Utilisation de cartes graphiques pour accélérer le Calcul Haute Performance

Cédric Augonnet
Samuel Thibault
Raymond Namyst

LaBRI, University of Bordeaux, INRIA Bordeaux

LaBRI - 4 Janvier 2011

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

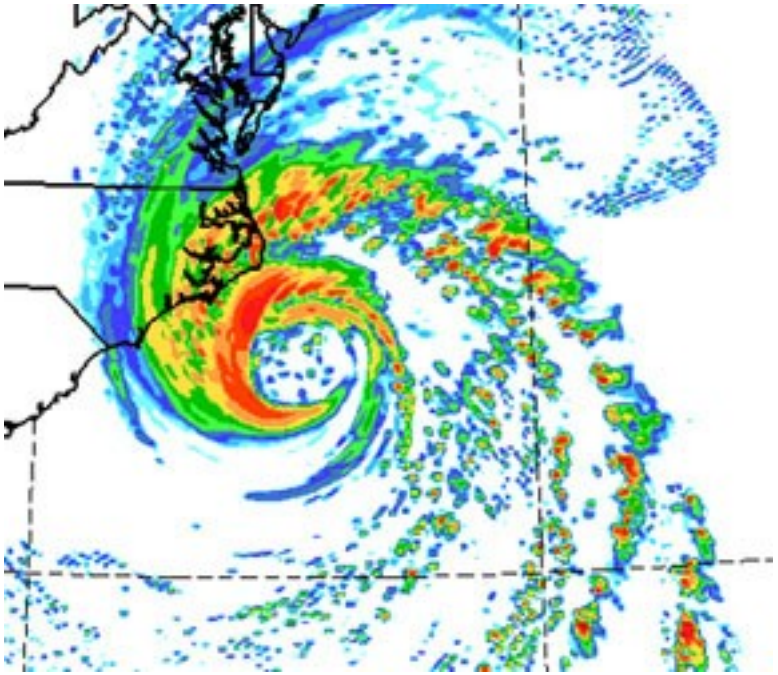
INRIA

centre de recherche
BORDEAUX - SUD-OUEST

Introduction

High Performance Computing

- Simulation
  - Weather prediction
  - Seismology
  - Finance ...
- Growing needs
  - Faster results
  - Better accuracy
  - Larger problem



Introduction

The RUNTIME team

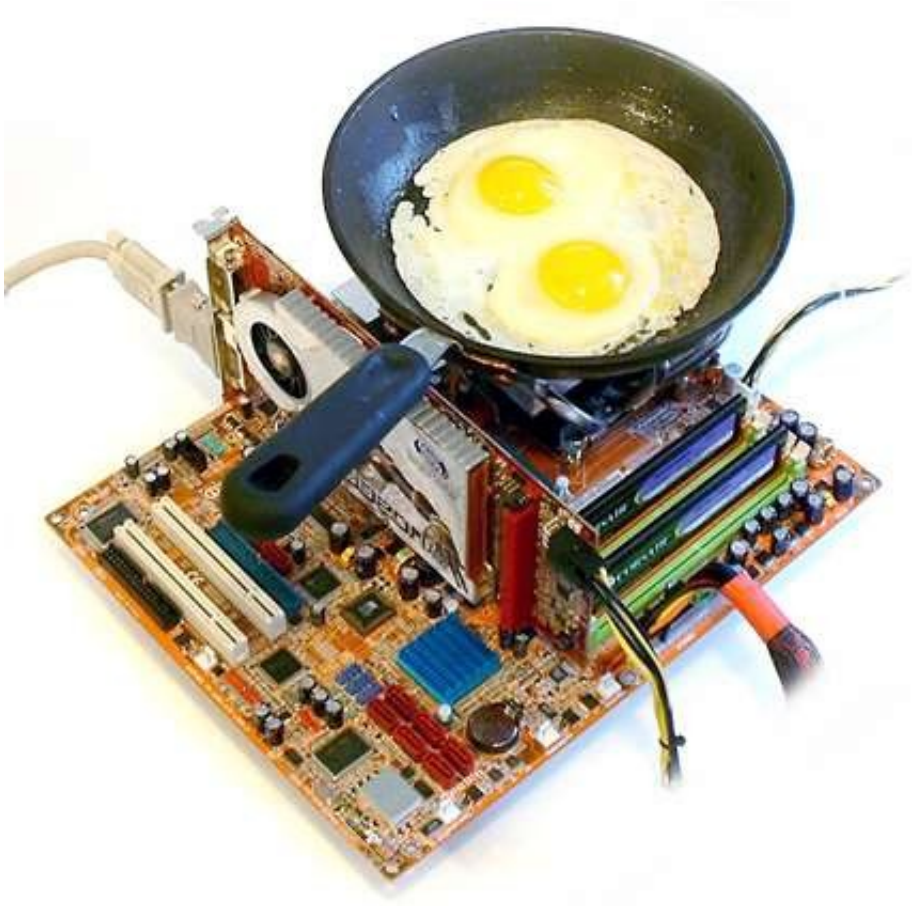
- High Performance Runtime Systems for Parallel Architectures
  - "*Runtime Systems perform dynamically what cannot be not statically*"

- Main research directions
  - Exploiting shared memory machines
    - Thread scheduling over hierarchical multicore architectures
    - Task scheduling over accelerator-based machines
  - Communication over high speed networks
    - Multicore-aware communication engines
    - Multithreaded MPI implementations
  - Integration of multithreading and communication
    - Runtime support for hybrid programming

Introduction

Multicore architectures

- The Free lunch is over !
  - Reached physical limits
    - Clock speed, pipeline, cache …
    - Energy is a concern
  - Era of parallelism
    - SMP, Multicore, Clusters



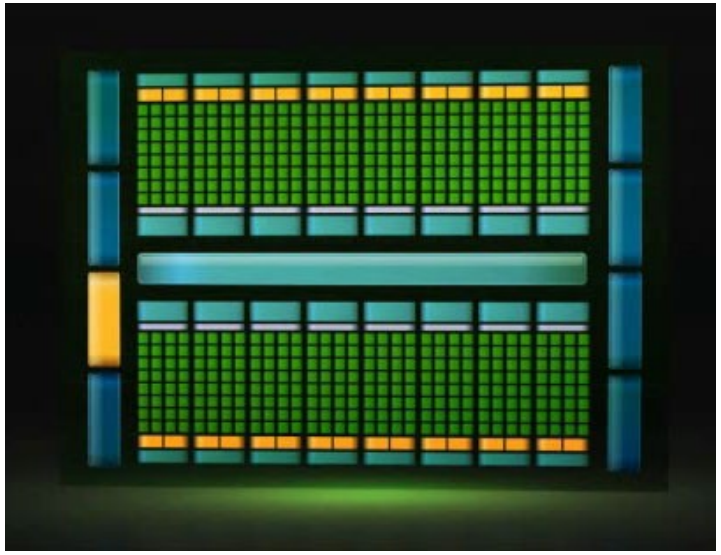
- Multicore architectures
  - Hierarchical chips
  - Getting really complex

