Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Exploiting CUDA Dynamic Parallelism for low power ARM based prototypes

Vishal Mehta
Engineer, Barcelona Supercomputing Center
vishal.mehta@bsc.es

BSC/UPC CUDA Centre of Excellence (CCOE)

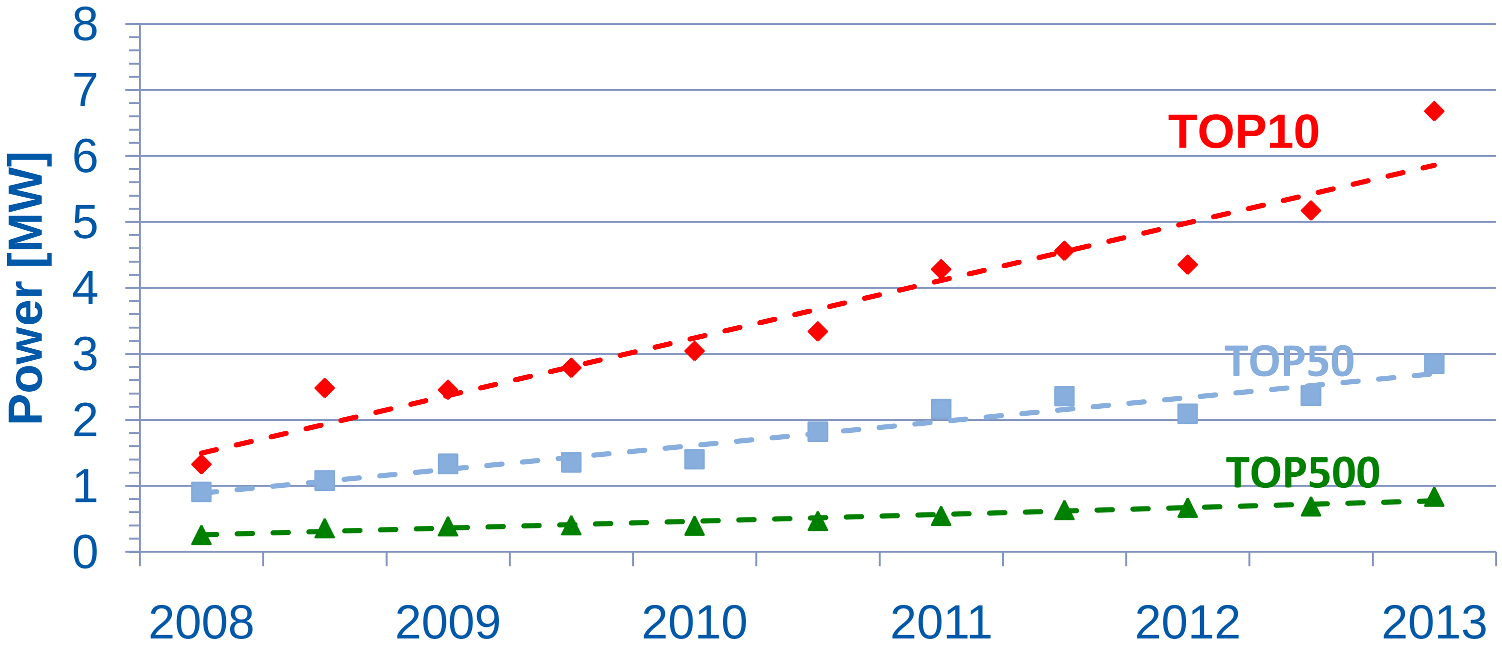
Training

- Build an education program on parallel programming using CUDA, OpenCL and OmpSs
- PUMPS summer school 2010-2015, courses at BSC and UPC

Research

- Generation, Simulation and Rendering of Large Varied Animated Crowds that attendees can get a presentation using OmpSs at current GTC
- HERTA Security GPU-based machine learning for real-time face recognition, and bio-Marketing, also presented at this GTC.
- Exploring the potential of low-power GPU clusters as high-performance platforms involved in Mont-Blanc and PRACE prototypes

Top500 Power Consumption Evolution



x5.04 in 5y

x3.13 in 5y

x3.25 in 5y

- Higher performance, at the expense of higher power

Mont-Blanc Project

MONT-BLANC

http://www.montblanc-project.eu

European approach for energy efficient HPC systems.

Objectives:

•To develop a full energy-efficient HPC prototype using low-power commercially available embedded technology.

•To develop a portfolio of exascale applications to be run on this new generation of HPC systems.

• Exploring different alternatives for the compute node (from low-power mobile sockets to special-purpose high-end ARM chips), and its implications on the rest of the system

Partners:

Euroserver Project

EURO
SERVER

http://www.euroserver-project.eu

European approach for energy efficient data servers.

Objectives:

•Reduced Energy consumption by: (i) using ARM (64-bit) cores (ii) drastically reducing the core-to-memory distance (iii) improving on the "energy proportionality".

•Reduced Cost to build and operate each microserver, (i) improved manufacturing yield (ii) reduced physical volume of the packaged interposer module (iii) and energy efficient semiconductor process (FDSOI) .

Partners:

