

# Integración de tecnologías de virtualización de GPUs en el planificador de recursos SLURM

---

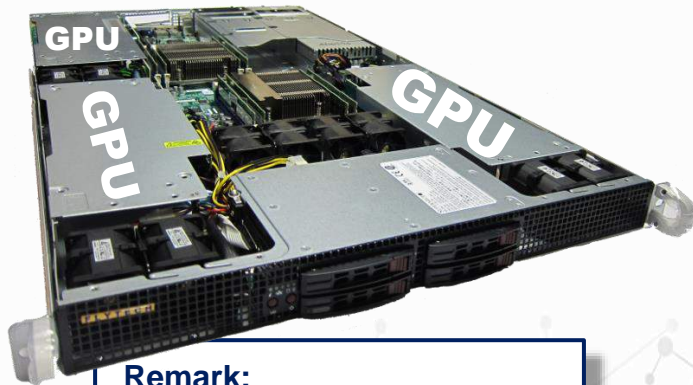
Federico Silla

Universitat Politècnica de València

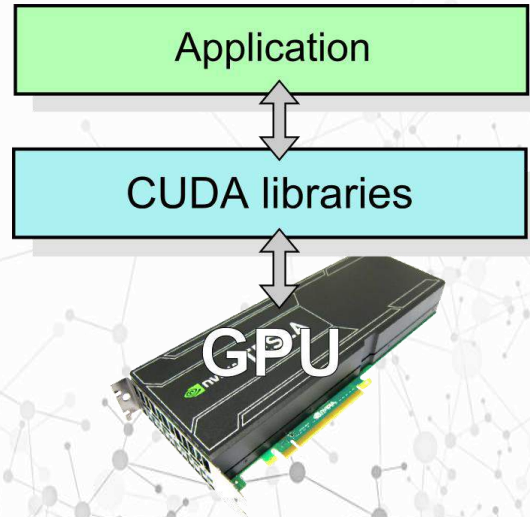
## What is rCUDA?

# Basics of GPU computing

Basic behavior of CUDA

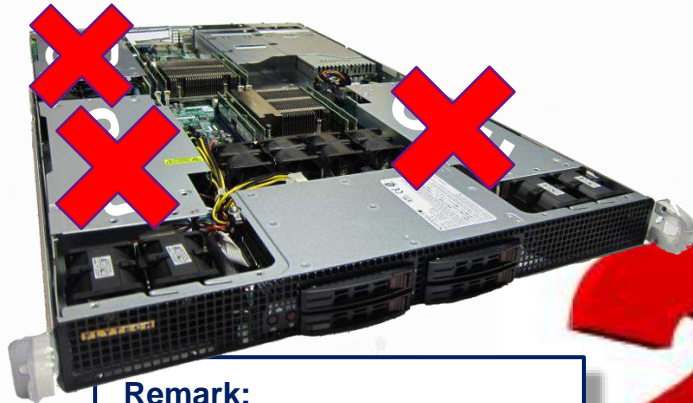


**Remark:**  
GPUs can only be used within  
the node they are attached to

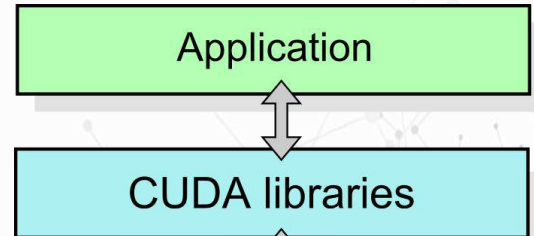


# Basics of GPU computing

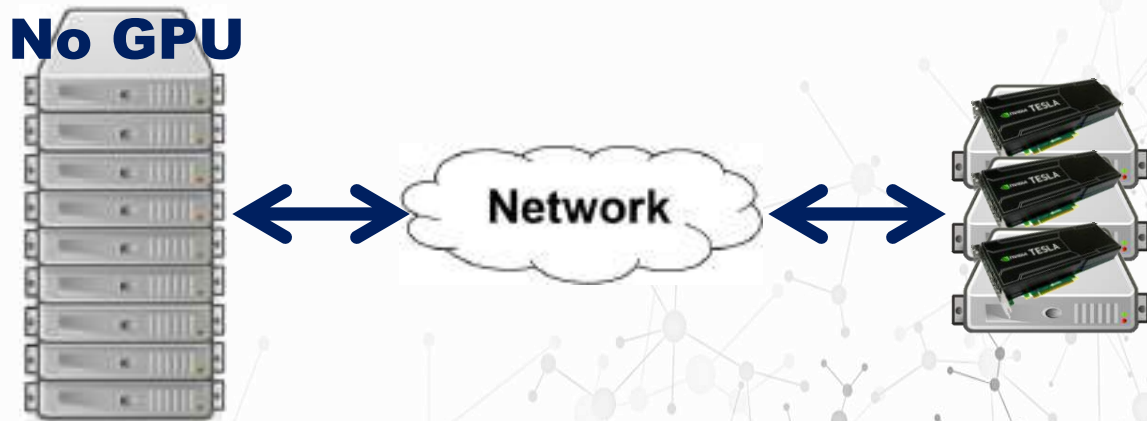
## Basic behavior of CUDA



**Remark:**  
GPUs can only be used within  
the node they are attached to



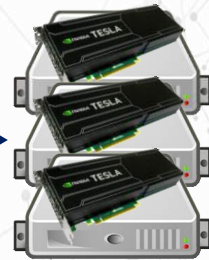
# A different approach: remote GPU virtualization



