Optimizing a 3D-FWT code in a cluster of CPUs+GPUs

Gregorio Bernabé Javier Cuenca Domingo Giménez

Universidad de Murcia
Scientific Computing and Parallel Programming Group

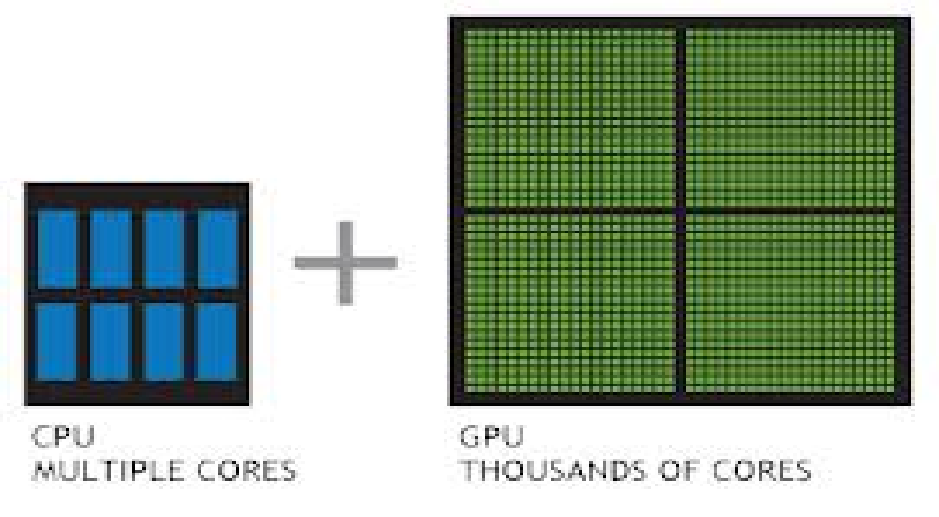
XXIX Simposium Nacional de la Unión Científica Internacional de Radio
Valencia, 3-5 septiembre 2014

Outline

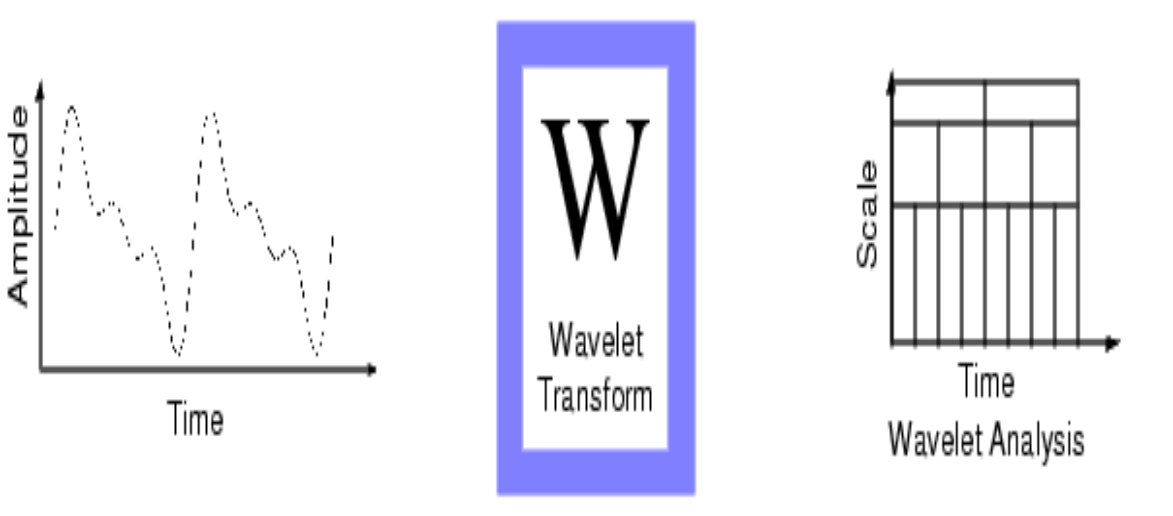
1. Introduction and Motivation
2. 3D-FWT basics
3. Autotuning architecture to run the 3D-FWT kernel automatically on clusters
4. Experimental results
5. Conclusion and Future works

Introduction and Motivation

Cluster of nodes:



⇒ To solve scientific problems like 3D-FWT:

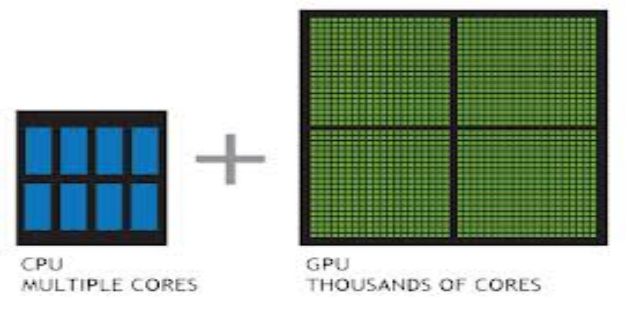


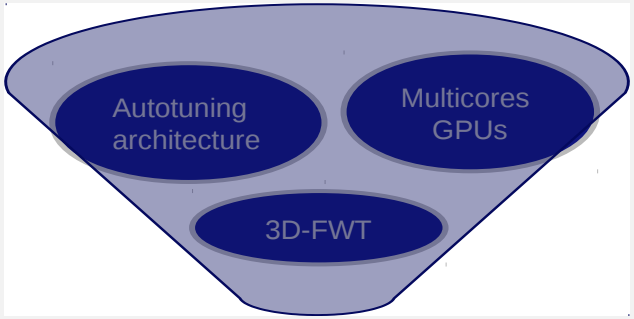
Introduction and Motivation

The development and optimization of parallel code is a complex task:

- Deep knowledge of the different components to exploit their computing capacity
- Programming and combining efficiently different programming paradigms (message passing, shared memory and SIMD GPU)

⇒ Autotuning architecture to run the 3D-FWT kernel automatically on clusters of multicore+GPUs

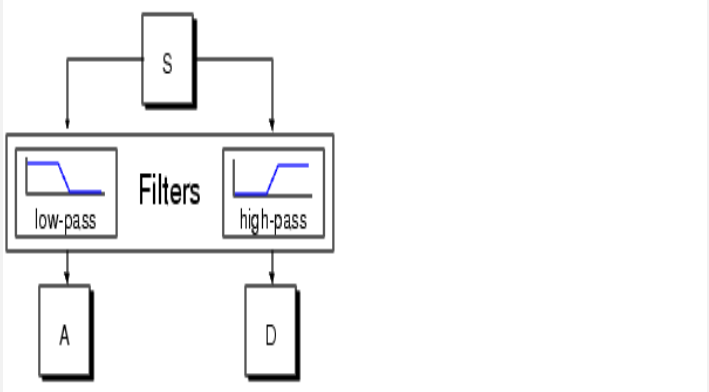




3D-FWT basics

1D-FWT

- The wavelet transform uses simple filters for fast computing
- The filters are applied to the signal.
  The filter output is downsampled by generating two bands