ARM

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

FORTH
Foundation for Research & Technology - Hellas

onapp

ST
life.augmented

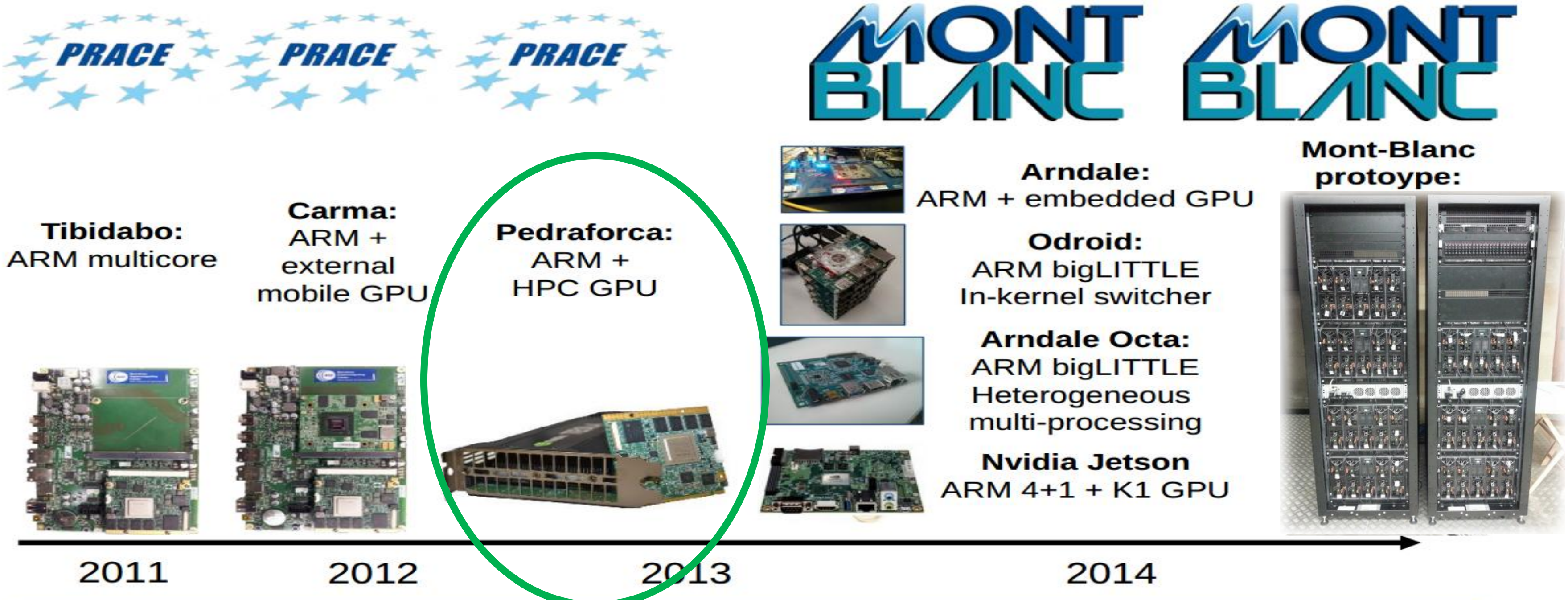
TECHNISCHE UNIVERSITÄT DRESDEN

cea

CHALMERS

EUROTECH
Imagine. Build. Succeed.

Mont-Blanc Prototype Ecosystem



Tibidabo:
ARM multicore

Carma:
ARM +
external
mobile GPU

Pedraforca:
ARM +
HPC GPU

Arndale:
ARM + embedded GPU

Odroid:
ARM bigLITTLE
In-kernel switcher

Arndale Octa:
ARM bigLITTLE
Heterogeneous
multi-processing

Nvidia Jetson
ARM 4+1 + K1 GPU

Mont-Blanc protoype:

Prototypes are critical to accelerate software development
System software stack + applications

Outline

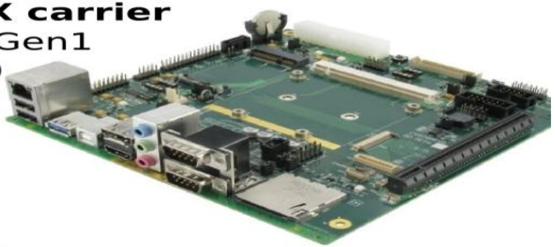
1. Pedraforca Prototype Architecture
2. Evaluation application
3. Exploiting Dynamic Parallelism
4. Some benchmarks and results
5. Limitations & Conclusions

Pedraforca : Prototype Node Architecture

Tegra 3 Q7 Module
4x ARM Cortex A9 @ 1.3 GHz
2GB DDR3

2.5'' SSD
250GB SATA 3

Mini-ITX carrier
4x PCIe Gen1
SATA 2.0
1 GbE



NVIDIA K20
16x PCIe Gen3
1170 GFLOPS
(peak)

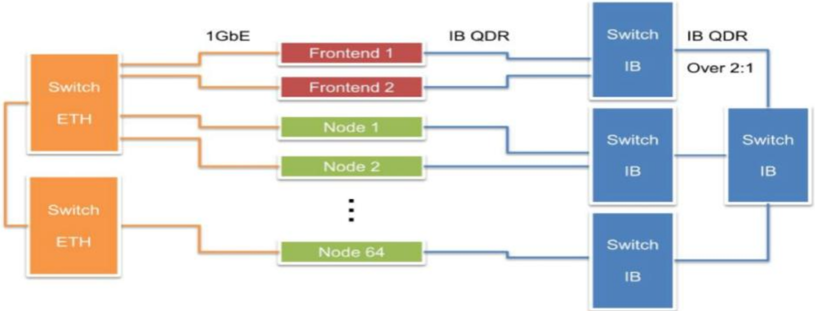
Mellanox ConnectX-3
8x PCIe Gen3 QDR

E4 ARKA single node desktop unit

Pedraforca: Cluster

3 × bullx 1200 rack

78 compute nodes
2 login nodes
4 36-port InfiniBand switches (MPI)
2 50-port GbE switches (storage)



Comparing Power Budgets

- X86_64 based system

Component	Max power usage
Tesla K20	235
Board	80
CPU	90
Total	405

Quad core Intel i5-3570K @3.4GHz , ASUS P8Z77 V-pro

- Low power ARM

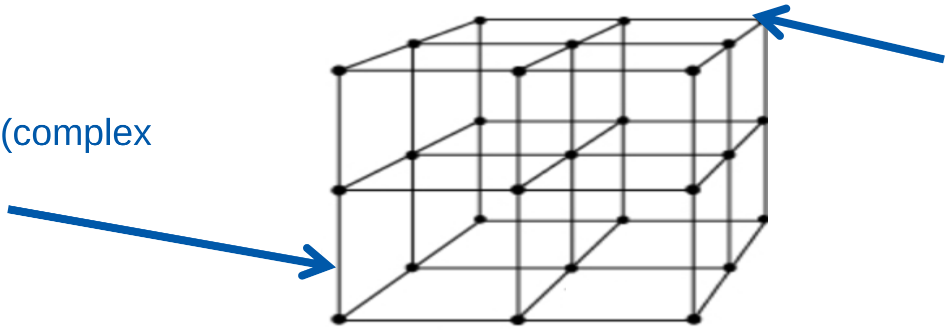
Component	Max power usage
Tesla K20	235
Board	25
CPU	5
Total	265

Tegra 3 (quad core ARM A9 @ 1.3 GHz), Mini ITX – Carrier

Outline

1. Pedraforca Prototype Architecture
2. Evaluation application
3. Exploiting Dynamic Parallelism
4. Some benchmarks and results
5. Limitations & Conclusions

Thick restarted Lanczos Algorithm in Lattice QCD



At time ‘t’

