rCUDA: hybrid CPU-GPU clusters

Federico Silla
Technical University of Valencia
Spain
1. Hybrid CPU-GPU clusters
2. Concerns with hybrid clusters
3. One possible solution: virtualize GPUs!
4. rCUDA …what’s that?
5. What can I do with rCUDA?
6. Additional activities
1. Hybrid CPU-GPU clusters
2. Concerns with hybrid clusters
3. One possible solution: virtualize GPUs!
4. rCUDA …what’s that?
5. What can I do with rCUDA?
6. Additional activities
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
Introducing GPUs into clusters
Basics of GPU computing

Basic behavior of CUDA

Application

CUDA libraries

GPU

GPU

GPU

GPU

GPU

HPC ADMINTECH 2017
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
1. Hybrid CPU-GPU clusters
2. Concerns with hybrid clusters
3. One possible solution: virtualize GPUs!
4. rCUDA …what’s that?
5. What can I do with rCUDA?
6. Additional activities
First concern with accelerated clusters

- 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
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

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

All these GPUs 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 …
GPU usage of GPU-Blast

The graph illustrates the utilization of GPU resources over time. The top graph shows the core utilization and memory utilization (accesses) over time. The bottom graph displays the power consumption in watts.

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

The red arrows indicate periods where GPUs were assigned but not used:
- 00:00:05 to 00:00:10
- 00:00:33 to 00:00:40

These periods highlight inefficiencies in resource allocation and possible opportunities for optimization.
GPU usage of LAMMPS

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

**GPU assigned but not used**
Sharing a GPU among jobs

K20 GPU

- LAMMPS: 876 MB
- mCUDA-MEME: 151 MB
- BarraCUDA: 3319 MB
- MUMmerGPU: 2104 MB
- GPU-LIBSVM: 145 MB
1. Hybrid CPU-GPU clusters
2. Concerns with hybrid clusters
3. **One possible solution: virtualize GPUs!**
4. rCUDA …what’s that?
5. What can I do with rCUDA?
6. Additional activities
Remote GPU virtualization allows a new vision of a GPU deployment, moving from the usual cluster configuration:

**Logical configuration**

**Physical configuration**

Interconnection Network
Remote GPU virtualization envision

Without GPU virtualization

With GPU virtualization

Virtualized remote GPUs

GPU virtualization allows all nodes to share all GPUs
Basics of remote GPU virtualization

Client side | Server side

Application

CUDA API

client engine

server engine

CUDA libraries

Network

Software

Hardware
1. Hybrid CPU-GPU clusters
2. Concerns with hybrid clusters
3. One possible solution: virtualize GPUs!
4. rCUDA …what’s that?
5. What can I do with rCUDA?
6. Additional activities
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 8.0)</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

HPC ADMINTECH 2017
P2P copy support within rCUDA

CUDA P2P
rCUDA P2P

FDR InfiniBand + K20 !!
Application performance with rCUDA

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

Lower is better
1. Hybrid CPU-GPU clusters
2. Concerns with hybrid clusters
3. One possible solution: virtualize GPUs!
4. rCUDA …what’s that?
5. What can I do with rCUDA?
6. Additional activities
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
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 (644213264 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 (dim) 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: 65535 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

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: GPU task migration

Job granularity instead of GPU granularity
• The GPU-Blast application is migrated up to 5 times among K40 GPUs
  • The aggregated volume of GPU data is 1300 MB (consisting of 9 memory regions)

The “Reference” line is the execution time of the application when using CUDA with a local GPU and without any migration.
3: 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
3: virtual machines can easily access GPUs

High performance network available
Low performance network available
3: virtual machines can easily access GPUs

FDR InfiniBand + K20 !!
rCUDA can provide advanced support

- rCUDA can be used to provide VMs with concurrent access to one or more GPUs with advanced support

- Why is more support for VMs needed?
  - Because shared GPUs may run out of memory and cause applications to abort
  - VMs blindly allocate and release GPU memory independently from each other without any kind of coordination among them

Computer hosting several KVM virtual machines

[KVM diagram with labels: KVM Guest Linux 1, vETH, SW BRIDGE, Gb ETH, GPU, KVM Guest Linux 2, vETH, ... KVM Guest Linux n, vETH, KVM Host Linux, Host HW]
rCUDA can provide advanced support

- rCUDA can be used to provide VMs with concurrent access to one or more GPUs with advanced support.