# A different approach: remote GPU virtualization

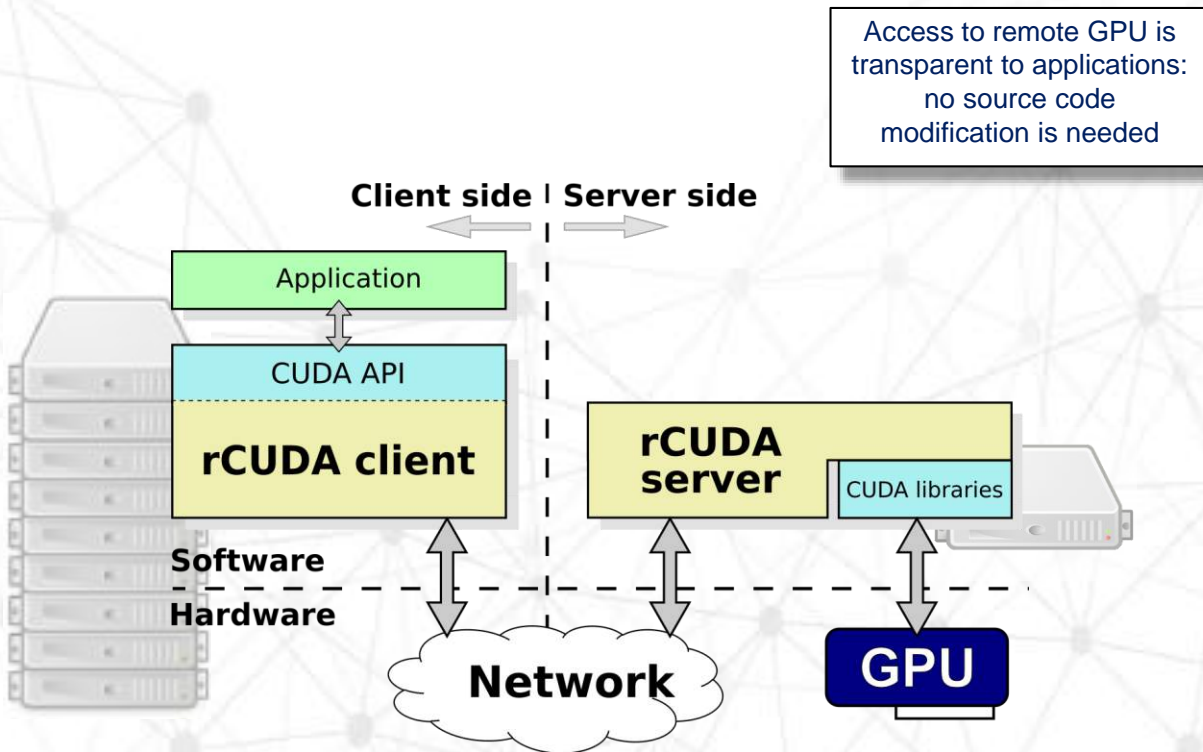
A software technology that enables a more flexible use of GPUs in computing facilities