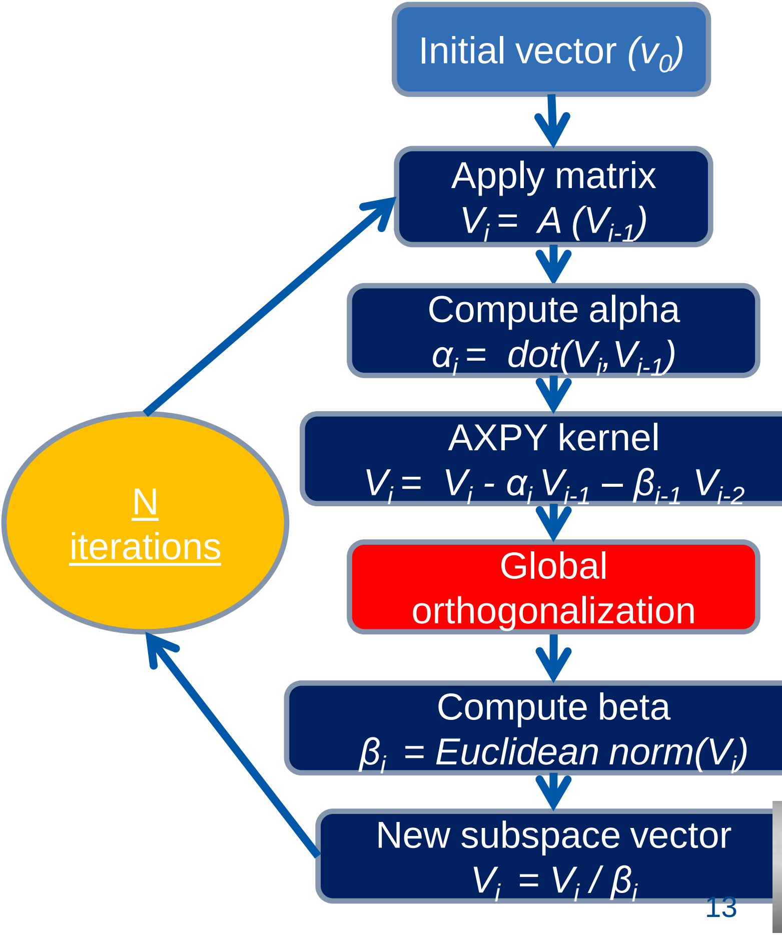
SU(3) vector (complex double)

SU(3 x 3) matrix(complex double)

- Each point on lattice is SU(3) vector and links connecting points are SU(3) matrix.
- Using thick restarted Lanczos algorithm for generating eigenpairs of the lattice
- **80 % cuBLAS routines**
- **Average number of cuBLAS calls: 60000 – 90000 depending on lattice configuration**
- **Process lattice from multiple time steps in parallel**

Evaluation Example – Lanczos Iteration

- Large number of BLAS operations
- Dominated by global orthogonalization module which includes BLAS
- Implemented using cuBLAS, highly modularized and easy to use
- Iterations are not independent of each other



Initial vector *(v_0)*

Apply matrix
$V_i = A(V_{i-1})$

Compute alpha
$\alpha_i = dot(V_i, V_{i-1})$

AXPY kernel
$V_i = V_i - \alpha_i V_{i-1} - \beta_{i-1} V_{i-2}$

Global orthogonalization

Compute beta
$\beta_i = Euclidean\ norm(V_i)$

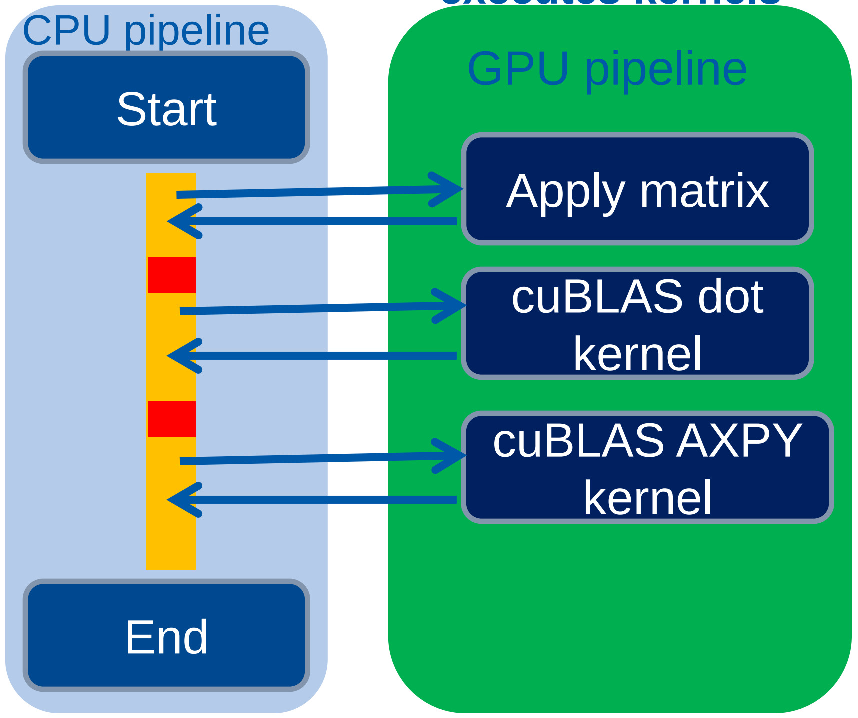
New subspace vector
$V_i = V_i / \beta_i$

N iterations

Algorithm Implementation for the Prototype

CPU works as coordinator

GPU slave executes kernels



CPU pipeline

Start

End

GPU pipeline

Apply matrix

cuBLAS dot kernel

cuBLAS AXPY kernel

🟥 Serial Dependency

Bottlenecks

- Large number of calls to cuBLAS.
- Overall algorithm is serial
- Dominated by CPU’s capability of launching cuBLAS kernels
- ARM processor is not fast enough to quickly launch kernels on GPU. GPU in underutilized

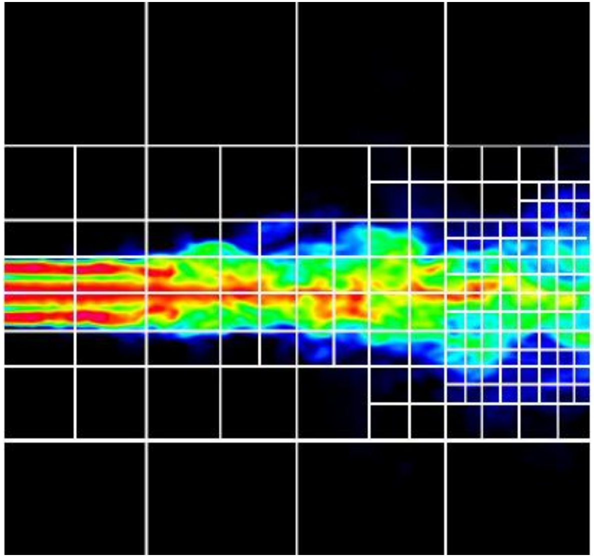
Outline

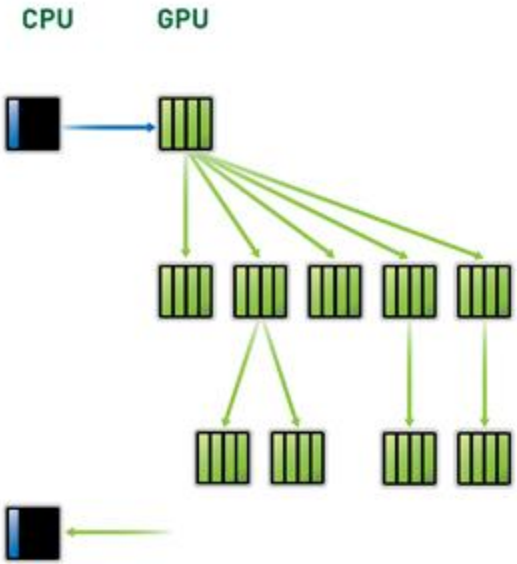
1. Pedraforca Prototype Architecture
2. Evaluation application
3. Exploiting Dynamic Parallelism
4. Some benchmarks and results
5. Limitations & Conclusions

