

StarPU: A Runtime system for accelerator-based multicore machines

Cédric Augonnet Samuel Thibault
Raymond Namyst Pierre-André Wacrenier

INRIA Bordeaux, LaBRI, Université de Bordeaux

19th October 2009 – COST – Lisbon (ES)

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE



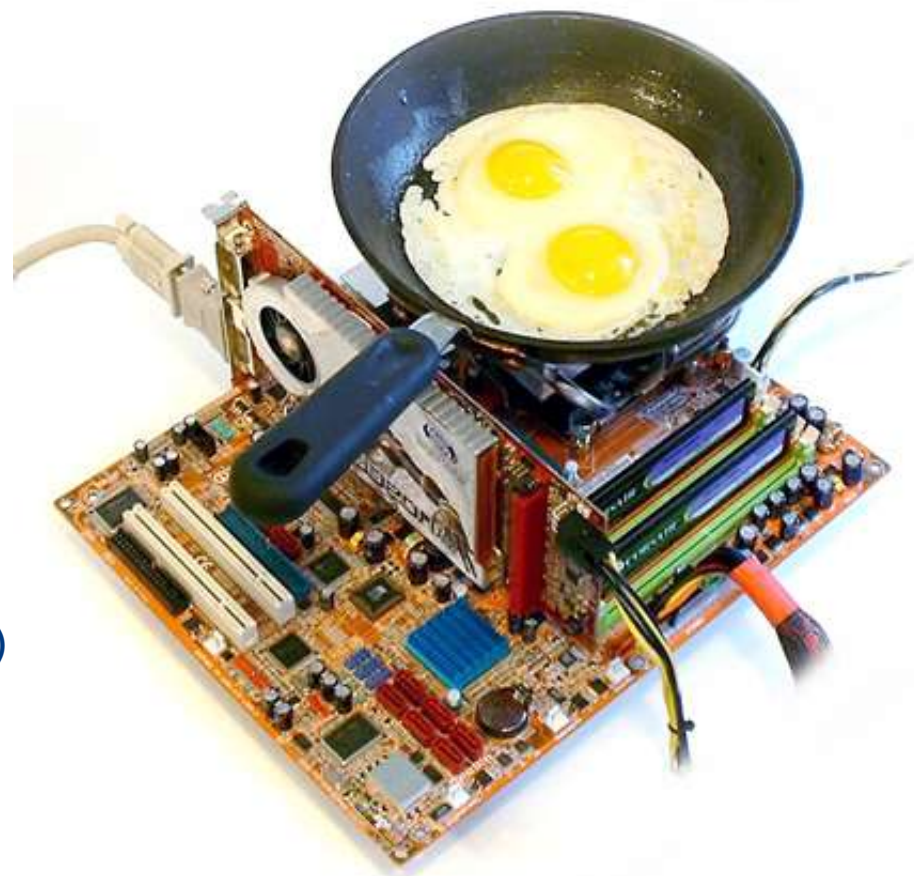
centre de recherche
BORDEAUX - SUD-OUEST

- Introduction
- StarPU
- A portable scheduling engine
- Evaluation
- Conclusion



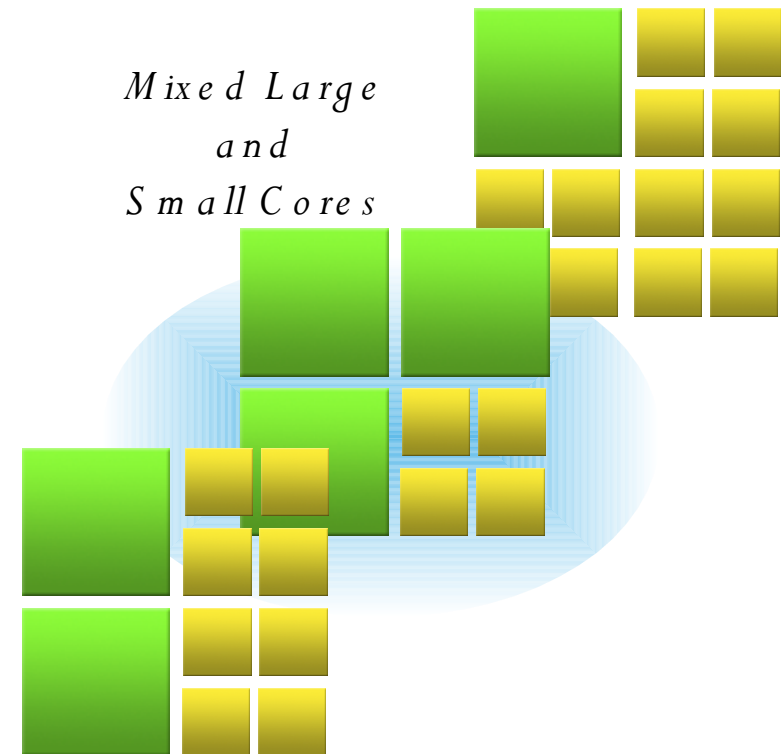
Toward heterogeneous multi-core architectures

- Multicore is here
 - Hierarchical architectures
 - Manycore is coming
 - Power is a major concern
- Architecture specialization
 - Now
 - Accelerators (GPGPUs, FPGAs)
 - Coprocessors (Cell's SPUs)
 - In the (near?) Future
 - Many simple cores
 - A few full-featured cores



Toward heterogeneous multi-core architectures

- Multicore is here
 - Hierarchical architectures
 - Manycore is coming
 - Power is a major concern
- Architecture specialization
 - Now
 - Accelerators (GPGPUs, FPGAs)
 - Coprocessors (Cell's SPUs)
 - In the (near?) Future
 - Many simple cores
 - A few full-featured cores

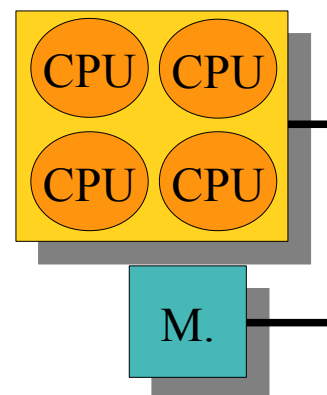
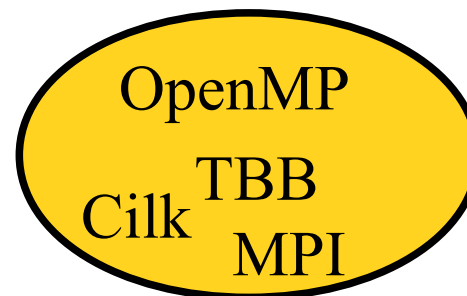


Introduction

How to program these architectures?

- Multicore programming
 - pthreads, OpenMP, TBB, ...

Multicore

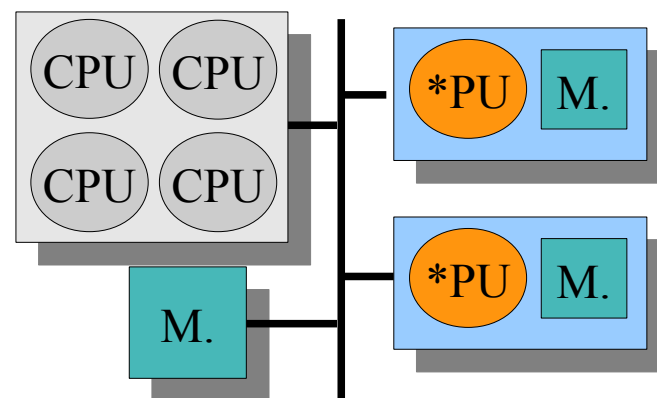


Introduction

How to program these architectures?

- Multicore programming
 - pthreads, OpenMP, TBB, ...
- Accelerator programming
 - Still no consensus (OpenCL?)
 - Pure offloading model

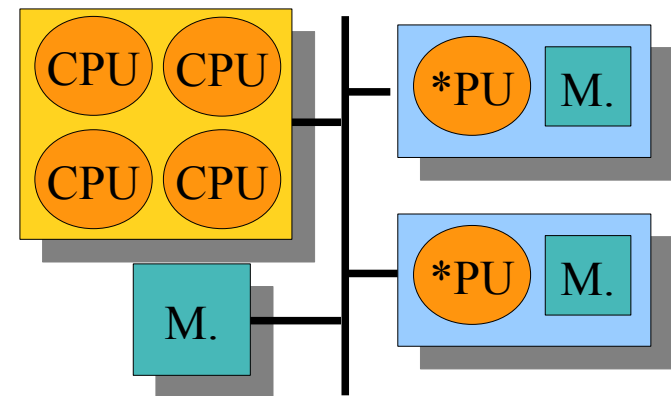
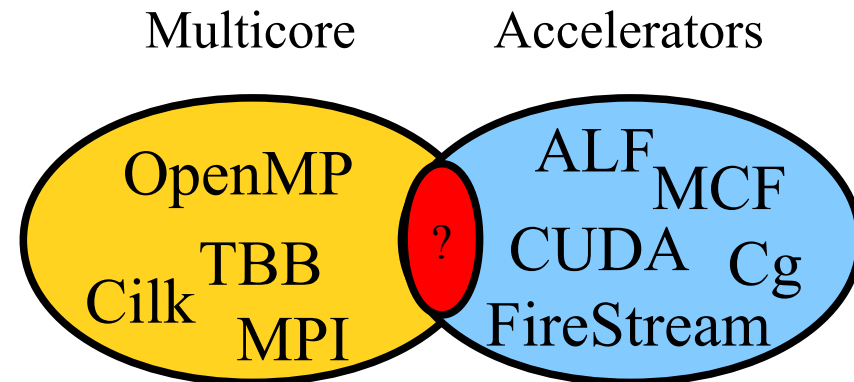
Accelerators



Introduction

How to program these architectures?