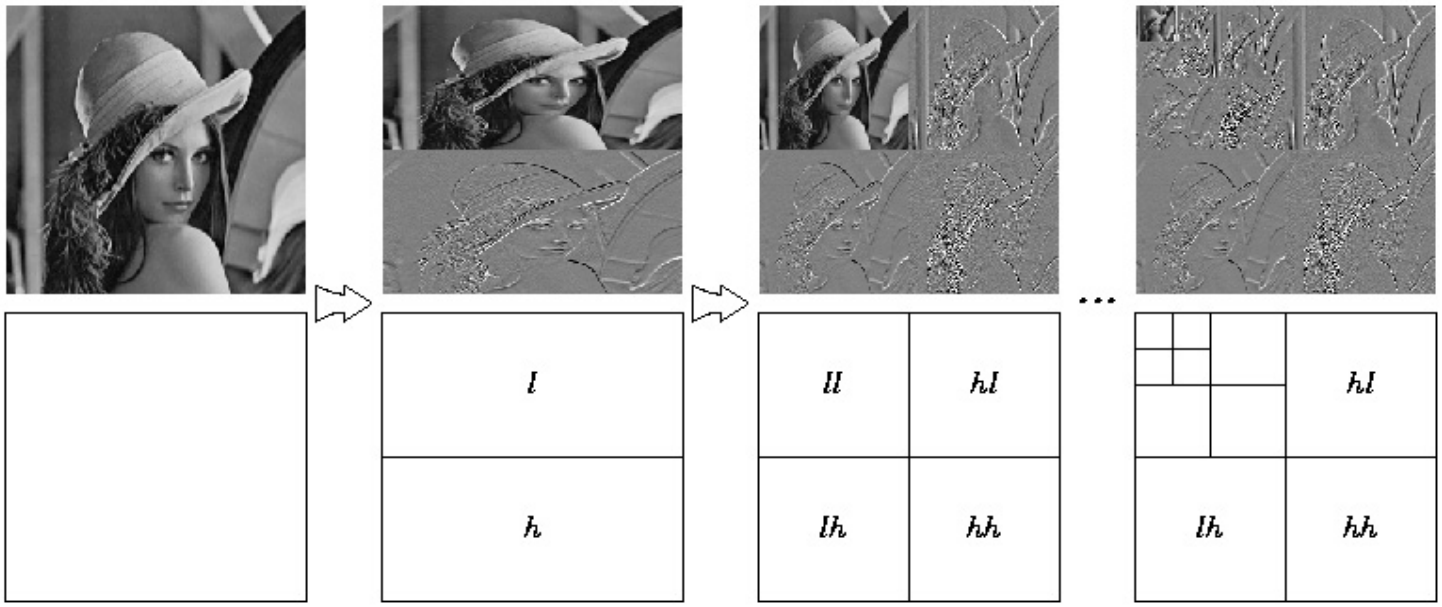
- Maintaining the amount of data on each additional level
- Access pattern is determined by the mother wavelet function

3D-FWT basics

2D-FWT

- Generalize the 1D-FWT for an image (2D)
- Applying the 1D-FWT to each row and to each column of the image

Original image Columns transformed Rows transformed ... after a three level application of the filters

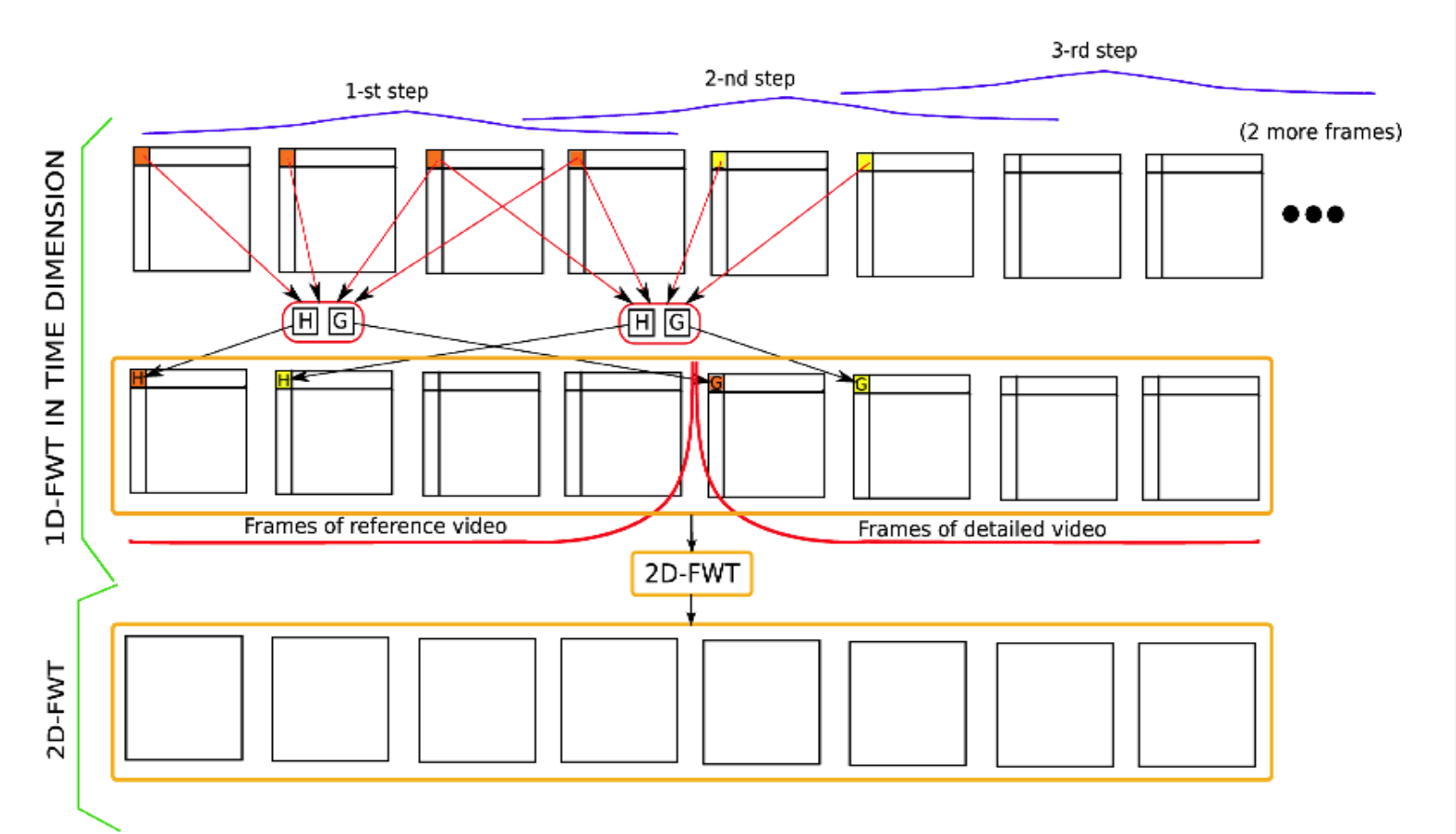


3D-FWT basics

3D-FWT

Generalize the 1D-FWT for a sequence of video (3D)