Exploiting Dynamic Parallelism

The reason for dynamic parallelism, is to make GPU adapt to data

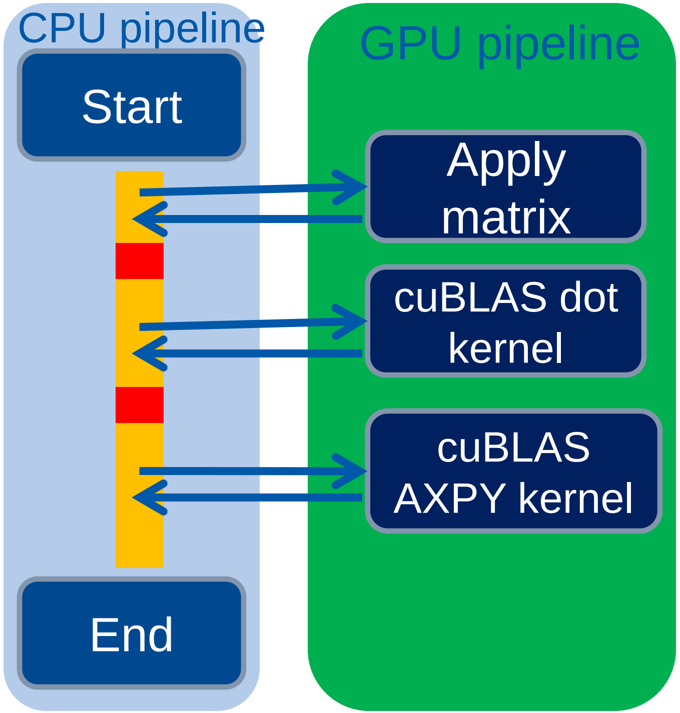
Can we exploit further to solve bottlenecks and save power ?





Approach for Exploiting Dynamic Parallelism for Low Power Prototype

CPU works as coordinator **GPU slave executes kernels**



CPU pipeline

Start

End

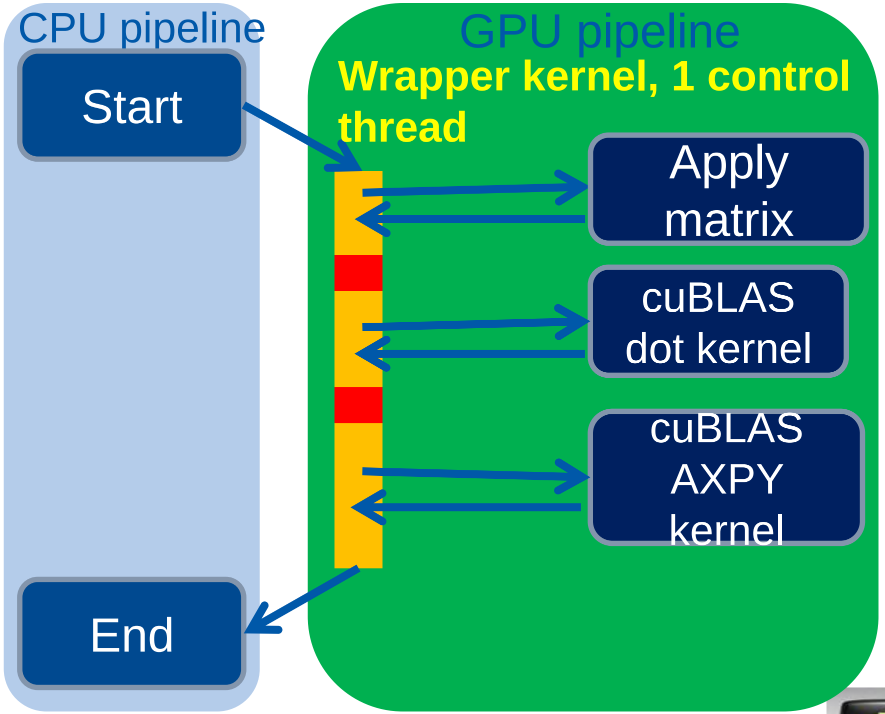
GPU pipeline

Apply matrix

cuBLAS dot kernel

cuBLAS AXPY kernel

CPU starts and ends wrapper **GPU wrapper coordinates the tasks**



CPU pipeline

Start

End

GPU pipeline

Wrapper kernel, 1 control thread

Apply matrix

cuBLAS dot kernel

cuBLAS AXPY kernel

🟥 Serial Dependency

Example code:1 - Simple Wrapper

Original code

```
__global__ Applymatrix(..,..)

int main()
{
   copytoGPU();

   Applymatrix<<<…,…>>>();
   cublasZdot();
   cublasZAXPY();

   copyfromGPU();
}
```

Code with wrapper

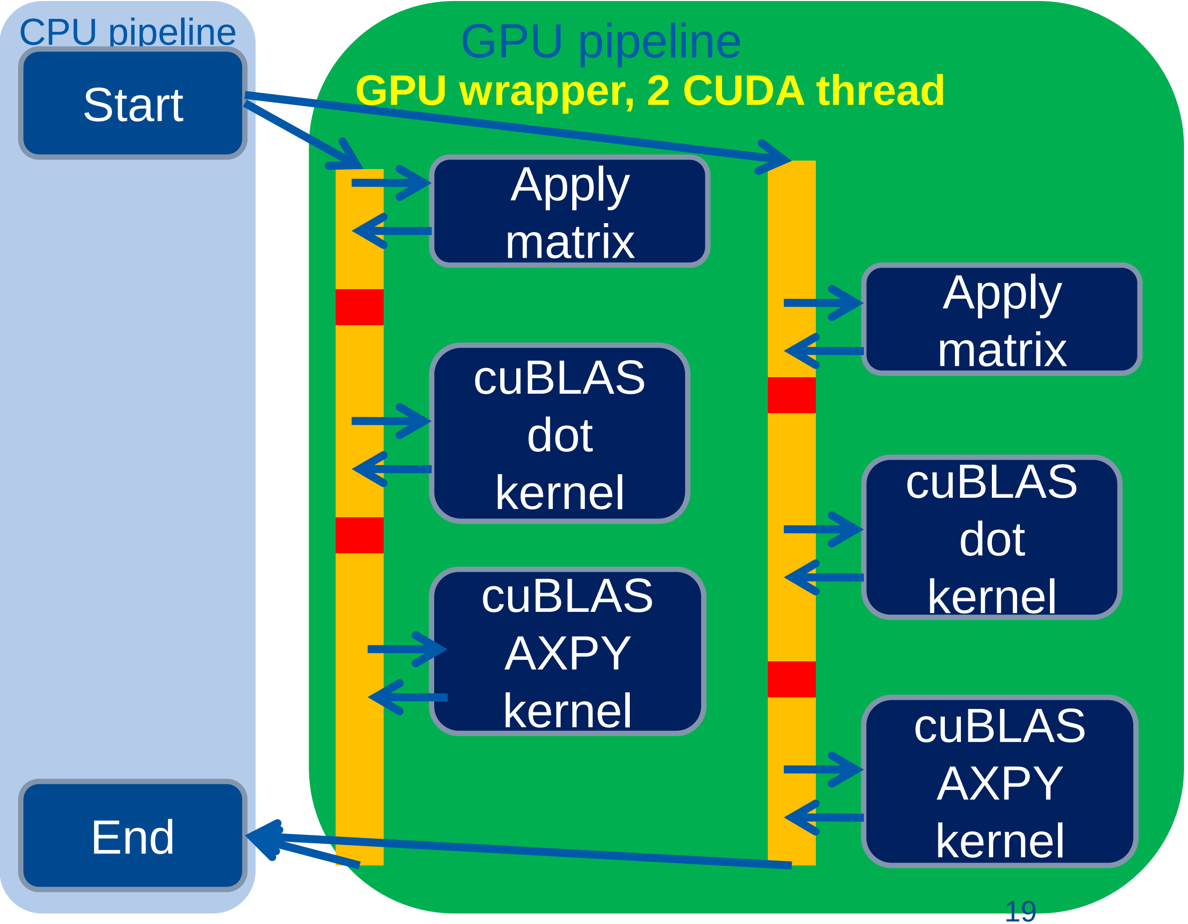
```
__global__ Applymatrix(..,..)
__global__ wrapper(..,..)
{
  Applymatrix<<<…,…>>>();
  cublasZdot();
  cublasZAXPY();
}

int main()
{
  copytoGPU();
  wrapper<<<1,1>>>();
  copyfromGPU();
}
```