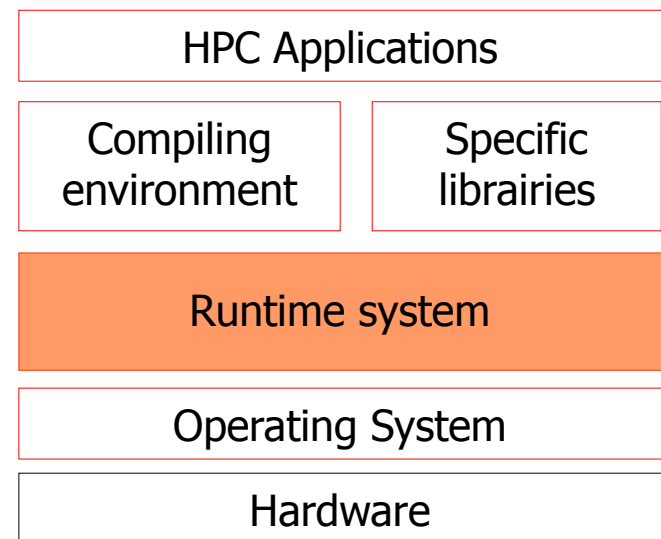
- Multicore programming
 - pthreads, OpenMP, TBB, ...
- Accelerator programming
 - Still no consensus (OpenCL?)
 - Pure offloading model
- Hybrid models?
 - **Take advantage of all resources** 😊
 - Complex interactions ☹️



Introduction

Challenging issues at all stages

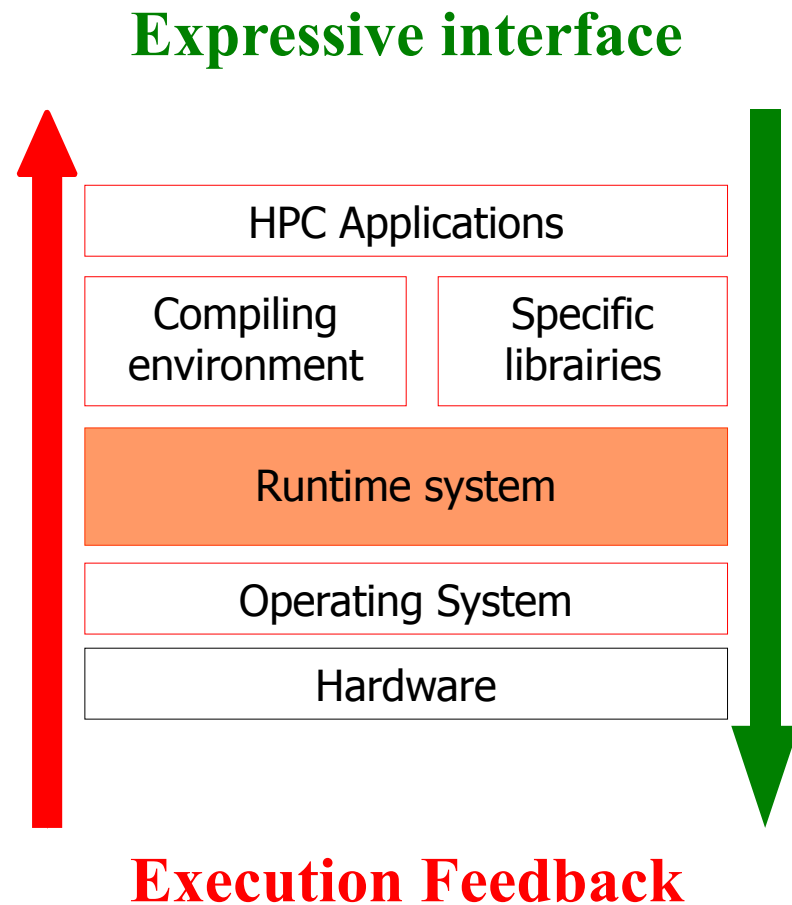
- Applications
 - Programming paradigm
 - BLAS kernels, FFT, ...
- Compilers
 - Languages
 - Code generation/optimization
- **Runtime systems**
 - **Resources management**
 - **Task scheduling**
- Architecture
 - Memory interconnect



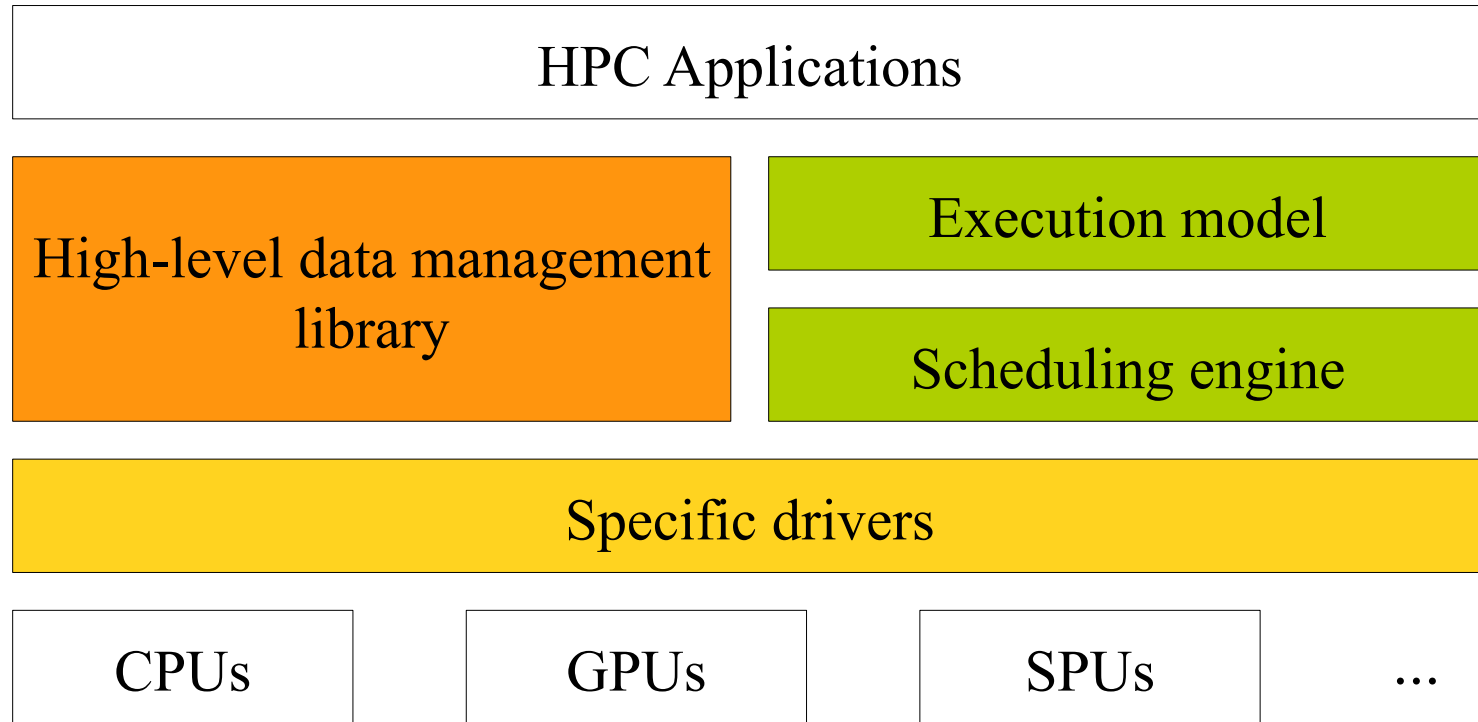
Introduction

Challenging issues at all stages

- Applications
 - Programming paradigm
 - BLAS kernels, FFT, ...
- Compilers
 - Languages
 - Code generation/optimization
- Runtime systems
 - Resources management
 - Task scheduling
- Architecture
 - Memory interconnect



The StarPU runtime system¹⁰

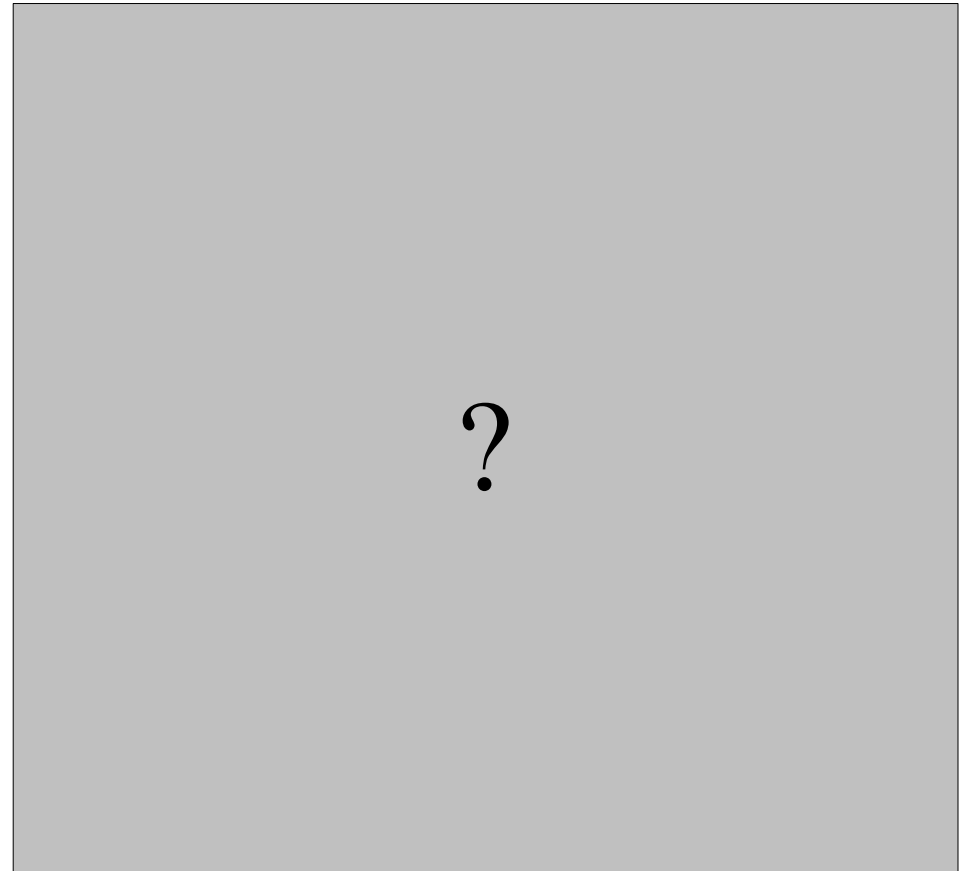
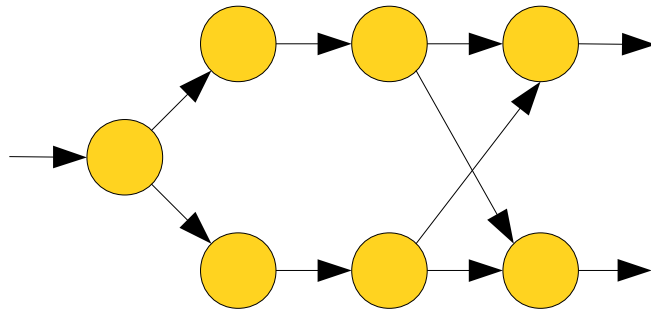


Mastering CPUs, GPUs, SPUs ... ***PUs**



The StarPU runtime system

Motivations



- How to map this application ?

