Remote GPU virtualization from the rCUDA point of view

Federico Silla
Technical University of Valencia
Spain
Remote GPU virtualization from the rCUDA point of view?
What is “remote GPU virtualization”?
It has to do with GPUs, obviously!
Basics of GPU computing

Application

CUDA libraries

GPU
Basics of GPU computing

Application

CUDA libraries
Remote GPU virtualization

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

Client side

Application

CUDA Runtime API

client engine

Software

Hardware

Network

Server side

server engine

CUDA libraries

GPU
Basics of remote GPU virtualization

Client side

Application

CUDA Runtime API

client engine

Software

Hardware

Network

Server side

server engine

CUDA libraries

GPU
Basics of remote GPU virtualization

Client side

Application

CUDA Runtime API

client engine

Server side

server engine

CUDA libraries

Software

Hardware

Network

GPU
Remote GPU virtualization envision

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

to the following one:
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 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.
Second concern with accelerated clusters

- Applications can only use the GPUs located within their node:
  - non-MPI multi-GPU applications running on a node cannot make use of the tremendous GPU resources available at other cluster nodes (even if they are idle)
One more concern with accelerated clusters

- Do applications \textbf{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 …
GPU usage of GPU-Blast

- Core Utilization
- Memory Utilization (accesses)

GPU assigned but not used

Power (W)

Time
GPU usage of CUDA-MEME

GPU utilization is far away from maximum
GPU usage of LAMMPS

- GPU assigned but not used
GPU allocation vs GPU utilization

- Normalized Workload Execution Time
- Normalized GPU Allocation Time
- GPU Utilization

GPUs assigned but not used
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

Two concurrent instances of GPU-Blast

First instance

Utilization

Memory Utilization (accesses)

Power (W)

Time
Sharing a GPU among jobs: GPU-Blast

Two concurrent instances of GPU-Blast

First instance

Second instance

Core Utilization
Memory Utilization (accesses)

Power (W)
Sharing a GPU among jobs

K20 GPU

- LAMMPS: 876 MB
- mCUDA-MEME: 151 MB
- BarraCUDA: 3319 MB
- MUMmerGPU: 2104 MB
- GPU-LIBSVM: 145 MB
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
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).
2nd

What is rCUDA?
A software technology that enables a more flexible use of GPUs in computing facilities.

rCUDA is a development by Technical University of Valencia.
rCUDA ... slower CUDA?
The main concern with rCUDA is the reduced bandwidth to the remote GPU.

No GPU

Network
Using InfiniBand networks
- CUDASW++

Bioinformatics software for Smith-Waterman protein database searches
Initial transfers with InfiniBand

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 with InfiniBand

Almost 100% of available BW

HPC Knowledge Meeting’16
rCUDA optimizations on applications

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

Outline

3rd
1: more GPUs for a single application

- 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

Logical connections: Interconnection Network
1: more GPUs for a single application

Detected 64 CUDA Capable device(s)

64 GPUs!
1: more GPUs for a single application

- Monte Carlo Multi-GPU (from NVIDIA samples)

FDR InfiniBand + NVIDIA Tesla K20

The chart shows the performance comparison between CUDA and rCUDA (red and blue bars) for different numbers of GPUs (1 to 14). Higher is better for Options per second, while Lower is better for Computation Time (ms).
2: busy CPU cores do not block GPUs

Interconnection Network

Physical configuration

Logical connections

Logical configuration
• 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

No GPU

GPU-enabled
Approach 2: augment the cluster with some rCUDA servers → all nodes can execute accelerated applications

GPU-enabled
3: cheaper cluster upgrade

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

16 nodes without GPU + 1 node with 4 GPUs
### More workloads for studying rCUDA+Slurm

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

Job granularity instead of GPU granularity
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

rCUDA server

GPU

KVM Guest Linux 1

rCUDA client

vGPU

vETH

IB

KVM Guest Linux n

rCUDA client

vGPU

vETH

IB

High performance network available

Low performance network available

HPC Knowledge Meeting’16
Overhead of rCUDA within VMs

**KVM**

<table>
<thead>
<tr>
<th>Application</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>LAMMPS</td>
<td>1.6%</td>
<td>2.5%</td>
<td>0.5%</td>
<td>0.07%</td>
</tr>
<tr>
<td>CUDA-MEME</td>
<td>6.1%</td>
<td>2.9%</td>
<td>0.7%</td>
<td>2.3%</td>
</tr>
<tr>
<td>CUDASW++</td>
<td>0.07%</td>
<td>0.7%</td>
<td>2.3%</td>
<td></td>
</tr>
<tr>
<td>GPU-BLAST</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

**Xen**

FDR InfiniBand + K20 !!
• GPUs can be shared among jobs running in remote clients
  • Job scheduler required for coordination
  • **Slurm** was selected
6: rCUDA at the cluster level

- Dual socket E5-2620v2 Intel Xeon + 32GB RAM + K20 GPU
- FDR InfiniBand based cluster
Applications for 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 (2 nodes) (167 seconds)
  - NAMD (4 nodes) (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
## Workloads for studying rCUDA+Slurm

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

**Execution Time (s)**

<table>
<thead>
<tr>
<th>Workload</th>
<th>CUDA</th>
<th>rCUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>8000</td>
<td>4000</td>
</tr>
<tr>
<td>Set 2</td>
<td>12000</td>
<td>7300</td>
</tr>
<tr>
<td>Set 1+2</td>
<td>9000</td>
<td>6300</td>
</tr>
</tbody>
</table>

**Energy (kWh)**

<table>
<thead>
<tr>
<th>Workload</th>
<th>CUDA</th>
<th>rCUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>4</td>
<td>3.2</td>
</tr>
<tr>
<td>Set 2</td>
<td>8</td>
<td>6.4</td>
</tr>
<tr>
<td>Set 1+2</td>
<td>6</td>
<td>4.8</td>
</tr>
</tbody>
</table>

**GPU Utilization**

<table>
<thead>
<tr>
<th>Workload</th>
<th>CUDA</th>
<th>rCUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>0.15</td>
<td>0.25</td>
</tr>
<tr>
<td>Set 2</td>
<td>0.35</td>
<td>0.55</td>
</tr>
<tr>
<td>Set 1+2</td>
<td>0.25</td>
<td>0.45</td>
</tr>
</tbody>
</table>
... in summary ...
Remote GPU virtualization with 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?

rCUDA is a development by Technical University of Valencia