- Jaguar Machine (Oak Ridge, USA)
  - 224162 cores (AMD Opteron)
  - 6.9 MW

Introduction

Toward Multi-GPU clusters

- GPUs are the new kids on the block
  - Very powerful data-parallel accelerators
  - Specific Instruction Set
  - No hardware memory consistency



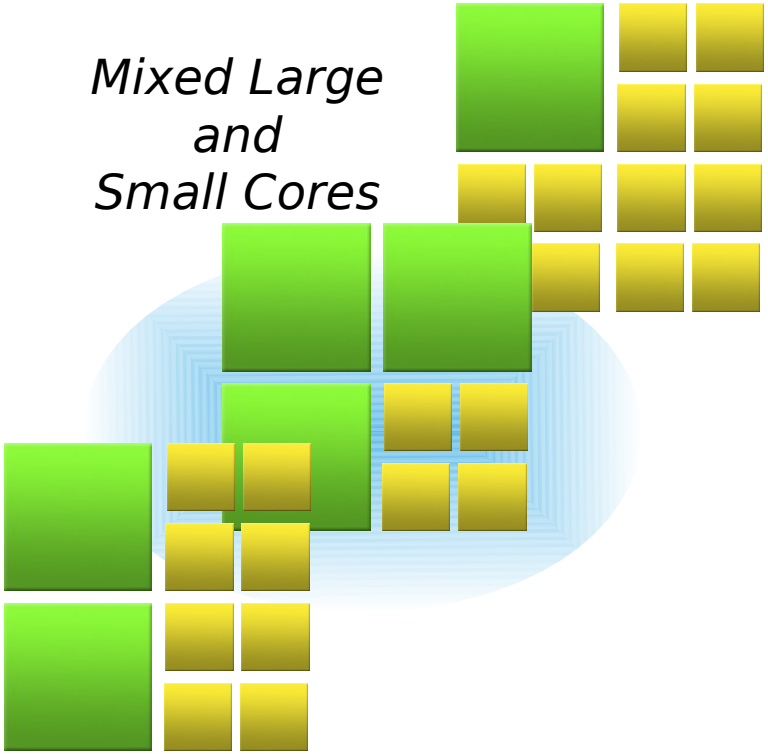
- Other chips already feature specialized hardware
  - IBM Cell/BE (PS3)
    - 1 PPU + 8 SPUs
  - Intel Larrabee / MIC
    - 48-core with SIMD units

- Are we happy with that ?
  - No, but it's probably unavoidable!

Introduction

Heterogeneity is also a solid trend

- One interpreation of « Amdalh's law »
  - We will always need powerful, general purpose cores to speed up sequential parts of our applications
- Future processors will be a mix of general purpose and specialized cores
  - [Intel]
- Study today's accelerators to be ready for tomorrow's processors



Performance Portability

Run fast everywhere

- Portability
  - Codes run anywhere
- Performance portability
  - Codes run as fast as possible anywhere

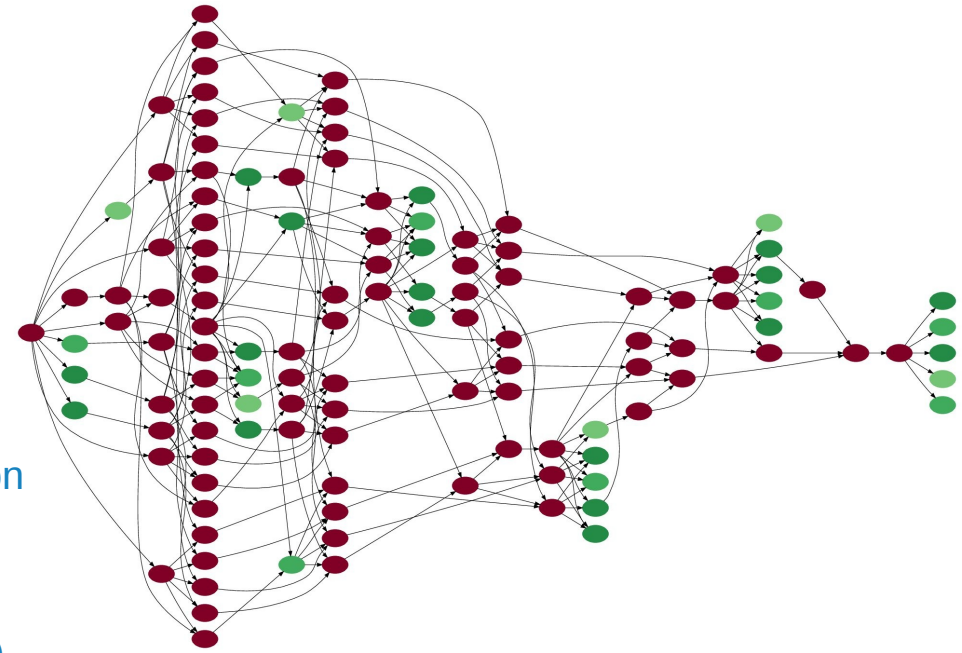
- Architectures are evolving FAST
- HPC has a lot of inertia
  - Millions of line of code
  - Codes may run for decades
  - Impossible to rewrite codes every 5 years

- Which programming models to use?