- ✗ Vendor specific programming interfaces

- ✗ Many possible architectures

- A unified execution model

- ✓ Programmability

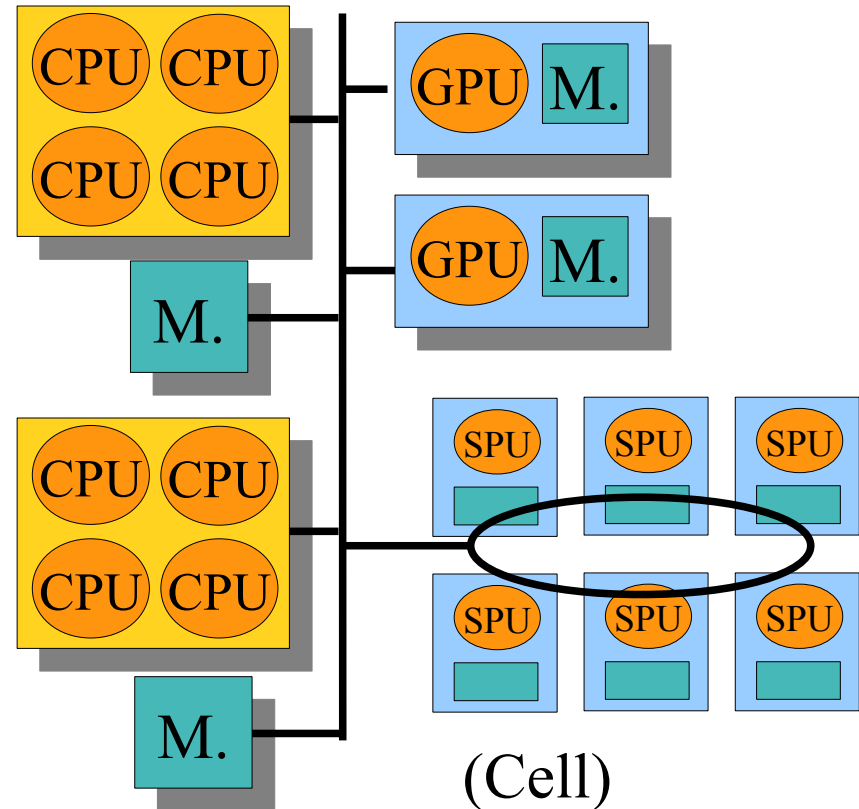
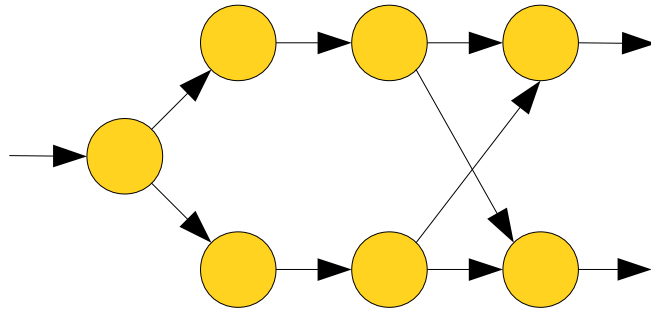
- ✓ Portability of performance

Unknown execution Platform



The StarPU runtime system

Motivations



- How to map this application ?

- ✗ Vendor specific programming interfaces

- ✗ Many possible architectures

- A unified execution model

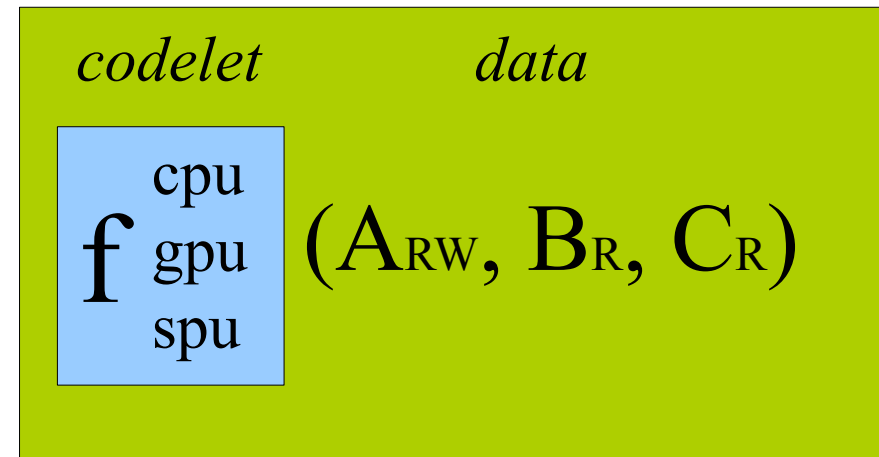
- ✓ Programmability

- ✓ Portability of performance

The StarPU runtime system

The Task structure

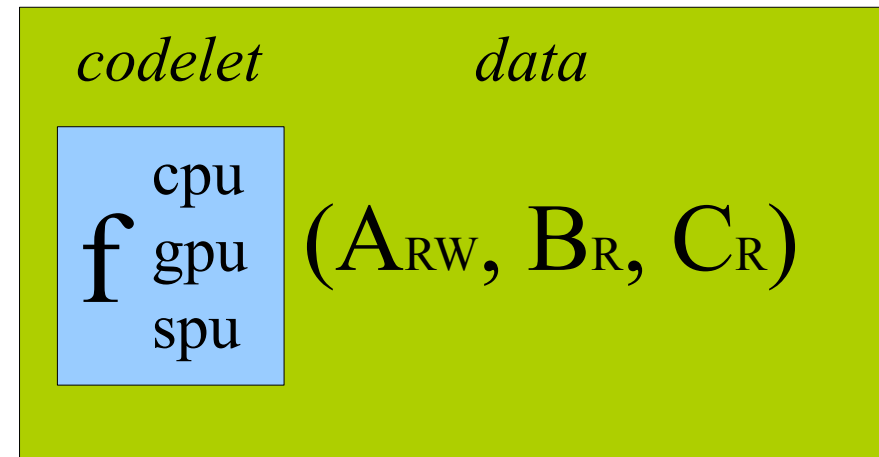
- A generic task structure
 - Codelet
 - Per-arch implementation
 - Explicit all accessed data
 - High level DSM
 - Access mode
- Asynchronous
 - Callback function
- Optional parameters
 - Ease programming
 - Task deps.
 - Hints for the scheduler
 - Priorities, cost models, ...



The StarPU runtime system

The Task structure

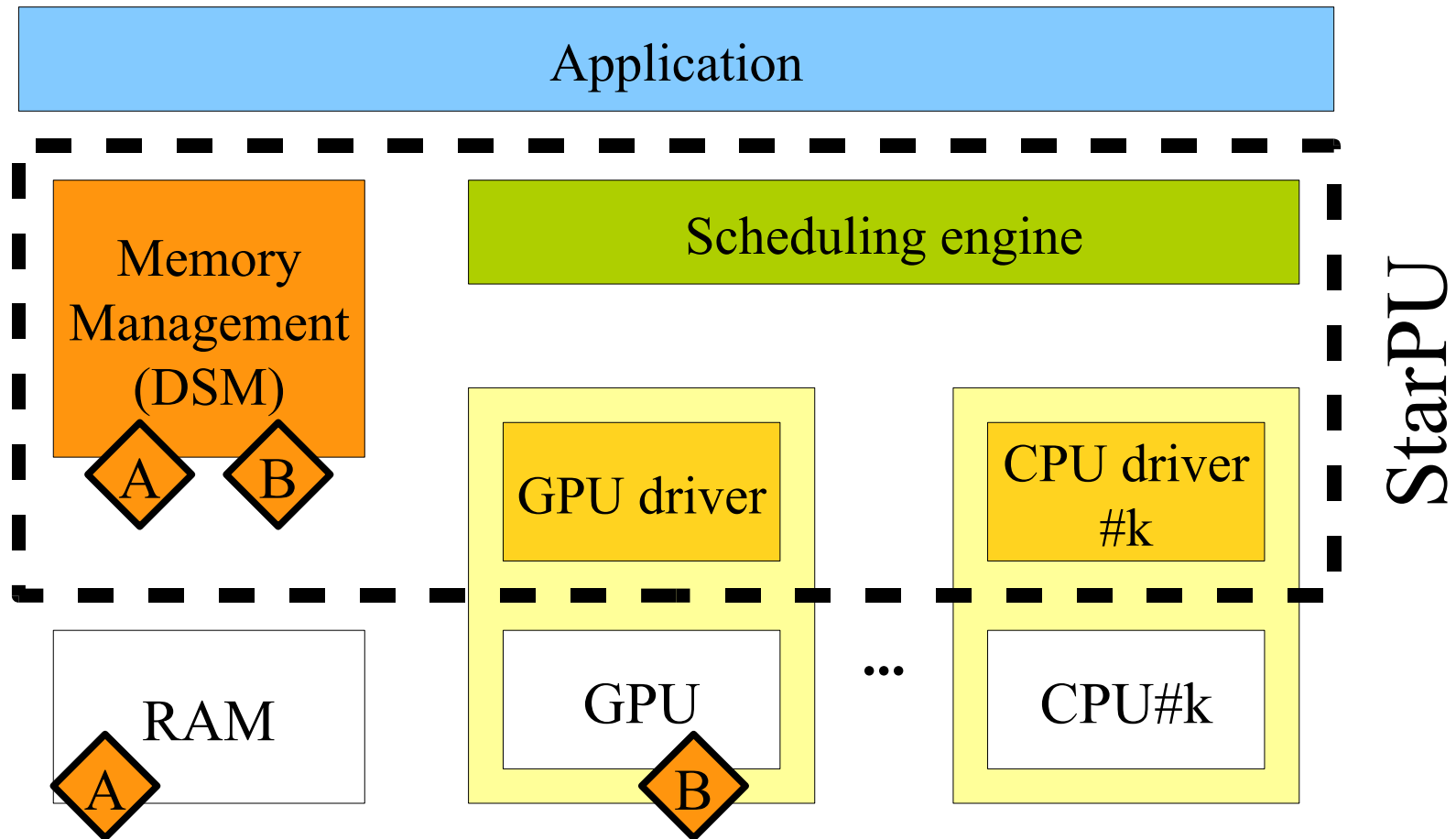
- A generic task structure
 - Codelet
 - Per-arch implementation
 - Explicit all accessed data
 - High level DSM
 - Access mode
- Asynchronous
 - Callback function
- Optional parameters
 - Ease programming
 - Task deps.
 - Hints for the scheduler
 - Priorities, cost models, ...
 - Sub-data management



- Who generates the codelets ?
 - NOT StarPU !
 - Compiling environnements
 - CUDA / CUBLAS
 - CellSs (BSC)
 - HMPP (CAPS)