- Why is more support for VMs needed?
  - Because shared GPUs may run out of memory and cause applications to abort.
  - VMs blindly allocate and release GPU memory independently from each other without any kind of coordination among them.

Computer hosting several KVM virtual machines:

- KVM Guest Linux 1
- KVM Guest Linux 2
- KVM Guest Linux 3
- ... KVM Guest Linux n

- vETH
- SW BRIDGE
- 4GB
- Gb ETH
- GPU
- K40 (12GB)

Host HW

KVM Host Linux
rCUDA can provide advanced support

- rCUDA can be used to provide VMs with concurrent access to one or more GPUs with advanced support

- Why is more support for VMs needed?
  - Because shared GPUs may run out of memory and cause applications to abort
  - VMs blindly allocate and release GPU memory independently from each other without any kind of coordination among them
rCUDA can provide advanced support

- rCUDA can be used to provide VMs with concurrent access to one or more GPUs with advanced support.

- Why is more support for VMs needed?
  - Because shared GPUs may run out of memory and cause applications to abort.
  - VMs blindly allocate and release GPU memory independently from each other without any kind of coordination among them.

![Diagram of KVM virtual machines and GPUs](image)
rCUDA can provide advanced support

- rCUDA can be used to provide VMs with concurrent access to one or more GPUs with advanced support

- Why is more support for VMs needed?
  - Because shared GPUs may run out of memory and cause applications to abort
  - VMs blindly allocate and release GPU memory independently from each other without any kind of coordination among them

![Diagram of computer hosting several KVM virtual machines](image)
rCUDA can provide advanced support

- rCUDA can be used to provide VMs with concurrent access to one or more GPUs **with advanced support**

- **Why is more support for VMs needed?**
  - Because **shared GPUs may run out of memory** and cause applications to abort
  - VMs blindly allocate and release GPU memory independently from each other without any kind of coordination among them

**Conclusion**: In addition to access the GPU, it is required to coordinate how VMs access the accelerator
4: 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**”
4: cheaper cluster upgrade

Lower is better

Execution Time (s)

Energy (kWh)

Average GPU Utilization

Lower is better

Higher is better

Large Workload

CUDA  rCUDA

Large Workload

CUDA  rCUDA

Large Workload

CUDA  rCUDA
• GPUs can be shared among jobs running in remote clients
  • Job scheduler required for coordination
  • Slurm was enhanced
5: increased cluster performance with Slurm

Higher is better

Lower is better

Lower is better

Workload size

Execution Time (s)

Energy (kWh)

Average GPU Utilization

CUDA  rCUDA

Large

Large

Large

5: increased cluster performance with Slurm

Higher is better

Lower is better

Lower is better

Workload size

Execution Time (s)

Energy (kWh)

Average GPU Utilization

CUDA  rCUDA

Large

Large

Large

5: increased cluster performance with Slurm

Higher is better

Lower is better

Lower is better

Workload size

Execution Time (s)

Energy (kWh)

Average GPU Utilization

CUDA  rCUDA

Large

Large

Large

5: increased cluster performance with Slurm

Higher is better

Lower is better

Lower is better

Workload size

Execution Time (s)

Energy (kWh)

Average GPU Utilization

CUDA  rCUDA

Large

Large

Large

5: increased cluster performance with Slurm

Higher is better

Lower is better

Lower is better

Workload size

Execution Time (s)

Energy (kWh)

Average GPU Utilization

CUDA  rCUDA

Large

Large

Large

5: increased cluster performance with Slurm

Higher is better

Lower is better

Lower is better

Workload size

Execution Time (s)

Energy (kWh)

Average GPU Utilization

CUDA  rCUDA

Large

Large

Large
1. Hybrid CPU-GPU clusters
2. Concerns with hybrid clusters
3. One possible solution: virtualize GPUs!
4. rCUDA …what’s that?
5. What can I do with rCUDA?
6. Additional activities
Additional activities

• Latest rCUDA release @ SC’16:
  • Support for CUDA 8.0
  • P2P memory copies
  • Improved support for cuDNN

• On going developments:
  • Improved support for deep learning
  • Support for 64-bit ARM architectures
  • Support for Matlab
  • Support for popular graphics rendering suites

• Developments we are considering
  • Support for PowerPC architectures

• ... and much more!!

rCUDA is a development by Technical University of Valencia
Get a free copy of rCUDA at
http://www.rcuda.net
More than 750 requests world wide

@rcuda_

rCUDA is a development by Technical University of Valencia
Thanks!

Questions?

rCUDA is a development by Technical University of Valencia