Performance Portability

Task parallelism

- Which Programming model?
  - There is no perfect model !
  - Should be Architecture independent
  - Need to express parallelism
- Task parallelism
  - Task
    - Describes a piece of computation
    - Accessed data
  - Graph of tasks
    - Directed Acyclic Graphs (DAGs)
  - Task scheduling
    - Static or Dynamic
    - Use Runtime systems

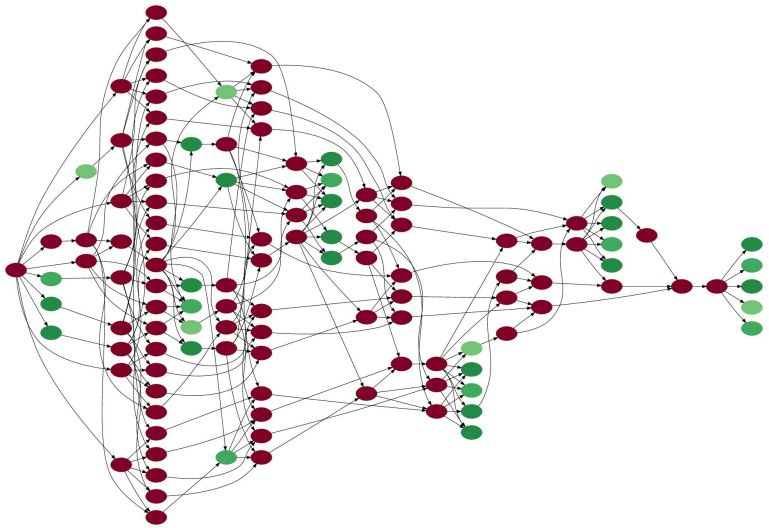


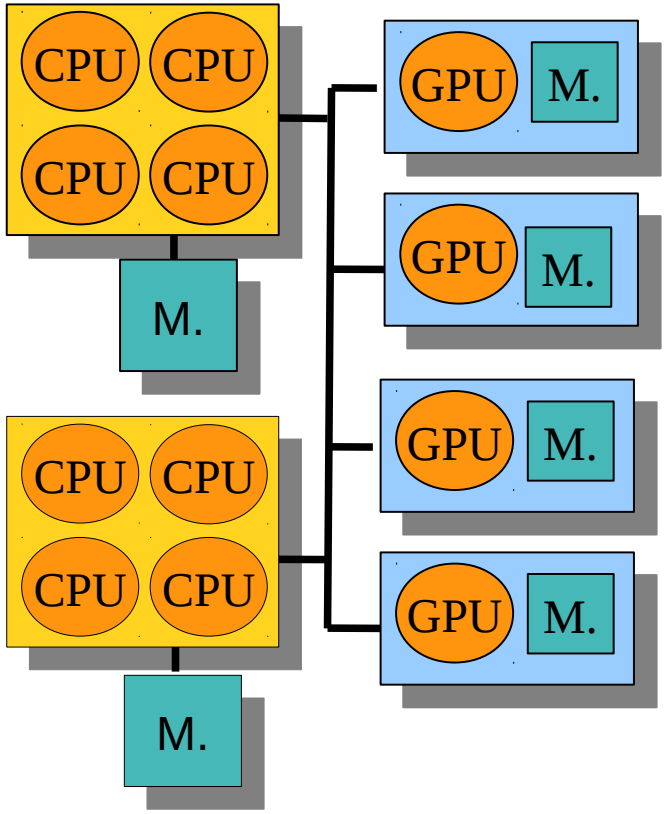
Task Scheduling

Major challenges

- Map task graphs onto the different processing units

- Main issues
  - Load Balancing
  - Data locality



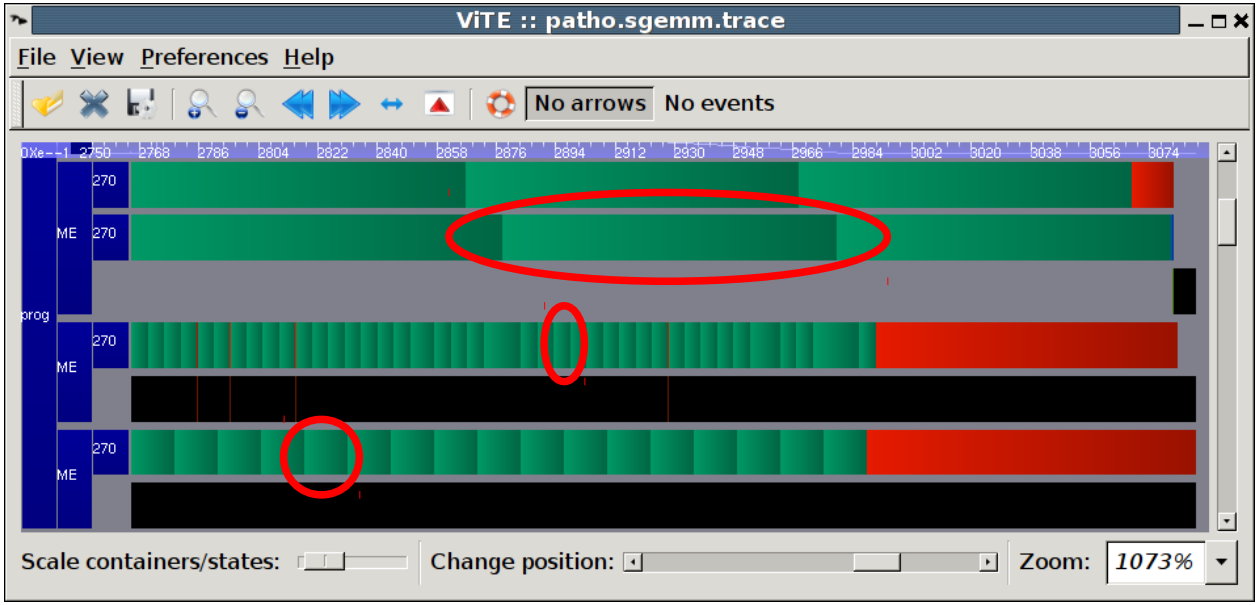


Task scheduling

Load balancing

Things can go (really) wrong even on trivial problems !

- Greedy scheduler
  - Processing units get tasks when they are idle
- Matrix multiplication
  - All tasks are identical
  - Heterogeneous performance



