Make your heterogeneous cluster more flexible with remote GPU virtualization

Federico Silla & Carlos Reaño
Technical University of Valencia
Spain
Tutorial Schedule

- 14.00
  - Basics of remote GPU virtualization
  - Hands on rCUDA
- 15.30
  - Coffee Break
- 16.00
  - Advanced features of rCUDA
  - rCUDA lab
Make your heterogeneous cluster more flexible with remote GPU virtualization

Federico Silla
Universidad Politécnica de Valencia
Current computing needs

• Many applications require a lot of computing resources
• Execution time is usually increased
• Applications are accelerated to get their execution time reduced
• GPU computing has experienced a remarkable growth in the last years
GPUs reduce energy and time

```
blastp --db sorted_env_nr --query SequenceLength_00001300.txt -num_threads X -gpu [t|f]
```

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.
**GPUs reduce energy and time**

```bash
blastp -db sorted_env_nr -query SequenceLength_00001300.txt -num_threads X -gpu [t|f]
```

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

- **Green zone:** GPUs are **better** than CPUs
- **Red zone:** GPUs are **worse** than CPUs

GPUs are not the magic solution for all applications.
Current GPU computing facilities

The basic building block is a node with one or more GPUs
Heterogeneous clusters

- From the programming point of view:
  - A set of nodes, each one with:
    - one or more CPUs (with several cores per CPU)
    - one or more GPUs (typically between 1 and 4)
  - An interconnection network
A GPU-enabled cluster is a set of independent self-contained nodes that leverage 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
Basics of GPU computing

Basic behavior of CUDA

Application

CUDA libraries

GPU

NVIDIA CUDA

rCUDA Tutorial PACT’15

8/63
What is “remote GPU virtualization”?

Make your heterogeneous cluster more flexible with remote GPU virtualization.
Has to do with GPUs, obviously!
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 allows a new vision of a GPU deployment, moving from the usual cluster configuration:

Remote GPU virtualization envision to the following one:
Why is “remote GPU virtualization” needed?
Which is the problem with GPU-enabled clusters?
The problem is the cluster architecture

- A GPU-enabled cluster is a set of independent self-contained nodes that leverage 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
First concern with accelerated clusters

- Applications can only use the GPUs located within their node:
  - Non-accelerated applications keep GPUs idle in the nodes where they use all the cores

A CPU-only application spreading over these four nodes would 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 GPUs](image_url)

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>1 GPU node</th>
<th>4 GPUs node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-7</td>
<td>Idle Power (Watts)</td>
<td></td>
</tr>
<tr>
<td>14-21</td>
<td></td>
<td>25%</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 1 GPU node: Two E5-2620V2 sockets and 32GB DDR3 RAM. One Tesla K20 GPU
- 4 GPU node: Two E5-2620V2 sockets and 128GB DDR3 RAM. Four Tesla K20 GPUs
• Applications can only use the GPUs located within their node.

• Non-MPI multi-GPU applications cannot make use of the tremendous GPU resources available at other cluster nodes (even if they are idle).

All these GPUs cannot be used by the multi-GPU application being executed.

Second concern with accelerated clusters
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 …
Sharing a given GPU among jobs

- Several **GPU-Blast** instances concurrently executed on the same GPU. Each instance uses about 1.5GB of GPU memory.
In summary …

- There are scenarios where GPUs are available but cannot be used
- Accelerated applications do not make use of GPUs 100% of the time

In conclusion …

- GPU 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 across nodes in the cluster

Physical configuration

Logical configuration

Interconnection Network

Logical connections
Once GPUs are shared, their amount can be reduced to match real workload.

This would increase GPU utilization, also lowering power consumption, at the same time that initial acquisition costs are reduced.
Remote GPU virtualization envision

Without GPU virtualization

With GPU virtualization

Virtualized remote GPUs

GPU virtualization allows all nodes to access all GPUs
What is needed for increased flexibility?