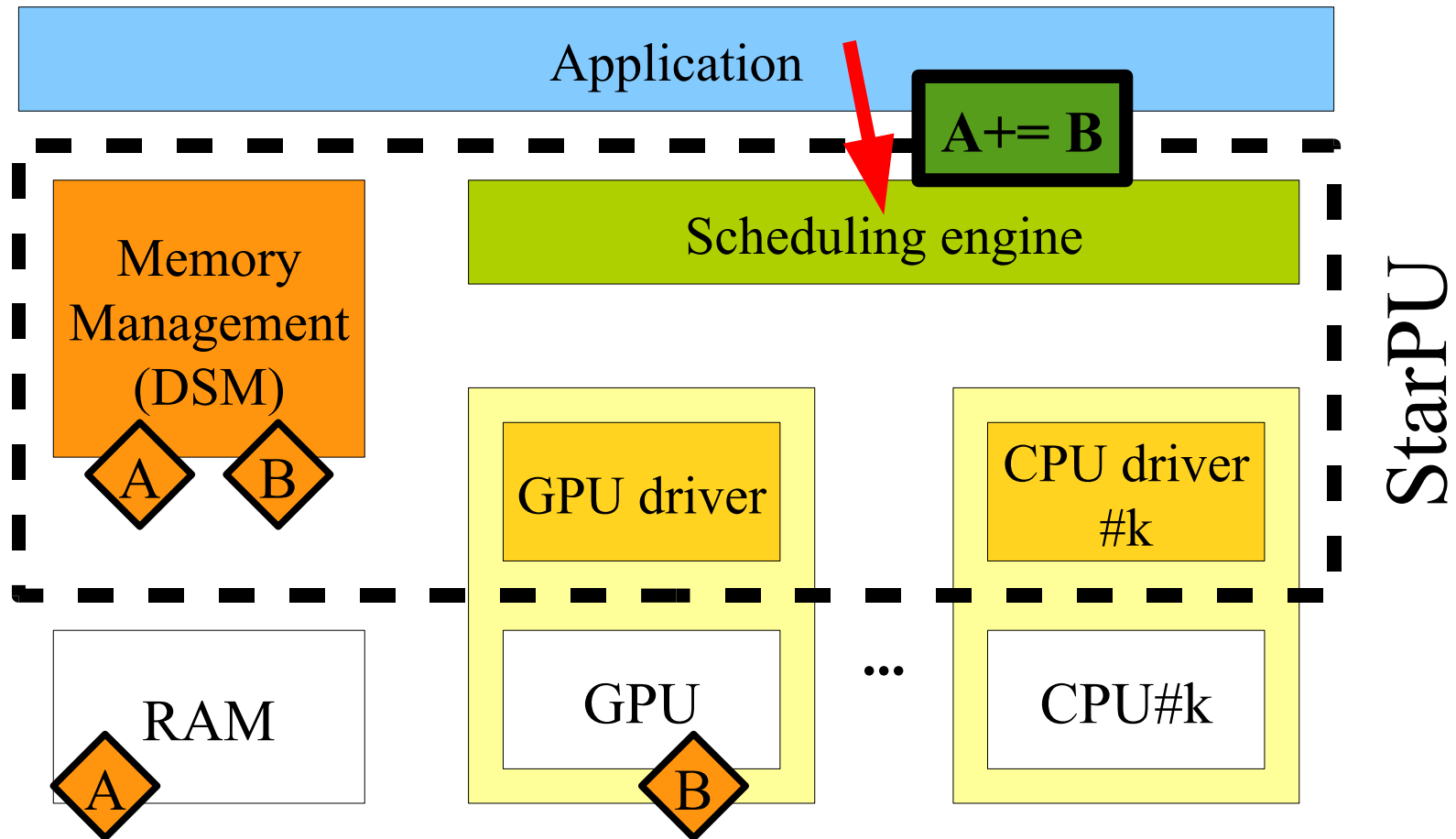
The StarPU runtime system

Execution model



The StarPU runtime system

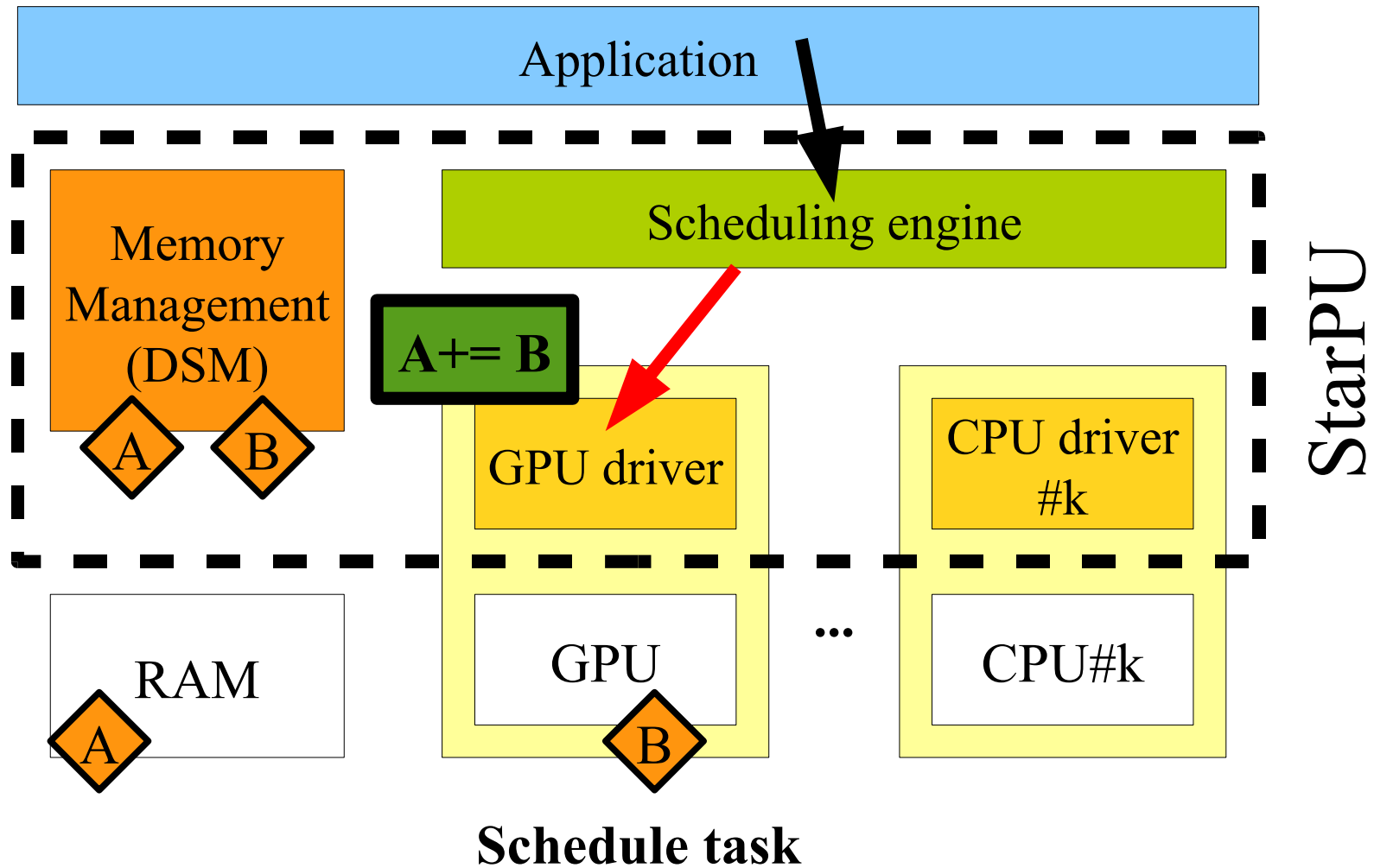
Execution model



Submit task « $A += B$ »