2 Xeon cores

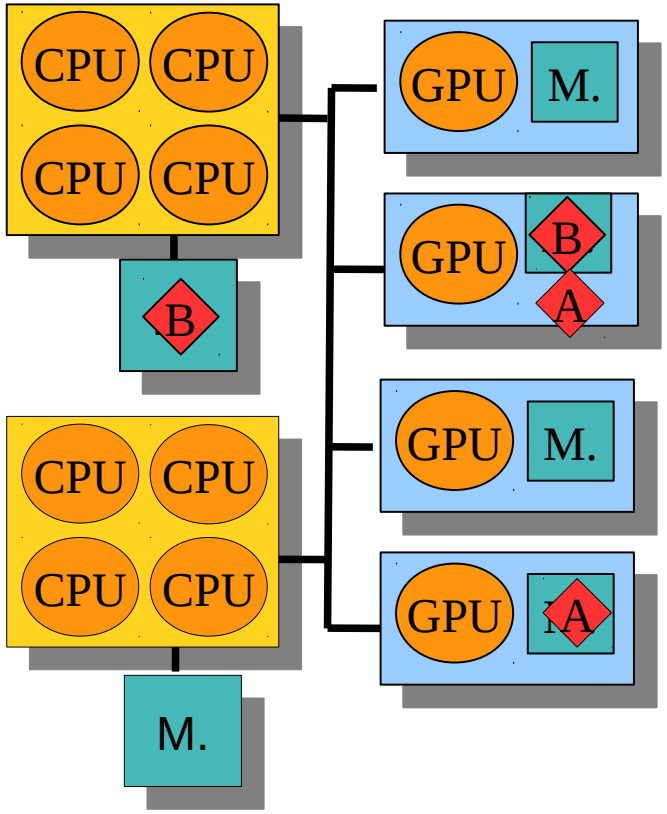
Quadro FX5800

Quadro FX4600

Task scheduling

Data locality

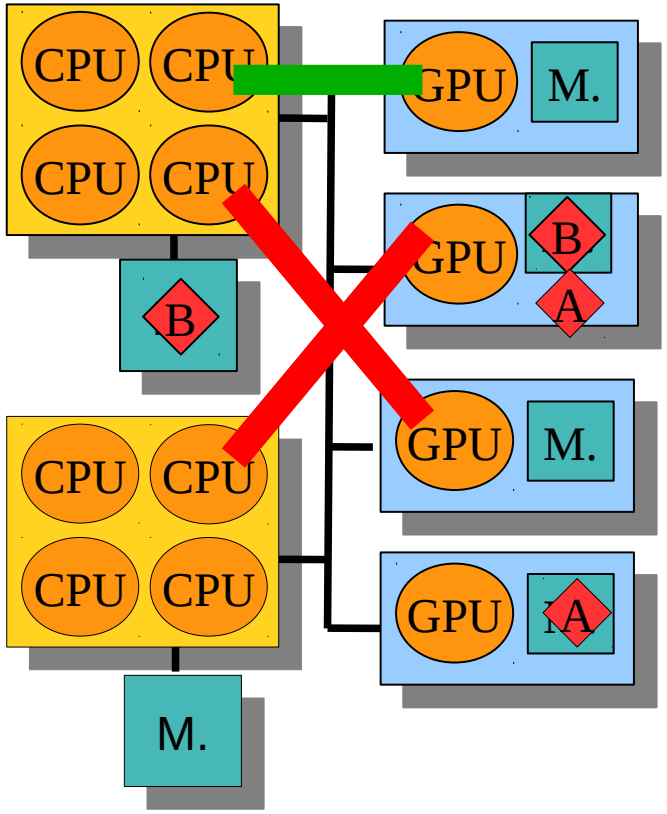
- Explicit Data transfers
- PCI Bus = Bottleneck
  - ~150 GB/s within a GPU
  - ~10 GB/s on PCI
- Non Uniform Memory Accesses
  - Significant impact on scalability
- Load Balancing vs. Data locality
  - Cannot avoid all data transfers
  - Minimize them



Task scheduling

Data locality

- Explicit Data transfers
- PCI Bus = Bottleneck
  - ~150 GB/s within a GPU
  - ~10 GB/s on PCI
- Non Uniform Memory Accesses
  - Significant impact on scalability
- Load Balancing vs. Data locality
  - Cannot avoid all data transfers
  - Minimize them



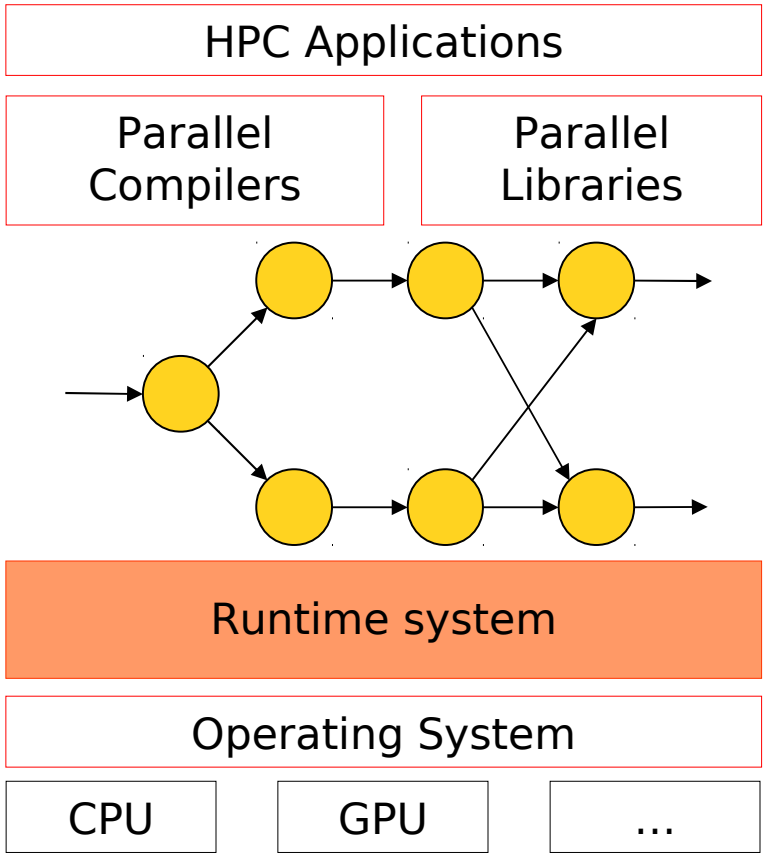
The StarPU runtime system

The need for runtime systems

- “do dynamically what can’t be done statically anymore”

