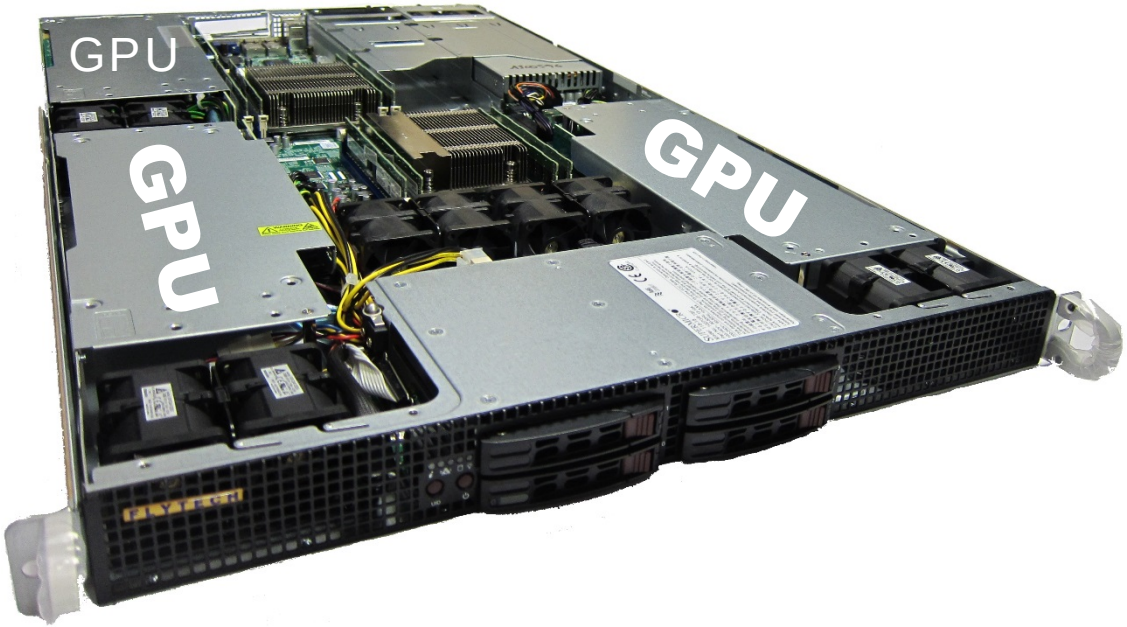
Increasing cluster throughput with Slurm and rCUDA

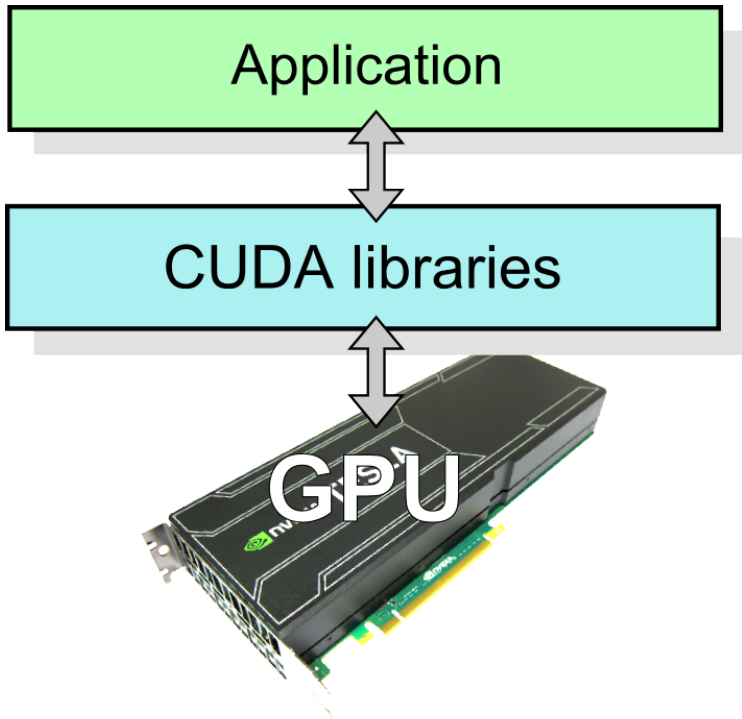
Federico Silla
Technical University of Valencia
Spain

Increasing cluster throughput with Slurm and rCUDA

What is "*rCUDA*"?