- This new cluster configuration requires:
  - A way of seamlessly sharing GPUs across nodes in the cluster (remote GPU virtualization)
  - Enhanced job schedulers that take into account the new virtual GPUs
Frameworks for remote GPU virtualization
Several efforts have been made to implement remote GPU virtualization during the last years:

- **rCUDA** (CUDA 7.0)
- GVirtuS (CUDA 3.2)
- DS-CUDA (CUDA 4.1)
- vCUDA (CUDA 1.1)
- GViM (CUDA 1.1)
- GridCUDA (CUDA 2.3)
- V-GPU (CUDA 4.0)

### Publicly available
- rCUDA

### NOT publicly available
- GVirtuS
- DS-CUDA
- vCUDA
- GViM
- GridCUDA
- V-GPU
Remote GPU virtualization frameworks

FDR InfiniBand + K20 !!

- H2D pageable
- D2H pageable
- H2D pinned
- D2H pinned
rCUDA uses optimized transfers

- rCUDA features **optimized data transfers:**
  - Pipelined transfers to improve performance
  - Preallocated pinned memory buffers
  - Optimal pipeline block size
Basic performance analysis

- Pipeline block size for InfiniBand Connect-IB

It was 2MB with IB QDR and 1MB with FDR X3

- NVIDIA Tesla K40; Mellanox Connect-IB + SX6036 Mellanox switch
rCUDA uses a proprietary communication protocol

Example:

1) initialization
2) memory allocation on the remote GPU
3) CPU to GPU memory transfer of the input data
4) kernel execution
5) GPU to CPU memory transfer of the results
6) GPU memory release
7) communication channel closing and server process finalization
How to declare remote GPUs

Environment variables are properly initialized in the client side and used by the rCUDA client (transparently to the application)

Server name/IP address: GPU

Amount of GPUs exposed to applications
Cons of “remote GPU virtualization”?
The main drawback of GPU virtualization is the reduced bandwidth to the remote GPU.
Using InfiniBand networks
CUDASW++

Bioinformatics software for Smith-Waterman protein database searches

Performance of rCUDA

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

Dual socket E5-2620 v2 Intel Xeon node with NVIDIA K20 GPU
Initial transfers of rCUDA with InfiniBand
Several optimizations applied to rCUDA:

- At the InfiniBand level:
  - Optimization intended for short transfers
  - Optimization intended for long transfers
- At the rCUDA level:
  - The internal pipeline has been redesigned
Applications use **InfiniBand** by creating **queue pairs (QPs)**:

- QPs do not store data but work requests (WRs)
- Work requests are descriptors of the transfer operation to be done
- Applications can **associate several QPs to the same network adapter**
  - The more QPs, the higher the complexity of keeping them synchronized
Improving communications within rCUDA

- Optimization for short transfers:
  - increase the capacity of the send/receive queues (i.e., number of work requests that can be allocated)
Improving communications within rCUDA

- Optimization for long transfers:
  - increase the number of QPs used for a single transfer

![Graph showing bandwidth test results for different QP counts.](image)
Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

D2H pinned

Optimized transfers within rCUDA

H2D pageable

Almost 100% of available BW

D2H pageable

Almost 100% of available BW

H2D pinned

Almost 100% of available BW

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
rCUDA optimizations on applications

- Two are the reasons for the better performance of rCUDA:
  1. Higher bandwidth for pageable memory
  2. Network polling interval to check for work completions: CUDA polls PCIe whereas rCUDA polls InfiniBand network adapter
Rodinia performance with rCUDA

InfiniBand EDR + K40 !!

Short execution time

Long execution time

Medium execution time

Execution Time (s)

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

backprop dwt2d gaussian hotspot hybridsort lavaMD leukocyte myocyte.out nn needle particlefilter_naive particlefilter_float srad_v1 srad_v2

Execution Time (s)

0 1 2 3 4 5

bfs euler3d euler3d_double heartwall kmeans pathfinder mummergpu
Rodinia performance with rCUDA

InfiniBand EDR + K40 !!
Pros of "remote GPU virtualization"?
GPU virtualization is useful for multi-GPU applications.

**Without** GPU virtualization:
- Only the GPUs in the node can be provided to the application.

**With** GPU virtualization:
- Many GPUs in the cluster can be provided to the application.
- Many GPUs in the cluster can be provided to the application.

