, 2, 3...

- 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/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:  
  ```bash
  export PATH=$PATH:/usr/local/cuda/bin
  export LD_LIBRARY_PATH=$HOME/rCUDA/lib:$LD_LIBRARY_PATH
  export RCUDADEVICE_COUNT=1
  export RCUDADEVICE_0=<server_name_or_ip_address>:0
  ```

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

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

- 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
  ```

- Compile CUDA program using dynamic libraries:

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

- Run the CUDA program as usual:

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

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

- 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
```

- Compile CUDA program using dynamic libraries:

```bash
cd $HOME/NVIDIA_CUDA_Samples/1_Utilsities/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
Starting rCUDA server using IB:

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

Run CUDA program using rCUDA over IB:

```bash
export RCUDAPROTO=IB
cd $HOME/NVIDIA_CUDA_Samples/1.Utilities/bandwidthTest
./bandwidthTest
```
Starting rCUDA server using IB:

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

Tell rCUDA we want to use IB

Run CUDA program using rCUDA over IB:

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

Also in the client!!
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
  ```

- 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
Three main types of applications:

- Bandwidth bounded: more transfers, more rCUDA overhead
- Computations bounded: more computations, less rCUDA overhead
- Intermediate
LAMMPS Application

- GPU vs. remote GPU
  - Overhead of remote GPUs?
- Live demonstration:
  - LAMMPS with CUDA
  - LAMMPS with rCUDA
CPU vs. remote GPU
- What is better: a local CPU or a remote GPU?

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

Live demonstration:
  - LAMMPS on CPU (without CUDA)
  - LAMMPS on CPU (using all the cores!)
CPU vs. remote GPU
- What is better: a local CPU or a remote GPU?

Live demonstration:
- LAMMPS on CPU (without CUDA)
- LAMMPS on CPU (using all the cores!)

What if I am using TCP instead of IB?
- LAMMPS using rCUDA over TCP (1GbE)
- GPU vs. remote GPU
  - Overhead of remote GPUs?
- Live demonstration:
  - MAGMA with CUDA
  - MAGMA with rCUDA
CPU vs. remote GPU
  What is better: a local CPU or a remote GPU?
Live demonstration:
  MAGMA on CPU (without CUDA)
CPU vs. remote GPU
  - What is better: a local CPU or a remote GPU?

Live demonstration:
  - MAGMA on CPU (without CUDA)

What if I am using TCP instead of IB?
  - MAGMA using rCUDA over TCP (1GbE)
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
- **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
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
```

- Set device count:

```bash
export RCUDA_DEVICE_COUNT=5
```

- Set device configurations:

```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 RCUDADEVICE_4=node4:0
```

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

- `export 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
Configure rCUDA for Multi-GPU:

- Check configuration by running deviceQuery sample
Multi-GPU Configuration

- Live demonstration:
  - deviceQuery sample with multiple GPUs
  - LAMMPS multiple remote GPUs
  - MAGMA multiple remote GPUs
  - MonteCarloMultiGPU
Outline

- What is rCUDA?
- Installing and using rCUDA
- rCUDA over HPC networks
  - InfiniBand
- How taking benefit from rCUDA
  - Sample scenarios
- Questions & Answers
Get a free copy of rCUDA at
http://www.rcuda.net

@rcuda_r
Hands-on Session
(Part II)

Carlos Reaño
Technical University of Valencia
Spain
Outline

- Short intro to CUDA programming
- Cluster for testing
- Installing rCUDA
- Running some codes
- Questions & Answers
• Short intro to CUDA programming
• Cluster for testing
• Installing rCUDA
• Running some codes
• Questions & Answers
CUDA programming language = C extended:
- For running functions in the GPU: called **kernels**
- ...

CUDA API:
- Functions to copy data from main memory to GPU memory and vice versa
- ...

---

![Diagram of CPU, GPU, and memory connections](image_url)
CUDA typical code layout:

- Copy input data from main memory to GPU memory
- Perform some computation in GPU (launch kernel)
- Copy output data from GPU memory to main memory
Sample code:

```c
// CUDA Kernel
__global__ void myCUDAKernel(...) {
    // Perform some computation in the GPU
    ...
}

int main() {
    ...
    // Copy input data to GPU mem
    cudaMemcpy(gpuMemPtr, memPtr, size, cudaMemcpyHtoD);
    ...
    // Launch kernel
    myCUDAKernel<<<nThreads, nBlocks>>>(...);
    ...
    // Copy output data to main mem
    cudaMemcpy(memPtr, gpuMemPtr, size, cudaMemcpyDtoH);
}
```
● Short intro to CUDA programming

● **Cluster for testing**

● Installing rCUDA

● Running some codes

● Questions & Answers
“rcu” cluster:

rcu3, rcu4, rcu5, ... rcu29

Network (1 GbE)

rcu15, rcu16

GPU, GPU, GPU, GPU, GPU
"rcu" cluster:

Nodes without GPUs for users (rCUDA clients)
“rcu” cluster:

- Nodes without GPUs for users (rCUDA clients)
- Nodes with GPUs for rCUDA servers
Outline

- Short intro to CUDA programming
- Cluster for testing
- **Installing rCUDA**
- Running some codes
- Questions & Answers
Installing and using rCUDA

- Log into rcu cluster

- rCUDA software package already in $HOME folder

- Installing rCUDA
  - Just untar the tarball in $HOME folder

- Package contents. Important folders:
  - doc: rCUDA user guide
  - bin: rCUDA server daemon
  - lib: rCUDA library
Installing and using rCUDA

- CUDA must be installed in GPU nodes
- In the nodes running rCUDA servers
- Already installed in our cluster
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:
  
  ```bash
  export PATH=$PATH:/usr/local/cuda/bin
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
  ```

- Start rCUDA server:
  
  ```bash
  cd $HOME/rCUDA/bin
  ./rCUDAd
  ```
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:

  ```bash
  cd $HOME/rCUDA/bin
  ./rCUDAstart
  ```
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
Running a CUDA program with rCUDA:

- 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=rcu16:0
```

- Compile CUDA program using dynamic libraries:

```bash
cd $HOME/NVIDIA_CUDA_Samples/1_Utilsites/deviceQuery
make
```

- 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=rcu16:0
  ```

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

- 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=rcu16:0
  ```

- Compile CUDA program using dynamic libraries:
  ```
  cd $HOME/NVIDIA_CUDA_Samples/1_Utilsites/deviceQuery
  make
  ```

- 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=rcu16:0
    ```
  - Compile CUDA program using dynamic libraries:
    ```
    cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
    make
    ```
  - Run the CUDA program as usual:
    ```
    ./deviceQuery
    ...```
Running a CUDA program with rCUDA:

- 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=rcu16:0
  ```

- Compile CUDA program using dynamic libraries:
  
  ```bash
  cd $HOME/NVIDIA_CUDA_Samples/1_Utilsities/deviceQuery
  make
  ```

- Run the CUDA program as usual:
  
  ```bash
  ./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=rcu16:0
    ```
  - Compile CUDA program using dynamic libraries:
    ```
    cd $HOME/NVIDIA_CUDA_Samples/1_Utilsities/deviceQuery
    make
    ```
  - 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 RCUDADEVICECOUNT=1
export RCUDADEVICE0=rcu16:0
```

- Compile CUDA program using dynamic libraries:

```
cd $HOME/NVIDIA_CUDA_Samples/1_Utilsities/deviceQuery
make
```

- 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/1_Utilities/deviceQuery
    make
    ```

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

- 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
```

- Compile CUDA program using dynamic libraries:

```bash
cd $HOME/NVIDIA_CUDA_Samples/1_Utilities/deviceQuery
make
```

- Run the CUDA program as usual:

```
./deviceQuery
... Error! Why?
```
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
...
```

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

- 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
  ```

- Compile CUDA program using dynamic libraries:

  ```bash
  cd $HOME/NVIDIA_CUDA_Samples/6.5/1_Utilities/deviceQuery
  make EXTRA_NVCCFLAGS=--cudart=shared
  ```

- Run the CUDA program as usual:

  ```bash
  ./deviceQuery
  ```
Outline

- Short intro to CUDA programming
- Cluster for testing
- Installing rCUDA
- Running some codes
- Questions & Answers
Impact of multiple users sharing the same GPU

- Measure time of running deviceQuery sample 10 times when the remote GPU is only used by one user

- Measure time of running deviceQuery sample 10 times when multiple users are sharing the same GPU

- Does execution time increases? With how many concurrent users the increase is more noticeable?
Impact of multiple users sharing the same GPU

- Measure time of running deviceQuery sample 10 times when the remote GPU is only used by one user

- Measure time of running deviceQuery sample 10 times when multiple users are sharing the same GPU

- Does execution time increases? With how many concurrent users the increase is more noticeable?

