Creating new horizons with rCUDA: the power of remote GPU virtualization

Federico Silla
Universitat Politècnica de València
Spain
GPUs are great!
Basics of GPU computing (single node)

Basic behavior of CUDA
Basics of GPU computing (single node)

GPUs can only be used within the node they are attached to.

Basic behavior of CUDA
A GPU-enabled cluster is a set of independent self-contained nodes that share nothing among them:

- MPI is required for aggregating resources within the cluster (included GPUs)

Using GPUs across the cluster

Interconnection Network
GPUs are great!

How can we make an even better usage of GPUs?
How can we make an even better usage of GPUs?

Which characteristics do we miss from GPUs?
How can we make an even better usage of GPUs?

Which characteristics do we miss from GPUs?

1. Many GPUs in a single box
2. Easily sharing a given GPU (or GPUs)
Characteristics missing in GPUs

1. Why many GPUs in a single box

Traditionally, in order to use many GPUs, applications had to use MPI:

- GPUs can only be used within the node they are attached to
- Nothing is directly shared among nodes (MPI required for aggregating computing resources within the cluster)

A non-MPI application running in this node can only use the GPUs in this node

Interconnection Network
Characteristics missing in GPUs

1. Many GPUs in a single box
1. Many GPUs in a single box

The amount of GPUs is limited by the physical space inside the node
1. Having many GPUs in a single box

MonteCarlo multi-GPU program running in 10 NVIDIA Tesla K40 GPUs
Characteristics missing in GPUs

1. Having many GPUs in a single box

![CUDA Device Query Output]

- Detected 64 CUDA Capable device(s)
- Device: "Tesla M2090"
- CUDA Driver Version / Runtime Version: 5.0 / 5.0
- 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 54
- 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)

64 GPUs !!
1. Many GPUs in a single box

How many GPUs are “many GPUs in a single box”?
1. Many GPUs in a single box

How many GPUs are “many GPUs in a single box”?

As many GPUs as they can be installed in the cluster

Interconnection Network
How can we make an even better usage of GPUs?

Which characteristics do we miss from GPUs?

1. Many GPUs in a single box
2. Easily sharing a given GPU (or GPUs)
2. Easily sharing a given GPU

Why should we be interested on sharing GPUs among applications?
GPU usage of GPU-Blast

NVIDIA Tesla K20 GPU
GPU usage of CUDA-MEME

GPU utilization is far away from maximum

NVIDIA Tesla K20 GPU
GPU usage of LAMMPS

- GPU assigned but not used
- NVIDIA Tesla K20 GPU
Sharing a GPU among jobs: GPU-Blast

One instance required about 51 seconds

Two concurrent instances of GPU-Blast

Core Utilization
Memory Utilization ( accesses )
Sharing a GPU among jobs: GPU-Blast

First instance

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

Two concurrent instances of GPU-Blast

First instance

Second instance

Utilization vs. Time

Power (W) vs. Time
Sharing a GPU among applications

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

K20 GPU (5GB memory)
Sharing a GPU among applications

K20 GPU (5GB memory)

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

The main concern for sharing a GPU is the memory limitation.
How can we make an even better usage of GPUs?

Which characteristics do we miss from GPUs?

1. Many GPUs in a single box
2. Easily sharing a given GPU (or GPUs)
Easily sharing a GPU among VMs

- A GPU is assigned to a VM by using PCI passthrough.
- Assignment is done exclusively to a single virtual machine. Concurrent usage of the GPU is not possible.
Easily sharing a GPU among VMs

Computer hosting several KVM virtual machines

KVM Host Linux
SW BRIDGE

Gb ETH
IB PF

KVM Guest Linux 1

rCUDA client
vGPU
vETH
IB

PCI PT

IB VF

InfiniBand Fabric

rCUDA server
GPU

Low performance network available

High performance network available

Computer hosting several KVM virtual machines

KVM Host Linux
SW BRIDGE

Gb ETH

KVM Guest Linux 1

rCUDA client
vGPU
vETH

rCUDA server
GPU

VM Guest Linux n

rCUDA client
vGPU
vETH
Characteristics missing in GPUs

Which characteristics do we miss from GPUs?

1. Many GPUs in a single box
2. Easily sharing a given GPU (or GPUs)
Which characteristics do we miss from GPUs?

1. Many GPUs in a single box
2. Easily sharing a given GPU (or GPUs)

The remote GPU virtualization technique can efficiently address these concerns
Characteristics missing in GPUs

The remote GPU virtualization technique can efficiently address these concerns.
Characteristics missing in GPUs

The remote **GPU virtualization** technique can efficiently address these concerns.
The remote GPU virtualization technique can efficiently address these concerns.
The remote **GPU virtualization** technique can efficiently address these concerns.
What is “remote GPU virtualization”? 
Basics of GPU computing

Application

CUDA libraries

Basic behavior of CUDA
Basics of GPU computing

Application

CUDA libraries

HPC Advisory Council Perth Conference 2017
Remote GPU virtualization

No GPU

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

rCUDA is a development by Technical University of Valencia.
rCUDA is a development by Universitat Politècnica de València, Spain.
Basics of rCUDA

rCUDA is a development by Universitat Politècnica de València, Spain
Remote GPU virtualization allows a new vision of a GPU deployment, moving from the usual cluster configuration:

![Physical configuration diagram]

...to the following one:

![Logical configuration diagram]
Performance of rCUDA

Guy Kawasaki, marketing specialist and Silicon Valley venture capitalist

“Ideas Are Easy, Implementation Is Hard”
Performance of rCUDA

CPU to GPU
Higher is better

GPU to CPU
Performance of rCUDA

CPU to GPU

Higher is better

GPU to CPU
Performance of rCUDA

CPU to GPU

Higher is better

GPU to CPU
Performance of rCUDA

CUDA

rCUDA

rCUDA scenario 1

rCUDA scenario 2
"Ideas Are Easy, Implementation Is Hard"
Guy Kawasaki, marketing specialist and Silicon Valley venture capitalist
Performance of applications using rCUDA

K40 GPUs and EDR InfiniBand

MonteCarlo multi-GPU program running in 10 NVIDIA Tesla K40 GPUs
Performance of applications using rCUDA

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 (6442123254 bytes)
- (32) 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
- Cache Size: 786432 bytes
- Texture Dimension Size (x,y,z): 1D=(65536), 2D=(65536,65535), 3D=(2048,2048,2048)
- 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) >
Performance of applications using rCUDA

- K20 GPU and FDR InfiniBand
- K40 GPU and EDR InfiniBand

Lower is better
Performance of applications using rCUDA

EDR InfiniBand and P100 GPU

BarraCUDA

CUDA-MEME

Lower is better
Get a free copy of rCUDA at http://www.rcuda.net
More than 850 requests world wide

@rcuda_

rCUDA is a development by Technical University of Valencia
rCUDA is a development by Technical University of Valencia
Thanks! Questions?

rCUDA is a development by Technical University of Valencia