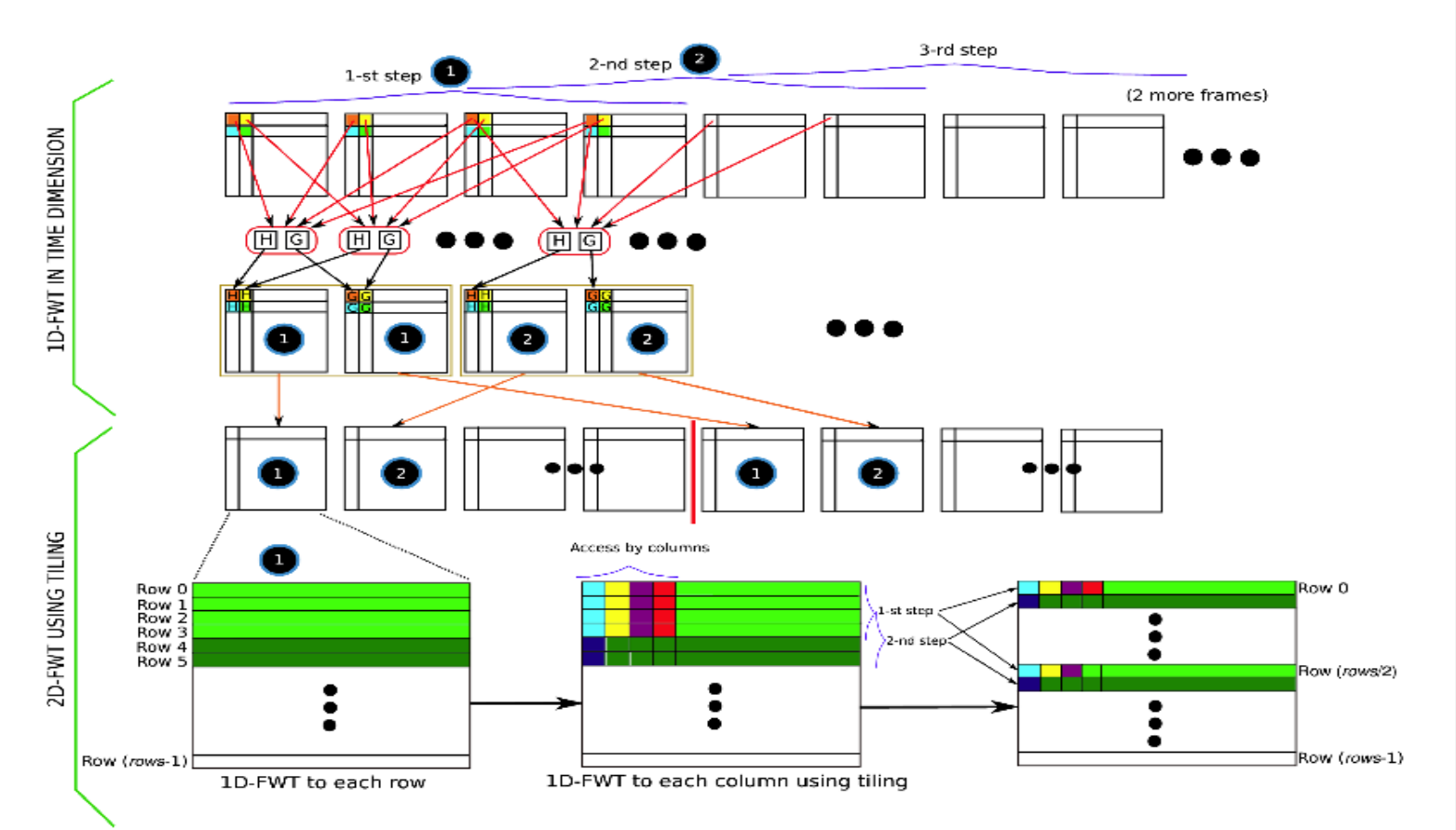
- *Nrows* $\times$ *Ncols* calls to 1D-FWT on frames
- Each of *Nframes* calls to 2D-FWT