- Library that provides
  - Task scheduling
  - Memory management

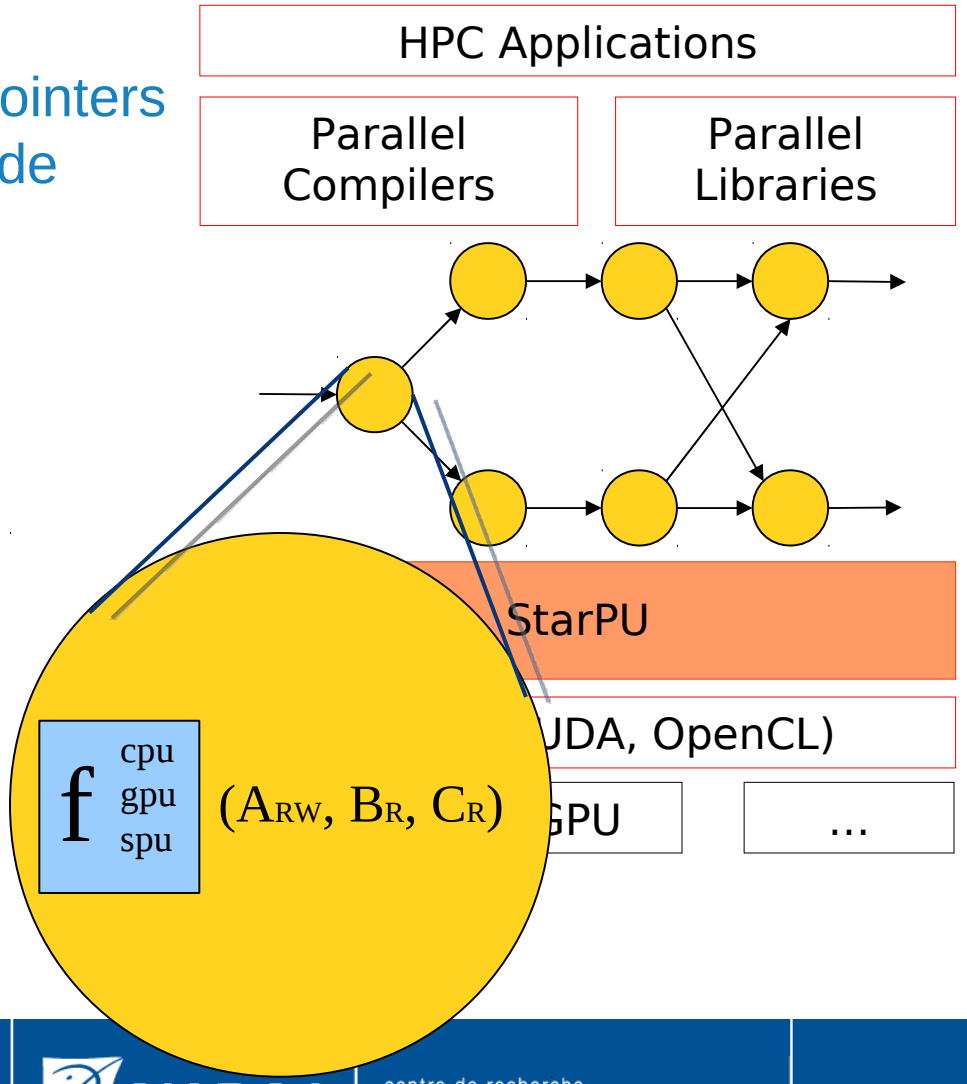
- Compilers and libraries generate (graphs of) parallel tasks
  - Additional information is welcome!



The StarPU runtime system

Task scheduling

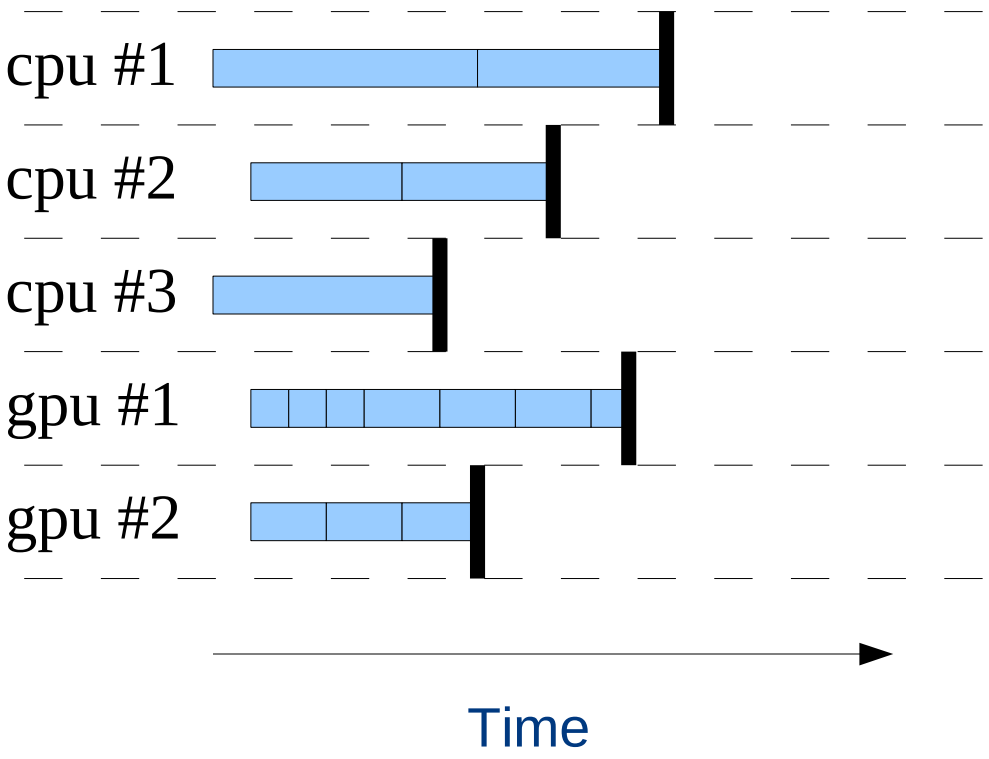
- Who generates the code ?
  - StarPU Task = ~function pointers
  - StarPU don't generates code
- Manual programming
- Libraries era
  - PLASMA + MAGMA
  - FFTW + CUFFT...
- Rely on compilers
  - PGI accelerators
  - CAPS HMPP...