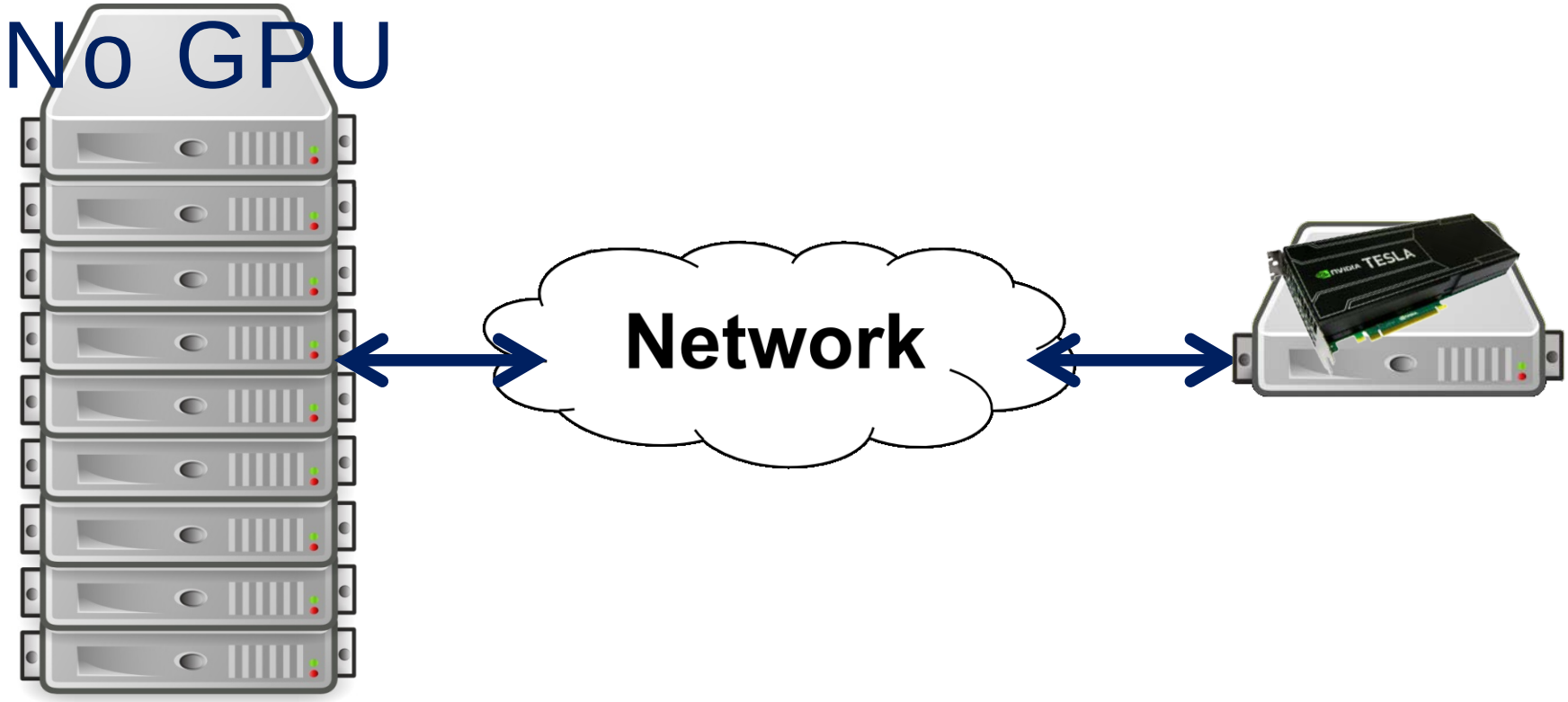
Basic behavior of CUDA



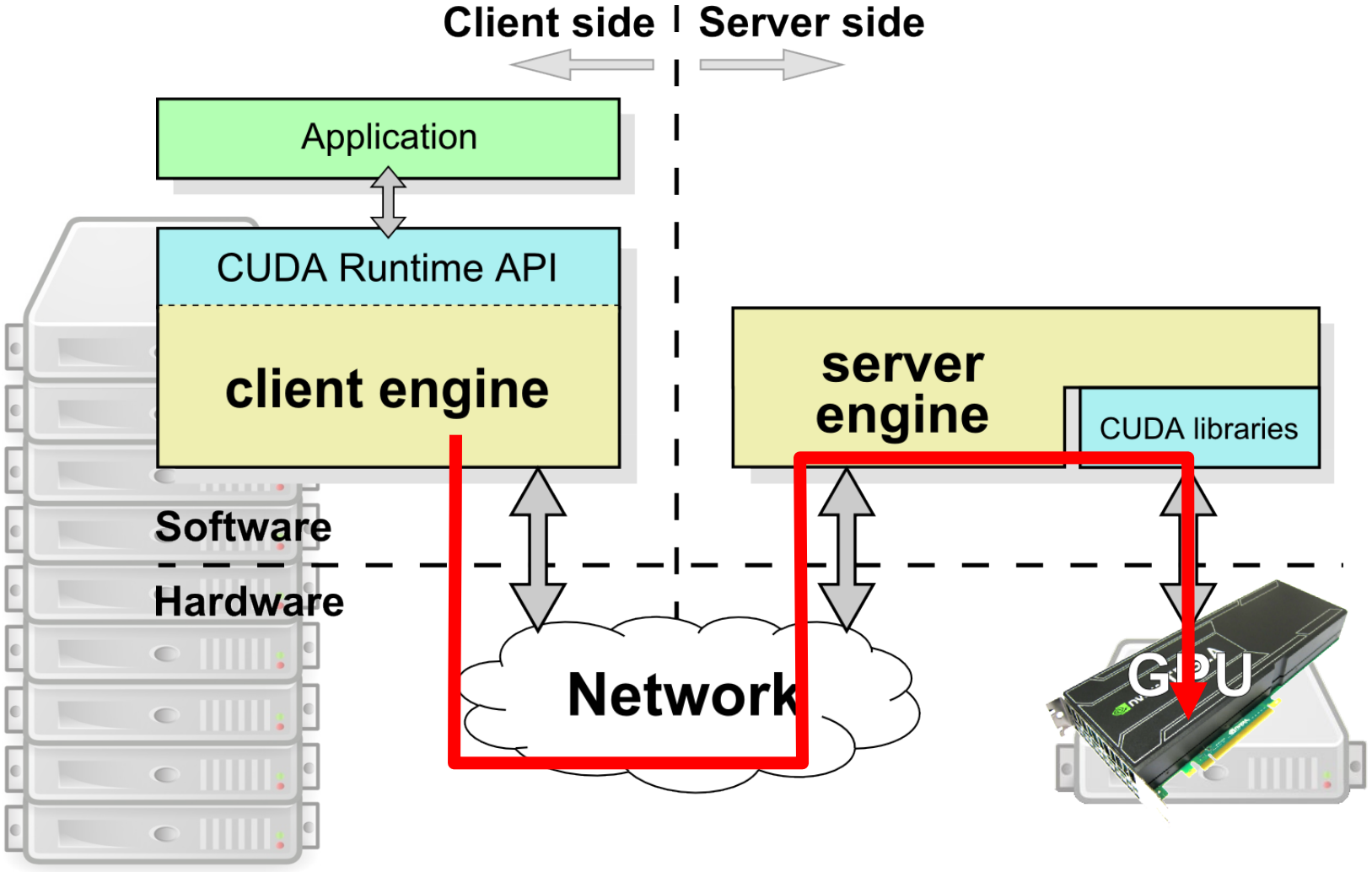


rCUDA: remote GPU virtualization

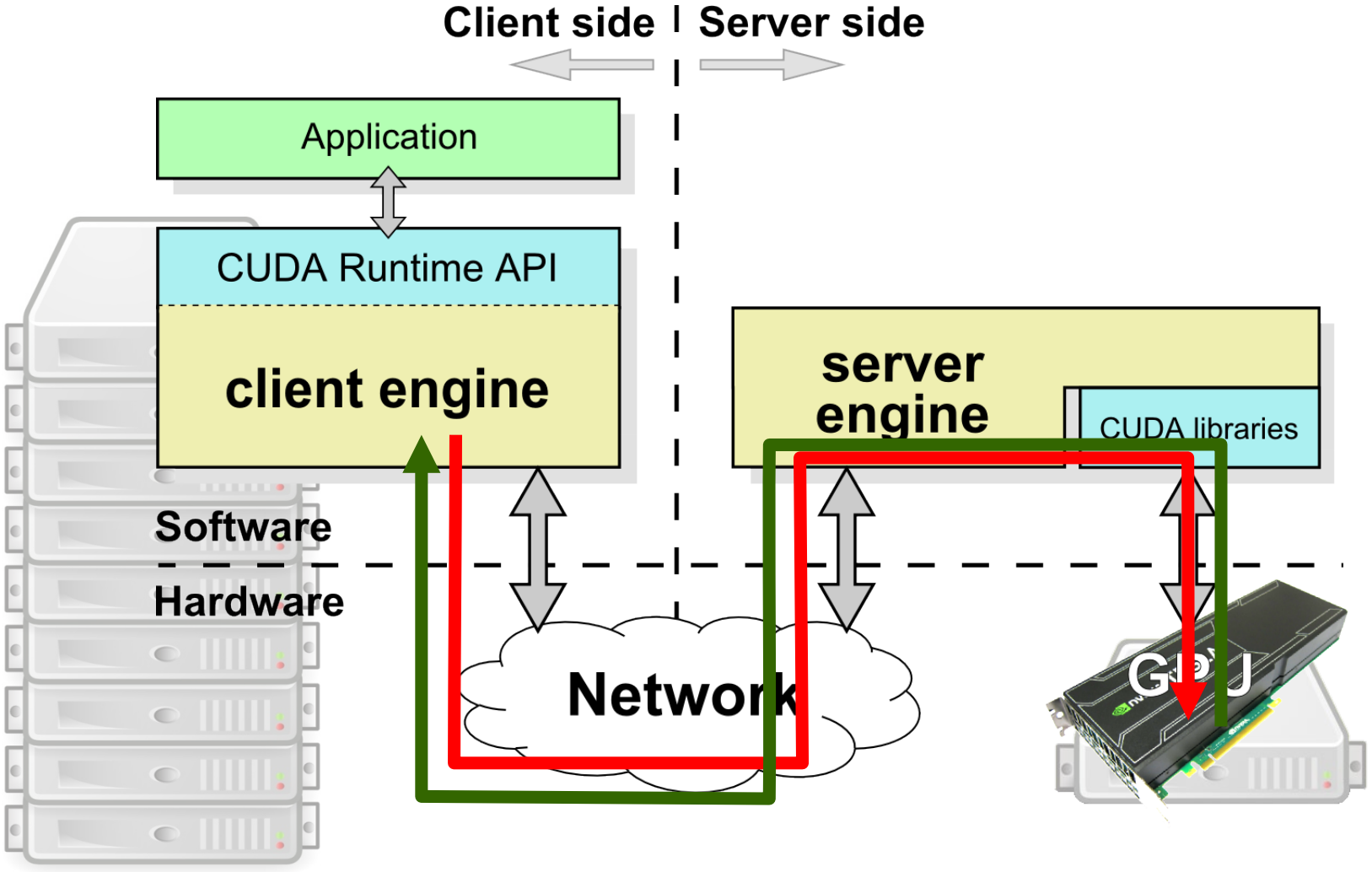
A **software** technology that enables **a more flexible use of GPUs** in computing facilities



Basics of remote GPU virtualization

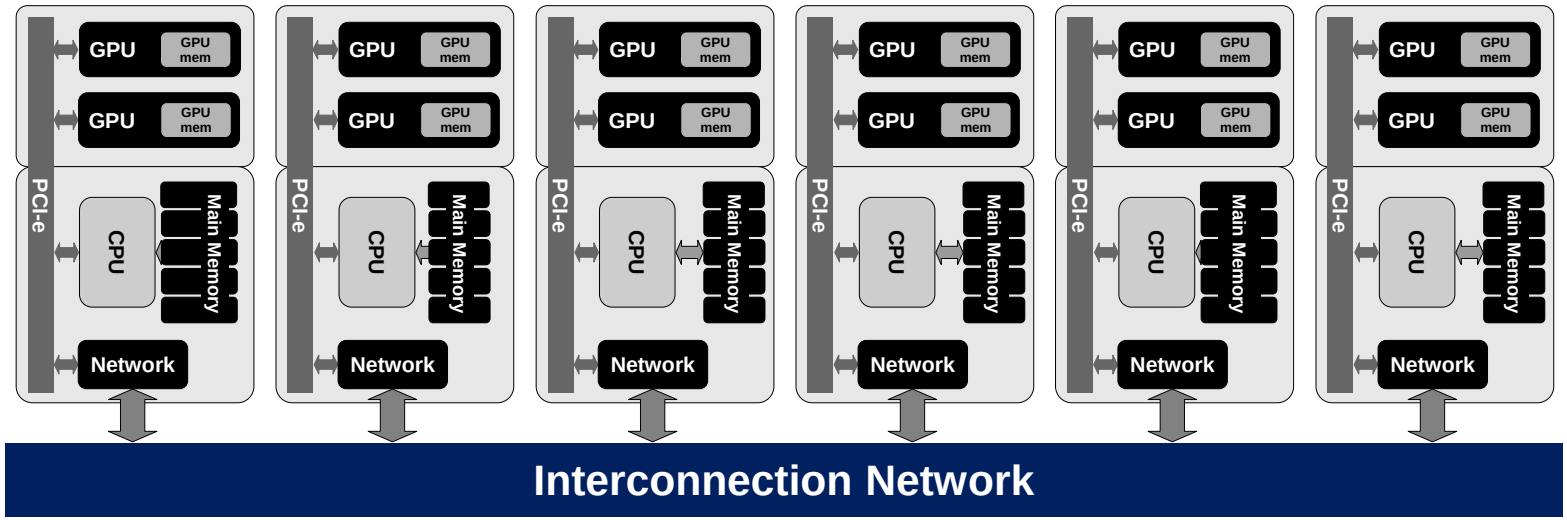


Basics of remote GPU virtualization



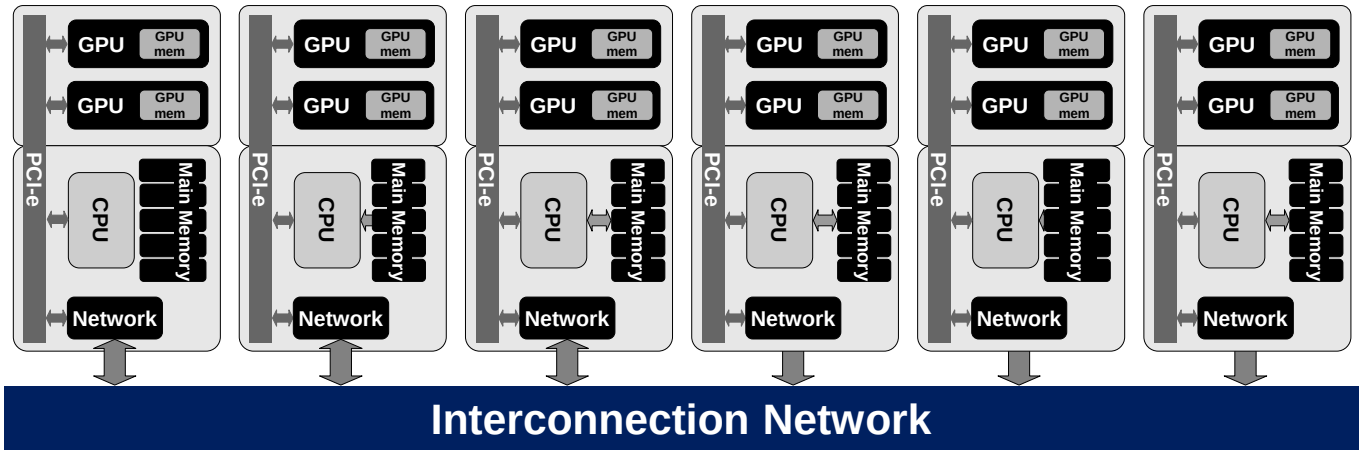
Remote GPU virtualization envision

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

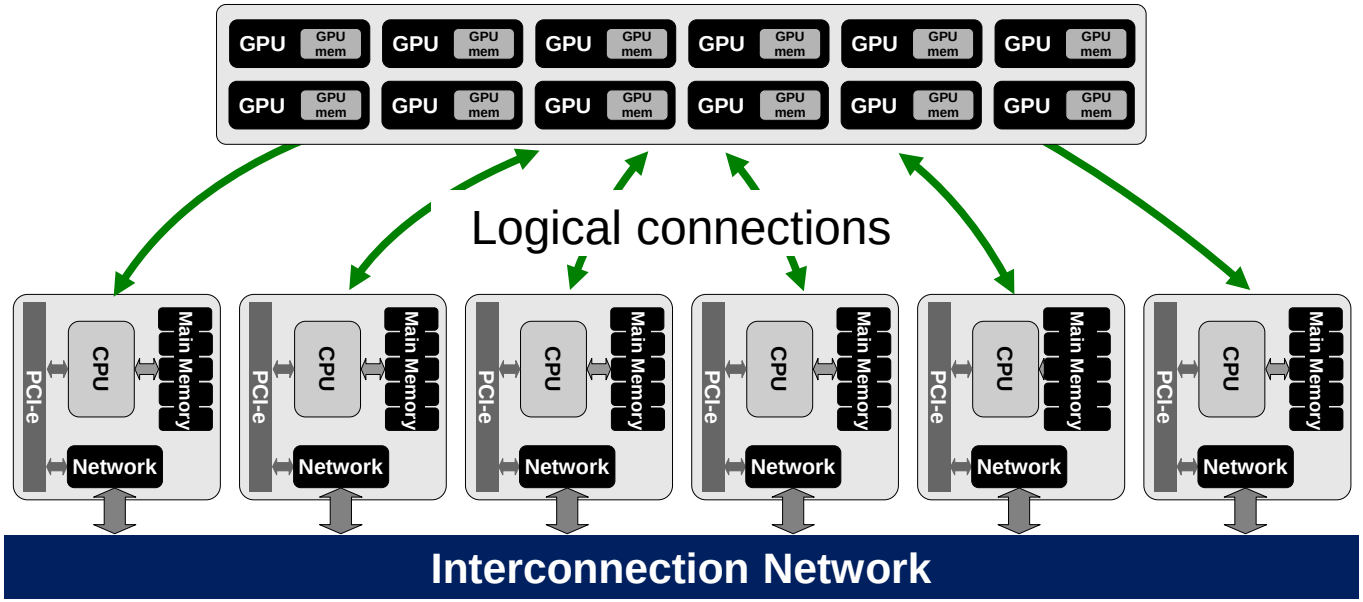


to the following one ….