**No GPU**



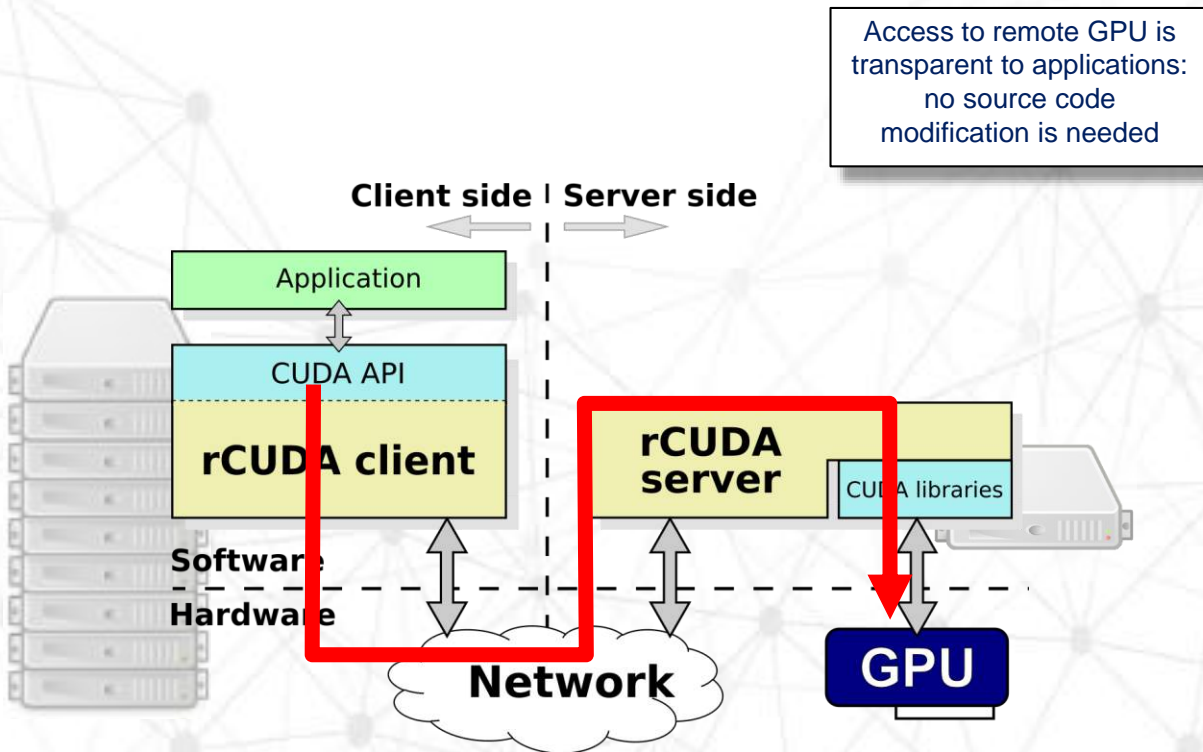
**rCUDA ... remote CUDA**

# Basics or rCUDA



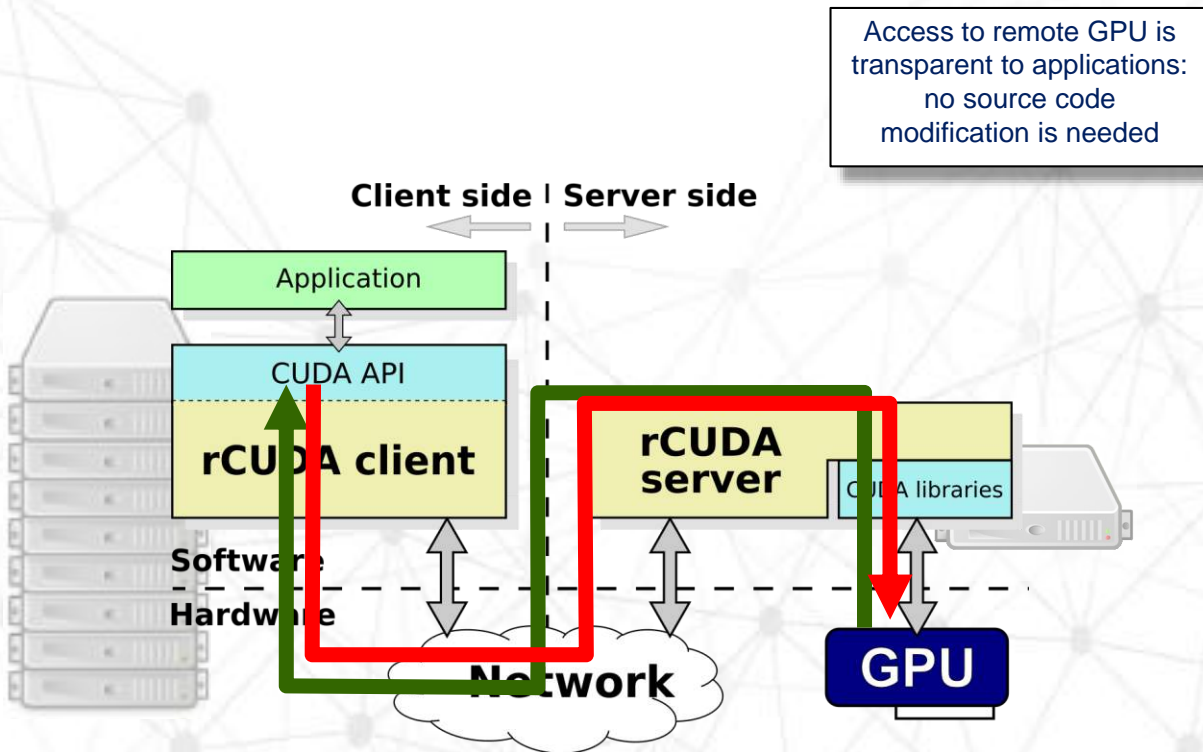


# Basics or rCUDA



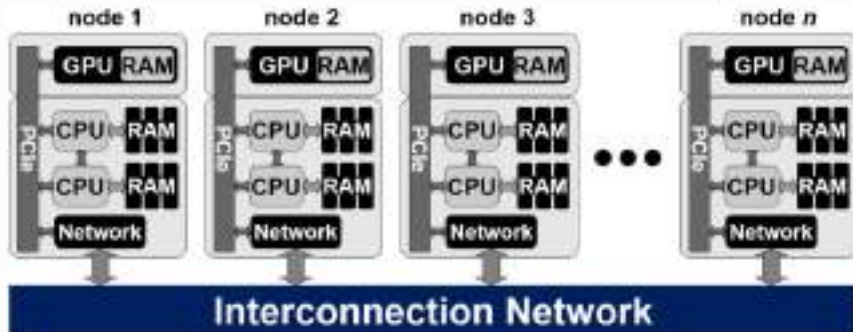


# Basics of rCUDA



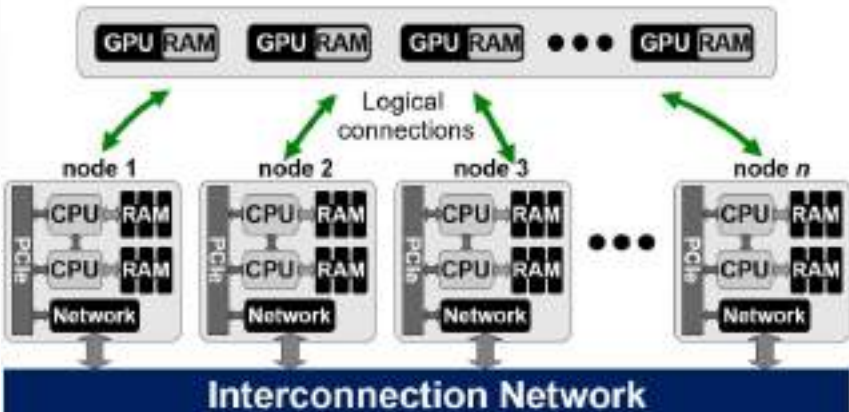