Multiple Threads in Wrapper

When wrapper executed with more than one thread to process multiple instances.

Wrapper<<<1,2>>>()
PROBLEM

Threads in same block launch kernels one after another. Multiple instances are not executed simultaneously.

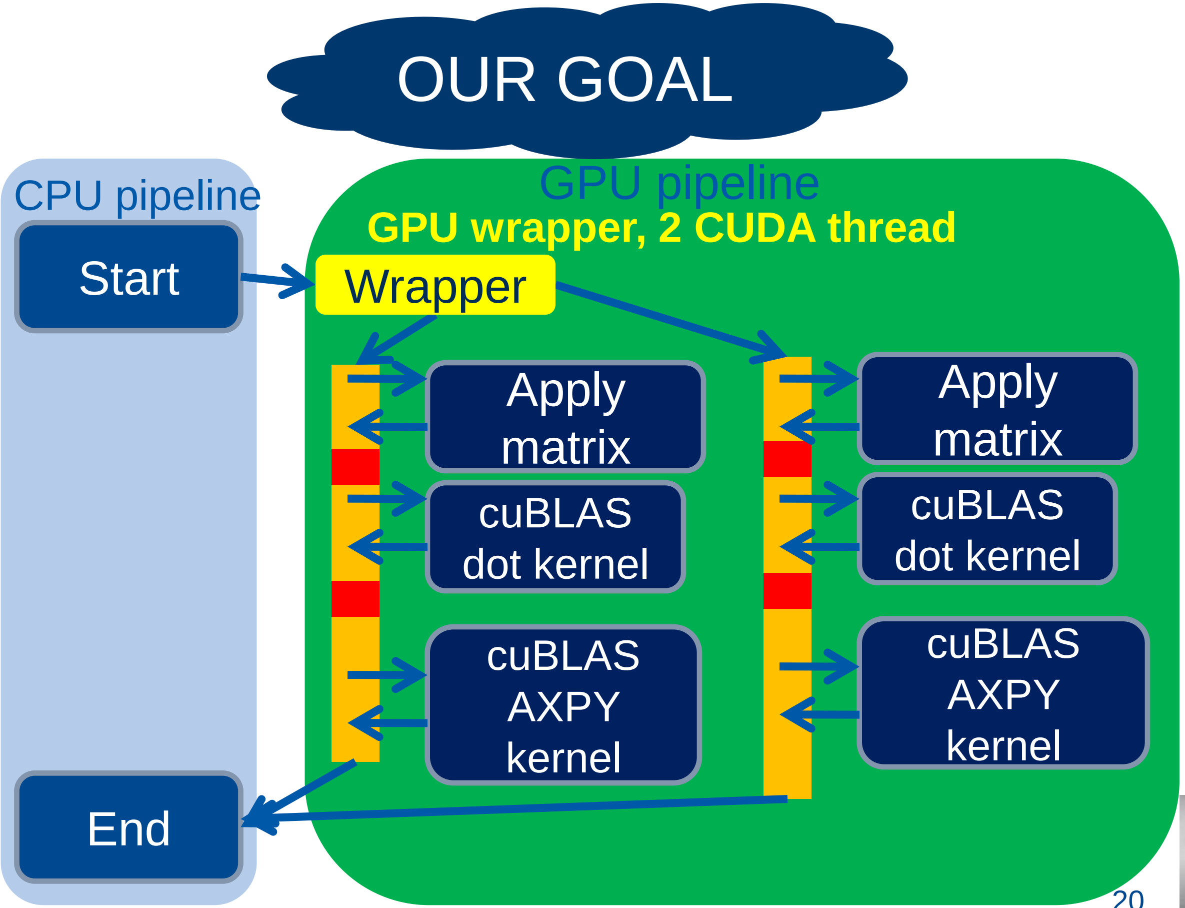


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Bottleneck caused by multiple threads in wrapper