Remote GPU virtualization envision



Physical configuration



Logical configuration

Remote GPU virtualization frameworks

- Several efforts have been made to **implement** remote GPU virtualization during the last years:

• **rCUDA** (CUDA 7.0) • GVirtuS (CUDA 3.2) • DS-CUDA (CUDA 4.1)	**Publicly available**
• vCUDA (CUDA 1.1) • GViM (CUDA 1.1) • GridCUDA (CUDA 2.3) • V-GPU (CUDA 4.0)	**NOT publicly available**

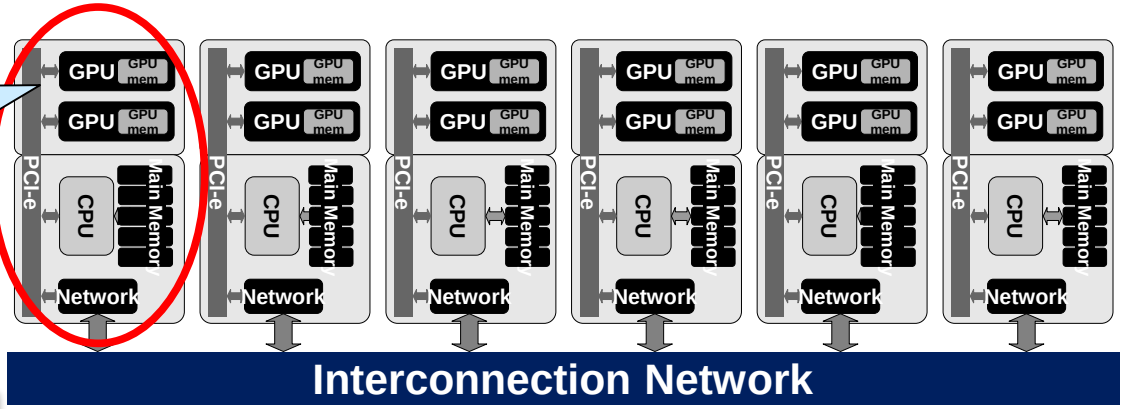
2nd

Is "*remote GPU virtualization*" useful?

1: more GPUs for a single application

- GPU virtualization is useful for multi-GPU applications

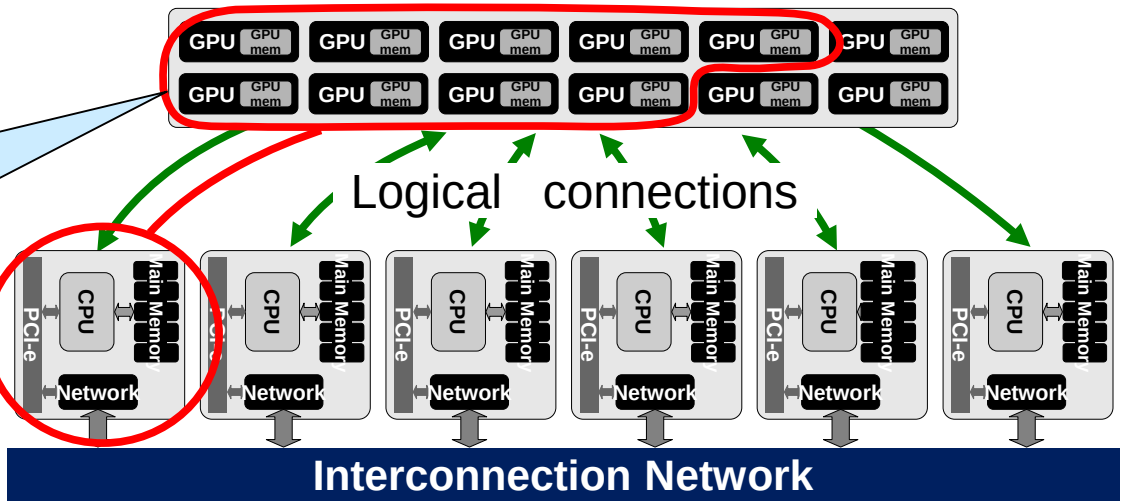
Only the GPUs in the node can be provided to the application



Without GPU virtualization

With GPU virtualization

Many GPUs in the cluster can be provided to the application



1: more GPUs for a single application

64 GPUs!

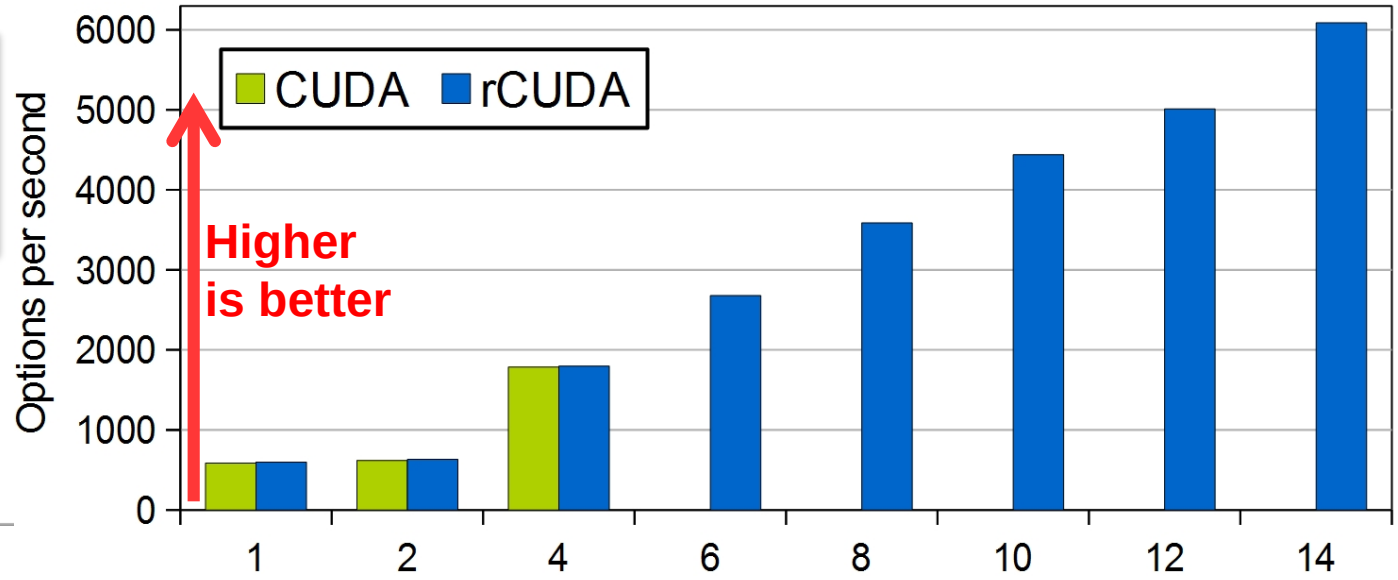
```
bsc19421@nvb127:~
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART 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=(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
```

1: more GPUs for a single application

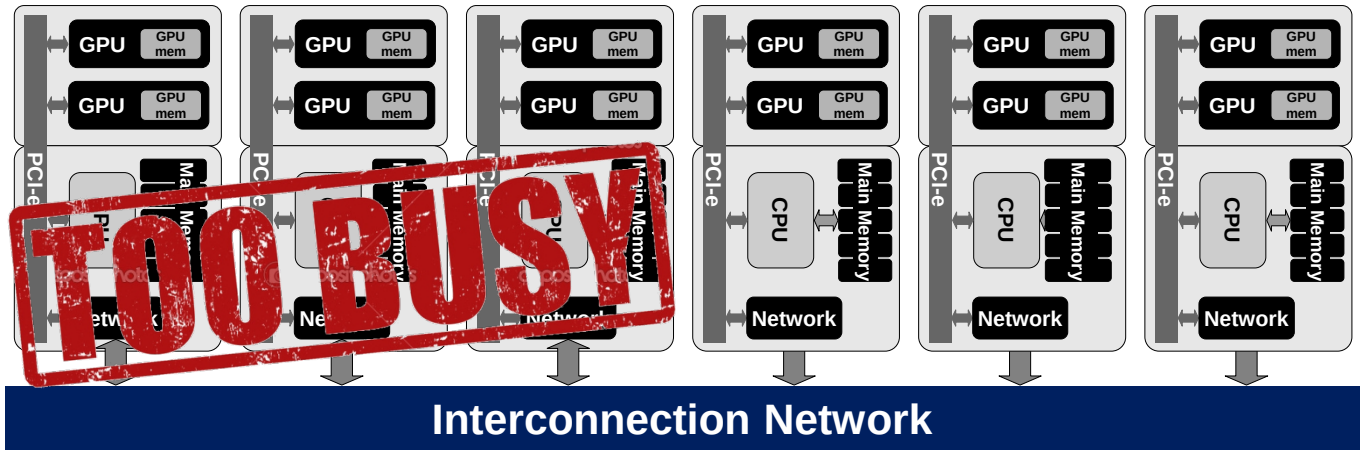
- MonteCarlo Multi-GPU (from NVIDIA samples)