# rCUDA envision

- **rCUDA** allows a new vision of a GPU deployment, moving from the usual cluster configuration ...



Physical configuration

... to the following one:

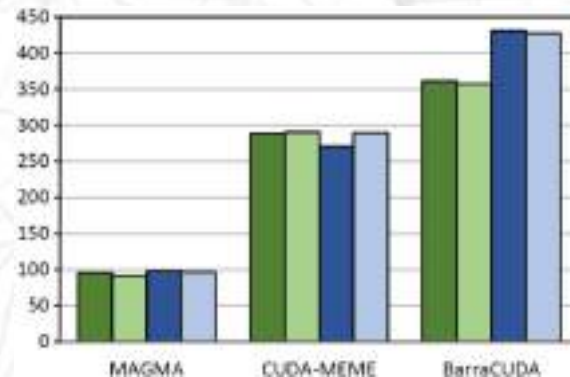
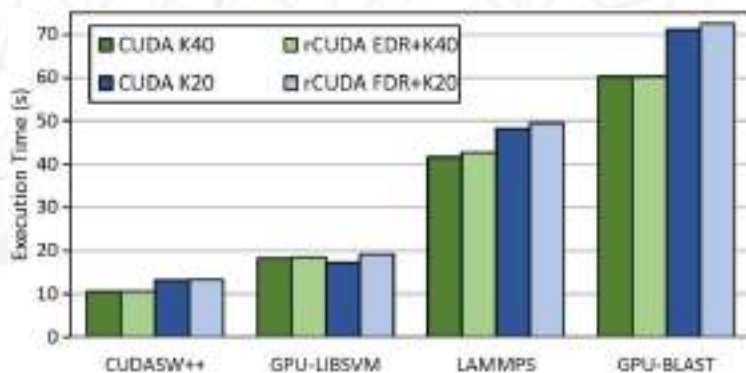
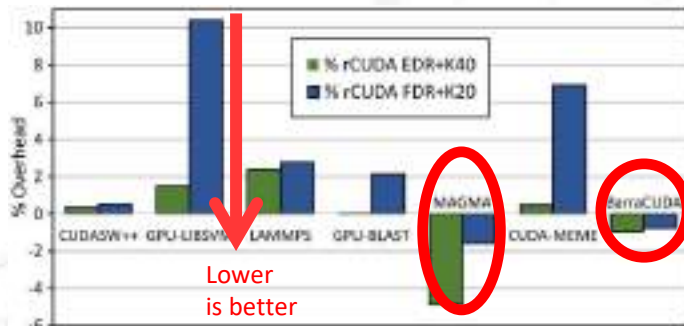