SOLUTION

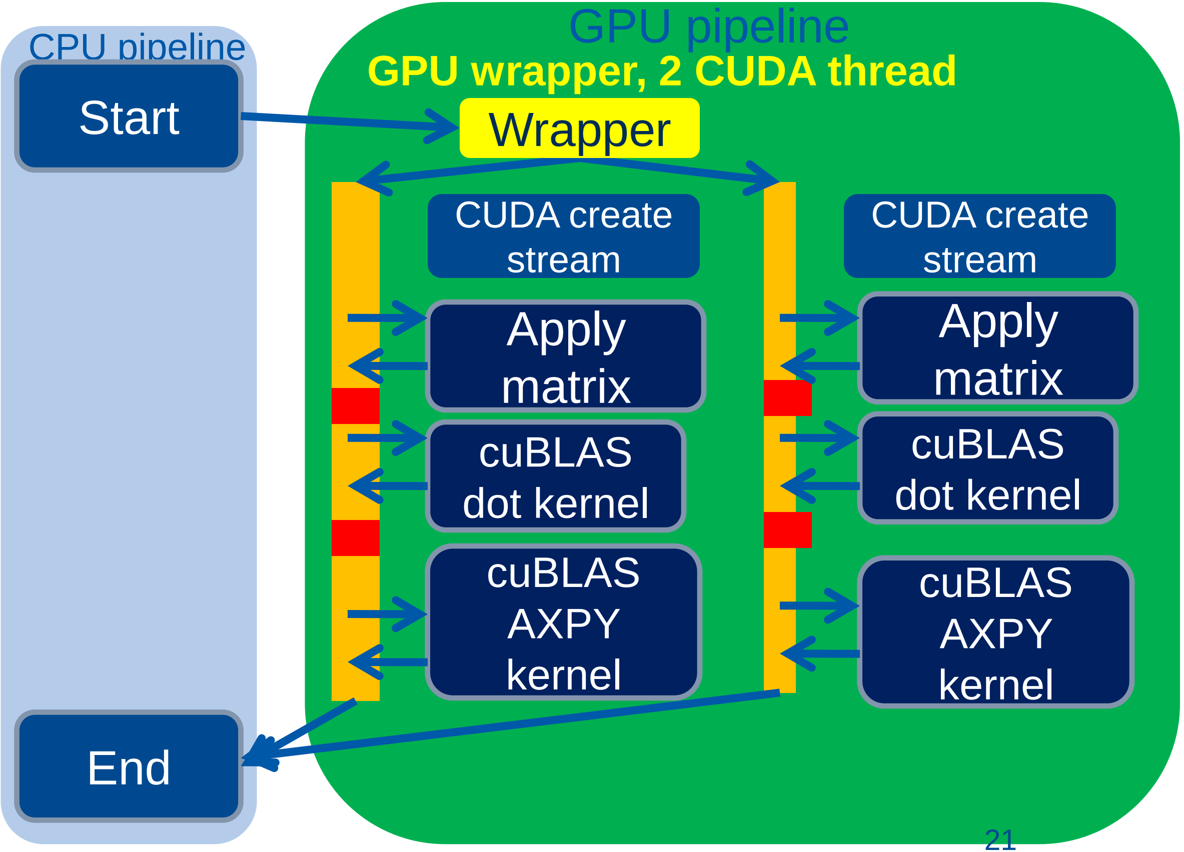
CUDA streams created on GPU side



Solution for processing multiple instances by CUDA streams

Modification to code

```
__global__ wrapper(..,..)
{
cudaStream_t  stream;
cudaStreamCreateWithFlags(&stream,cudaStreamNonBlocking);

cublasSetStream(….,stream);
Applymatrix<<<…,…stream>>>();
cublasZdot();
 cublasZAXPY();

cudaStreamDestroy(stream);
}
```



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Outline

1. Pedraforca Prototype Architecture
2. Evaluation application
3. Exploiting Dynamic Parallelism
4. Some benchmarks and results
5. Limitations & Conclusions

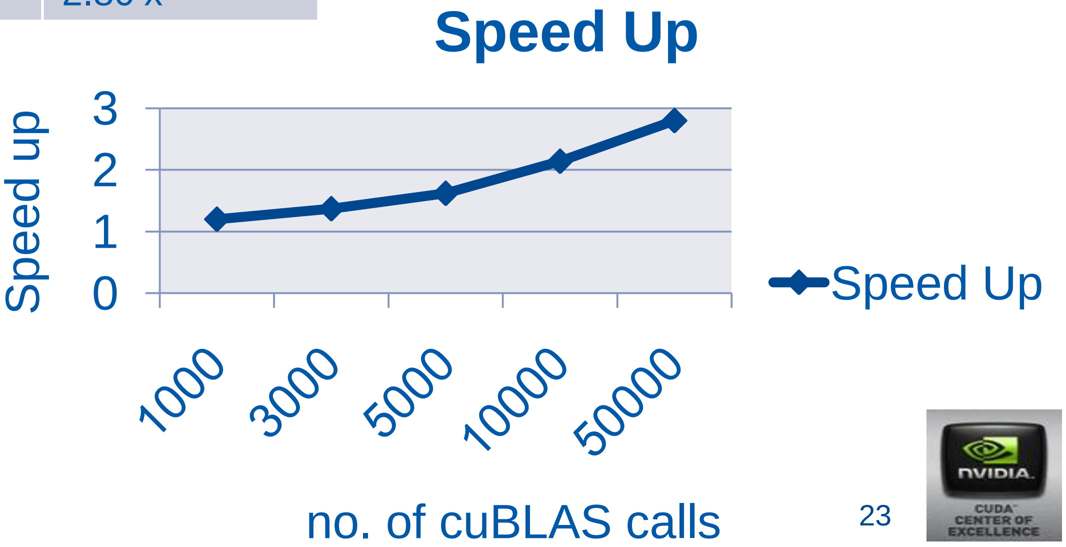
cuBLAS kernel launch scaling

No of kernel calls	cuBLAS calls by CPU (seconds)	cuBLAS calls GPU thread (seconds)	Speed up
1×10^3	1.72	1.43	1.20 x
3×10^3	2.23	1.62	1.37 x
5×10^3	4.7	2.9	1.62 x
10×10^3	7.52	3.5	2.14 x
50×10^3	11.78	4.2	2.80 x