3D-FWT basics

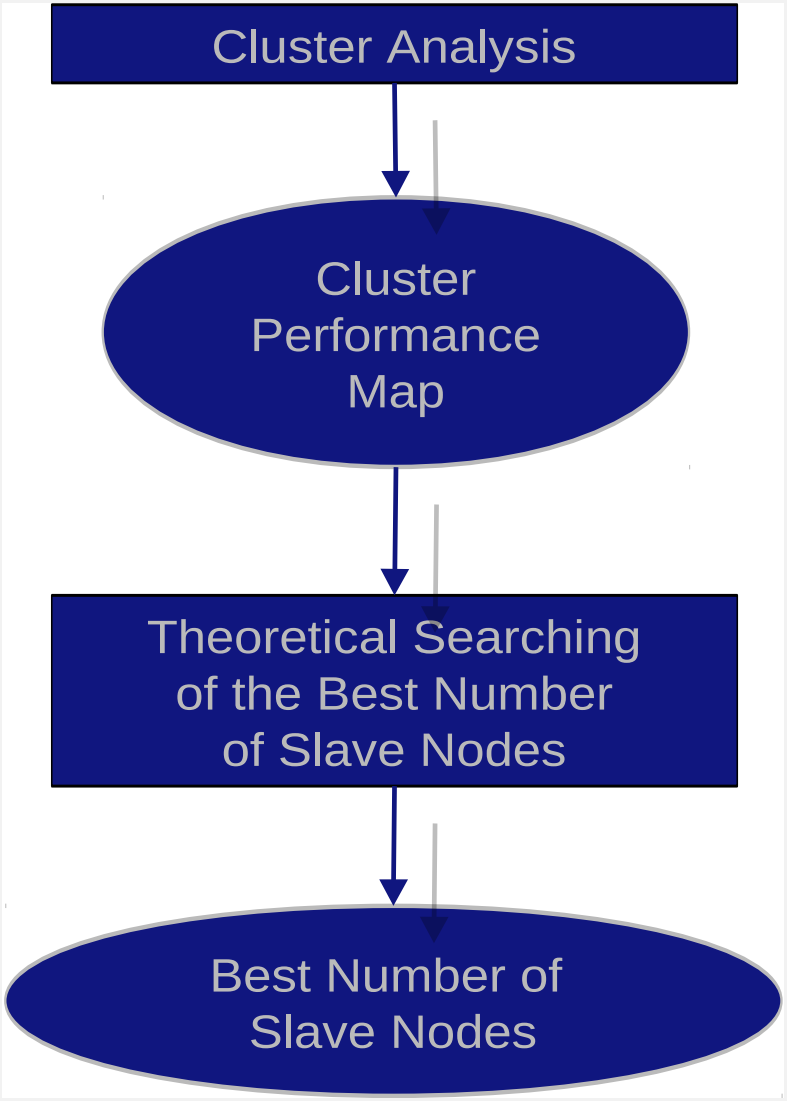
3D-FWT with tiling

- Decompose frames processing ⇒ Improving data locality
- Instead of calculating two entire frames in the time, we split them into small blocks, on which the 2D-FWT with tiling is applied



Autotuning Architecture

The architecture consists of the main objects



Cluster Analysis

- Detects automatically the available GPUs and CPUs in each node
- For each platform in each node (GPU or CPU) do:

  `If` GPU Nvidia: CUDA 3D-FWT

  $\longrightarrow$ calculates automatically block size

  `If` GPU ATI: OpenCL 3D-FWT

  $\longrightarrow$ computes automatically work-group size

  `If` CPU: Tiling and pthreads

  $\longrightarrow$ fast analysis to obtain the best number of threads

  Send one sequence

  $\longrightarrow$ Computer performance of the 3D-FWT kernel
- Measure the performance of the interconnection network among the nodes

Cluster Performance Map

For each platform (CPU or GPU)

- Number and type of CPUs and GPUs
- Computer performance of the 3D-FWT kernel
- Network bandwidth

Searching for the Best Number of Slave Nodes

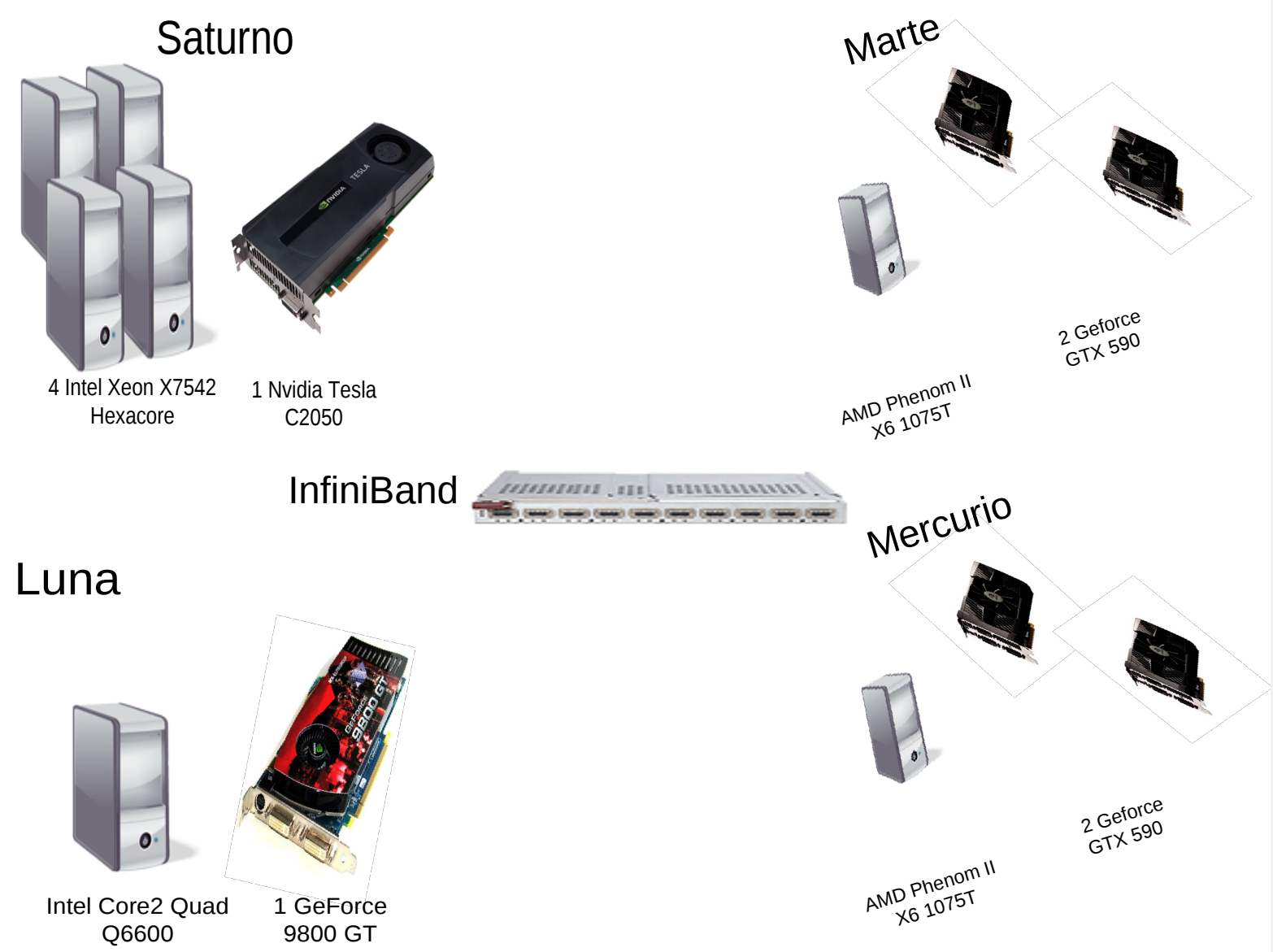
- Inputs
  - Cluster Performance Map
  - Routines of the 3D-FWT
  - Input data (the video sequences)
- Outputs
  - The temporal distribution scheme for different number of slave nodes
  - Selects the Best Number of Slave Nodes

Searching for the Best Number of Slave Nodes in a cluster of $N + 1$ nodes

- A temporal distribution scheme, $TS(n)$, for the n most powerful slave nodes working jointly with the master node (`for` $n = 1$ `to` N)
- Taking into account a double balance goal:
  - Inter-node coarse grain balance: Distribution of the workload between the slaves according to their relative performances when solving data sequences
  - Intra-node fine grain balance: Inside each node, balance the CPU - GPU workload

Execution Environment

The master process is in Saturno and the other nodes are the candidates to be slaves