FDR InfiniBand +

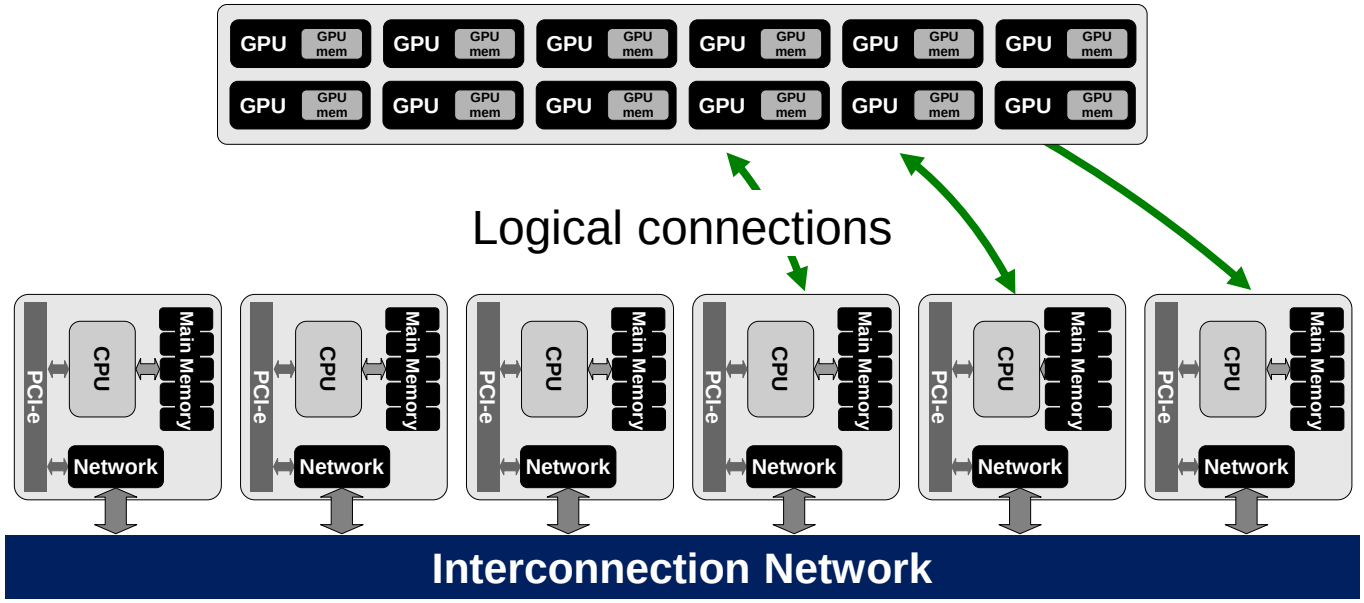
NVIDIA Tesla K20



2: increased cluster performance



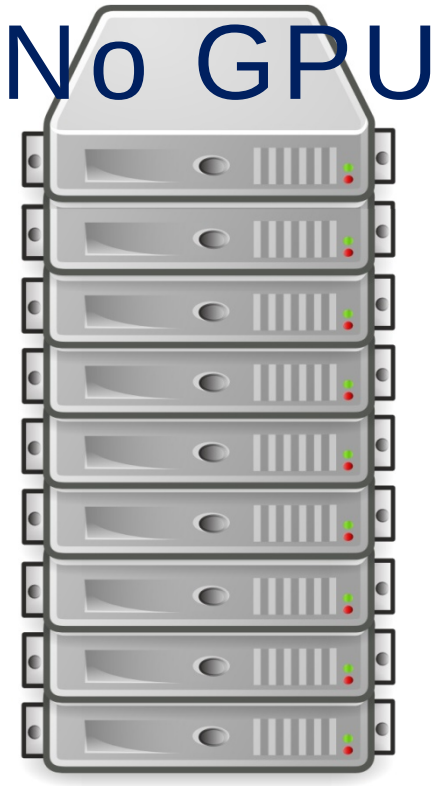
Physical configuration



Logical configuration

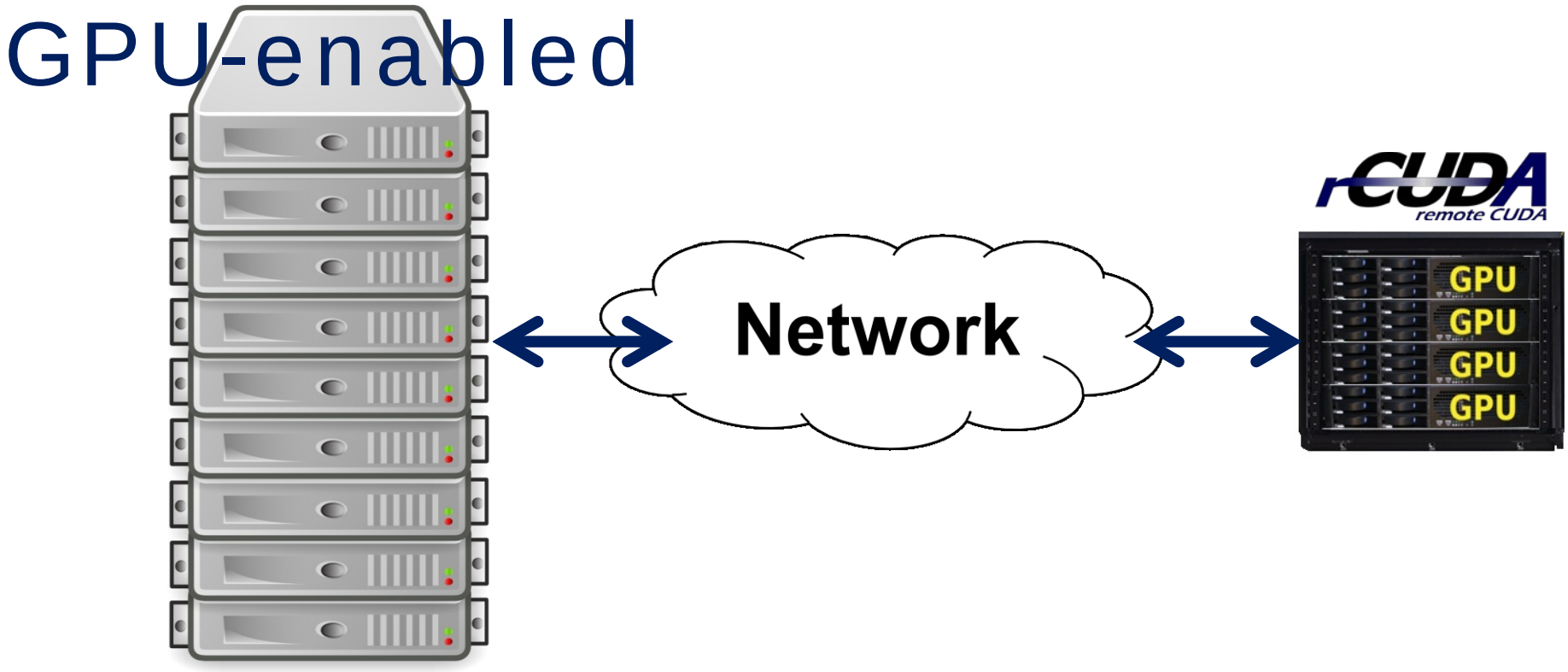
3: easier cluster upgrade

- A cluster without GPUs may be easily upgraded to use GPUs with rCUDA



3: easier cluster upgrade

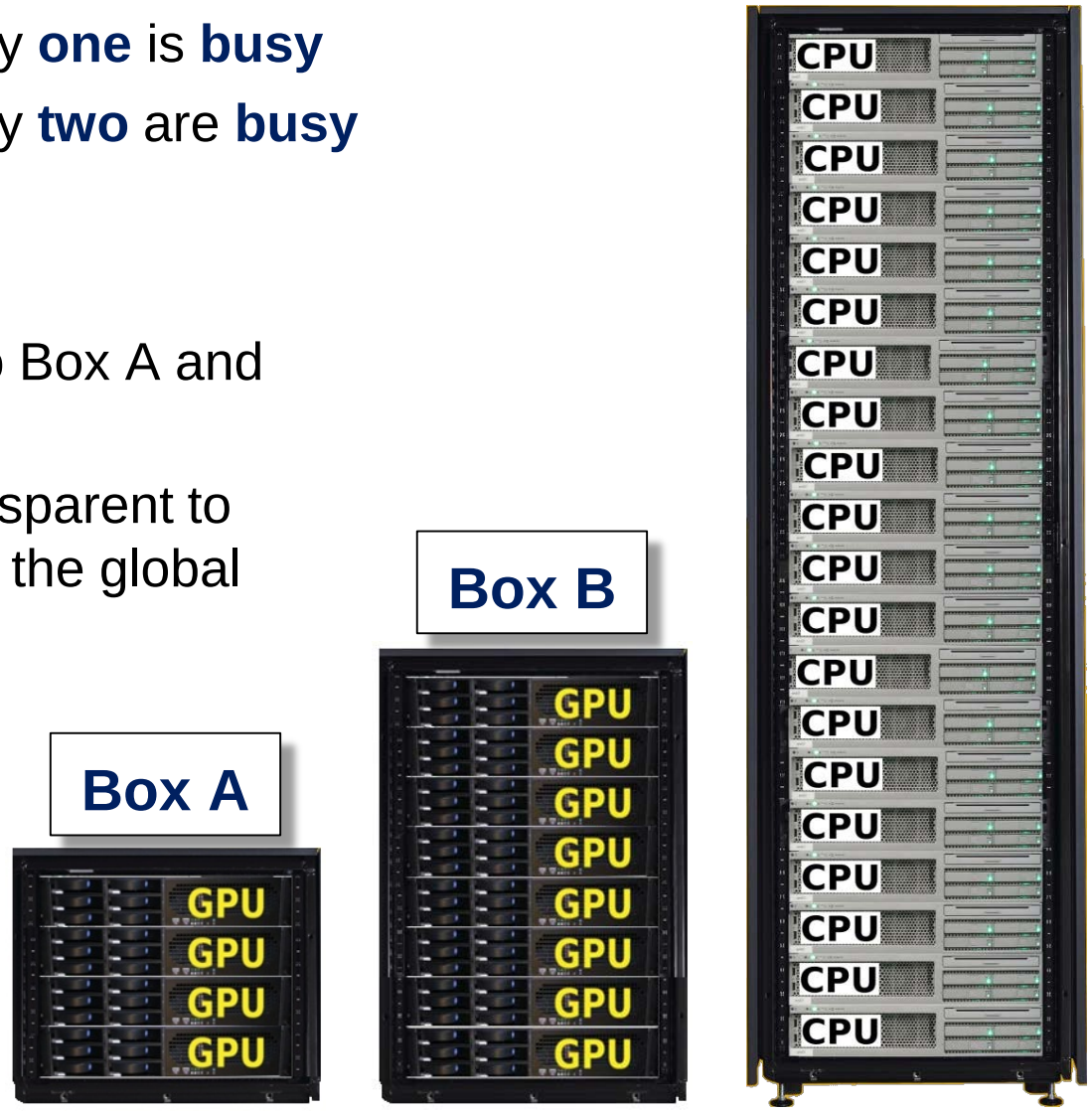
- A cluster without GPUs may be easily upgraded to use GPUs with rCUDA



4: GPU task migration

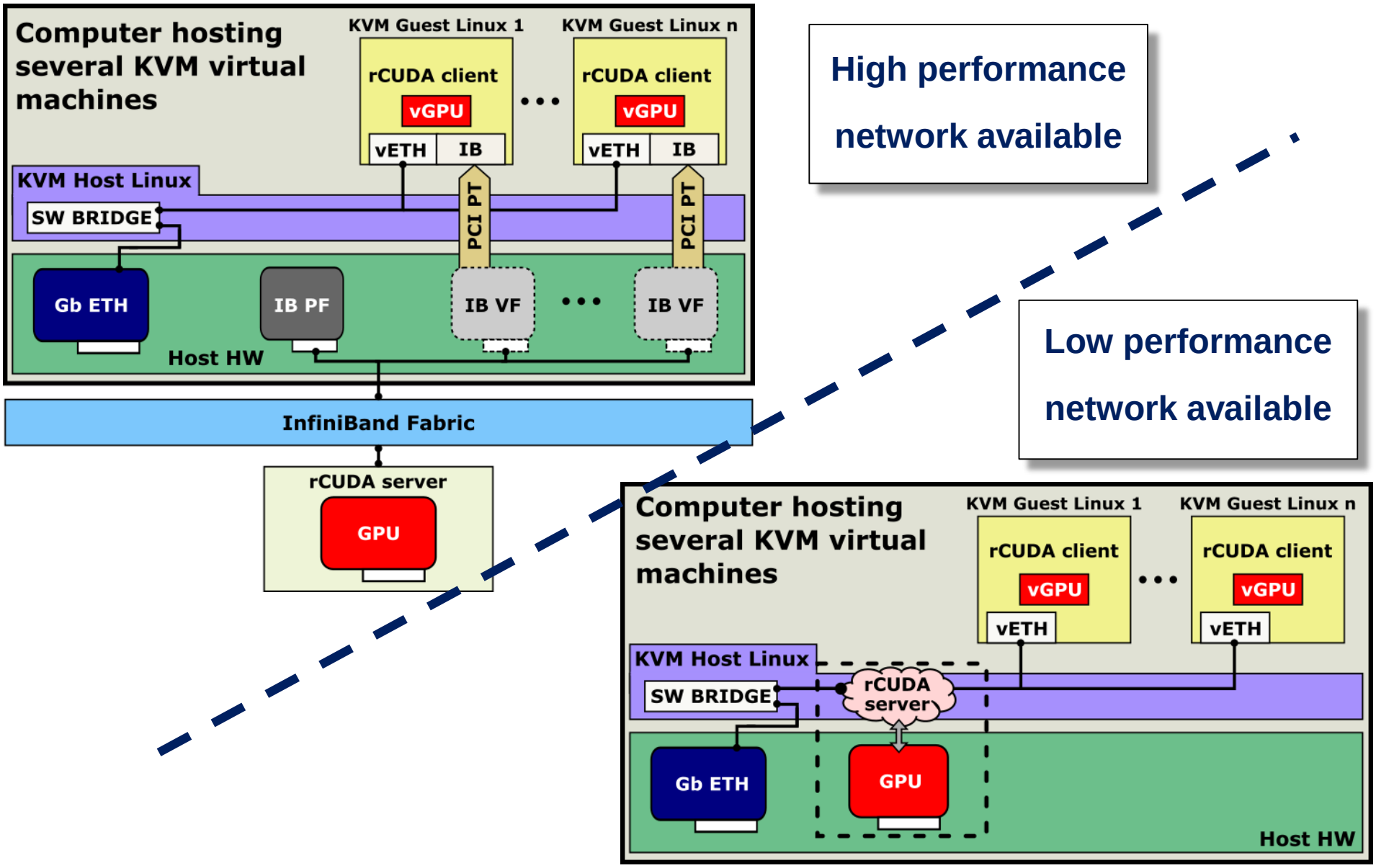
- Box A has **4 GPUs** but only **one** is **busy**
- Box B has **8 GPUs** but only **two** are **busy**