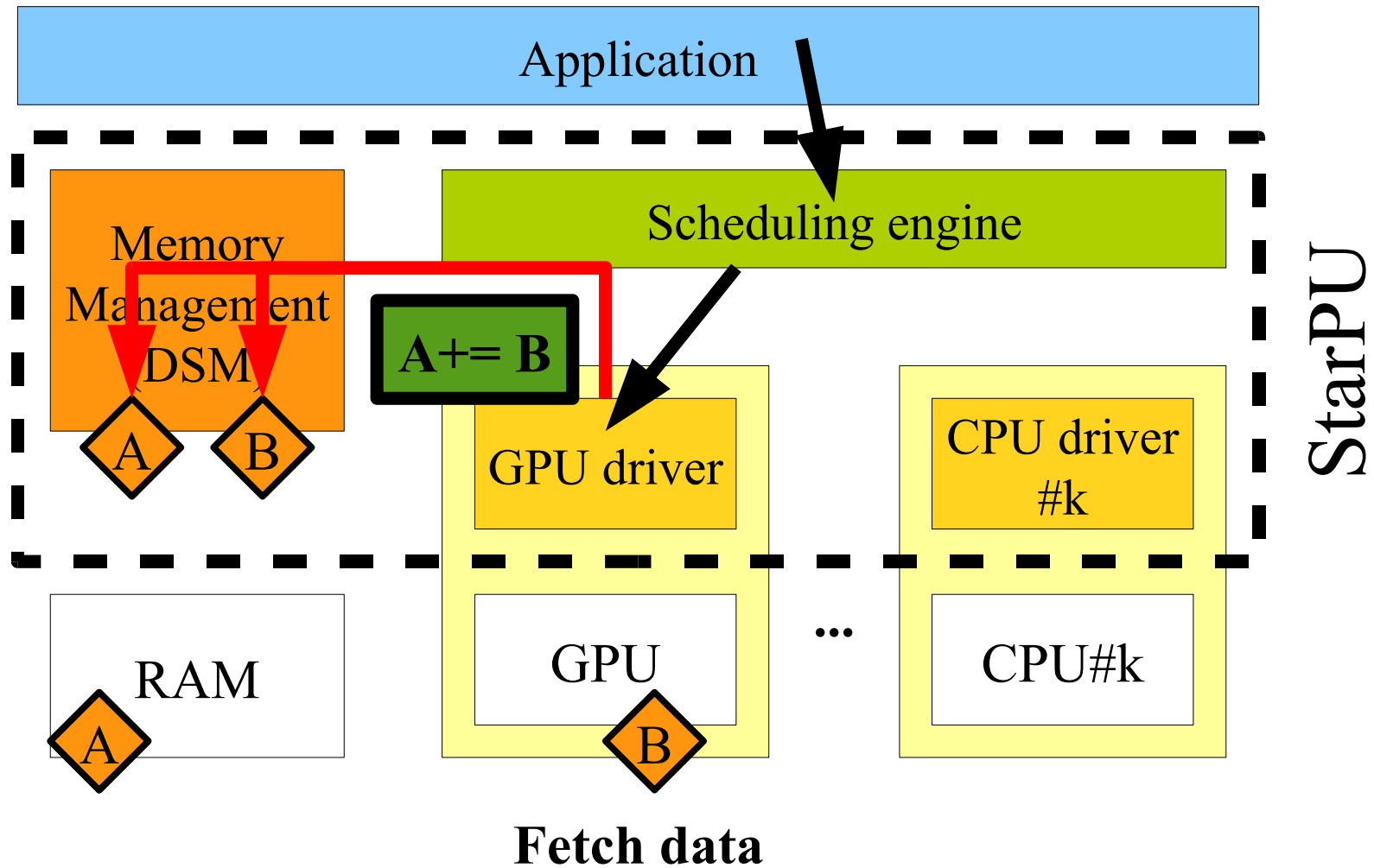
The StarPU runtime system

Execution model



The StarPU runtime system

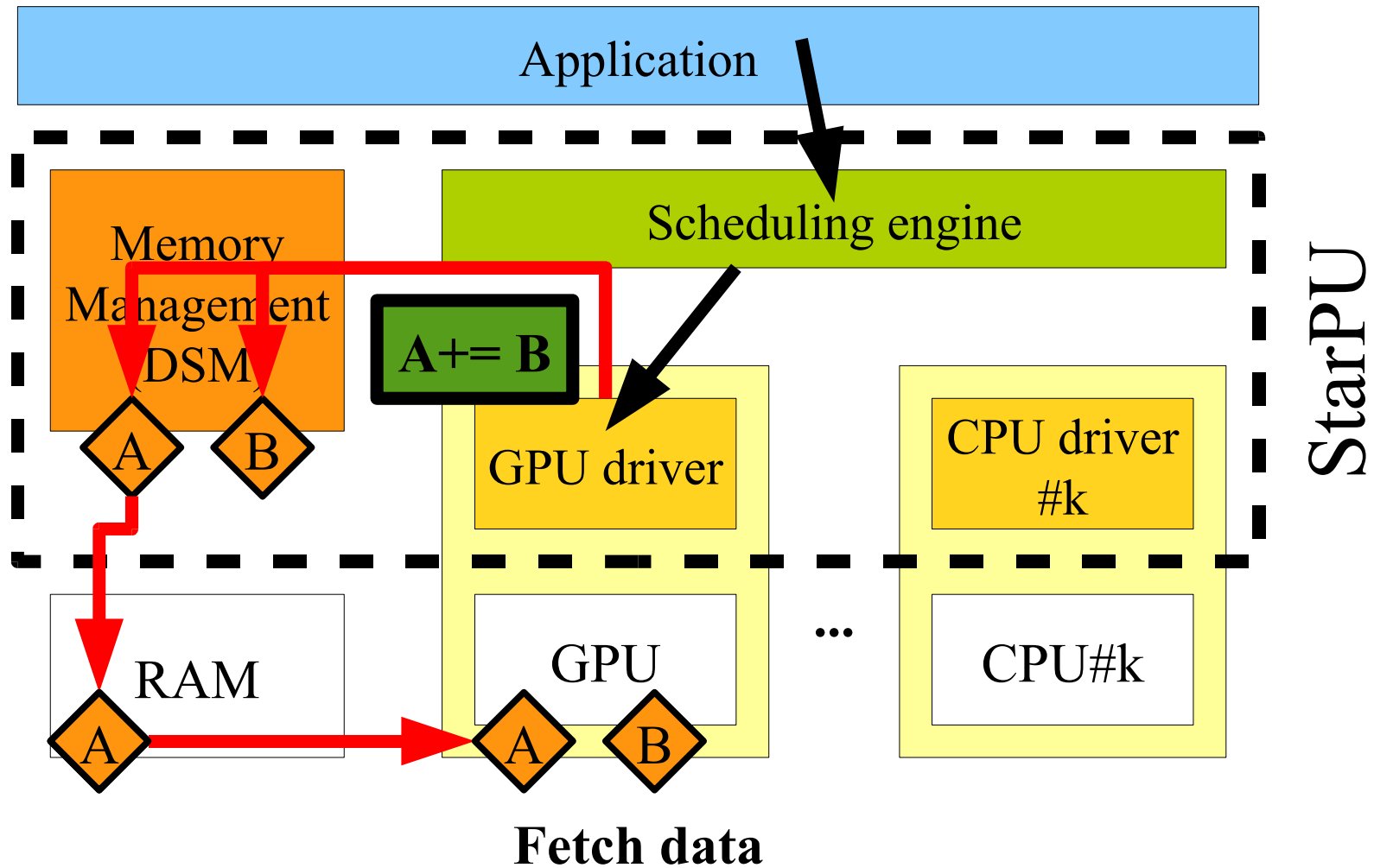
Execution model



StarPU

The StarPU runtime system

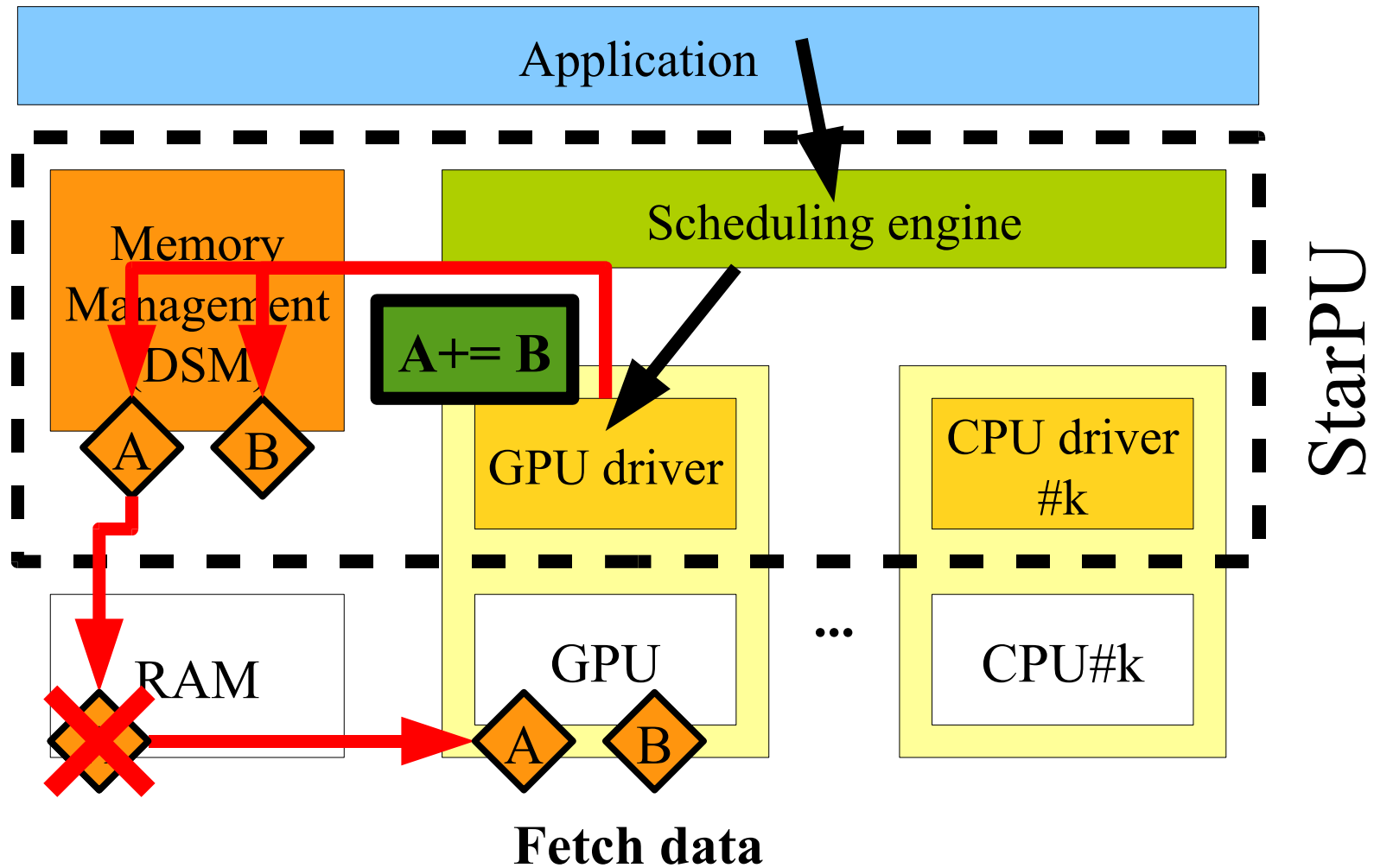
Execution model



StarPU

The StarPU runtime system

Execution model

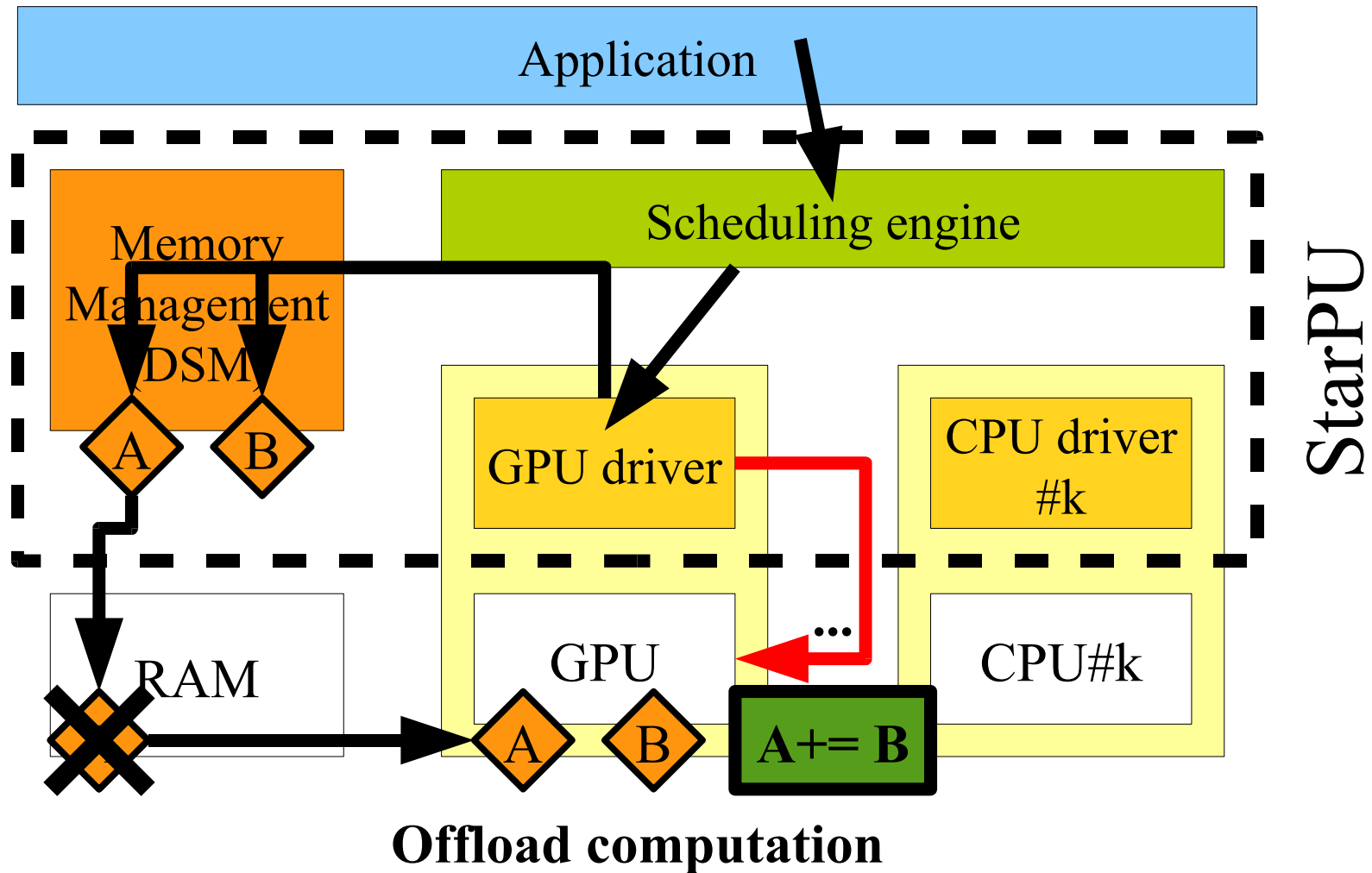


StarPU

Fetch data

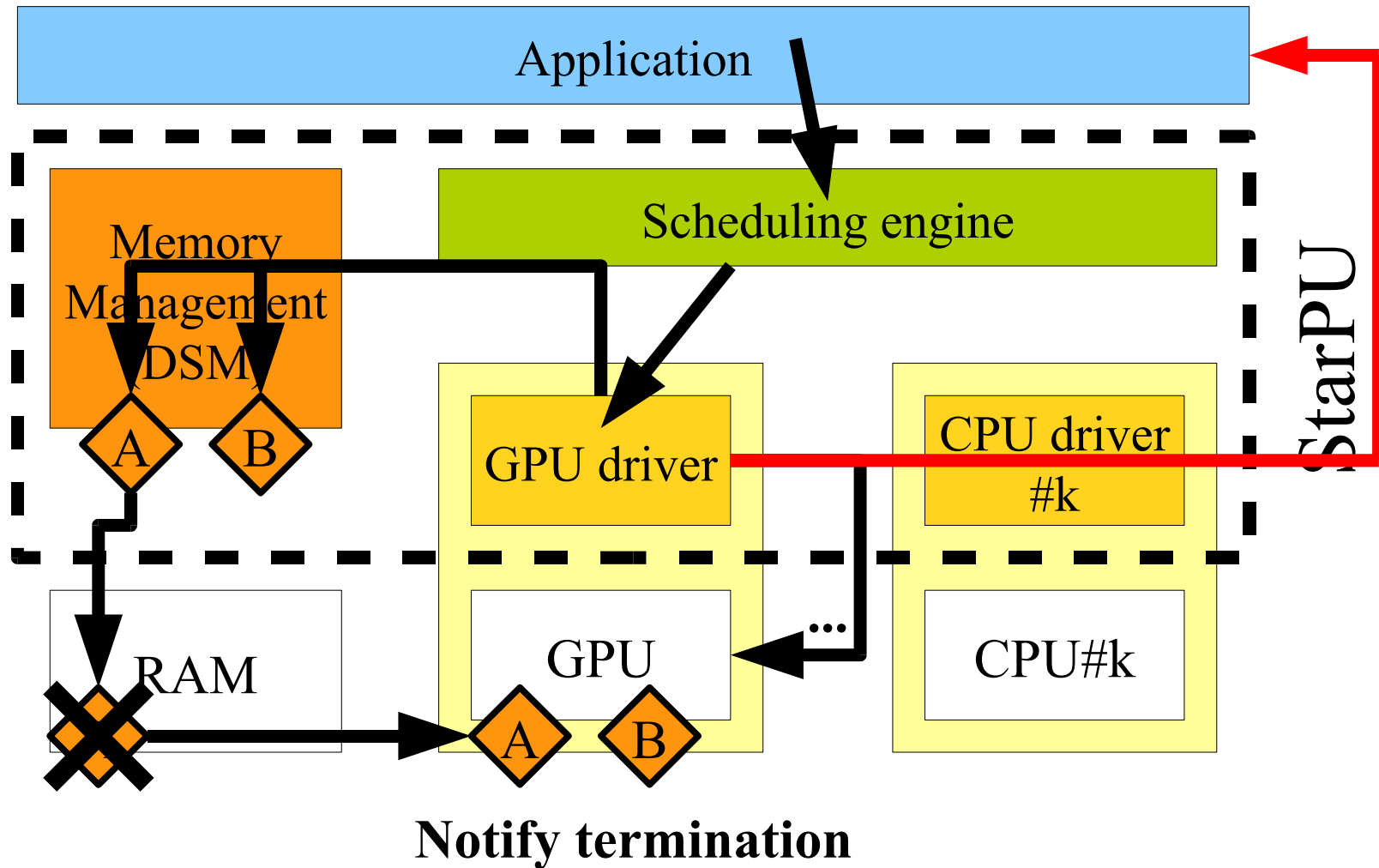
The StarPU runtime system

Execution model



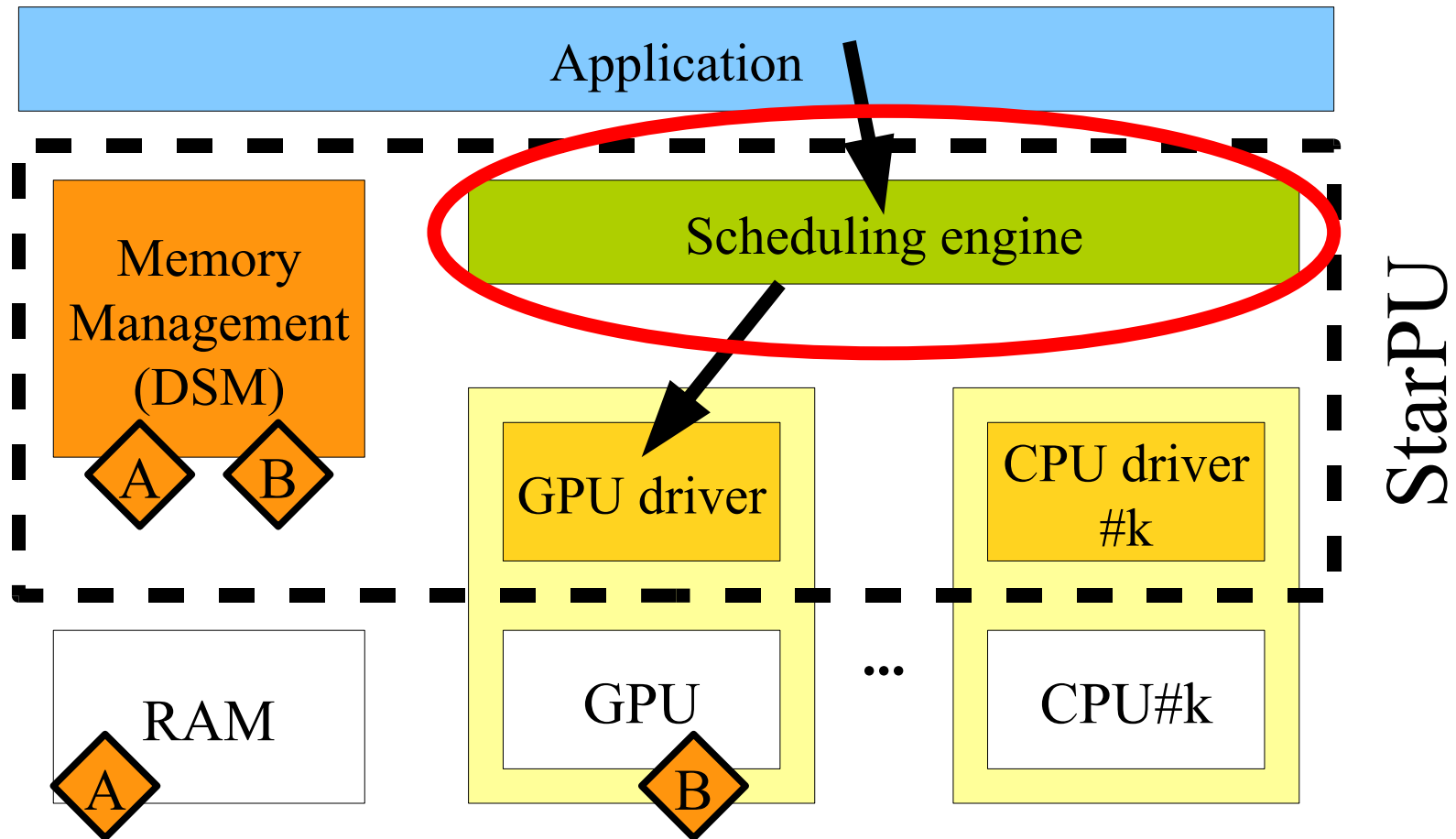
The StarPU runtime system

Execution model



The StarPU runtime system