Logical configuration

# Performance of rCUDA?

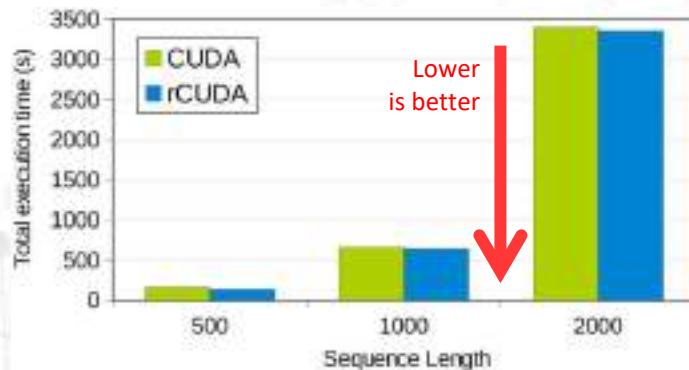
# Performance of rCUDA

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

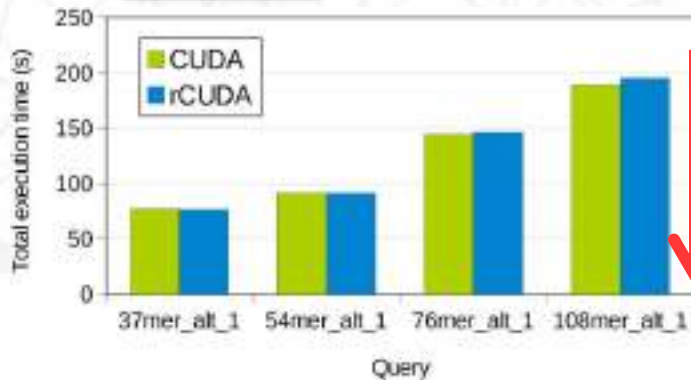


# Performance of rCUDA

P100 GPU and EDR InfiniBand



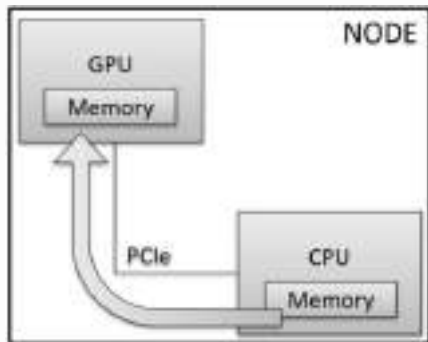
BarraCUDA



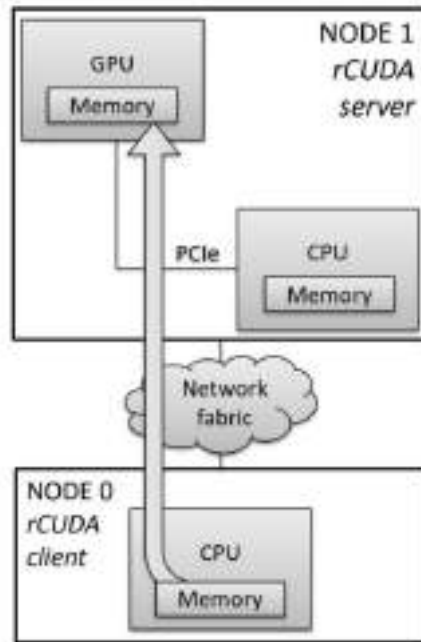
CUDA-MEME

# Performance of data movements to/from GPUs

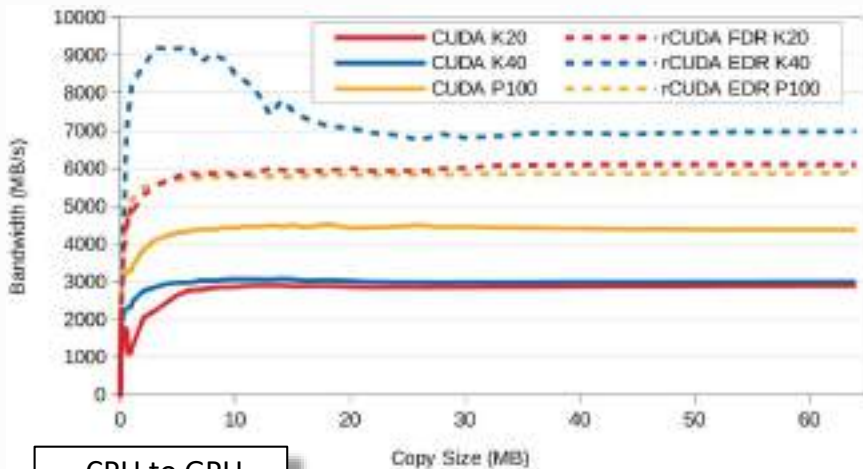
CUDA



rCUDA

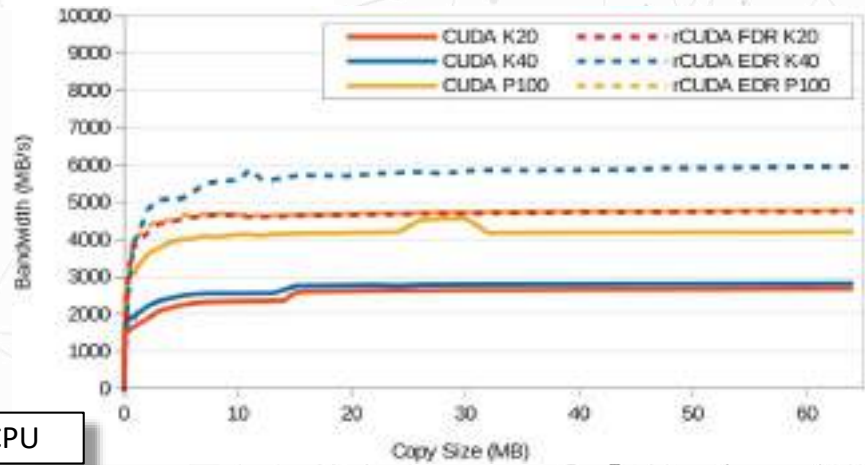


# Performance of data movements to/from GPUs



Higher is better

CPU to GPU



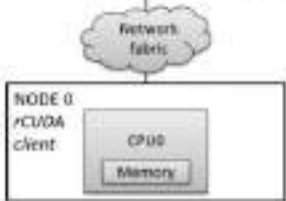
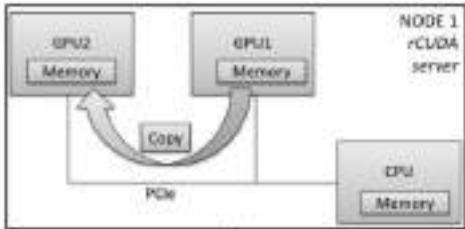
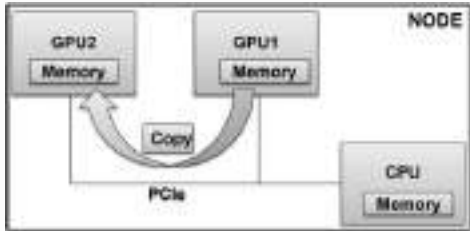
GPU to CPU