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)



TRUE GREEN GPU COMPUTING

5: virtual machines can easily access GPUs

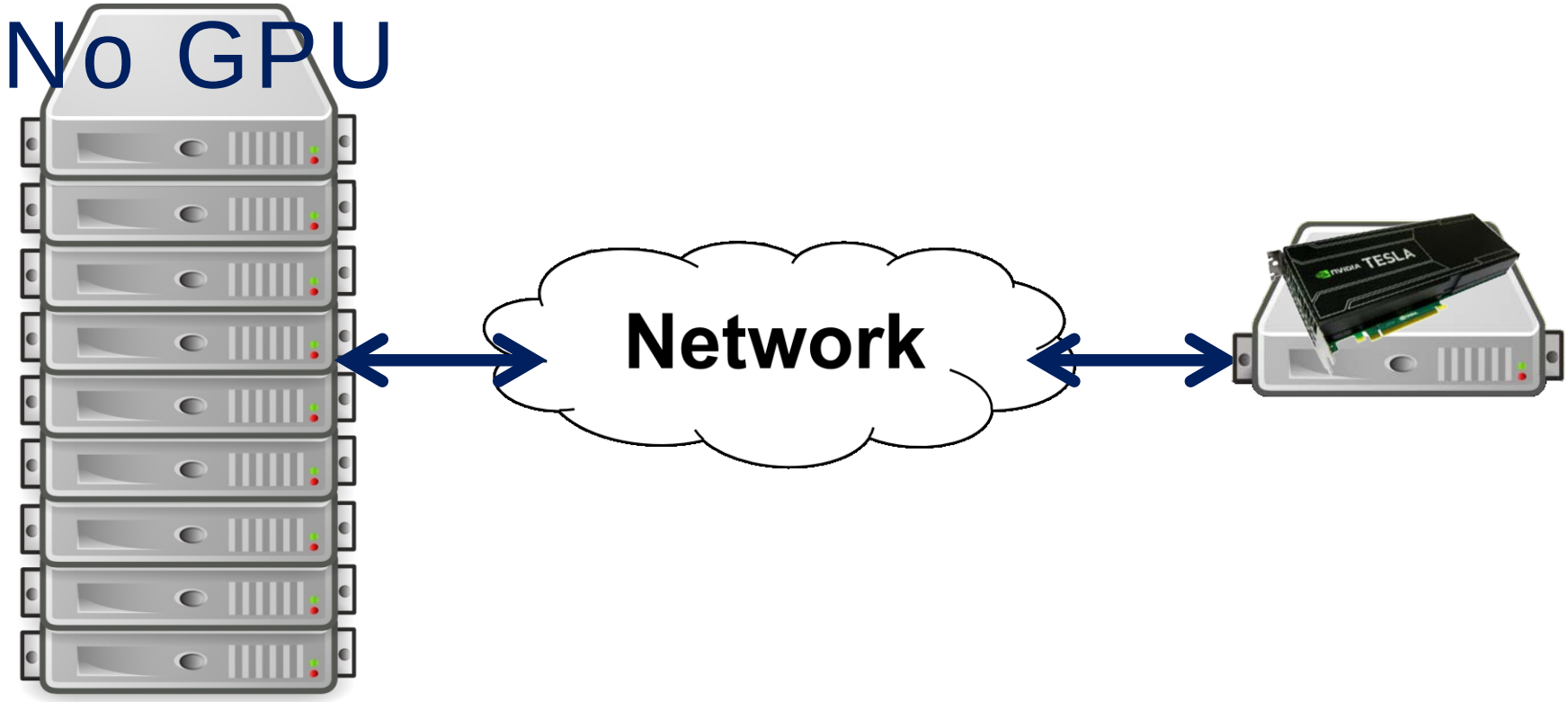


3rd

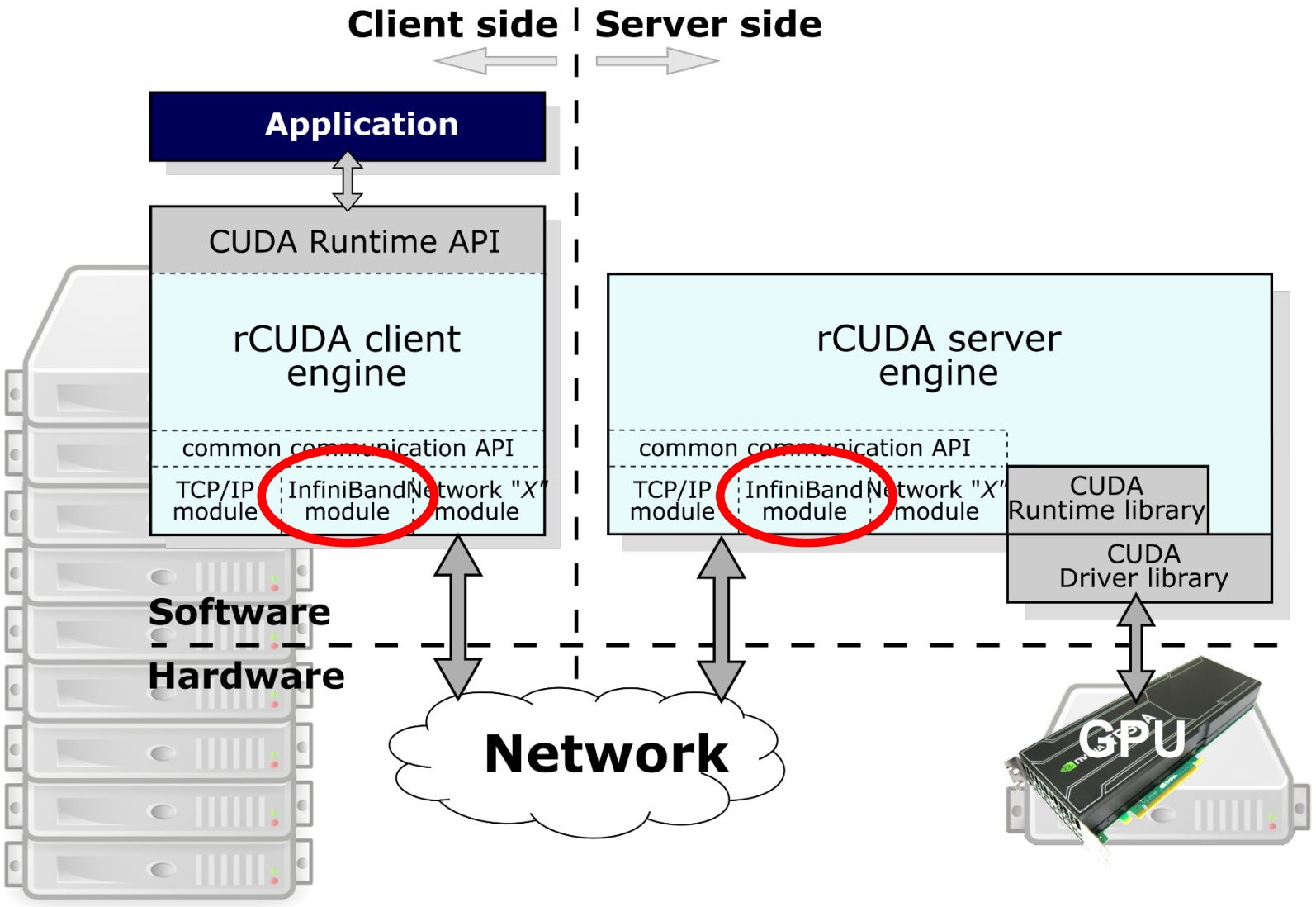
Cons of "*remote GPU virtualization*"?

Problem with remote GPU virtualization

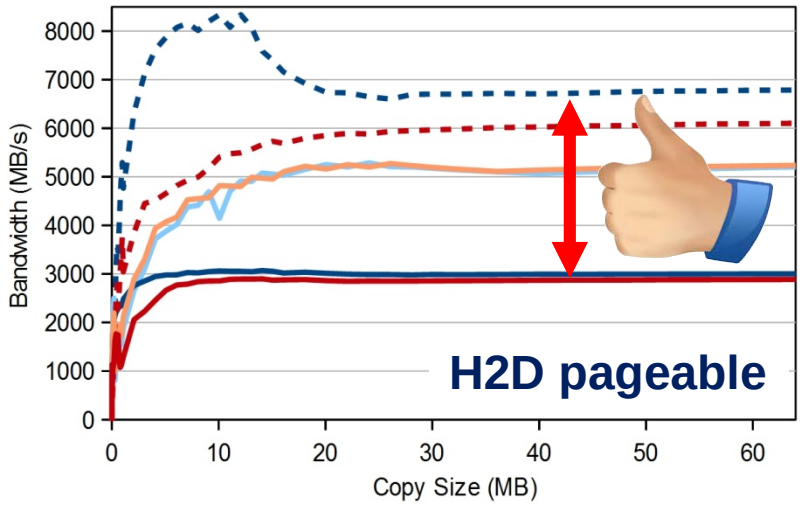
The main GPU virtualization drawback is the reduced bandwidth to the remote GPU