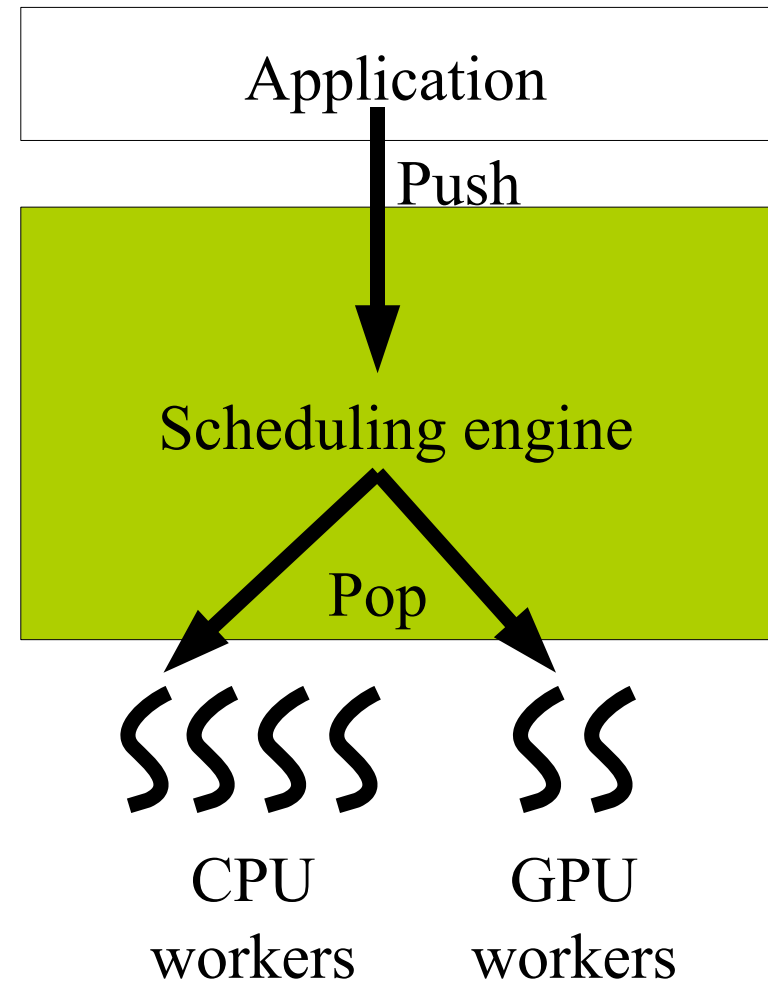
Execution model



StarPU Scheduling Engine

Writing portable policies

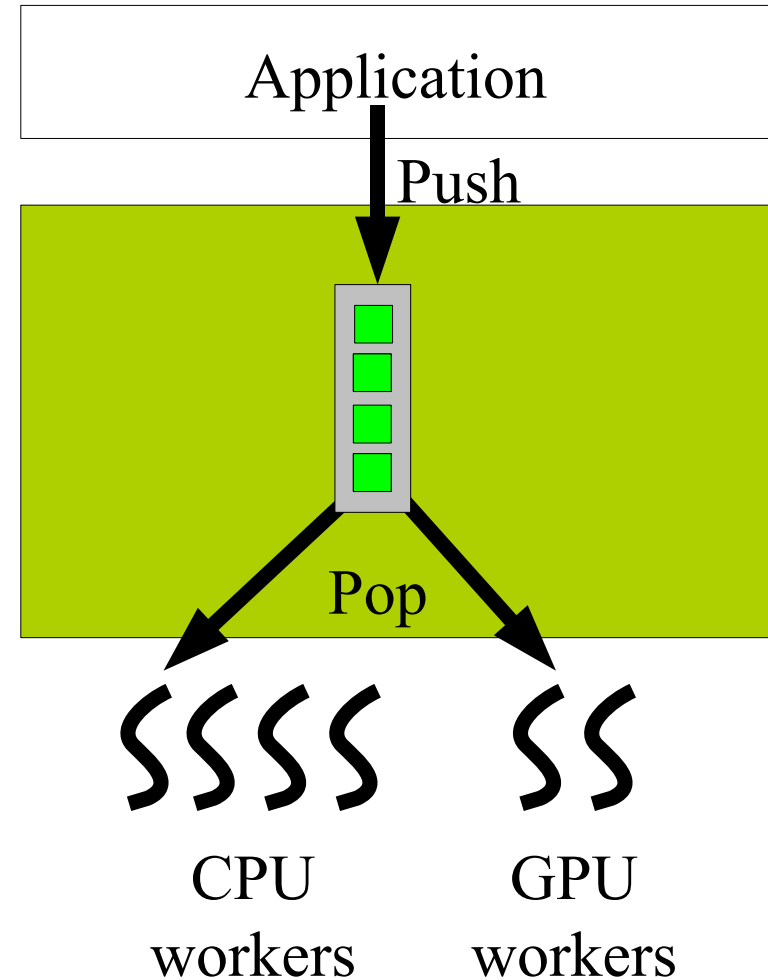
- A Scheduling Black-Box
 - Application's perspective
 - Submit tasks
 - Workers' perspective
 - Request tasks
- A portable scheduler engine
 - Architecture independent
 - CPUs, GPUs, Cell ...
 - Transparent to applications
 - Common interface
 - Select strategy at runtime



StarPU Scheduling Engine

Writing portable policies

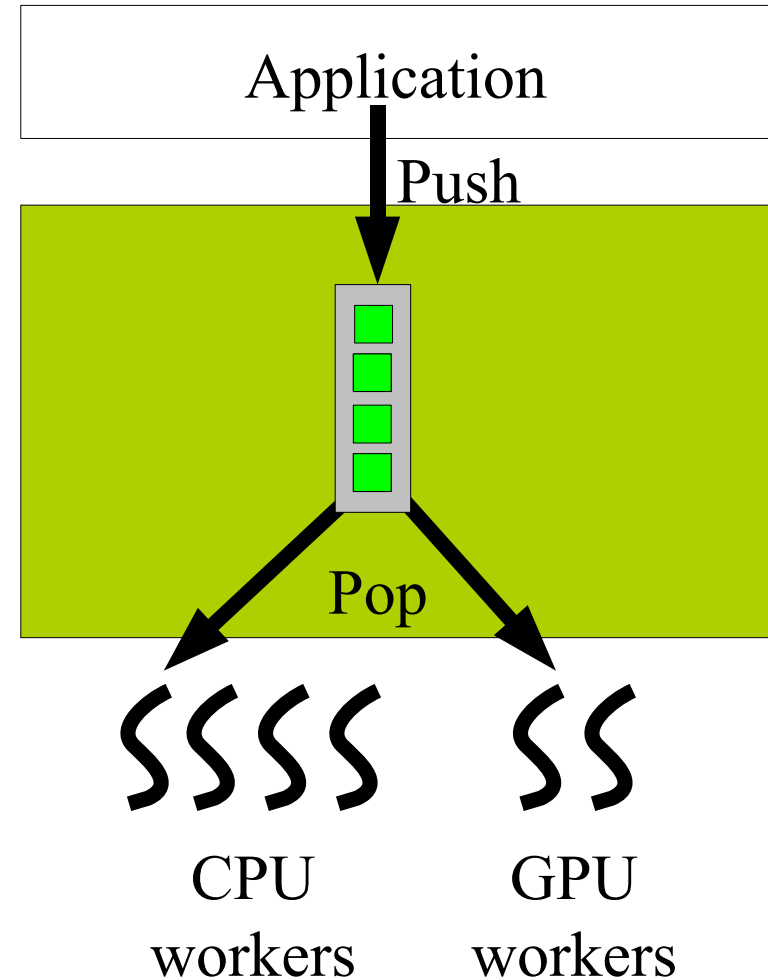
- A Scheduling Black-Box
 - Application's perspective