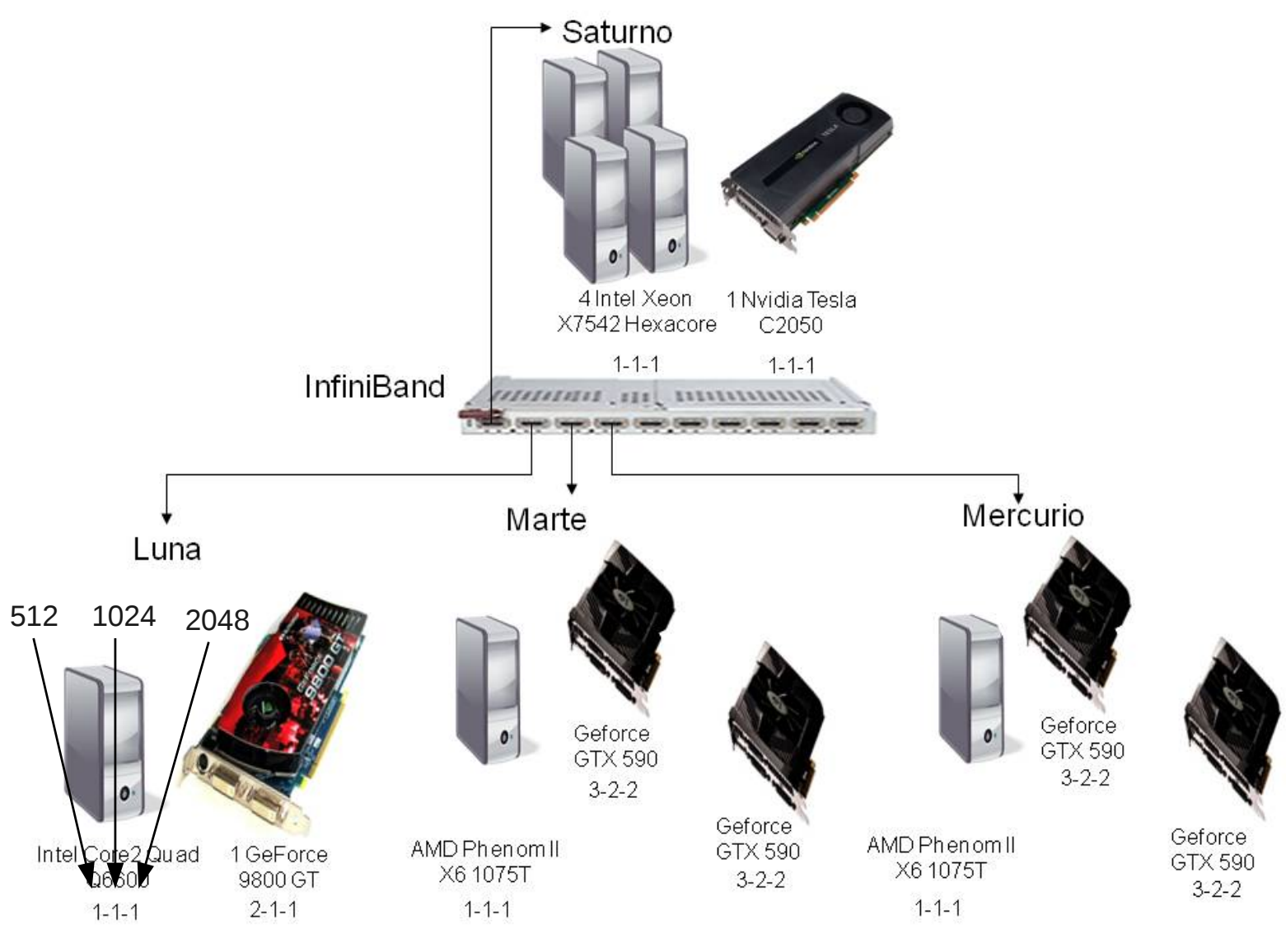


Cluster Analysis

- Detects automatically the available GPUs and CPUs in each node
- Selects the CUDA implementation for the Nvidia GPUs
- Studies the problem for different block sizes and varying the number of threads:
  - Selects block size 192 in the Tesla C2050, GeForce GTX 590 and GeForce 9800 GT GPU
  - the 3D-FWT tiling version with pthreads for the CPUs, with number of threads:
    4 for Luna, 12 for Saturno and 12 for Mercurio and Marte
- Executes an MPI program to determine the bandwidth (1.5 Gbytes/s)
- Computes the performance of the 3D-FWT kernel for each node and platform

