Increasing cluster performance by combining rCUDA with Slurm

Federico Silla
Technical University of Valencia
Spain
rCUDA ... what's that?
Basics of CUDA

Application

CUDA libraries

GPU

Multiple GPUs in a server
No GPU

Network
A **software** technology that enables a more flexible use of **GPUs** in computing facilities.
Basics of rCUDA

Client side | Server side

Application

CUDA Runtime API

client engine

server engine

CUDA libraries

Network

Software

Hardware

GPU
Basics of rCUDA
Basics of rCUDA
rCUDA allows a new vision of a GPU deployment, moving from the usual cluster configuration:

Physical configuration

Logical connections

Logical configuration
Two questions:

- Why should we need rCUDA?
- rCUDA ... slower CUDA?
Two questions:

• Why should we need rCUDA?
• rCUDA ... slower CUDA?
The main concern with rCUDA is the reduced bandwidth to the remote GPU.
Initial transfers within rCUDA

H2D pageable

D2H pageable

H2D pinned

D2H pinned
• CUDASW++

Bioinformatics software for Smith-Waterman protein database searches

Performance depending on network
Optimized transfers within rCUDA

H2D pageable

D2H pageable

Almost 100% of available BW

H2D pinned

D2H pinned

Copy Size (MB)

Bandwidth (MB/s)

CUDA K20
rCUDA FDR Orig
CUDA K40
rCUDA EDR Orig
Cuda K40
rCUDA FDR Opt
rCUDA EDR Opt

Copy Size (MB)

Bandwidth (MB/s)

Almost 100% of available BW
rCUDA optimizations on applications

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

Lower is better
Two questions:

• Why should we need rCUDA?

• rCUDA ... slower CUDA?
rCUDA improves cluster performance
Test bench for studying rCUDA+Slurm

- 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)
  - LAMMPPS (15 seconds; 4 GPUs; 876 MB)
  - MCUDA-MEME (165 seconds; 4 GPUs; 151 MB)
  - GROMACS (2 nodes) (167 seconds) **Set 1**
  - NAMD (4 nodes) (11 minutes) **Set 2**
  - BarraCUDA (10 minutes; 1 GPU; 3319 MB)
  - GPU-LIBSVM (5 minutes; 1GPU; 145 MB)
  - MUMmerGPU (5 minutes; 1GPU; 2804 MB)

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

Non-GPU

Short execution time

Long execution time
<table>
<thead>
<tr>
<th>Application</th>
<th>Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set 1</td>
</tr>
<tr>
<td>GPU-Blast</td>
<td>112</td>
</tr>
<tr>
<td>LAMMPS</td>
<td>88</td>
</tr>
<tr>
<td>mCUDA-MEME</td>
<td>99</td>
</tr>
<tr>
<td>GROMACS</td>
<td>101</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td></td>
</tr>
<tr>
<td>MUMmerGPU</td>
<td></td>
</tr>
<tr>
<td>GPU-LIBSVM</td>
<td></td>
</tr>
<tr>
<td>NAMD</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>400</strong></td>
</tr>
</tbody>
</table>
Performance of rCUDA+Slurm (I)

### Execution Time (s)
- CUDA
- rCUDA ex
- rCUDA sh

### Workload
- Set 1
- Set 2
- Set 1+2

### Energy (kWh)
- CUDA
- rCUDA ex
- rCUDA sh

### GPU Utilization
- CUDA
- rCUDA ex
- rCUDA sh

### Workload
- Set 1
- Set 2
- Set 1+2
## Workloads for studying rCUDA+Slurm (II)

<table>
<thead>
<tr>
<th>Application</th>
<th>Set 2</th>
<th>Set 1+2</th>
<th>New Set 2</th>
<th>New Set 1+2</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU-Blast</td>
<td>57</td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>LAMMPS</td>
<td>52</td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>mCUDA-MEME</td>
<td>55</td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>GROMACS</td>
<td>47</td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td>112</td>
<td>51</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>MUMmerGPU</td>
<td>88</td>
<td>52</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>GPU-LIBSVM</td>
<td>99</td>
<td>37</td>
<td>160</td>
<td>80</td>
</tr>
<tr>
<td>NAMD</td>
<td>101</td>
<td>49</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>400</strong></td>
<td><strong>400</strong></td>
<td><strong>400</strong></td>
<td><strong>400</strong></td>
</tr>
</tbody>
</table>
Performance of rCUDA+Slurm (II)

- **Execution Time (s)**
  - CUDA
  - rCUDA ex
  - rCUDA sh
  - Workloads: New Set 2, New Set 1+2

- **Energy (kWh)**
  - CUDA
  - rCUDA ex
  - rCUDA sh
  - Workloads: New Set 2, New Set 1+2

- **GPU Utilization**
  - CUDA
  - rCUDA ex
  - rCUDA sh
  - Workloads: New Set 2, New Set 1+2
Why does rCUDA improve cluster performance?
1st reason for improved performance

- 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.