 - Submit tasks
 - Workers' perspective
 - Request tasks
- A portable scheduler engine
 - Architecture independent
 - CPUs, GPUs, Cell ...
 - Transparent to applications
 - Common interface
 - Select strategy at runtime



StarPU Scheduling Engine

Writing portable policies

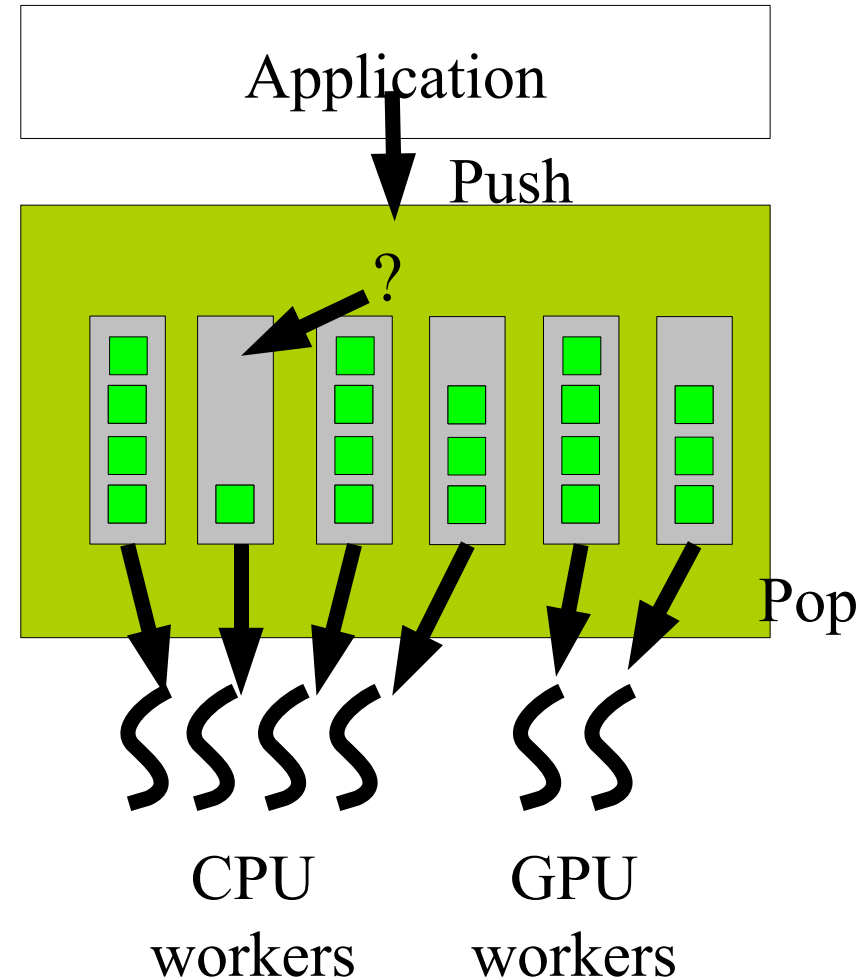
- Queue based scheduler
 - Each worker « pops » task in a specific queue
- Implementing a strategy
 - Easy !
 - Select queue topology
 - Implement « pop » and « push »
 - Priority tasks
 - Work stealing
 - Performance models, ...
- Scheduling algorithms testbed !