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

Detected 64 CUDA Capable device(s)

Device 0: "Tesla M2090"
CUDA Driver Version / Runtime Version: 5.0 / 5.0
CUDA Capability Major/Minor version number: 2.0
Total amount of global memory: 6144 MBytes (6442123264 bytes)
(16) Multiprocessors x (32) CUDA Cores/MP: 512 CUDA Cores
GPU Clock rate: 1301 MHz (1.30 GHz)
Memory Clock rate: 1848 MHz
Memory Bus Width: 384-bit
L2 Cache Size: 786432 bytes
Max Texture Dimension Size (x,y,z): 1D=(65536), 2D=(65536,65535), 3D=(2048,2048,2048)
Max Layered Texture Size (dln) x layers: 1D=(16384) x 2048, 2D=(16384,16384) x 2048
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 49152 bytes
Total number of registers available per block: 32768
Warp size: 32
Maximum number of threads per multiprocessor: 1536
Maximum number of threads per block: 1024
Maximum sizes of each dimension of a block: 1024 x 1024 x 64
Maximum sizes of each dimension of a grid: 65536 x 65535 x 65535
Maximum memory pitch: 2147483647 bytes
Texture alignment: 512 bytes
Concurrent copy and kernel execution: Yes with 2 copy engine(s)
Run time limit on kernels: No
Integrated GPU sharing Host Memory: No
Support host page-locked memory mapping: No
Alignment requirement for Surfaces: Yes
Device has ECC support: Disabled
Device supports Unified Addressing (UVA): Yes
Device PCI Bus ID / PCI location ID: 2 / 0
Compute Mode:
< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

Device 1: "Tesla M2090"
CUDA Driver Version / Runtime Version: 5.0 / 5.0
1: more GPUs for a single application

- Monte Carlo Multi-GPU (from NVIDIA samples)

![Graph showing performance comparison between CUDA and rCUDA](image)

Higher is better

Lower is better

FDR InfiniBand +
NVIDIA Tesla K20
2: busy CPU cores do not block GPUs

Interconnection Network

Physical configuration

Logical connections

Logical configuration
3: easier cluster upgrade

- A cluster without GPUs may be easily upgraded to use GPUs with rCUDA
3: easier cluster upgrade

- A cluster without GPUs may be easily upgraded to use GPUs with rCUDA
4: GPU task migration

- Consolidate 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)
5. Easier access from the inside of virtual machines to GPUs
6. Increased performance at the cluster level
   • Cluster throughput (jobs/hour) is **doubled**

More details in the second part of the tutorial
Get a free copy of rCUDA at
http://www.rcuda.net

More than 650 requests world wide

@rcuda_
Thanks!

Questions?
Advanced features of rCUDA

Federico Silla
Universidad Politécnica de Valencia
5. Easier access from the inside of virtual machines to GPUs
6. Increased performance at the cluster level
   • Cluster throughput (jobs/hour) is **doubled**
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
How to attach an InfiniBand card to a VM

- PCI pass-through is used to assign an IB card (either real or virtual) to a given VM
- The IB card manages the several virtual copies
5: virtual machines can easily access GPUs

High performance network available

Low performance network available
5: virtual machines can easily access GPUs

High performance network available

Low performance network available
Application performance with KVM

FDR InfiniBand + K20 !!

LAMMPS

CUDA-MEME

CUDASW++

GPU-BLAST
Application performance with Xen

FDR InfiniBand + K20 !!

LAMMPS

CUDA-MEME

CUDASW++

GPU-BLAST

CUDA
CUDA VM-PT
rCUDA non-VM
rCUDA VM IB
rCUDA VM Local

CUDA Tutorial PACT’15
Overhead of rCUDA within VMs

KVM

- LAMMPS: 2.5%
- CUDA-MEME: 1.6%
- CUDASW++: 0.5%
- GPU-BLAST: 0.07%

FDR InfiniBand + K20 !!