- deviceQuery sample: no computations, no transfers
Impact of multiple users sharing the same GPU

- Measure bandwidth when copying from main memory to remote GPU memory when the remote GPU is only used by one user

- Measure bandwidth when copying from main memory to remote GPU memory to the remote GPU when multiple users are sharing the same GPU

- Does bandwidth decreases? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the transfer?
Impact of multiple users sharing the same GPU

- Measure bandwidth when copying from main memory to remote GPU memory when the remote GPU is only used by one user.

- Measure bandwidth when copying from main memory to remote GPU memory to the remote GPU when multiple users are sharing the same GPU.

- Does bandwidth decreases? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the transfer?

- bandwidthTest sample: **no computations, only transfers**
Impact of multiple users sharing the same GPU

- Measure performance of vector addition when the remote GPU is only used by one user

- Measure performance of vector addition when multiple users are sharing the same GPU

- Does performance decreases? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the vector?
Impact of multiple users sharing the same GPU

- Measure performance of vector addition when the remote GPU is only used by one user

- Measure performance of vector addition when multiple users are sharing the same GPU

- Does performance decrease? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the vector?

- vectorAdd sample: computations and transfers
Impact of multiple users sharing the same GPU

- Measure performance of matrix multiplication when the remote GPU is only used by one user

- Measure performance of matrix multiplication when multiple users are sharing the same GPU

- Does performance decreases? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the matrix?
Impact of multiple users sharing the same GPU

- Measure performance of matrix multiplication when the remote GPU is only used by one user

- Measure performance of matrix multiplication when multiple users are sharing the same GPU

- Does performance decreases? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the matrix?

- matrixMul sample: *computations and transfers*
Impact of multiple users sharing the same GPU

- Measure performance of matrix multiplication when the remote GPU is only used by one user

- Measure performance of matrix multiplication when multiple users are sharing the same GPU

- Does performance decreases? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the matrix?
Impact of multiple users sharing the same GPU

- Measure performance of matrix multiplication when the remote GPU is only used by one user
- Measure performance of matrix multiplication when multiple users are sharing the same GPU
- Does performance decrease? With how many concurrent users the decrease is more noticeable? What happens if we increase the size of the matrix?
- `matrixMulCUBLAS` sample: computations and transfers
Multi-GPU scenario

- 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
Configure rCUDA for Multi-GPU:

```bash
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=rcu16:0
export RCUDA_DEVICE_1=rcu16:1
export RCUDA_DEVICE_2=rcu16:2
export RCUDA_DEVICE_3=rcu16:3
export RCUDA_DEVICE_4=rcu15:0
```
Configure rCUDA for Multi-GPU:

- `export PATH=$PATH:/usr/local/cuda/bin`
- `export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDA:$LD_LIBRARY_PATH`
- `export RCUDA_DEVICE_COUNT=5`
- `export RCUDADEVICE_0=rcu16:0`
- `export RCUDADEVICE_1=rcu16:1`
- `export RCUDADEVICE_2=rcu16:2`
- `export RCUDADEVICE_3=rcu16:3`
- `export RCUDADEVICE_4=rcu15:0`

Number of remote GPUs: 5
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=rcu16:0
export RCUDA_DEVICE_1=rcu16:1
export RCUDA_DEVICE_2=rcu16:2
export RCUDA_DEVICE_3=rcu16:3
export RCUDA_DEVICE_4=rcu15:0
```
Multi-GPU Configuration

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=rcu16:0
export RCUDA DEVICE_1=rcu16:1
export RCUDA DEVICE_2=rcu16:2
export RCUDA DEVICE_3=rcu16:3
export RCUDA DEVICE_4=rcu15:0
```

- Check configuration by running `deviceQuery` sample
Outline

- Short intro to CUDA programming
- Cluster for testing
- Installing rCUDA
- Running some codes
- Questions & Answers
Get a free copy of rCUDA at
http://www.rcuda.net

@rcuda_r