# Performance of data movements among GPUs

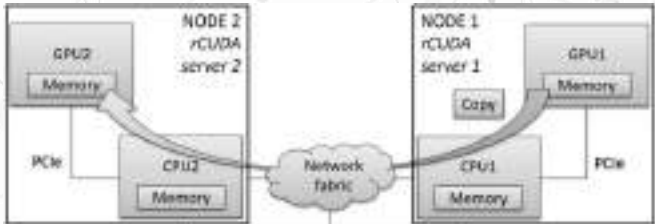
CUDA

rCUDA

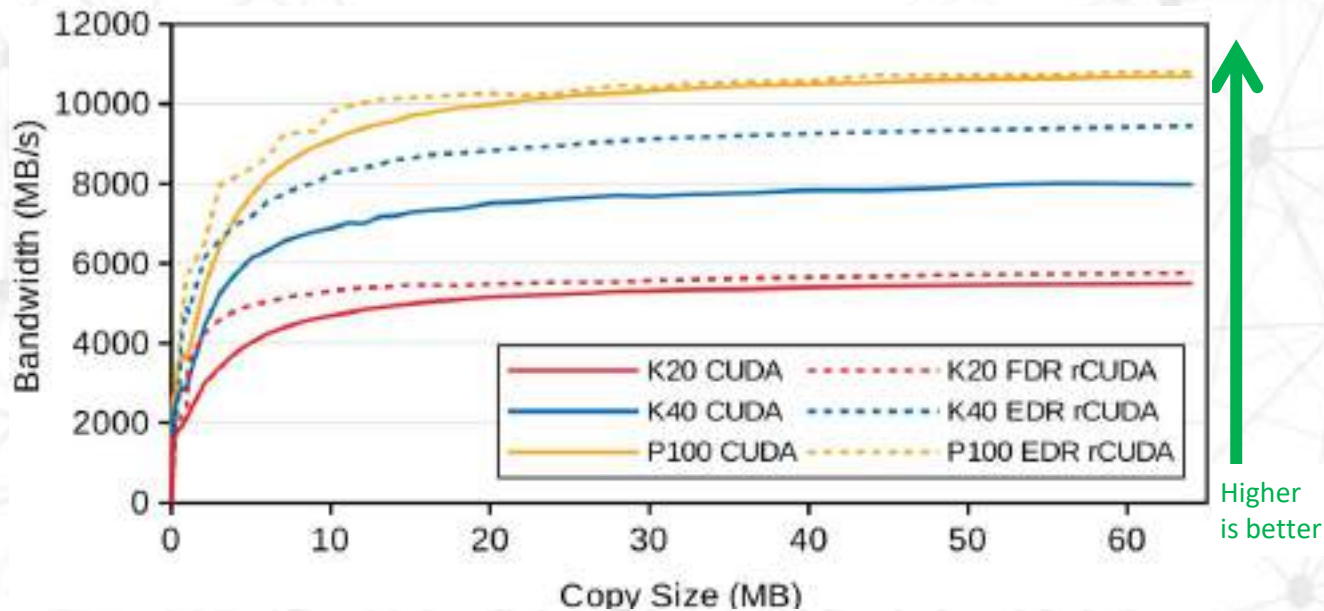


rCUDA scenario 1

rCUDA scenario 2



# Performance of data movements among GPUs

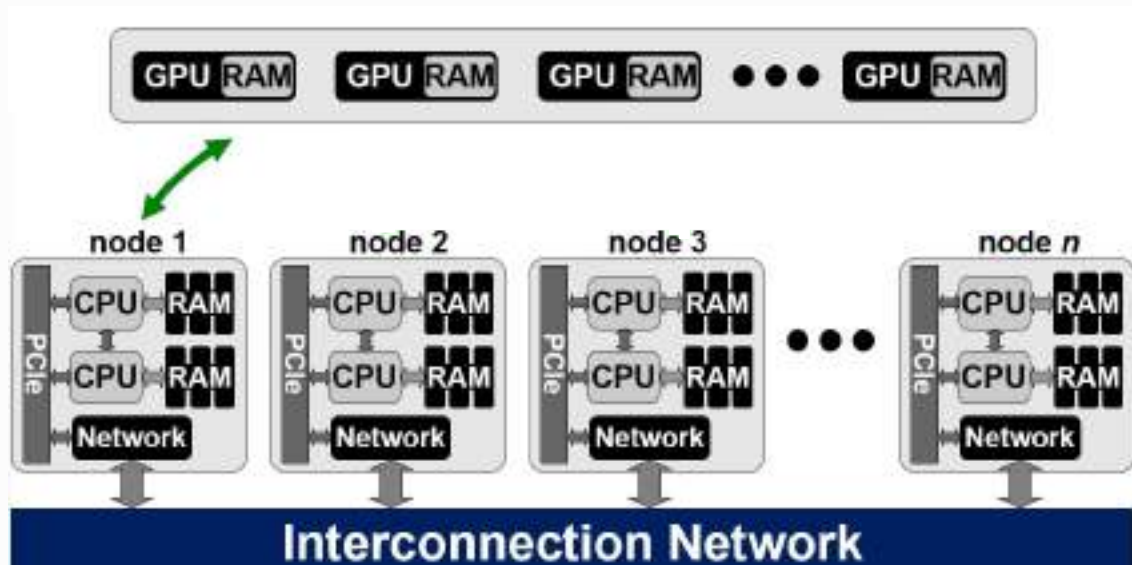


# Benefits of rCUDA?

## **Benefits of rCUDA?**

- 1. Many GPUs for an application**
- 2. Increased cluster throughput**

# Providing many GPUs to an application with rCUDA



# Providing many GPUs to an application with rCUDA

K20 GPUs and FDR InfiniBand



MonteCarlo multi-GPU program running in 14 NVIDIA Tesla K20 GPUs

# Providing many GPUs to an application with rCUDA

```
