Speeding up the execution of numerical computations and simulations with rCUDA

José Duato
Universidad Politécnica de Valencia
Spain
Outline

1. Introduction to GPU computing
2. What is remote GPU virtualization?
3. Why is remote GPU virtualization needed?
4. Frameworks for remote GPU virtualization
5. Cons of remote GPU virtualization
6. Pros of remote GPU virtualization
1st

Introduction to GPU computing
Current computing needs

- Many applications require huge computing power. This demand increases over time.
- Applications can be accelerated to reduce execution time (parallel processing, vector instructions).
- GPU computing has experienced a remarkable growth in the last years.
GPUs reduce energy and time

blaslp –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

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
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

- **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
Basics of GPU computing

Basic behavior of CUDA

Application

CUDA libraries

GPU

NVIDIA CUDA

OpenCL
Basics of GPU computing

```c
#include <cuda.h>
#include <stdio.h>

const int N = 8;

__global__ void my_gpu_function (int *a, int *b)
{
    b[threadIdx.x] = a[threadIdx.x] * 2;
}

int main ( )
{
    int a[N] = {0, 1, 2, 3, 4, 5, 6, 7};
    int *ad, *bd;
    const int isize = N * sizeof(int);
    // Perform some computations in the CPU
    ... CPU code ... CPU code ...

    // Allocate GPU memory
    cudaMalloc ((void**)&ad, isize);
    cudaMalloc ((void**)&bd, isize);

    // Copy data to GPU memory
    cudaMemcpy(ad, a, isize, cudaMemcpyHostToDevice);

    // Execute function in the GPU
    my_gpu_function<<<1, N>>>(ad, bd);

    // Copy results from GPU memory
    cudaMemcpy(b, bd, isize, cudaMemcpyDeviceToHost);

    // Free GPU memory
    cudaFree(ad);
    cudaFree(bd);
    return 0;
}
```
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
What is "remote GPU virtualization"?
It has to do with GPUs, obviously!
Remote GPU virtualization

A software technology that enables a more flexible use of GPUs in computing facilities.

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Same source code for using virtual GPUs

Application source code does not change.
The change is linking against the remote GPU virtualization libraries instead of using
the original CUDA libraries

```c
#include <cuda.h>
#include <stdio.h>

const int N = 8;

__global__ void my_gpu_function (int *a, int *b)
{
    b[threadIdx.x] = a[threadIdx.x] * 2;
}

int main ( )
{
    int a[N] = {0, 1, 2, 3, 4, 5, 6, 7};
    int *ad, *bd;
    const int isize = N * sizeof(int);
    // Perform some computations in the CPU
    … CPU code … CPU code …

    // Allocate GPU memory
    cudaMalloc ((void**)&ad, isize);
    cudaMalloc ((void**)&bd, isize);

    // Copy data to GPU memory
    cudaMemcpy(ad, a, isize, cudaMemcpyHostToDevice);

    // Execute function in the GPU
    my_gpu_function<<<1, N>>>(ad, bd);

    // Copy results from GPU memory
    cudaMemcpy(b, bd, isize, cudaMemcpyDeviceToHost);

    // Free GPU memory
    cudaFree(ad);
    cudaFree(bd);
    return 0;
}
```
Basics of remote GPU virtualization

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Basics of remote GPU virtualization

- **Client side**
  - Application

- **Server side**
  - CUDA Runtime API

- **client engine**
  - Software
  - Hardware

- **server engine**
  - CUDA libraries

- **Network**
Remote GPU virtualization envision

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

---

**Physical configuration**

**Logical configuration**

---

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Remote GPU virtualization envision

Without GPU virtualization

With GPU virtualization

GPU virtualization allows all nodes to share all GPUs

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
3rd

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
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
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

- 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
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

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 …

---

The diagram illustrates the interconnection network between nodes in an accelerated cluster. Each node contains a CPU, RAM, and a GPU connected through PCIe interfaces. The network interconnects these nodes, allowing for data exchange and computation distribution across the cluster.
Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017

**GPU usage of GPU-Blast**

![Graph showing GPU usage over time with annotations: GPU assigned but not used at specific intervals.](image-url)
GPU usage of CUDA-MEME

GPU utilization is far away from maximum
GPU usage of LAMMPS

Graph showing utilization over time:
- Core Utilization
- Memory Utilization (accesses)

Note: GPU assigned but not used.
Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017

GPU usage of CUDASW

- GPU assigned but not used

Graphs showing Core Utilization and Memory Utilization (accesses) over time, with a peak at 00:00:04 and a drop at 00:00:14.
GPU allocation vs GPU utilization

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

GPUs assigned but not used

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Sharing a GPU among jobs: GPU-Blast

First instance

Two concurrent instances of GPU-Blast

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Sharing a GPU among jobs: GPU-Blast

One instance required about 51 seconds

Two concurrent instances of GPU-Blast

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Sharing a GPU among jobs: GPU-Blast

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

One instance required about 51 seconds

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

K20 GPU

- LAMMPS: 876 MB
- mCUDA-MEME: 151 MB
- BarraCUDA: 3319 MB
- MUMmerGPU: 2104 MB
- GPU-LIBSVM: 145 MB
• 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

Interconnection Network

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
• 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.