The StarPU runtime system

Load Balancing

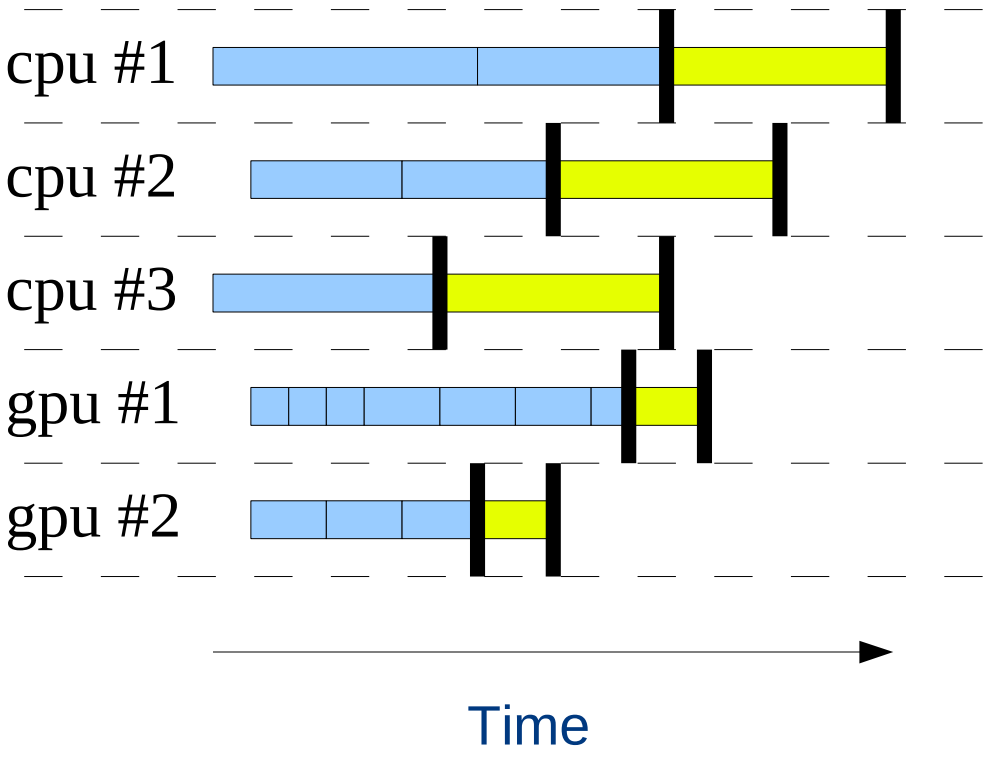
- Task completion time estimation
  - History-based
  - User-defined cost function
  - Parametric cost model
- Can be extended to improve data locality
  - Predict data transfer time



The StarPU runtime system

Load Balancing

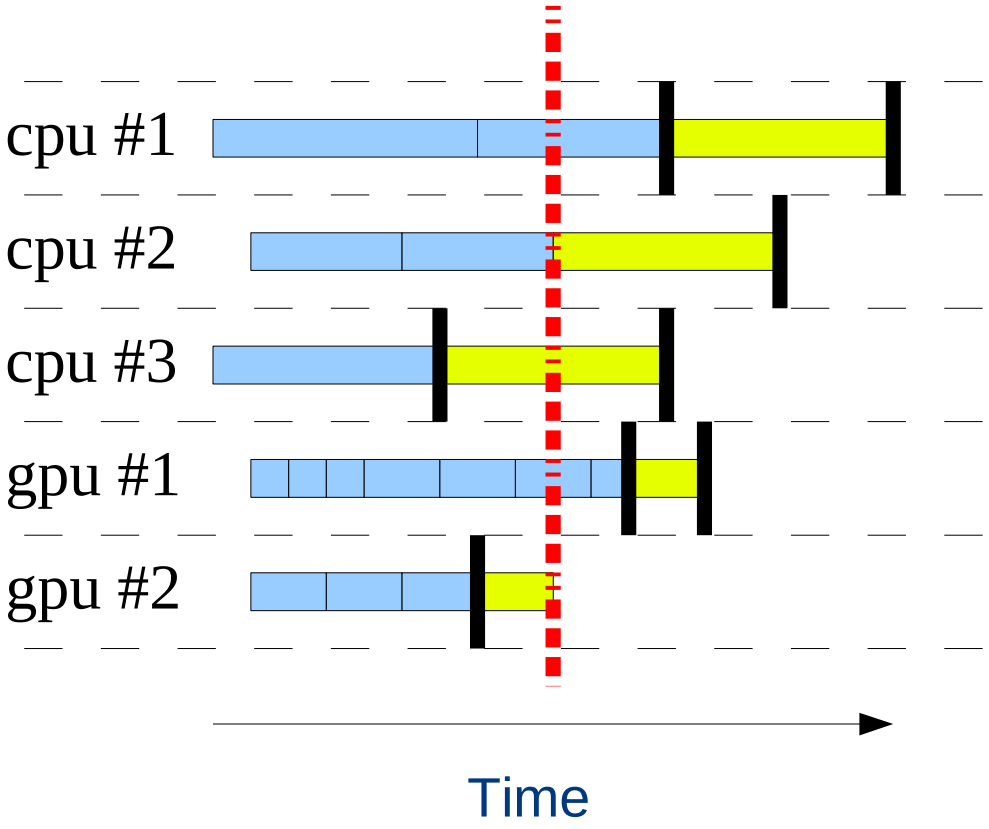
- Task completion time estimation
  - History-based
  - User-defined cost function
  - Parametric cost model
- Can be extended to improve data locality
  - Predict data transfer time



The StarPU runtime system

Load Balancing

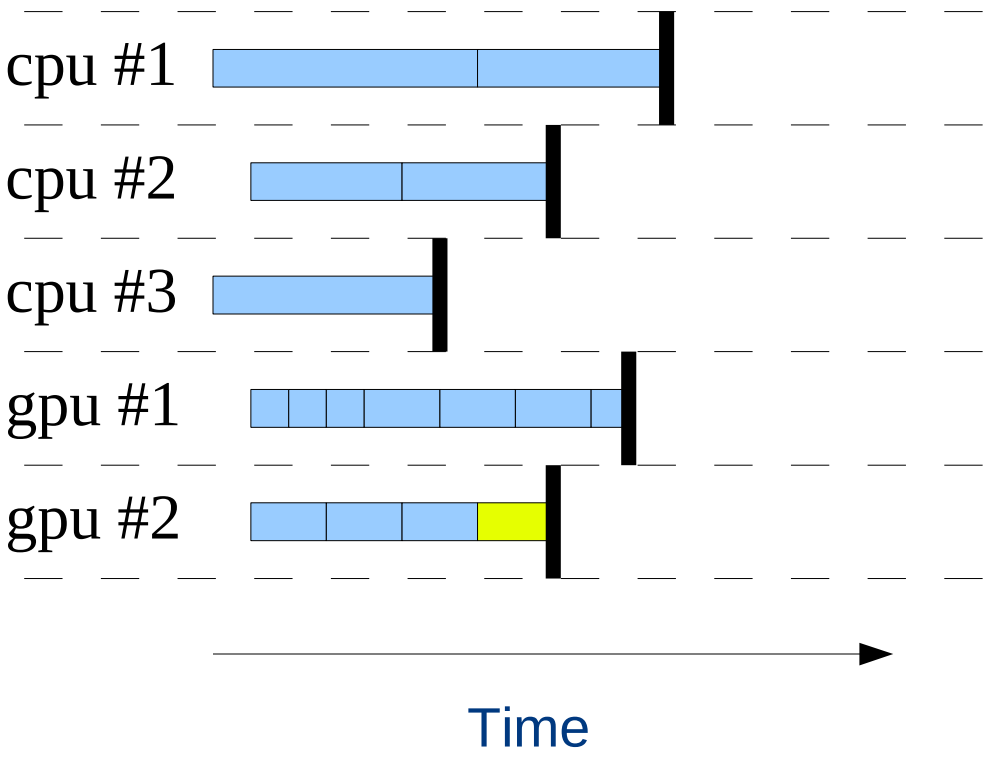
- Task completion time estimation
  - History-based
  - User-defined cost function
  - Parametric cost model
- Can be extended to improve data locality
  - Predict data transfer time