bsc19421@nrv127:~$ ./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDA static linking)

Detected 64 CUDA Capable device(s)

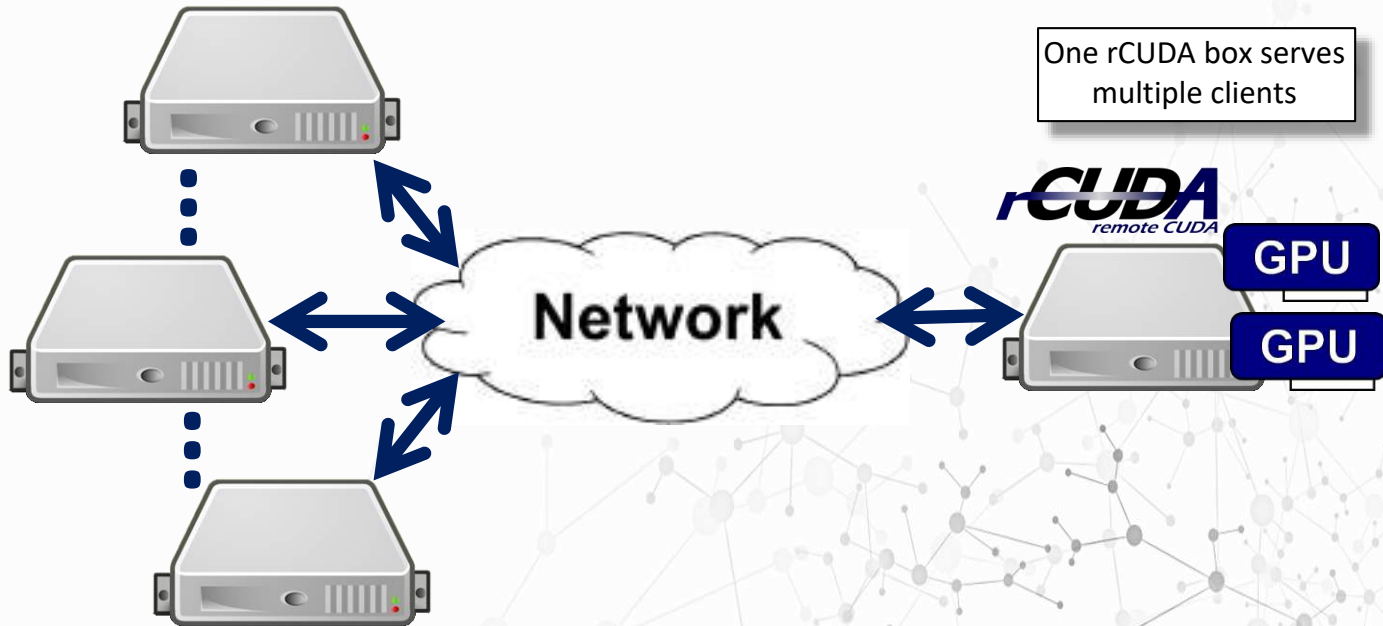
Device 0: "Tesla M2090"
  CUDA Driver Version / Runtime Version      5.0 / 5.0
  CUDA Capability Major/Minor version number: 2.0
  Total amount of global memory:             6144 MBytes (6442123264 bytes)
  (16) Multiprocessors x ( 32) CUDA Cores/MP: 512 CUDA Cores
  GPU Clock rate:                            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=(10304) x 2048, 2D=(10304,10304) 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
```