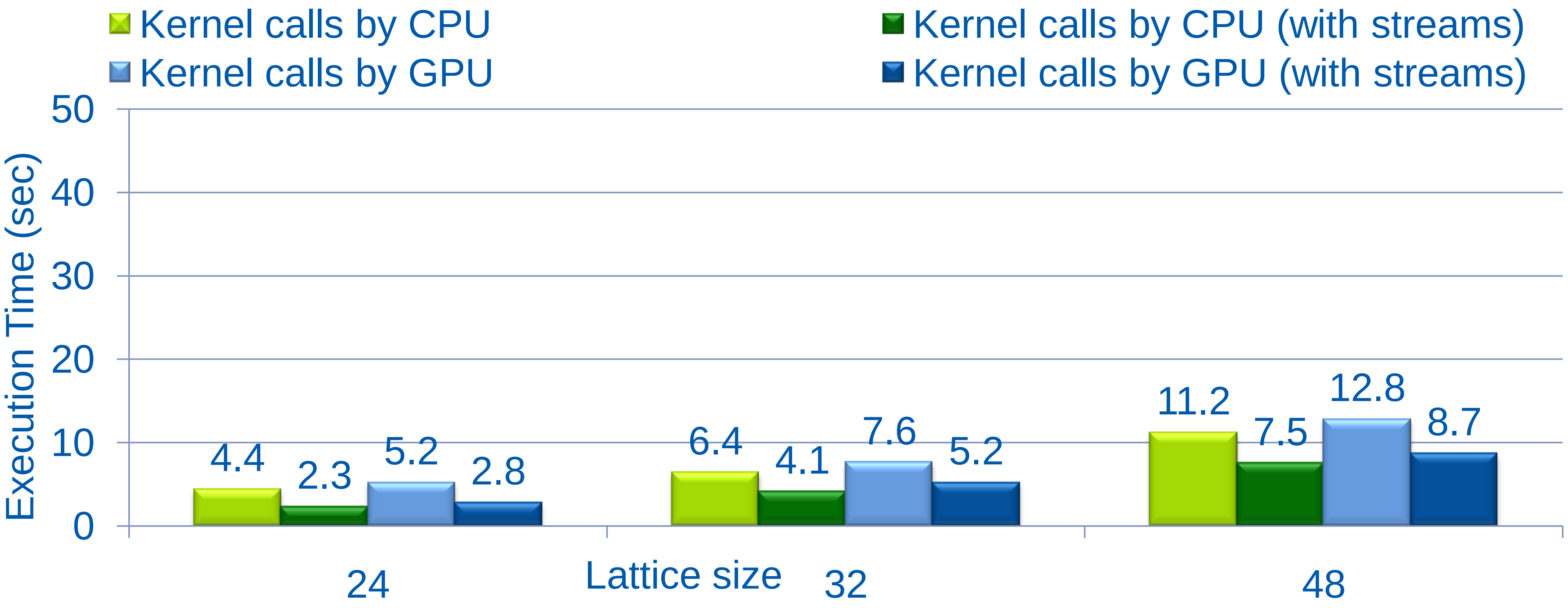
↑

cuBLAS level 1 routines

40% reduction kernel
30% AXPY kernel
30% dot product



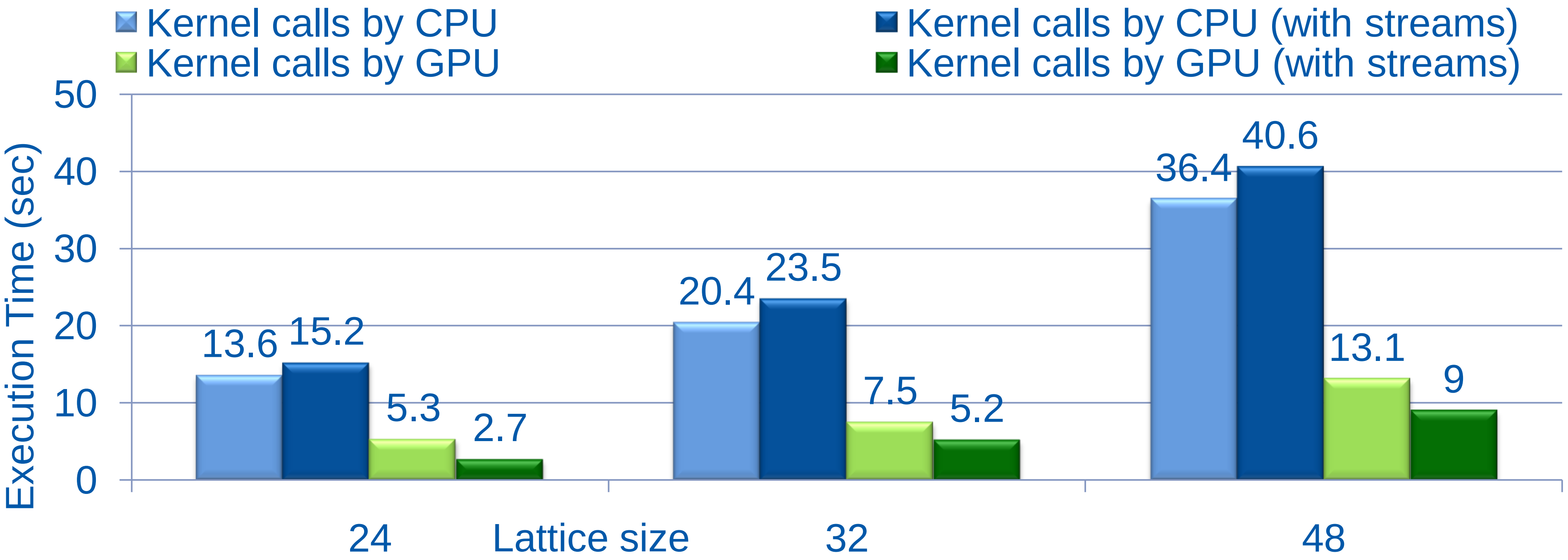
Application Performance (High Frequency CPU)



Code with wrapper may be slower on a system with fast CPU

Quad core intel i5-3570K @3.4GHz

Application Performance (Pedraforca Prototype)



Code with wrapper kernel performs better on ARM based system

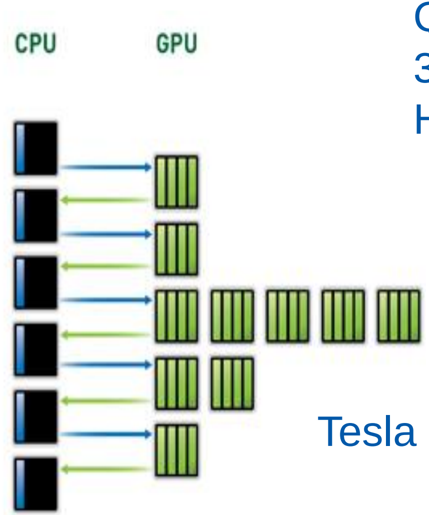
Tegra 3 - quad core ARM A9 @ 1.3 GHz

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

NVIDIA CUDA CENTER OF EXCELLENCE

Comparing systems

A

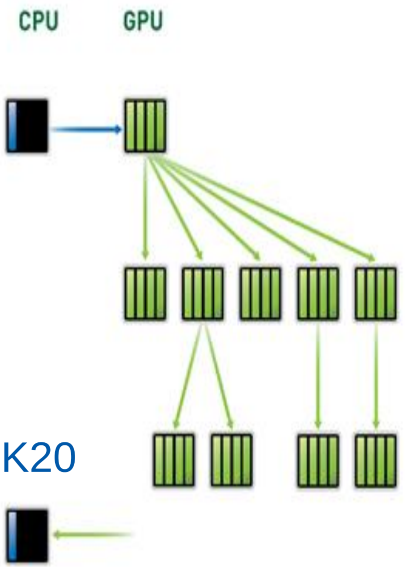


Quad core i5-3570K@3.4GHz

Tesla K20

B

Quad core ARM A9@1.3GHz



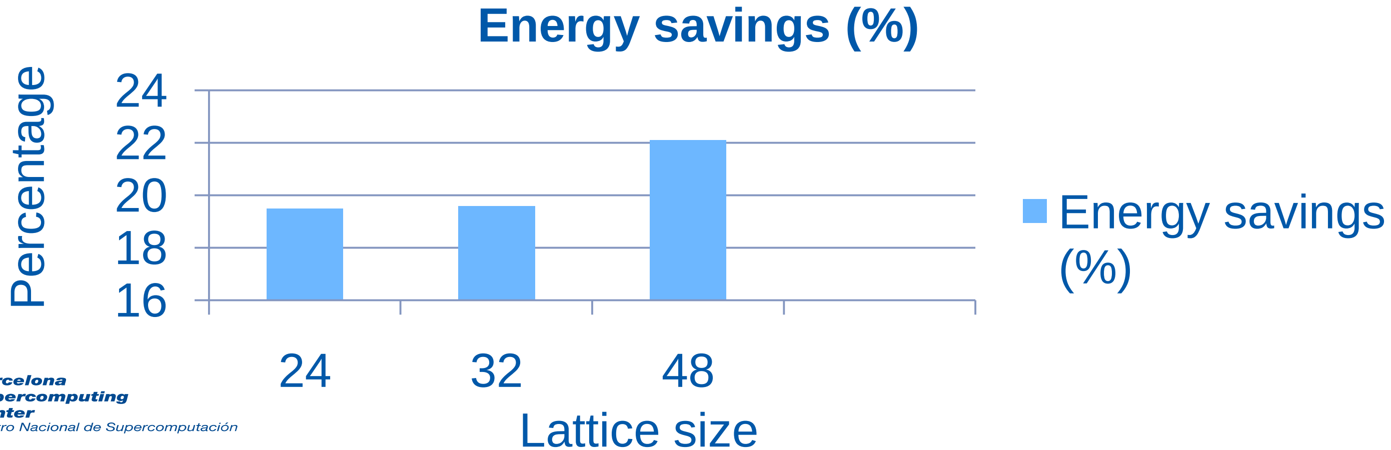
Tesla K20

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Comparing power footprint – Without CUDA streams