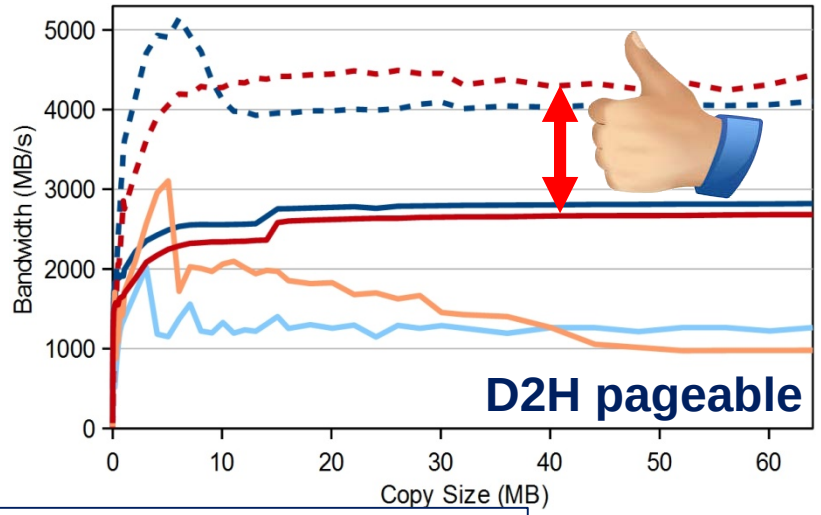


Using InfiniBand networks

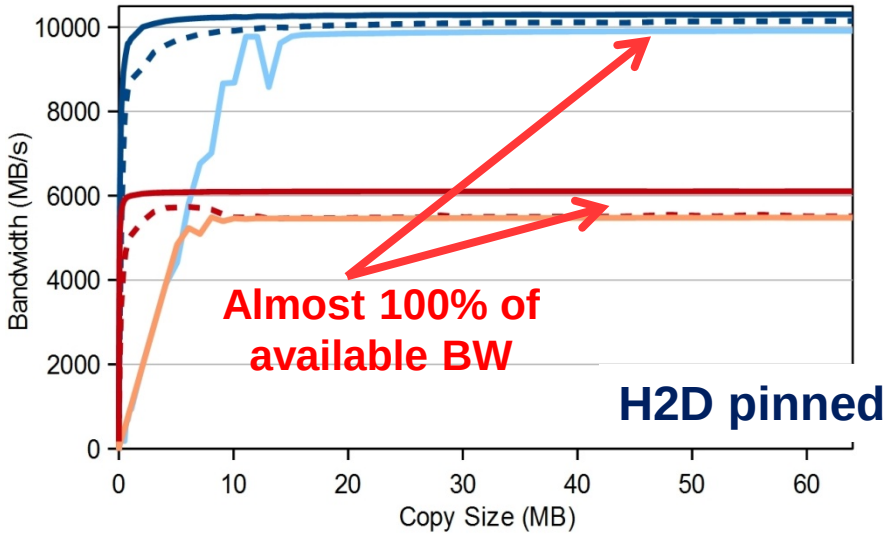


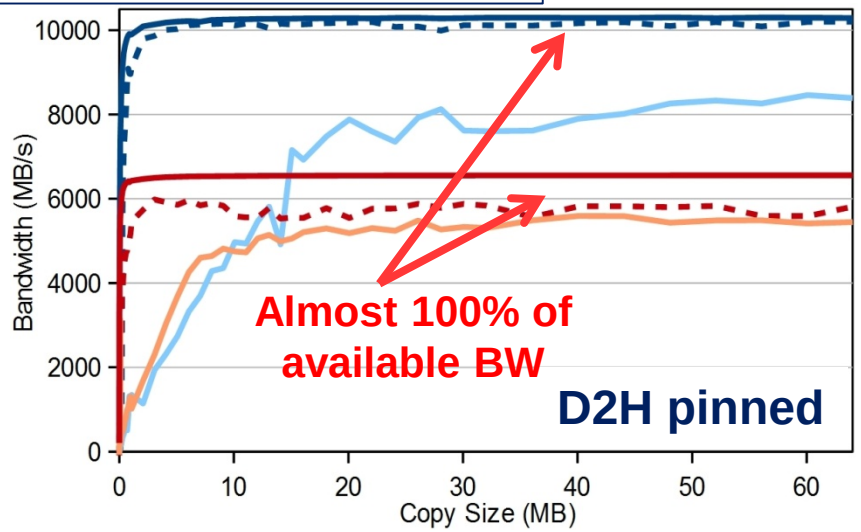
Impact of optimizations on rCUDA





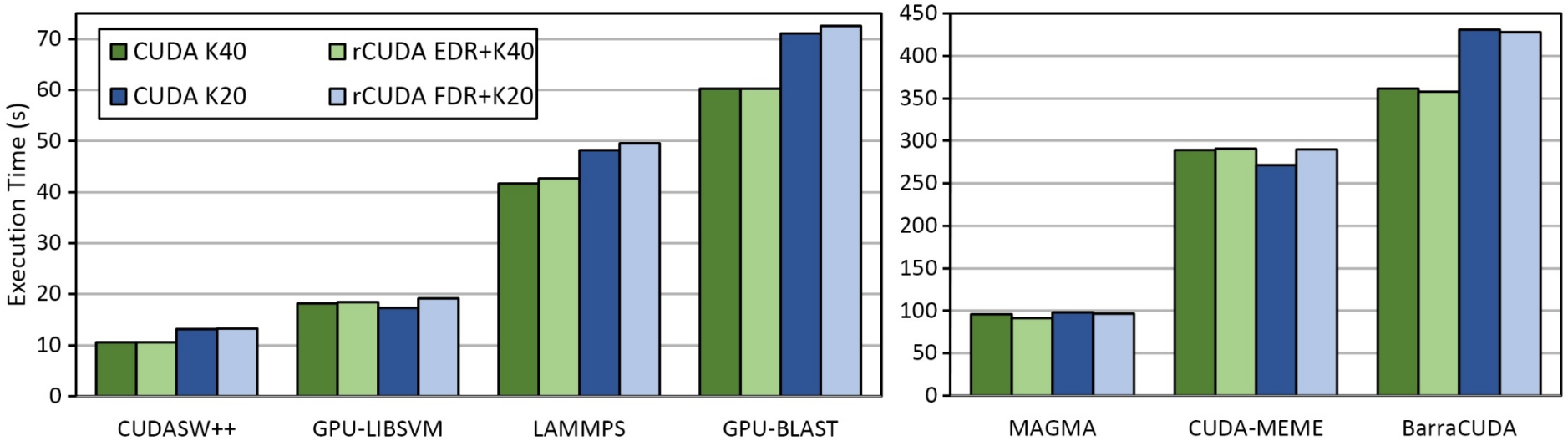
CUDA K20 — rCUDA X3 Orig — rCUDA X3 Opt
CUDA K40 — rCUDA EDR Orig — rCUDA EDR Opt





Effect of rCUDA optimizations on applications

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

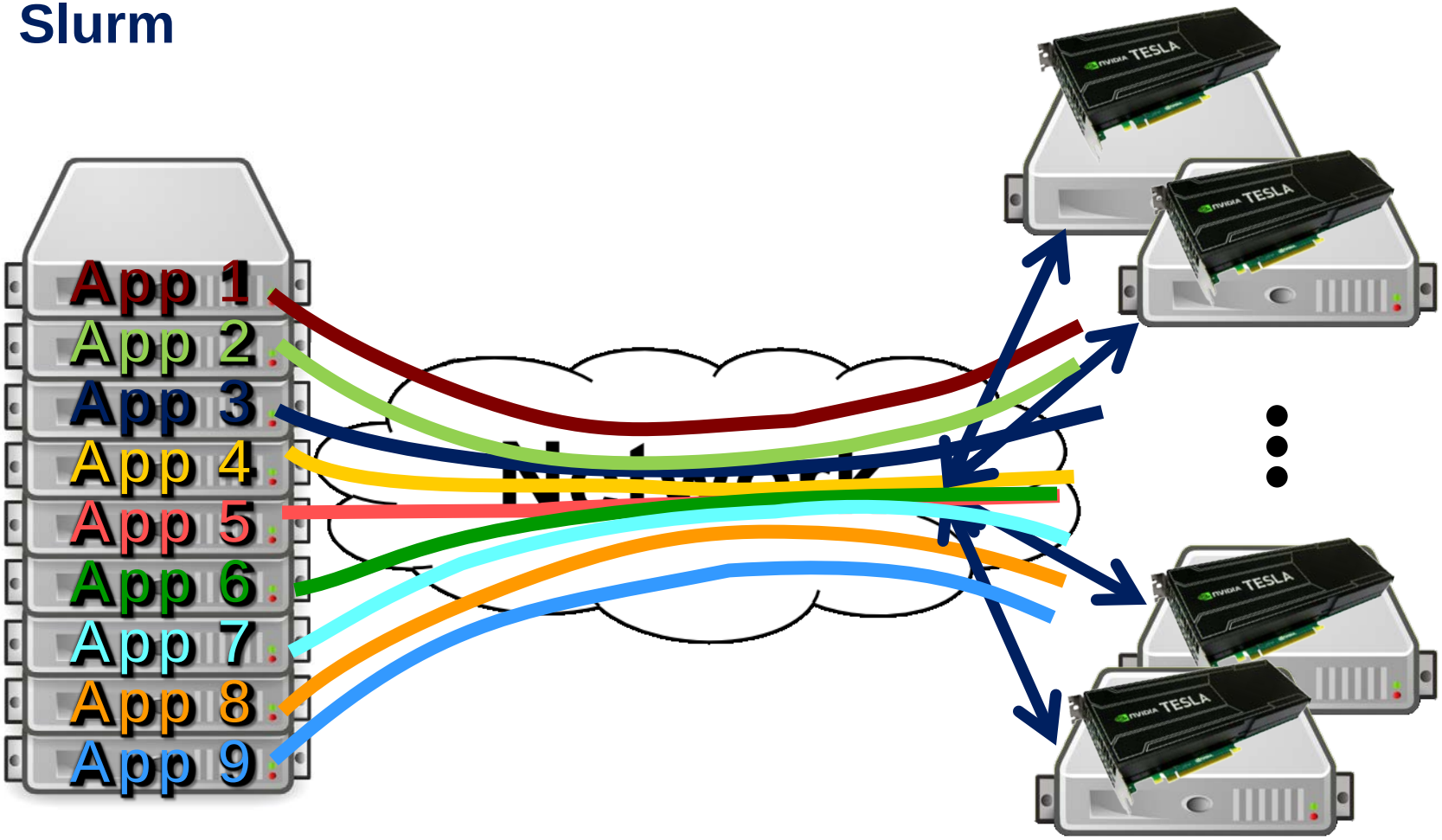


4th

What happens at the cluster level?

rCUDA at cluster level … Slurm

- **GPUs can be shared** among jobs running in remote clients
  - Job scheduler required for coordination
  - **Slurm**



rCUDA at cluster level … Slurm

- A new resource has been added: **rgpu**

gres.conf

Name = rgpu File =/ dev/ nvidia0 Cuda =3.5 Mem =4726 M
[Name =gpu File =/ dev/ nvidia0]

slurm.conf

SelectType = select / cons_rgpu
SelectTypeParameters = CR_CORE
GresTypes = rgpu [,gpu]

NodeName = node1 NodeHostname = node1
CPUs =12 Sockets =2 CoresPerSocket =6
ThreadsPerCore =1 RealMemory =32072
Gres = rgpu :1[, gpu :1]

New submission options:

--rcuda-mode=(shared|excl)
--gres=rgpu(:X(:Y)?(:Z)?)?
X = [1-9]+[0-9]*
Y = [1-9]+[0-9]*[kKmMgG]
Z = [1-9]\.[0-9](cc|CC)

Ongoing work: studying rCUDA+Slurm

- Analysis of different GPU assignment policies
  - Based on GPU-memory occupancy
  - Based on GPU utilization
- Applications used for tests:
  - GPU-Blast (21 seconds; 1 GPU; 1599 MB)
  - LAMMPS (15 seconds; 4 GPUs; 876 MB)
  - MCUDA-MEME (165 seconds; 4 GPUs; 151 MB)
  - GROMACS (167 seconds)