Idle CPU cores in nodes executing accelerated applications could be used for other GPU applications.
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)
Remote GPU virtualization

Interconnection Network

Physical configuration

Logical connections

Logical configuration
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.
In summary … 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
4th

Frameworks for remote GPU virtualization
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

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
rCUDA uses optimized transfers

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

---

**Diagram:**

- **Client Computer**
  - main memory
  - PCIe
  - IB

- **Network**
  - Stage 1
  - Stage 2

- **rCUDA Server**
  - IB
  - PCIe
  - GPU

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Moving data WITHOUT a pipeline

Pageable memory area to be copied to remote GPU

Destination memory area in the remote GPU
Moving data WITHOUT a pipeline

Pageable memory area to be copied to remote GPU

Memory area copied to network buffers for RDMA

Destination memory area in the remote GPU
Moving data WITHOUT a pipeline

Pageable memory area to be copied to remote GPU

Memory area copied to network buffers for RDMA

Memory area arrives at network buffers at remote server

Destination memory area in the remote GPU

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Moving data WITHOUT a pipeline

Pageable memory area to be copied to remote GPU

Memory area copied to network buffers for RDMA

Memory area arrives at network buffers at remote server

Destination memory area in the remote GPU

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Moving data WITHOUT a pipeline

Main memory to network buffers

Network from client to remote server

Network buffers to GPU memory

Transmission time without pipeline

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Moving data WITH a pipeline

CLIENT NODE

- Main memory
- CPU
- chipset
- InfiniBand

SERVER NODE

- Main memory
- CPU
- chipset
- GPU
- GPU memory

Pageable memory area to be copied to remote GPU

Destination memory area in the remote GPU
Moving data WITH a pipeline

**CLIENT NODE**

- Main memory
- CPU
- InfiniBand
- chipset

**SERVER NODE**

- Main memory
- CPU
- InfiniBand
- chipset
- GPU
- GPU memory

Pageable memory area to be copied to remote GPU

Memory area is split into chunks (pipeline blocks)

Destination memory area in the remote GPU
Moving data WITH a pipeline

Pageable memory area to be copied to remote GPU

Only two small network buffers at source and destination are required

Destination memory area in the remote GPU

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Moving data WITH a pipeline

Transmission starts by moving forward the first block

Pageable memory area to be copied to remote GPU

Destination memory area in the remote GPU
Moving data WITH a pipeline

Pageable memory area to be copied to remote GPU

2 pipeline stages concurrently move data forward

Destination memory area in the remote GPU
Moving data WITH a pipeline

The three stages of the pipeline are concurrently transferring data forward.
Moving data WITH a pipeline

CLIENT NODE

SERVER NODE

Pageable memory area to be copied to remote GPU

Destination memory area in the remote GPU

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Moving data WITH a pipeline

Pageable memory area to be copied to remote GPU

Destination memory area in the remote GPU
Moving data WITH a pipeline

Pageable memory area to be copied to remote GPU

Destination memory area in the remote GPU
Moving data WITH a pipeline

Pageable memory area to be copied to remote GPU

Destination memory area in the remote GPU

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
Moving data without a pipeline

Management of data transfer is much more complex

Transmission time without pipeline

Transmission time with pipeline

Improvement

Main memory to network buffers

Network from client to remote server

Network buffers to GPU memory

Without pipeline

Main memory to network buffers

Network from client to remote server

Network buffers to GPU memory

Transmission time without pipeline

Transmission time with pipeline

Improvement
Basic performance analysis

- Pipeline block size for InfiniBand Connect-IB

- NVIDIA Tesla K40; Mellanox Connect-IB + SX6036 Mellanox switch

<table>
<thead>
<tr>
<th>Copy Size (MB)</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 port</td>
<td></td>
</tr>
<tr>
<td>2 ports</td>
<td></td>
</tr>
</tbody>
</table>
How applications use 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 assigned to application

-bash-4.1$ export | grep DEVICE
declare -x RCUDA_DEVICE_0="mlxc2:0"
declare -x RCUDA_DEVICE_1="mlxc3:0"
declare -x RCUDA_DEVICE_2="mlxc6:0"
declare -x RCUDA_DEVICE_3="mlxc7:0"
declare -x RCUDA_DEVICE_4="mlxc8:0"
declare -x RCUDA_DEVICE_5="mlxc9:0"
declare -x RCUDA_DEVICE_6="mlxc10:0"
declare -x RCUDA_DEVICE_7="mlxf2:0"
declare -x RCUDA_DEVICE_COUNT="8"
-bash-4.1$
Cons of “remote GPU virtualization”
The main drawback of remote GPU virtualization is the reduced bandwidth to the remote GPU.
Using InfiniBand networks

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
**Performance of rCUDA**

- **CUDASW++**
  
  Bioinformatics software for Smith-Waterman protein database searches

![Graph showing performance of rCUDA](image)

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

---

*Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017*
6th

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**

- Only the GPUs in the node can be provided to the application
- Many GPUs in the cluster can be provided to the 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

64 GPUs!

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
1: more GPUs for a single application

- Monte Carlo Multi-GPU (from NVIDIA samples)

**FDR InfiniBand + NVIDIA Tesla K20**

**Diagram:**
- Bars representing CUDA and rCUDA for different numbers of GPUs.
- Higher is better for options per second.
- Lower is better for computation time (in ms).

**Legend:**
- CUDA
- rCUDA
2: busy CPU cores do not block GPUs

Interconnection Network

Physical configuration

Logical connections

Logical configuration

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
3: easier cluster upgrade

- 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?
**Approach 1:** augment the cluster with some CUDA GPU-enabled nodes → only those GPU-enabled nodes can execute accelerated applications.
3: easier cluster upgrade

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

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

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
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; 1GPU; 145 MB)
- MUMmerGPU (5 minutes; 1GPU; 2804 MB)
- GROMACS (167 seconds)
- NAMD (11 minutes)
### 3: easier cluster upgrade

<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>400</td>
</tr>
</tbody>
</table>
3: easier cluster upgrade

-68% - 60% execution time reduction
+131% + 119% increased GPU utilization
-63% - 56% energy reduction
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

Speeding up the execution of numerical computations and simulations. ICETE/PECCS 2017
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?