The StarPU runtime system

Load Balancing

- Task completion time estimation
  - History-based
  - User-defined cost function
  - Parametric cost model
- Can be extended to improve data locality
  - Predict data transfer time

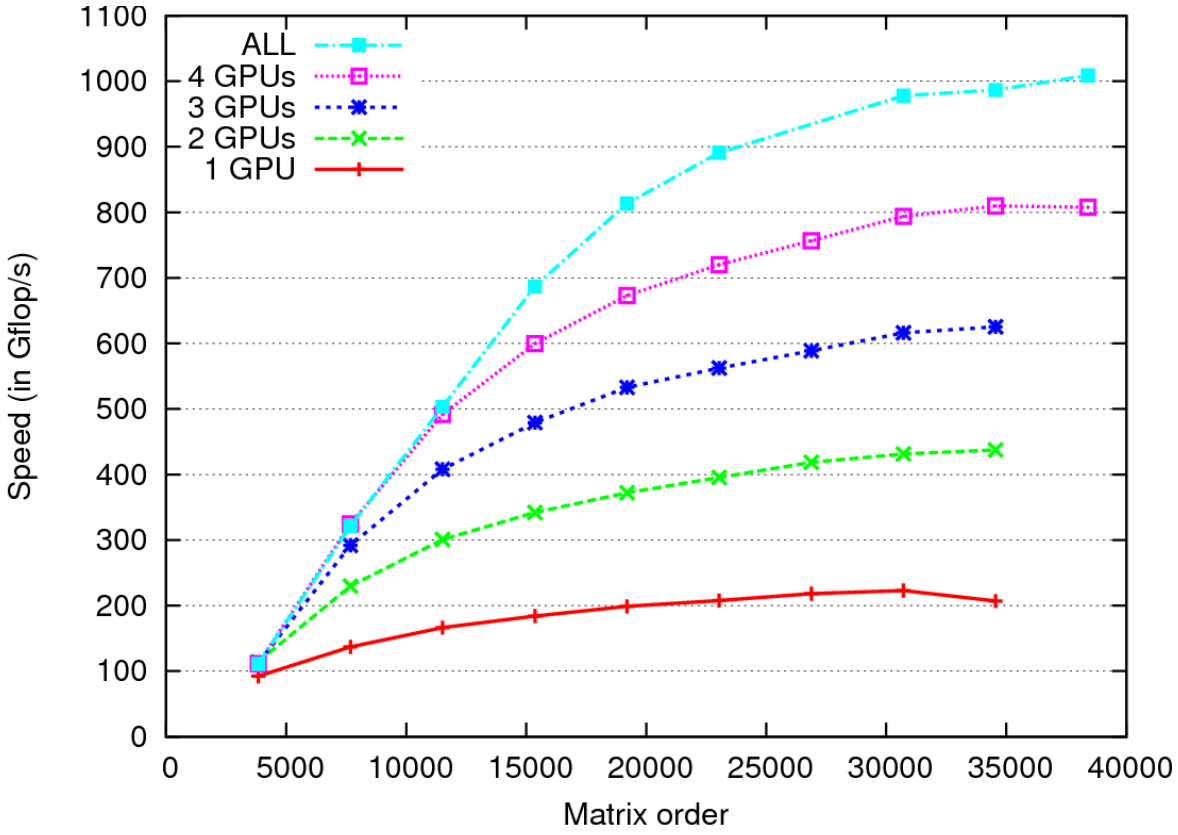


Mixing PLASMA and MAGMA with StarPU

- Dense Linear Algebra kernels
  - Building blocks for many HPC applications
- State of the art libraries
  - PLASMA (Multicore CPUs)
    - Dynamically scheduled with QUARK
  - MAGMA (Multiple GPUs)
    - Hand-coded data transfers, Static task mapping
- General SPLAGMA design
  - Use PLASMA algorithm, bypass their scheduler
  - PLASMA kernels on CPUs, MAGMA kernels on GPUs
- Programmability
  - QR : ~2 days of works, quick algorithmic prototyping