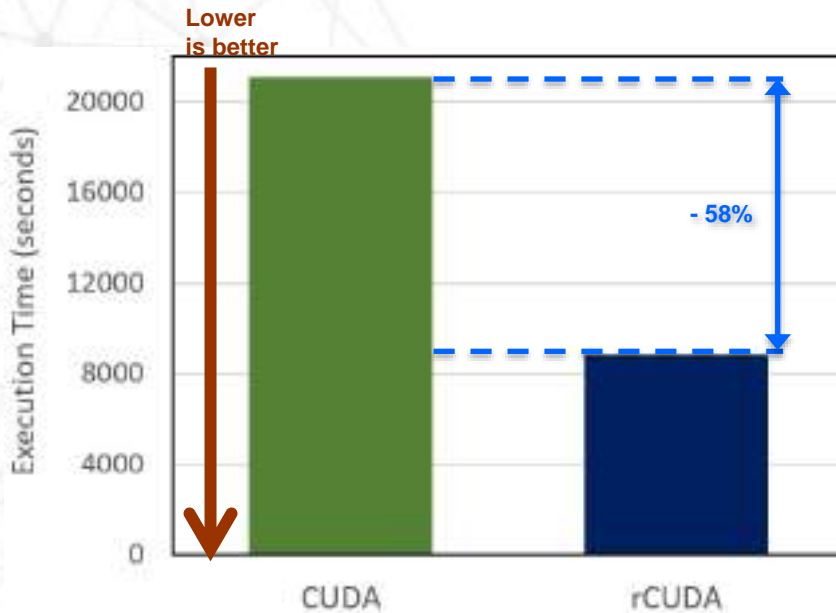
64 GPUs !!



# Increased cluster throughput

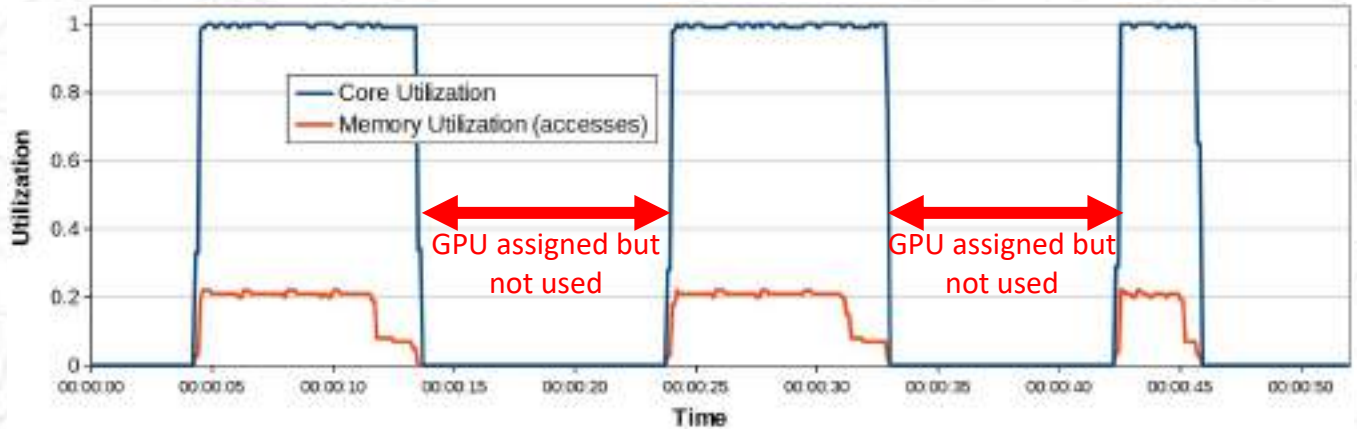


# Increased cluster throughput



1. BarraCUDA
2. CUDA-MEME
3. CUDASW++
4. GPU-Blast
5. Gromacs
6. Magma

# Increased cluster throughput



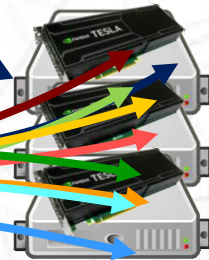
# Slurm and rCUDA

# Need of a resource scheduler

**No GPU**



GPUs are shared  
among applications



Which is the limit of GPU sharing?



Get a free copy of rCUDA at  
<http://www.rcuda.net>

More than 900 requests world wide



@rcuda\_