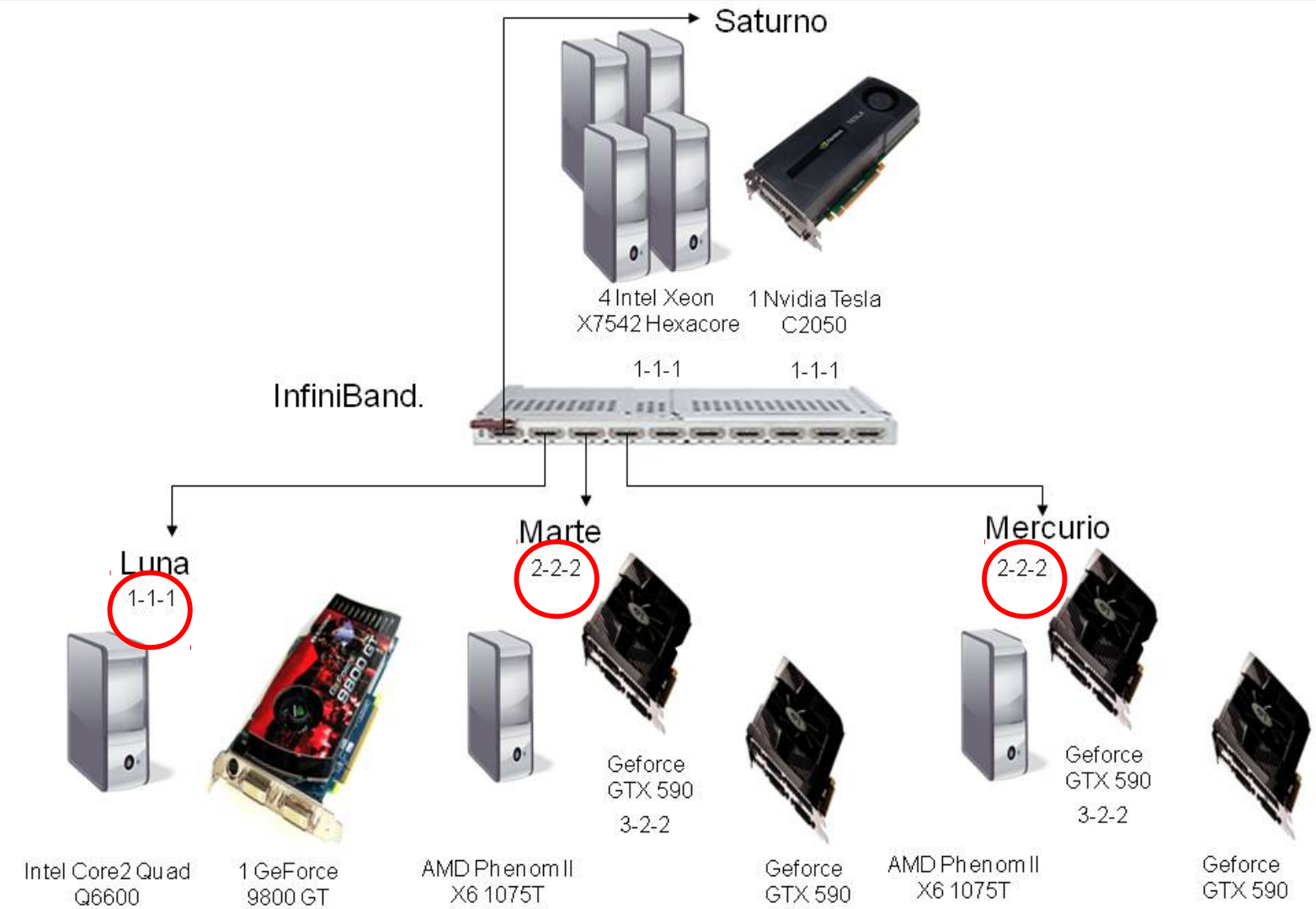
Cluster Performance Map

Under the name of each platform in a node, the three numbers represent the relative speedups of the platform in the node for different image resolutions



Cluster Performance Map

Under the name of the slave nodes, the three numbers represent the speedups of the nodes (Marte and Mercurio) with respect to the slowest slave node (Luna) for different image resolutions



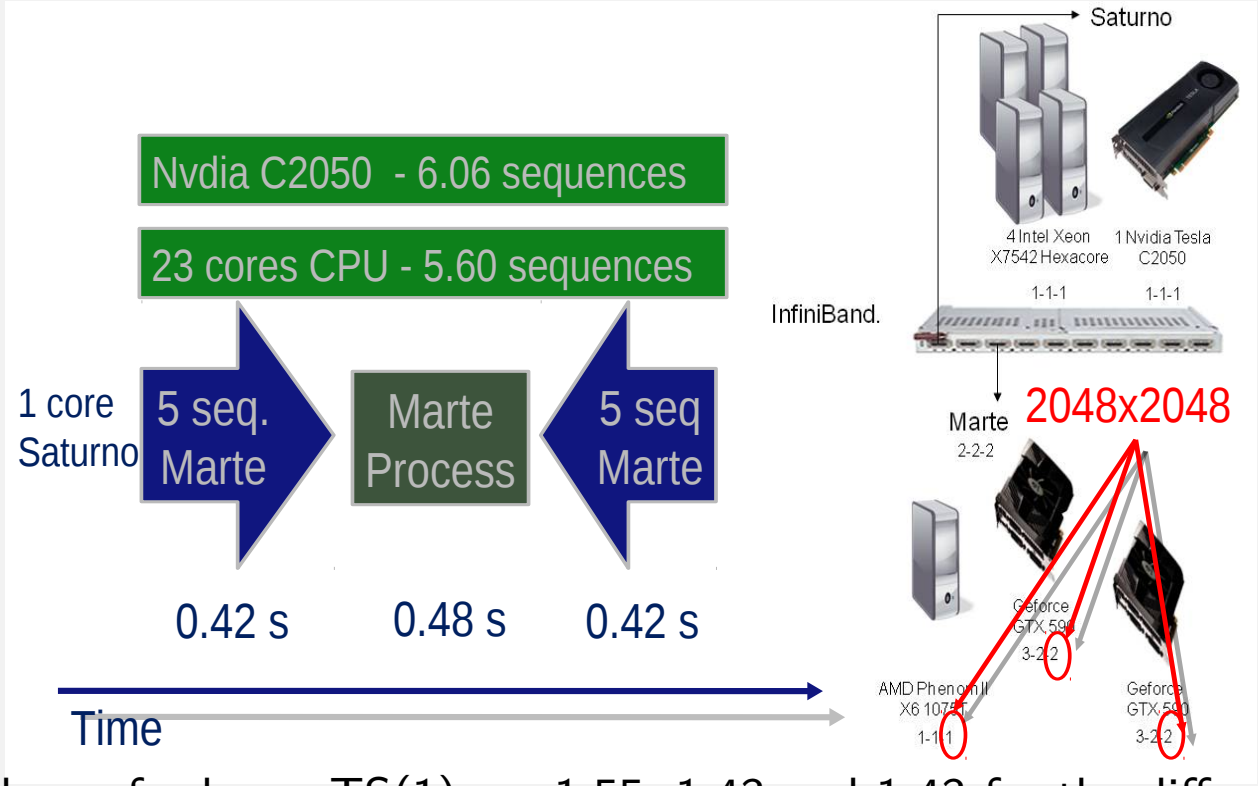
Searching of the Best Number of Slave Nodes

- Objectives:
  - Obtain a distribution to provide jobs for all platforms in the cluster
  - Maintain the proportion among the nodes
  - Once the sequences are received at each slave node, they are divided in proportion to the computer performance of each platform in the node
- Builds different possible Temporal distribution schemes for one, two and three slaves

TS for 1 slave node

For master node Saturno and slave node Marte, for images 2048 × 2048

- The master sends 5 sequences to the slave, because the computer performance between the platforms in Marte is 2 for each GPU and 1 for the multicore
- Marte processes the 5 sequences and the master node waits for the results
- During shipment, processing and reception, the GPU and the other cores of Saturno process sequences