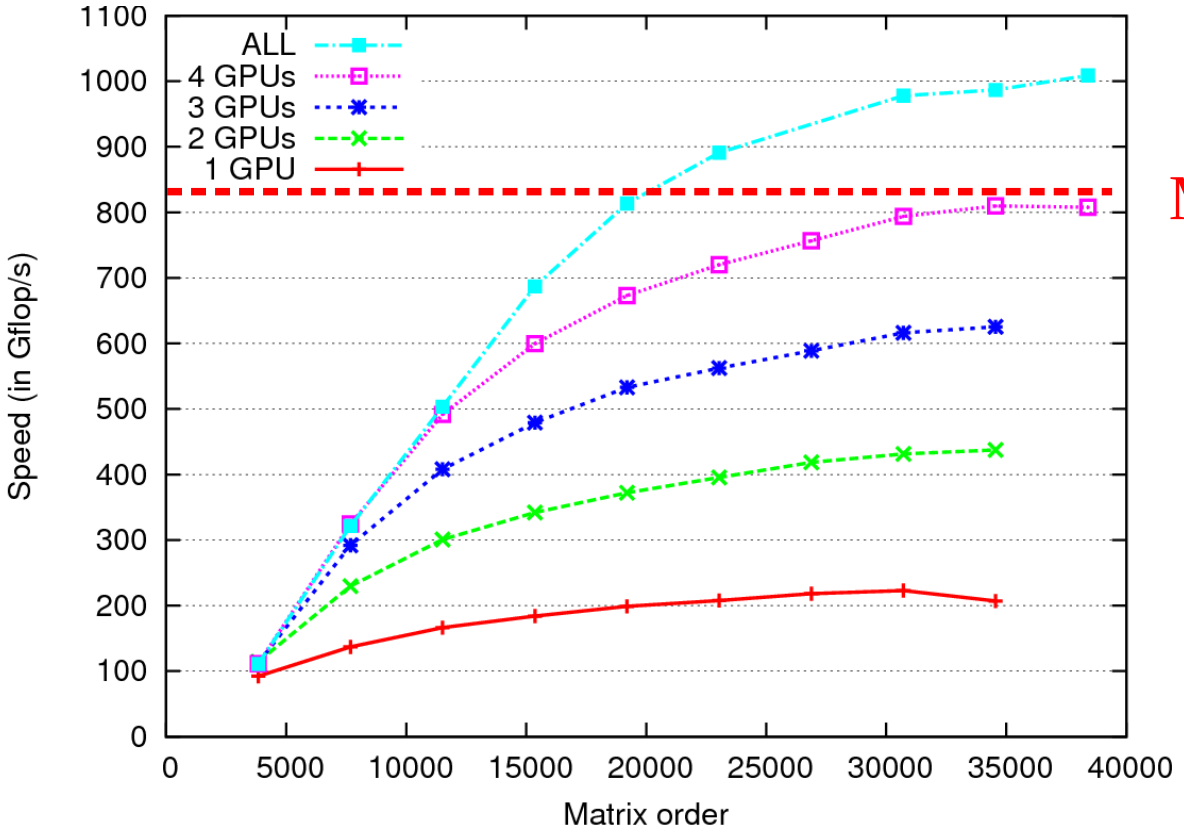
Mixing PLASMA and MAGMA with StarPU

- QR decomposition
  - Mordor8 (UTK) : 16 CPUs (AMD) + 4 GPUs (C1060)



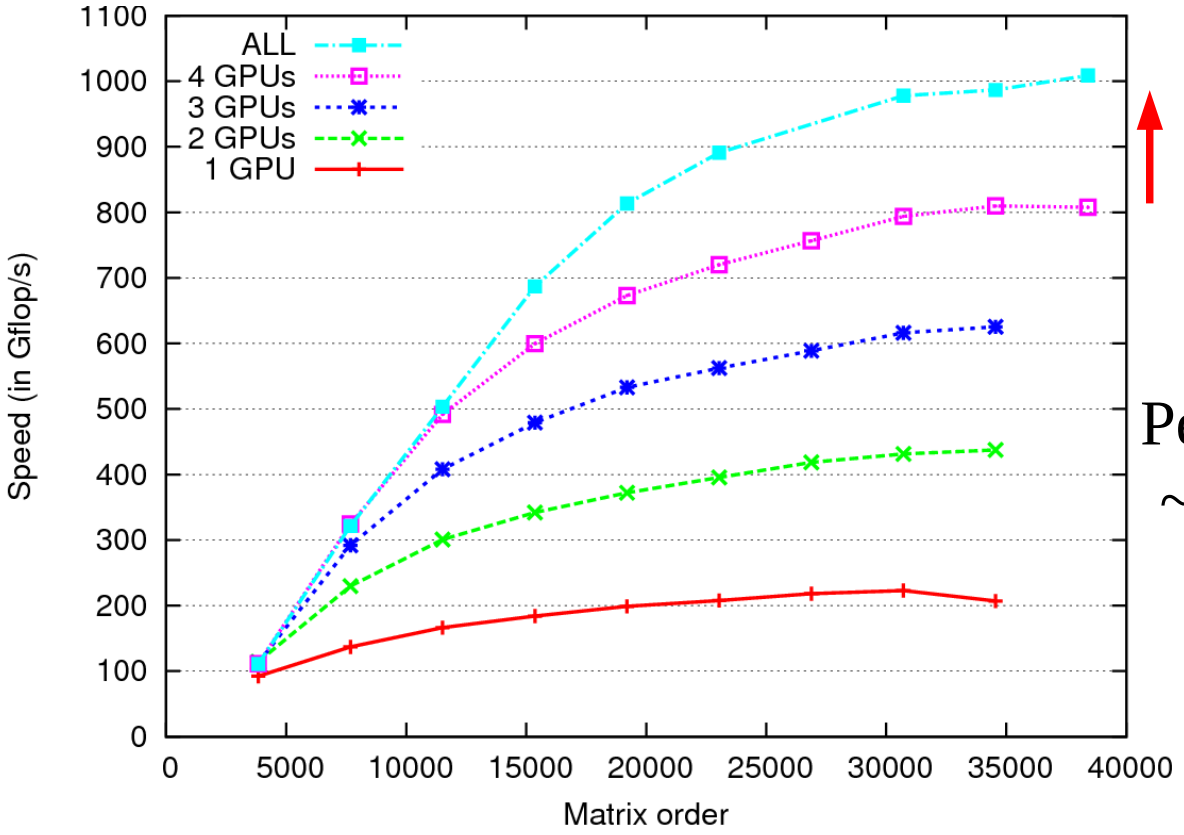
Mixing PLASMA and MAGMA with StarPU

- QR decomposition
  - Mordor8 (UTK) : 16 CPUs (AMD) + 4 GPUs (C1060)



Mixing PLASMA and MAGMA with StarPU

- QR decomposition
  - Mordor8 (UTK) : 16 CPUs (AMD) + 4 GPUs (C1060)



+12 CPUs
~200GFlops

Peak : 12 cores
~150 GFlops

Mixing PLASMA and MAGMA with StarPU

- « Super-Linear » efficiency in QR?
  - Kernel efficiency
    - sgeqrt
      - CPU: 9 Gflops GPU: 30 Gflops (Speedup : ~3)
    - stsqrt
      - CPU: 12Gflops GPU: 37 Gflops (Speedup: ~3)
    - somqr
      - CPU: 8.5 Gflops GPU: 227 Gflops (Speedup: ~27)
    - Sssmqr
      - CPU: 10Gflops GPU: 285Gflops (Speedup: ~28)
  - Task distribution observed on StarPU
    - sgeqrt: 20% of tasks on GPUs
    - Sssmqr: 92.5% of tasks on GPUs
  - Taking advantage of heterogeneity !
    - Only do what you are good for
    - Don't do what you are not good for

Conclusion

Summary

- StarPU
  - Freely available under LGPL
  - Open to external contributors
- A lot of opportunities
  - Enough work for the next decade!
  - Tons of Interactions
    - CEA, CAPS, Play ALL, …
    - PEPPHER European project
    - UTK, Urbana Champaign
    - FP3C project with Japan
    - INTEL, NVIDIA, ...