Non-GPU

Set 1

Short execution time

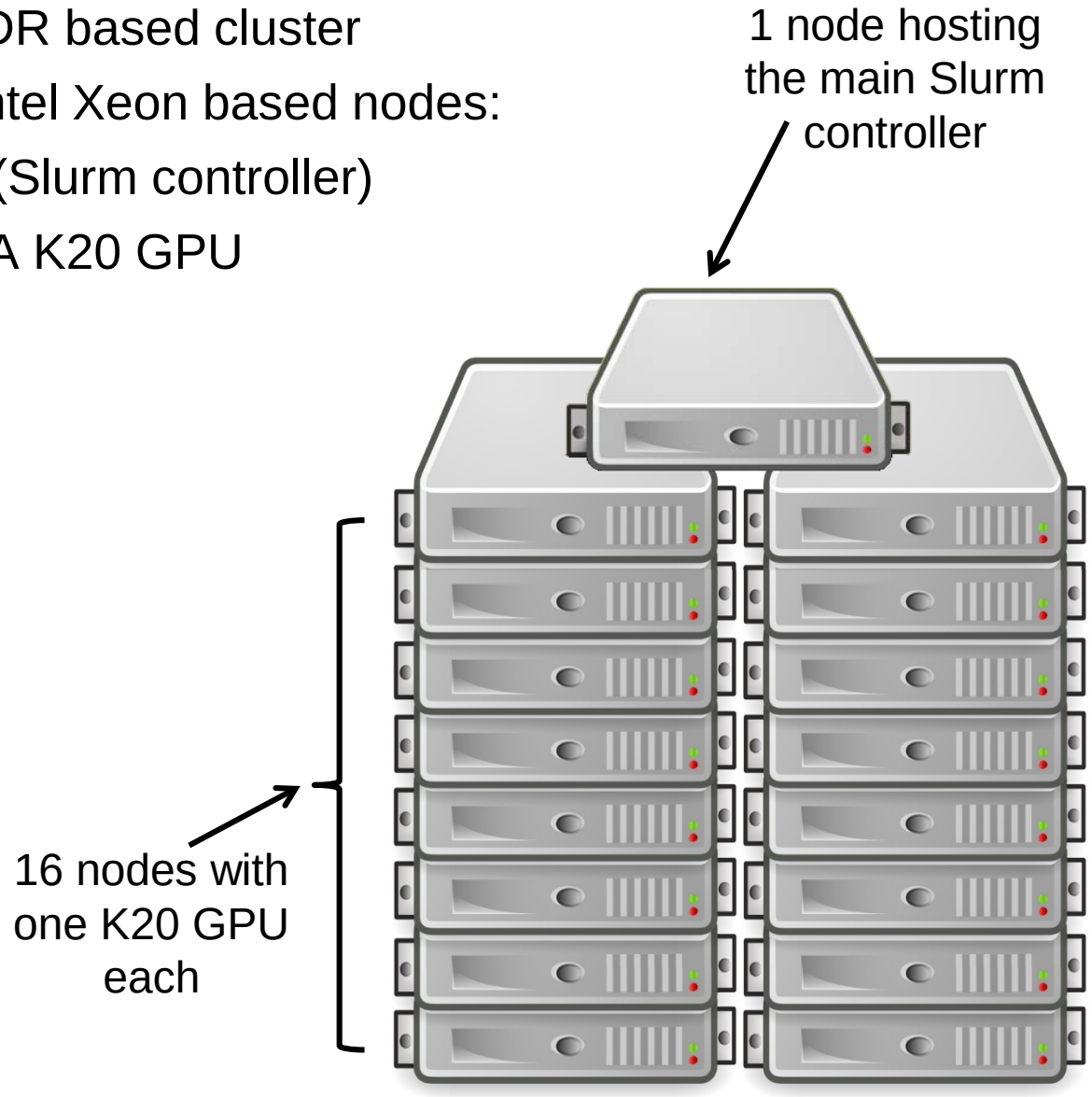
  - NAMD (11 minutes)
  - BarraCUDA (10 minutes; 1 GPU; 3319 MB)
  - GPU-LIBSVM (5 minutes; 1GPU; 145 MB)
  - MUMmerGPU (5 minutes; 1GPU; 2804 MB)

Set 2

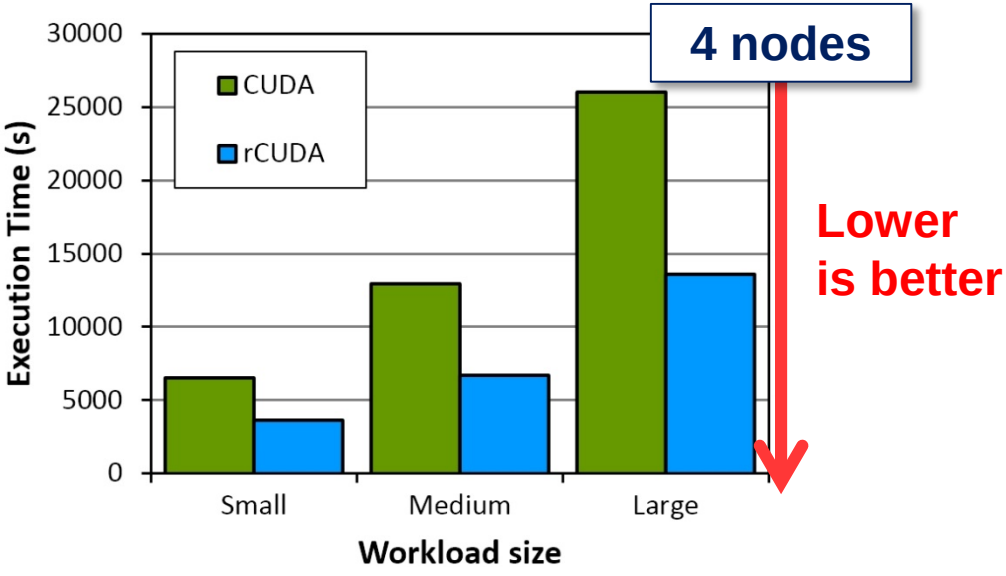
Long execution time

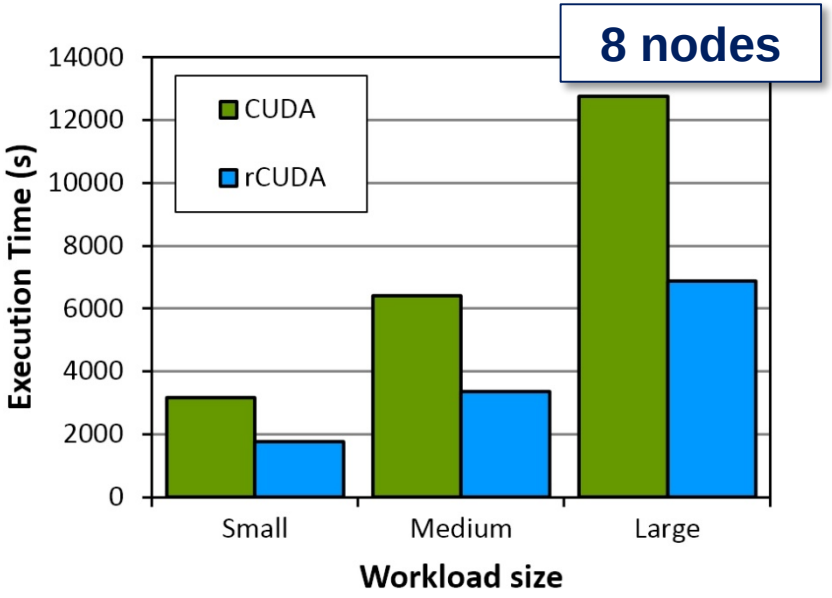
Test bench for studying rCUDA+Slurm

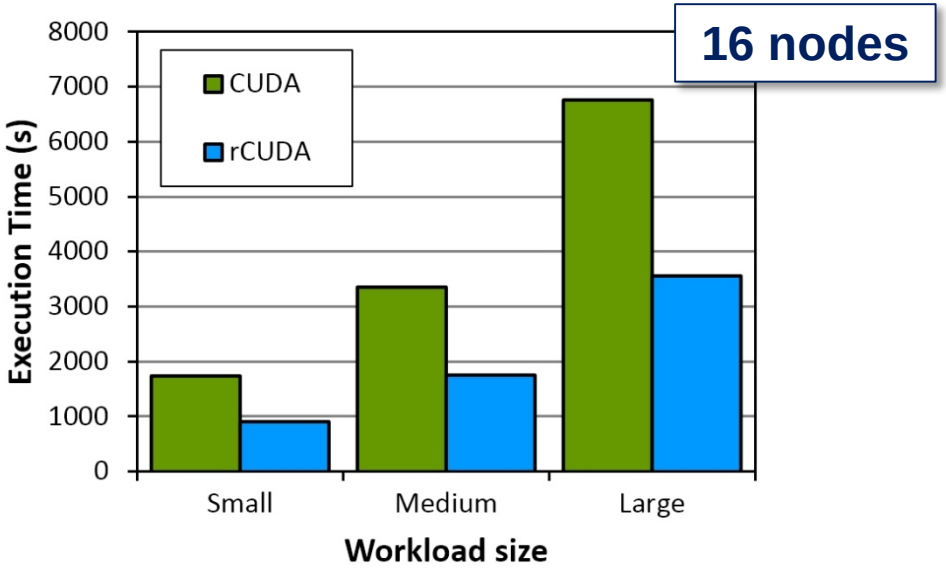
- InfiniBand ConnectX-3 FDR based cluster
- Dual socket E5-2620v2 Intel Xeon based nodes:
  - 1 node without GPU (Slurm controller)
  - 16 nodes with NVIDIA K20 GPU
- Three workloads:
  - Set 1
  - Set 2
  - Set 1 + Set 2
- Three workload sizes:
  - Small (100 jobs)
  - Medium (200 jobs)
  - Large (400 jobs)