Xen

- LAMMPS: 6.1%
- CUDA-MEME: 2.9%
- CUDASW++: 0.7%
- GPU-BLAST: 2.3%
5: increased performance for clusters
• **GPUs can be shared** among jobs running in remote clients
  • Job scheduler required for coordination
  • **Slurm** was selected
The basic idea about SLURM

Resources per job:
1. Nodes: 2 GPUs: 3
2. Nodes: 2 GPUs: 1
3. Nodes: 1 GPUs: 0
4. Nodes: 2 GPUs: 1
5. Nodes: 1 GPUs: 2

Diagram showing job allocation over time:
- NODE 0 (GPU 0, GPU 1)
  - t0: Job 1
  - t1: Job 2
  - t2: Job 4
  - t3: Job 4
  - t4: Job 5
  - t5: Job 5
  - t6: Job 5
  - t7: Job 5
  - t8: Job 5

Job queue:
1. Job 1
2. Job 2
3. Job 3
4. Job 4
5. Job 5
5. Job 5
5. Job 5

Error message: srund error: Unable to allocate resources
The basic idea about SLURM + rCUDA

resources per job

- 2 Nodes: 2 GPUs: 1
- 4 Nodes: 2 GPUs: 1
- 1 Nodes: 2 GPUs: 3
- 3 Nodes: 1 GPUs: 0
- 5 Nodes: 1 GPUs: 2

GPUs are decoupled from nodes

All jobs are executed in less time
Sharing remote GPUs among jobs

GPUs are decoupled from nodes. All jobs are executed even in less time.

GPU 0 is scheduled to be shared among jobs.
GPU memory is the limit

GPU memory

- App 1
- App 2
- App 3
- App 4
- App 5
- App 6
GPU memory is the limit

Does app 7 fit into the GPU?

GPU memory

App 1

App 2

App 5

App 4

App 6

App 3
Integrating rCUDA with SLURM

- SLURM does not know about virtualized GPUs
- SLURM must be enhanced in order to manage the new virtualized GPUs
- A new resource has been added: `rgpu`

### gres.conf

Name = `rgpu` File = `/dev/nvidia0` Cuda=3.5 Mem=4726 M
[ Name = `gpu` File = `/dev/nvidia0` ]

### slurm.conf

SelectType = select / cons_rgpu
SelectTypeParameters = CR_CORE
GresTypes = `rgpu`, `gpu`

NodeName = node1 NodeHostname = node1
CPUs=12 Sockets=2 CoresPerSocket=6
ThreadsPerCore=1 RealMemory=32072
Gres = `rgpu` :1[ , `gpu` :1]

**New submission options:**

--rcuda-mode=(shared|excl)
--gres=rgpu(X:(Y)?(:Z)?)?
  X = [1-9]+[0-9]*
  Y = [1-9]+[0-9]*[ kKmMgG]
  Z = [1-9].[0-9](cc|CC)
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)

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

Three workload sizes:
- Small (100 jobs)
- Medium (200 jobs)
- Large (400 jobs)
Test bench for studying rCUDA+Slurm

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

Results for Set 1

Lower is better
GPU utilization

Results for Set 1

Higher is better

Lower is better
Results for energy consumption

**Results for Set 1**

**4 nodes**

**Lower is better**

**8 nodes**

**Workload size**

- Small
- Medium
- Large

**Energy (kWh)**

- CUDA
- rCUDA
Reducing the amount of GPUs

Execution Time (s)

Lower is better

- 35% Less
- 39% Less
- 41% Less
Reducing the amount of GPUs

Cost (in Euros):

a) 16 nodes + 16 K20 ... 16*2500 + 16*2000 = 72000 Euros
b) 16 nodes + 4 K20 ... 16*2500 + 4*2000 = 48000 Euros

Reducing 33% acquisition expenses, performance is still 40% better
Energy when removing GPUs

Lower is better

39% Less

42% Less

44% Less
Energy when removing GPUs

- 39% Less
- 42% Less
- 44% Less

Lower is better

Costs:

- reducing 33% acquisition expenses, performance is still 40% better ...