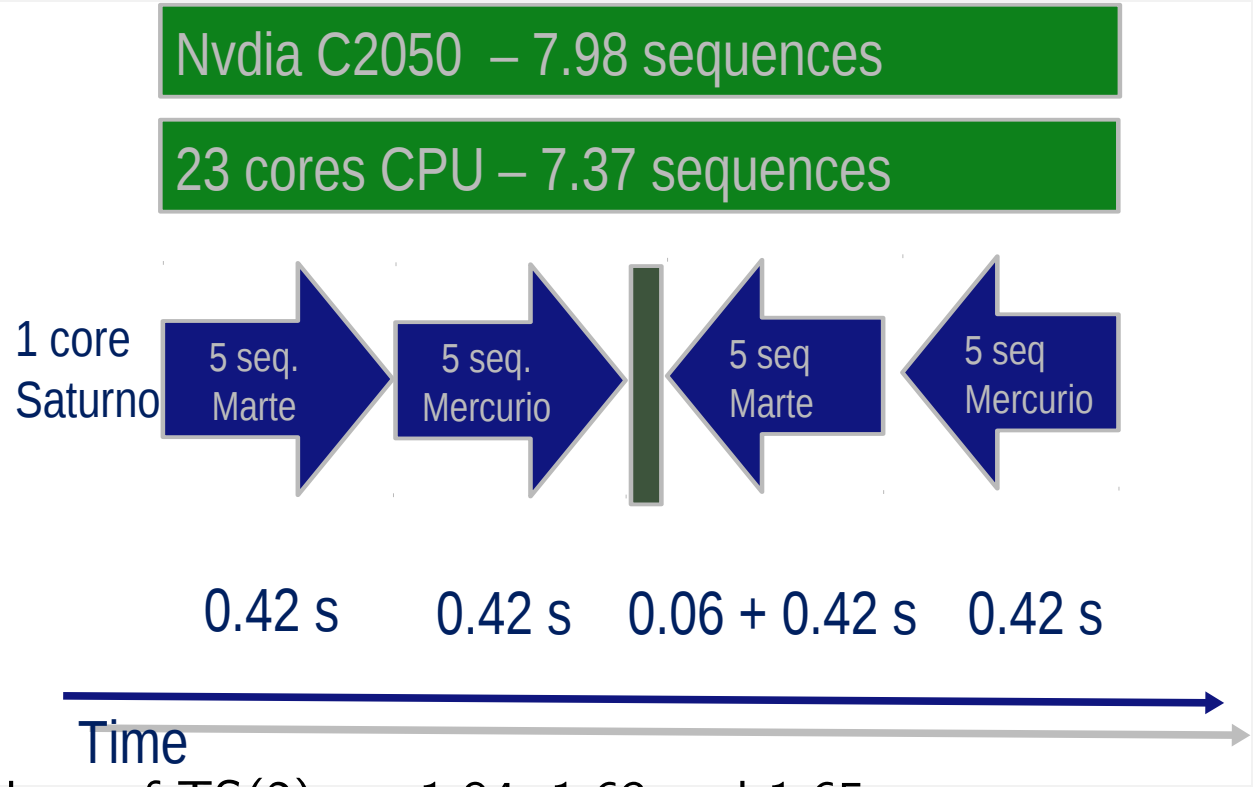


Speedups of scheme TS(1) are 1.55, 1.43 and 1.43 for the different image resolutions

TS for 2 slave nodes

For master node Saturno and slave nodes Marte and Mercurio

- The master sends 5 sequences to each slave
- Marte and Mercurio process the sequences
- During shipment, processing and reception, the GPU and the other cores of Saturno process sequences