A : All kernels launched by CPU(Quad core intel i5-3570K@3.4GHz)
B : All kernels launched by GPU (Tegra 3-quad core ARM A9@1.3 GHz)

	Execution time (seconds)		Average Power (W)		Energy Consumption (J)	
QCD lattice size	A	B	A	B	A	B
24	4.4	5.3	367	245	1614.8	1298.5
32	6.4	7.5	359	246	2297.6	1845
48	11.2	13.1	365	243	4088	3183.3



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

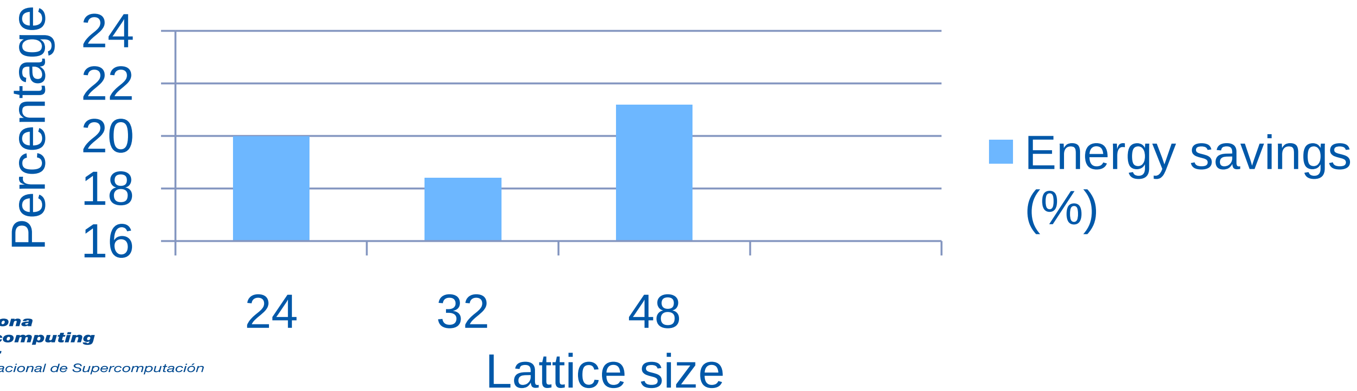
CUDA CENTER OF EXCELLENCE

Comparing power footprint – With CUDA streams

A : All kernels launched by CPU(Quad core intel i5-3570K@3.4GHz)
B : All kernels launched by GPU (Tegra 3-quad core ARM A9@1.3 GHz)

	Execution time (seconds)		Average Power (W)		Energy Consumption (J)	
QCD lattice size	A	B	A	B	A	B
24	2.3	2.7	420	286	966	772.2
32	4.1	5.2	426	287	1746.6	1392.4
48	7.5	9.0	425	282	3187.5	2538

Energy savings (%)



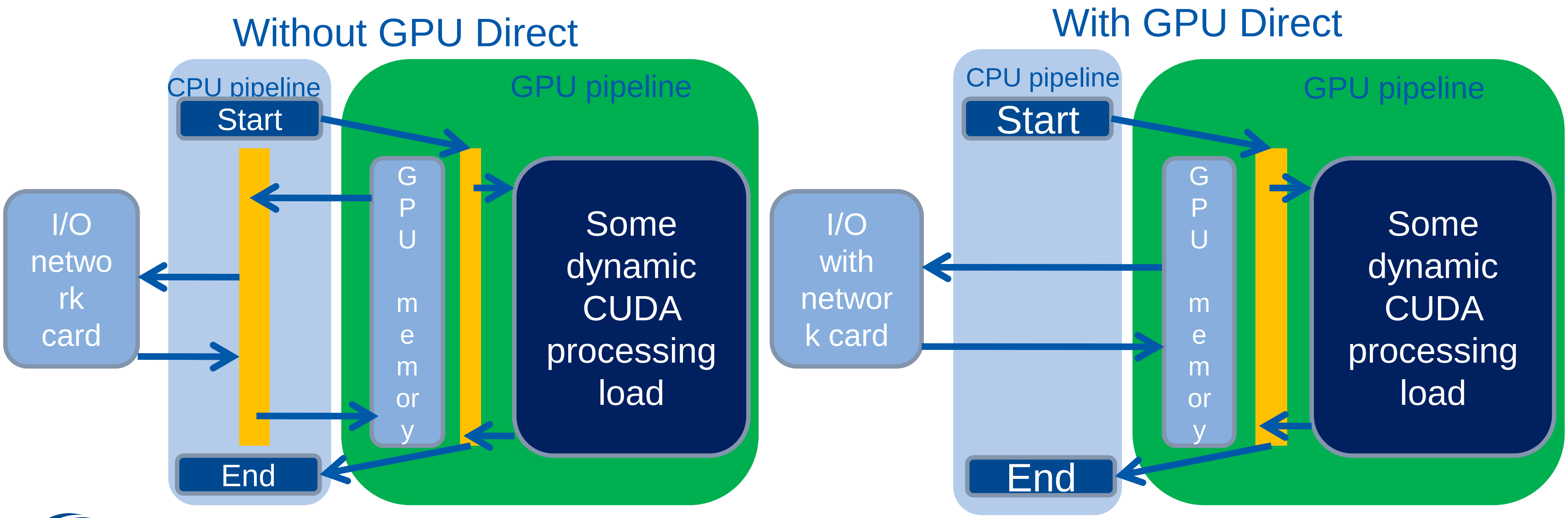
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CUDA CENTER OF EXCELLENCE

Scaling across Cluster

State of art technologies like GPU Direct, CUDA aware MPI can significantly improve data transfers among multiple nodes

Wrapper kernel ensures, low frequency CPU has sufficient time for communication.



Pedraforca limitations

Driver support for 32 bit

32 bit SoC

GOOD NEWS!!

64 bit SoC, upto 4GB support

Conclusions

- With CUDA dynamic parallelism and CUDA streams in action we are able to save roughly 20 % of power on Pedraforca prototype.

- CUDA Dynamic Parallelism helps reducing GPU-CPU communication, hence faster CPU is not always necessary.

- More libraries supporting dynamic parallelism have to be developed.

- Embedding ARM cores inside big accelerator like Tesla could be promising