---

**Interconnection Network**

- **node 1**: GPU x RAM
- **node 2**: GPU x RAM
- **node 3**: GPU x RAM
- **node n**: GPU x RAM

**Diagram Elements**
- **CPU**: Central Processing Unit
- **RAM**: Random Access Memory
- **GPUs**: Graphics Processing Units
- **PCIe**: Peripheral Component Interconnect Express
- **Network**: Interconnect network for communication between nodes

---

HPC Advisory Council Switzerland Conference 2016
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.
2nd reason for improved performance (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 MPI application using just one CPU core per node may keep part of the cluster busy.
3rd reason for improved performance

- 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 …
GPU usage of GPU-Blast

- Core Utilization
- Memory Utilization (accesses)

GPU assigned but not used

Power (W)
GPU usage of CUDA-MEME

GPU utilization is far away from maximum
GPU usage of LAMMPS

- GPU assigned but not used

Graph showing Core Utilization and Memory Utilization (accesses) over time, along with Power consumption.
GPU allocation vs 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
Sharing a GPU among jobs: GPU-Blast

First instance

Second instance

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

- LAMMPS: 876 MB
- mCUDA-MEME: 151 MB
- BarraCUDA: 3319 MB
- MUMmerGPU: 2104 MB
- GPU-LIBSVM: 145 MB
Other reasons for using rCUDA?
Cheaper 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
Cheaper cluster upgrade

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

GPU-enabled
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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WL 1</td>
</tr>
<tr>
<td>GPU-Blast</td>
<td>41</td>
</tr>
<tr>
<td>LAMMPS short</td>
<td>39</td>
</tr>
<tr>
<td>LAMMPS long 2p</td>
<td>20</td>
</tr>
<tr>
<td>LAMMPS long 4p</td>
<td>20</td>
</tr>
<tr>
<td>mCUDA-MEME short</td>
<td>39</td>
</tr>
<tr>
<td>mCUDA-MEME long 2p</td>
<td>20</td>
</tr>
<tr>
<td>mCUDA-MEME long 4p</td>
<td>20</td>
</tr>
<tr>
<td>GROMACS</td>
<td>40</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td>40</td>
</tr>
<tr>
<td>MUMmerGPU</td>
<td>41</td>
</tr>
<tr>
<td>GPU-LIBSVM</td>
<td>40</td>
</tr>
<tr>
<td>NAMD</td>
<td>40</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>400</strong></td>
</tr>
</tbody>
</table>
Performance

![Bar charts comparing CUDA and rCUDA performance metrics across different workloads (WL 1 and WL 2).]

Execution Time (s):
- CUDA: 80000, rCUDA: 20000
  - WL 1: -68%
  - WL 2: -60%

GPU Utilization:
- CUDA: 0.4, rCUDA: 0.3
  - WL 1: +131%
  - WL 2: +119%

Energy (kWh):
- CUDA: 40, rCUDA: 10
  - WL 1: -63%
  - WL 2: -56%
Outline

Additional reasons for using rCUDA?
#1: More GPUs for a single application

Detected 64 CUDA Capable device(s)

64 GPUs!
#1: More GPUs for a single application

- MonteCarlo Multi-GPU (from NVIDIA samples)

FDR InfiniBand + NVIDIA Tesla K20

Higher is better

Lower is better
#2: Virtual machines can share GPUs

- The GPU is assigned by using PCI passthrough exclusively to a single virtual machine
- Concurrent usage of the GPU is not possible

**Computer hosting several KVM virtual machines**
#2: Virtual machines can share GPUs

Computer hosting several KVM virtual machines

KVM Host Linux

SW BRIDGE

Gb ETH

IB PF

Host HW

InfiniBand Fabric

rCUDA server

GPU

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

Virtual machines can share GPUs
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)

Migration is performed at GPU granularity
Job granularity instead of GPU granularity
... 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?

rCUDA is a development by Technical University of Valencia