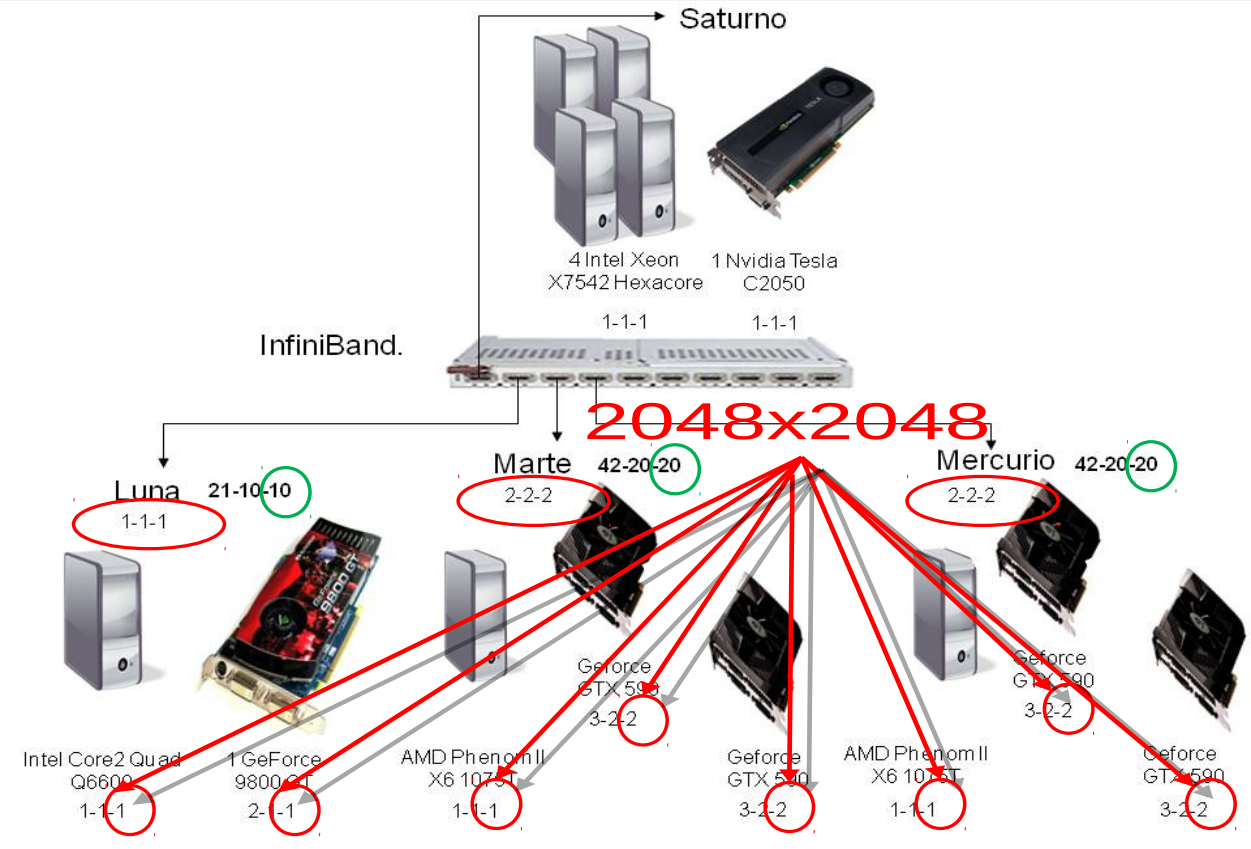


Speedups of TS(2) are 1.84, 1.63 and 1.65

TS for 3 slave nodes

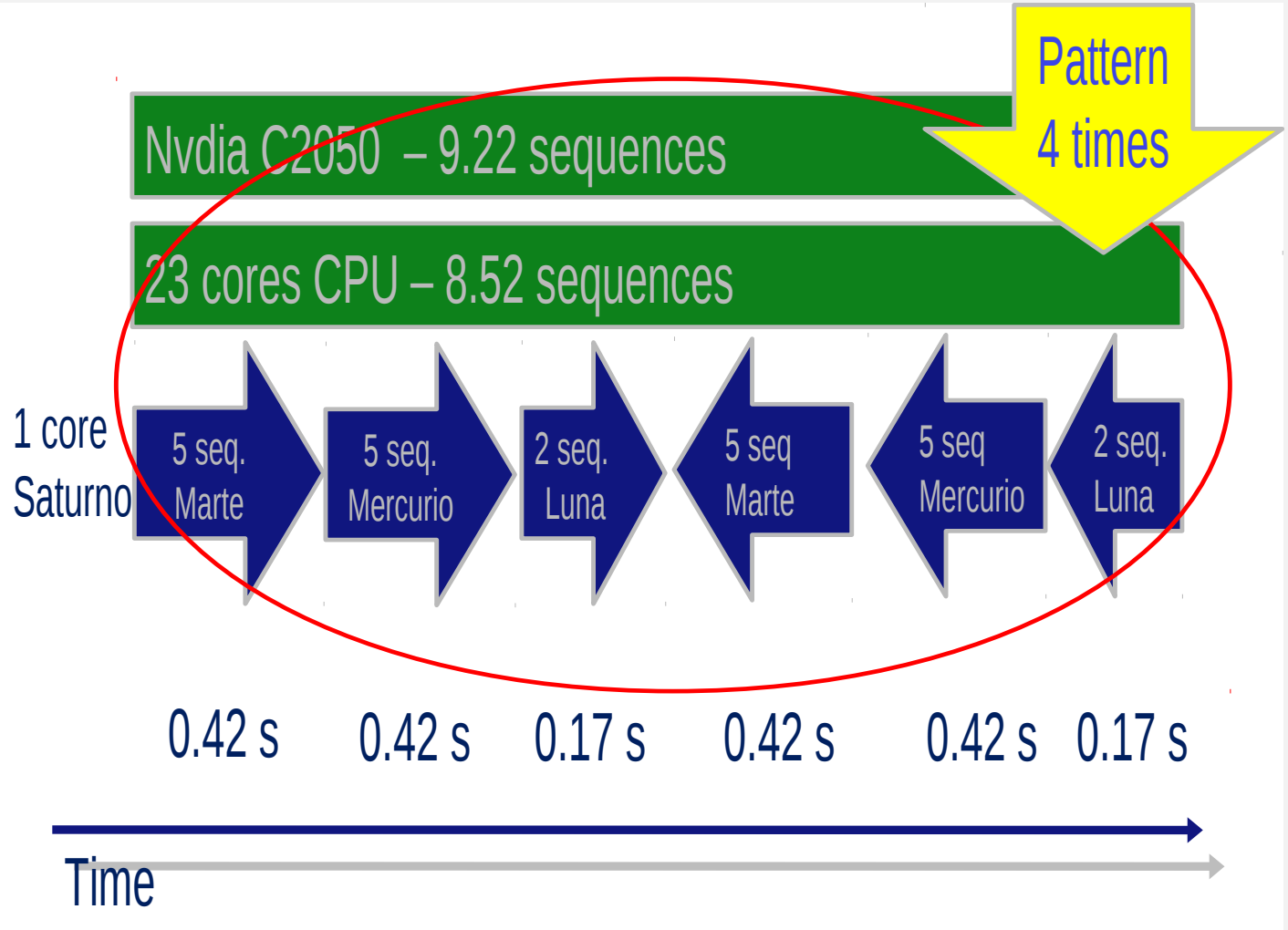
For master node Saturno and slave nodes Luna, Marte and Mercurio

- The proportions between Marte or Mercurio and Luna is 2 to 1
- The minimum common multiples of (1, 2, 2) and (2, 5, 5) are 2 and 10
- The master sends 20 (2×10) sequences to Marte and Mercurio and 10 to Luna



TS for 3 slave nodes

This pattern is repeated four times



Speedups of TS(3) are 1.89, 1.64 and 1.68

Comparison with GPU

Speedups of the 3D-FWT with autotuning with respect to the case in which all the sequences are sent to one or two of the GPUs of a node and the optimal block size is selected.
For a sequence of video of 2 hours with 25 frames per second, split in group of 32 frames.

speedup	512×512	1024×1024	2048×2048
AT vs Nvidia C2050 GPU	3.37	3.17	3.22
AT vs 2 GeForce GTX 590 GPU	1.58	1.60	1.66
AT vs GeForce 9800 GT GPU	8.35	8.53	8.96

An average speedup of $4.49\times$ versus a user who sends all the sequences to the GPU of a node and obtains the optimal block size.

Comparison with execution in a node

Speedups of the 3D-FWT with autotuning with respect to the case in which all the sequences are sent to a node (CPU+GPU) and the optimal block size is selected.
For a sequence of video of 2 hours with 25 frames per second, split in group of 32 frames.

speedup	512×512	1024×1024	2048×2048
AT vs AT on Saturno	2.06	1.81	1.74
AT vs AT on Marte/Mercurio	1.34	1.33	1.43
AT vs AT on Luna	5.57	6.34	6.15

An average speedup of $3.09\times$ versus processing all the sequences in only one node of the cluster and using the autotuning proposed in:
G. Bernabé, J. Cuenca, D. Giménez, Optimization techniques for 3D-FWT on systems with manycore GPUs and multicore CPUs. ICCS 2013

Conclusion

An autotuning engine to run automatically the 3D-FWT kernel on clusters of multicore+GPU:

- Detects the number and type of CPUs and GPUs, the 3D-FWT computer performance and the bandwidth of the interconnection network.
- The Theoretical Searching of the Best Number of Slave Nodes automatically computes the proportions at which the different sequences of video are divided among the nodes in the cluster.
- Searches for the Temporal distribution scheme for n slave nodes working jointly to a master node.
- Chooses the best number of slave nodes.

Average gains of 3.09× and 4.49× versus an optimization engine for a node (CPU+GPU) or a single GPU.

Future works

- The work is part of the development of an image processing library oriented to biomedical applications, allowing users the efficient executions of different routines automatically.
- The proposed autotuning methodology is applicable to other complex compute applications.
- It can be extended to more complex computational systems, multicore+GPU+MIC.

Selection of the number of threads for a sequence of video of 64 frames of 1024×1024 and 2048×2048 on Intel Xeon Phi.

#threads	1024×1024	2048×2048
1	3783.24	21774.10
32	317.34	903.40
64	389.16	1283.29
96	60.83	**35.78**
128	**41.09**	41.04
160	50.96	55.65
192	68.13	62.89
224	70.60	68.85

Optimizing a 3D-FWT code in a cluster of CPUs+GPUs

Gregorio Bernabé Javier Cuenca Domingo Giménez

Universidad de Murcia
Scientific Computing and Parallel Programming Group

XXIX Simposium Nacional de la Unión Científica Internacional de Radio
Valencia, 3-5 septiembre 2014