Results for execution time

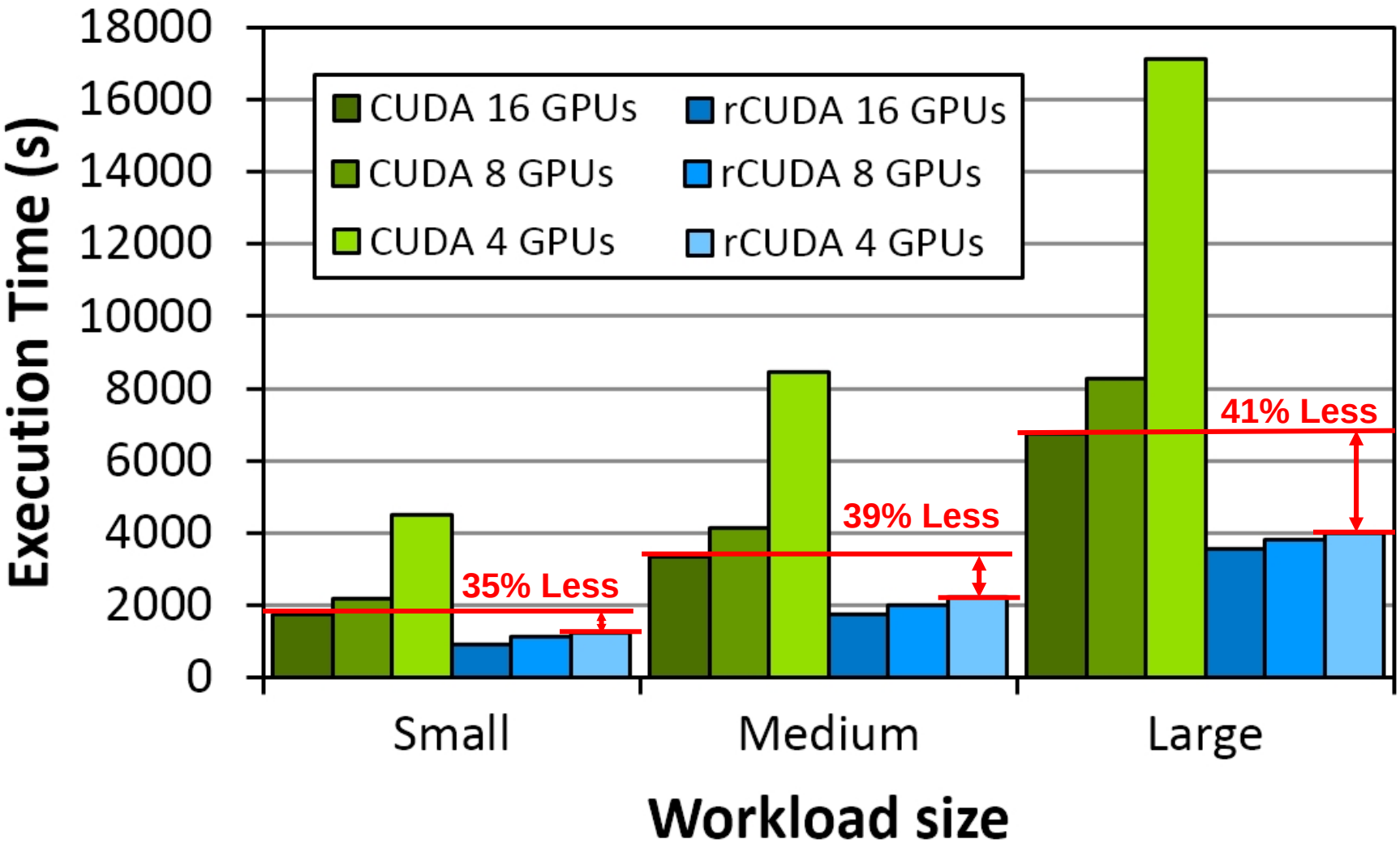




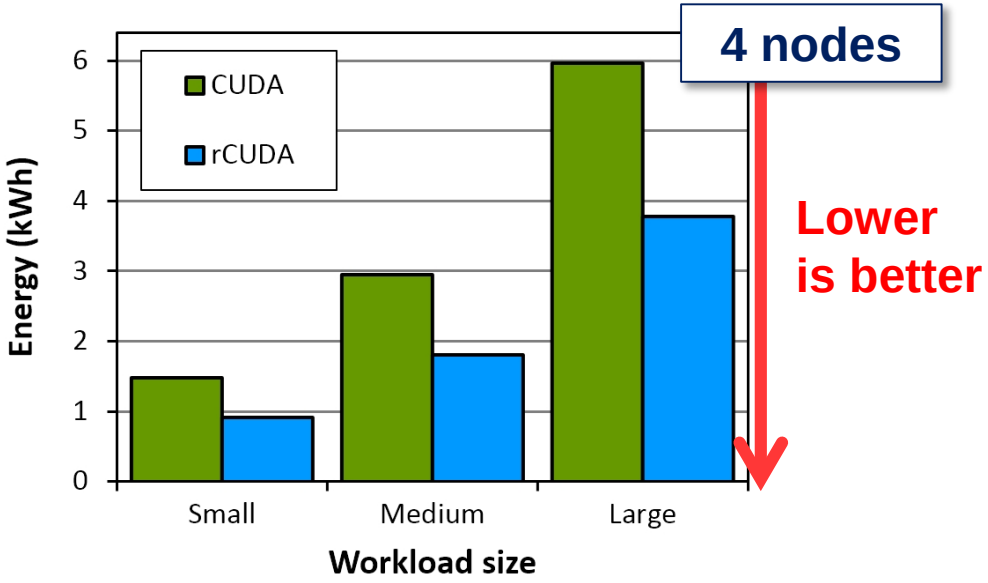


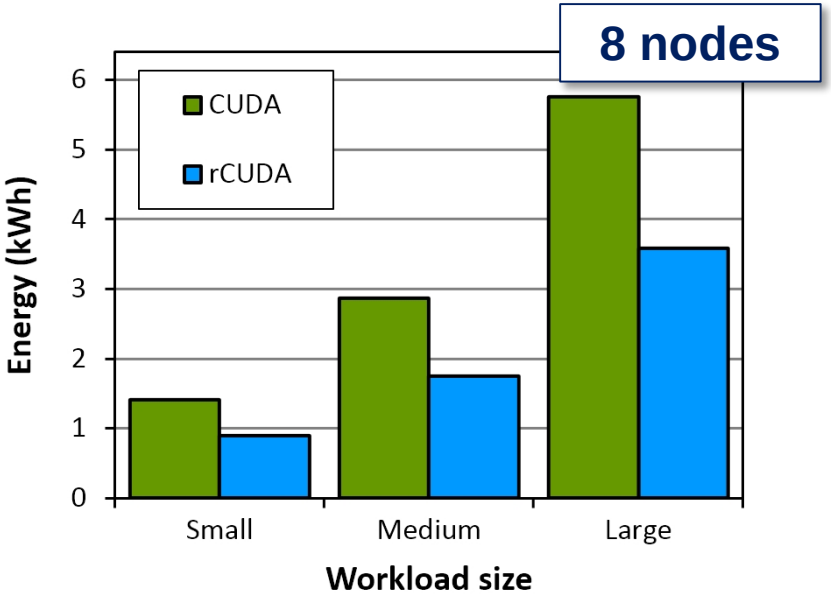
Results for Set 1

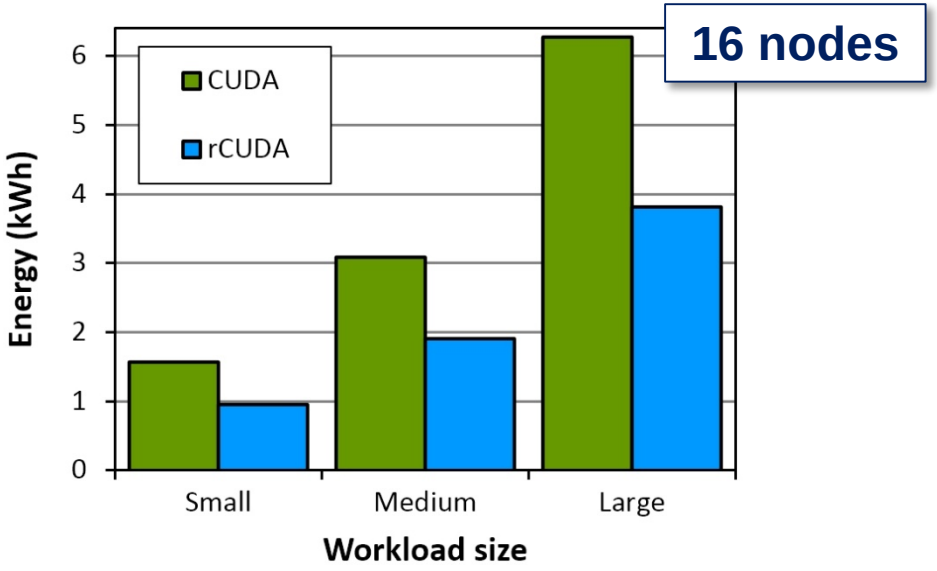
Reducing the amount of GPUs



Results for energy consumption

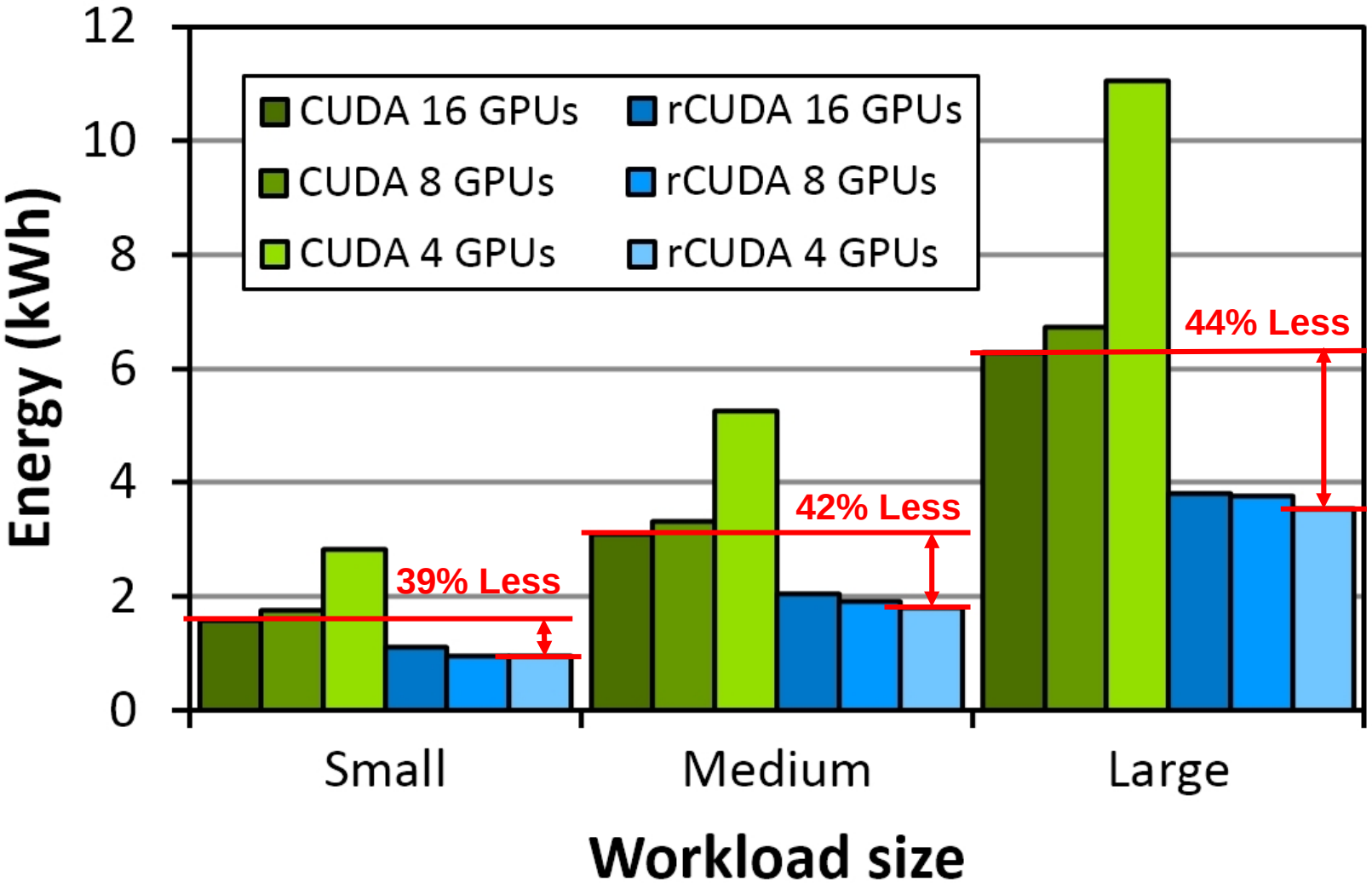






Results for Set 1

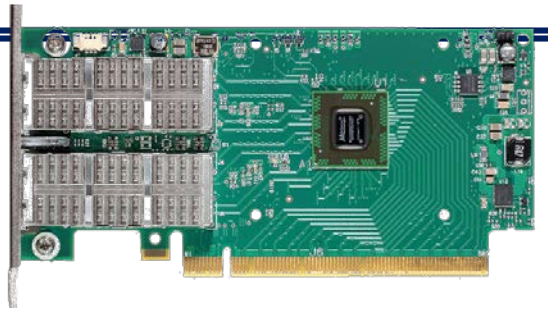
Energy when removing GPUs



5th

... in summary ...

Slurm + rCUDA allow …



- **High Throughput Computing**
  - Sharing remote GPUs makes applications to execute slower … **BUT** more throughput (jobs/time) is achieved
  - Datacenter administrators can **choose between HPC and HTC**

- **Green Computing**
  - GPU migration and application migration allow to devote just the required computing resources to the current workload

- **More flexible system upgrades**
  - GPU and CPU updates become independent from each other. Attaching GPU boxes to non GPU-enabled clusters is possible

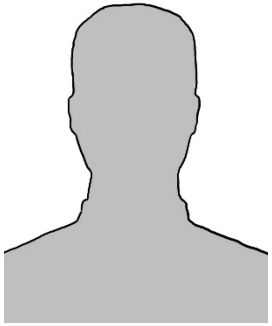
rCUDA remote CUDA

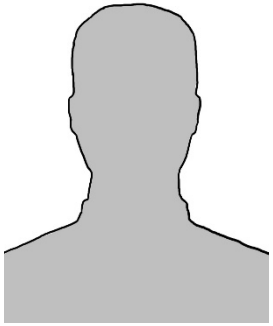
Get a free copy of rCUDA at

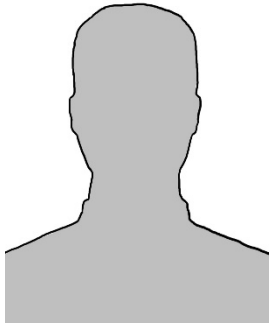
http://www.rcuda.net

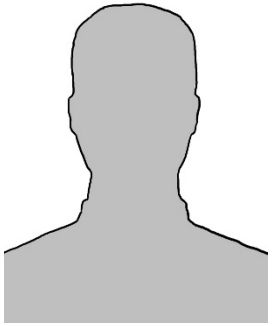
More than 650 requests world wide

@rcuda_


Sergio Iserte


Carlos Reaño


Javier Prades


Fernando Campos

rCUDA is owned by Technical University of Valencia

Thanks!

Questions?