Additionally, the electricity bill is significantly reduced
GPU utilization when removing GPUs

```
<table>
<thead>
<tr>
<th>Workload size</th>
<th>CUDA 16 GPUs</th>
<th>rCUDA 16 GPUs</th>
<th>CUDA 8 GPUs</th>
<th>rCUDA 8 GPUs</th>
<th>CUDA 4 GPUs</th>
<th>rCUDA 4 GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Higher is better
... 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
rCUDA is the enabling technology for …

- **High Throughput Computing**
  - Sharing remote GPUs makes applications to execute slower … **BUT** more throughput (jobs/time) is achieved
  - Datacenter administrators can **choose between HPC and HTC**

- **Green Computing**
  - GPU migration and application migration allow to devote just the required computing resources to the current load

- **More flexible system upgrades**
  - GPU and CPU updates become independent from each other. Adding GPU boxes to non GPU-enabled clusters is possible
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?

rCUDA is a development by Technical University of Valencia
Hands-on Session

Carlos Reaño
Technical University of Valencia
Spain
Outline

- 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:**
  - Node 1
  - Node 2
  - Network

- **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
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:

```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
```
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:
    ```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
    ```
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
    ```
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:
  - 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/5.5/1_Utilsities/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=$PATH:/usr/local/cuda/bin
    export LD_LIBRARY_PATH=$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/5.5/1.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`
  - Compile CUDA program using dynamic libraries:
    - `cd $HOME/NVIDIA_CUDA_Samples/5.5/1_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:

```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
```

- Compile CUDA program using dynamic libraries:

```bash
cd $HOME/NVIDIA_CUDA_Samples/5.5/1_Utils/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

- Run the CUDA program as usual:

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

Number of remote GPUs: 1, 2, 3...
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/5.5/1.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/5.5/1_Utilsites/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/5.5/1_Utilsites/deviceQuery
  make EXTRA_NVCCFLAGS=--cudart=shared
  ```

- Run the CUDA program as usual:

  ```
  ./deviceQuery
  ```

Very important!!!
Installing and using rCUDA

- 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/5.5/1Utilities/deviceQuery
make EXTRA_NVCCFLAGS=--cudart=shared
```

- Run the CUDA program as usual:

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

- Live demonstration:
  - deviceQuery
  - bandwidthTest
Installing and using rCUDA

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:

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

Run CUDA program using rCUDA over IB:

```
export RCUDAPROTO=IB
cd $HOME/NVIDIA_CUDA_Samples/5.5/1_Utilsities/bandwidthTest
./bandwidthTest
```
Starting rCUDA server using IB:

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

Tell rCUDA we want to use IB

Run CUDA program using rCUDA over IB:

```
export RCUDAPROTO=IB
cd $HOME/NVIDIA_CUDA_Samples/5.5/1_Utilities/bandwidthTest
./bandwidthTest
```

Also in the client!!
Starting rCUDA server using IB:

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

Run CUDA program using rCUDA over IB:

```
export RCUDAPROTO=IB
cd $HOME/NVIDIA_CUDA_Samples/5.5/1_Utils/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:

- **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

- 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
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=$PATH:/usr/local/cuda/bin`
- `export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDA1:$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`

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

- Export PATH:
  ```
  export PATH=$PATH:/usr/local/cuda/bin
  ```
- Export LD_LIBRARY_PATH:
  ```
  export LD_LIBRARY_PATH=$HOME/rCUDA/framework/rCUDAlib:$LD_LIBRARY_PATH
  ```
- Export RCUDA_DEVICE_COUNT:
  ```
  export RCUDA_DEVICE_COUNT=5
  ```
- Export RCUDA_DEVICE:
  ```
  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 RCUDADEVICE_4=rcu15:0
  ```

- Check configuration by running deviceQuery sample

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
```

- Check configuration by running deviceQuery sample

Location of each GPU
Sample scenarios:

- 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

- 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
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
Exercises

- Local CPU vs. Remote GPU using CUDA
- Local CPU vs. Remote GPU using CUDA Libs
- Aggregated BW of 4 local GPUs vs. 5 remote GPUs
- Multi-GPU App: 4 local GPUs vs. 5 remote GPUs
- Coordinate with attendees for sharing the same GPU at the same time
  - Performance decreases?