StarPU Scheduling Engine

Writing portable policies

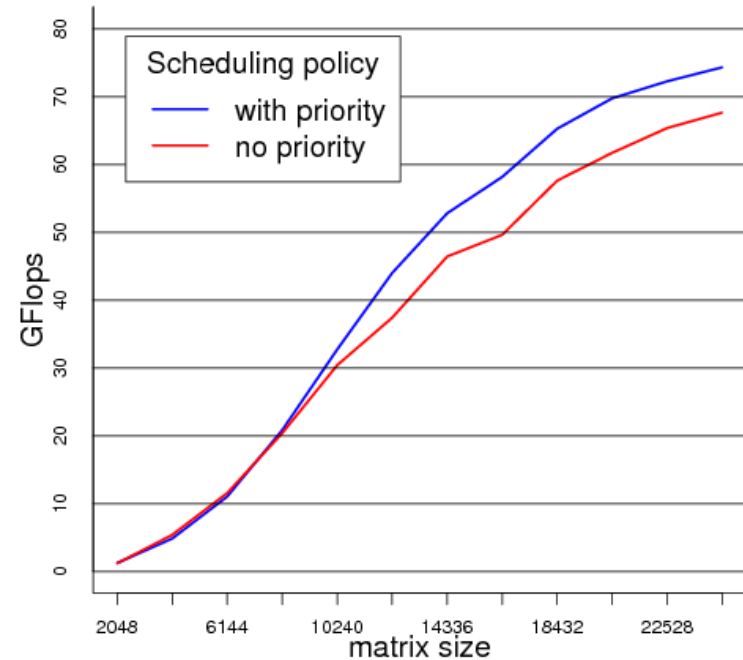
- Queue based scheduler
 - Each worker « pops » task in a specific queue
- Implementing a strategy
 - Easy !
 - Select queue topology
 - Implement « pop » and « push »
 - Priority tasks
 - Work stealing
 - Performance models, ...
- Scheduling algorithms testbed !



Evaluation

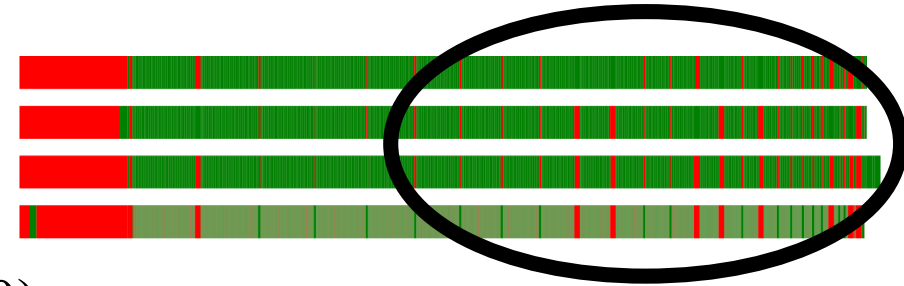
Cholesky decomposition

Cholesky decomposition



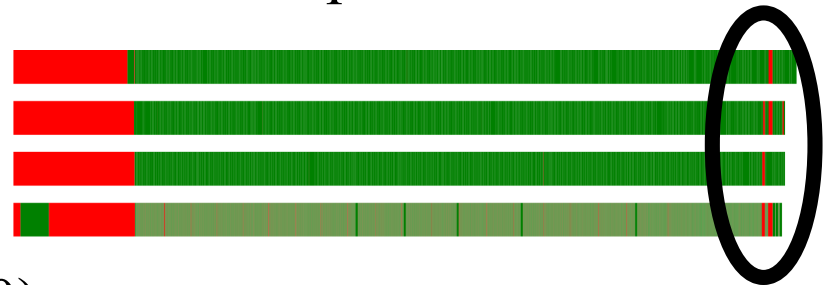
Without priorities

3 CPUs

1 GPU
(FX4600)

With priorities

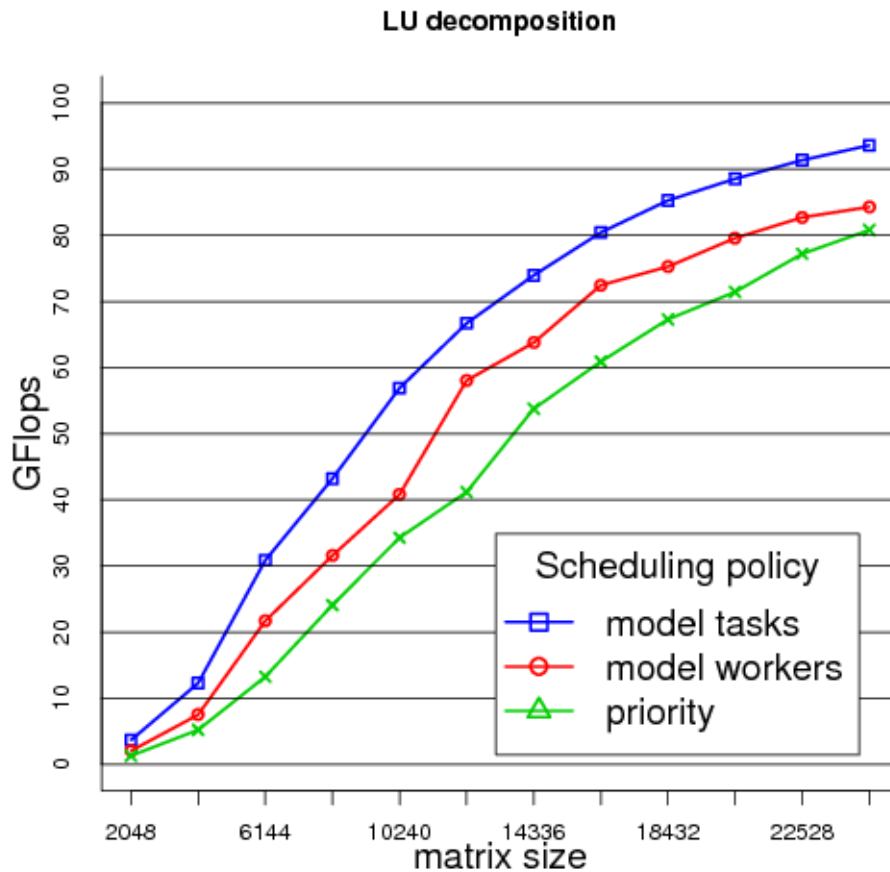
3 CPUs

1 GPU
(FX4600)

Priority tasks : gain $\sim 10\%$

Evaluation

Performance models



- Modeling workers' performance
 - 1 GPU = 10x faster than a CPU
 - Reduce load imbalance
 - All tasks are not identical
- Modeling tasks' execution time
 - Performance models
 - Explicit
 - Auto-tuned
 - Minimize termination time (HEFT)

Evaluation

Performance models

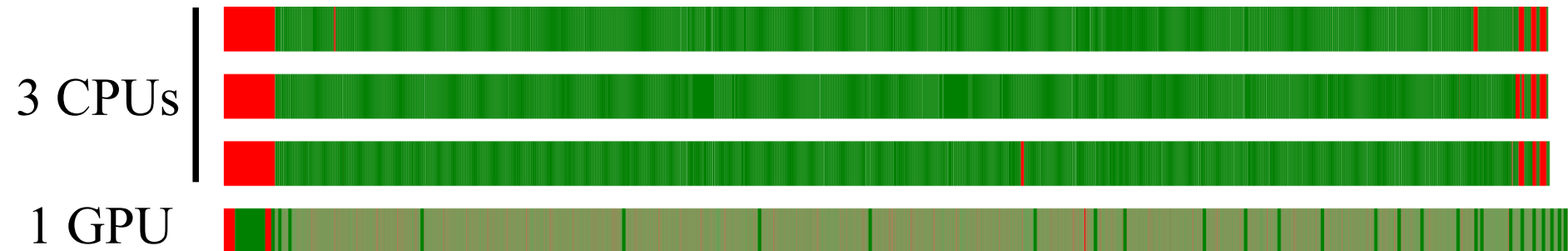
Simple model : “1 GPU = 10x faster than 1 CPU”

3 CPUs + 1 GPU
95.41 GFlops

3 CPUs
21.24 GFlops

1 GPU (FX4600)
75.04 GFlops

$$95.41 = 99.1\% (21.24 + 75.04)$$



Evaluation

Performance models

Simple model : “1 GPU = 10x faster than 1 CPU”

3 CPUs + 1 GPU
95.41 GFlops

3 CPUs
21.24 GFlops

1 GPU (FX4600)
75.04 GFlops

$$95.41 = \mathbf{99.1\%} (21.24 + 75.04)$$

Per-architecture performance model :

3 CPUs + 1 GPU
98.21 GFlops

3 CPUs
21.68 GFlops

1 GPU (FX4600)
75.07 GFlops

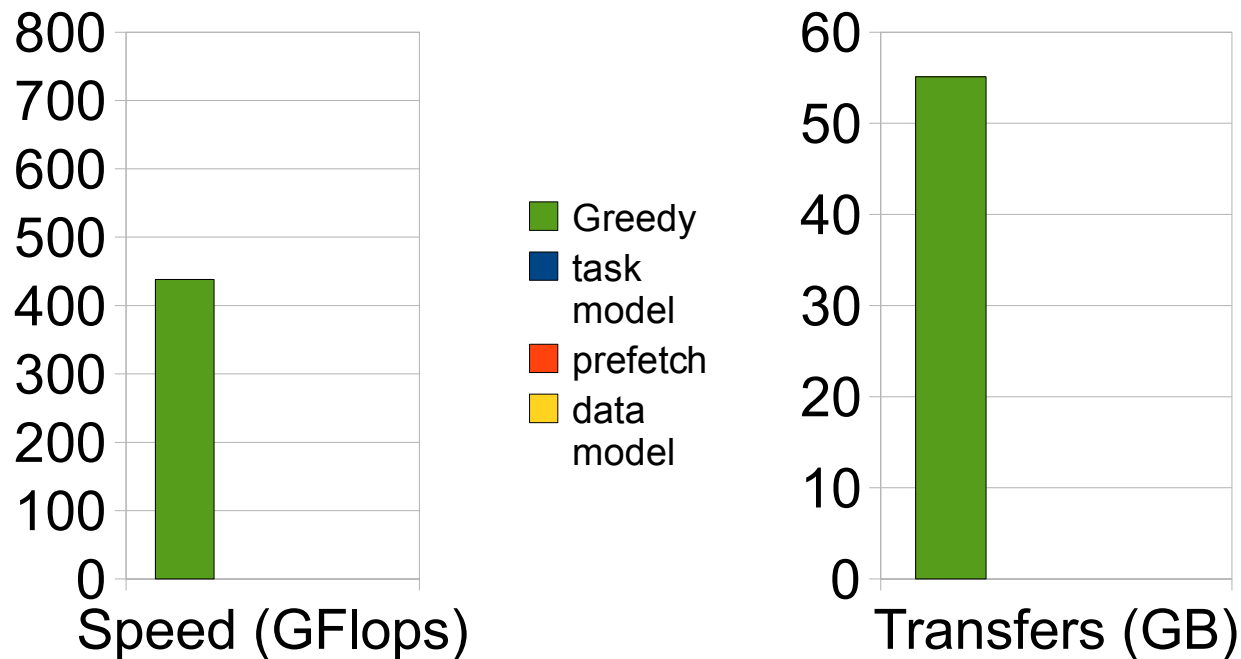
$$98.21 = \mathbf{101.5\%} (21.68 + 75.07)$$



Evaluation

Performance models

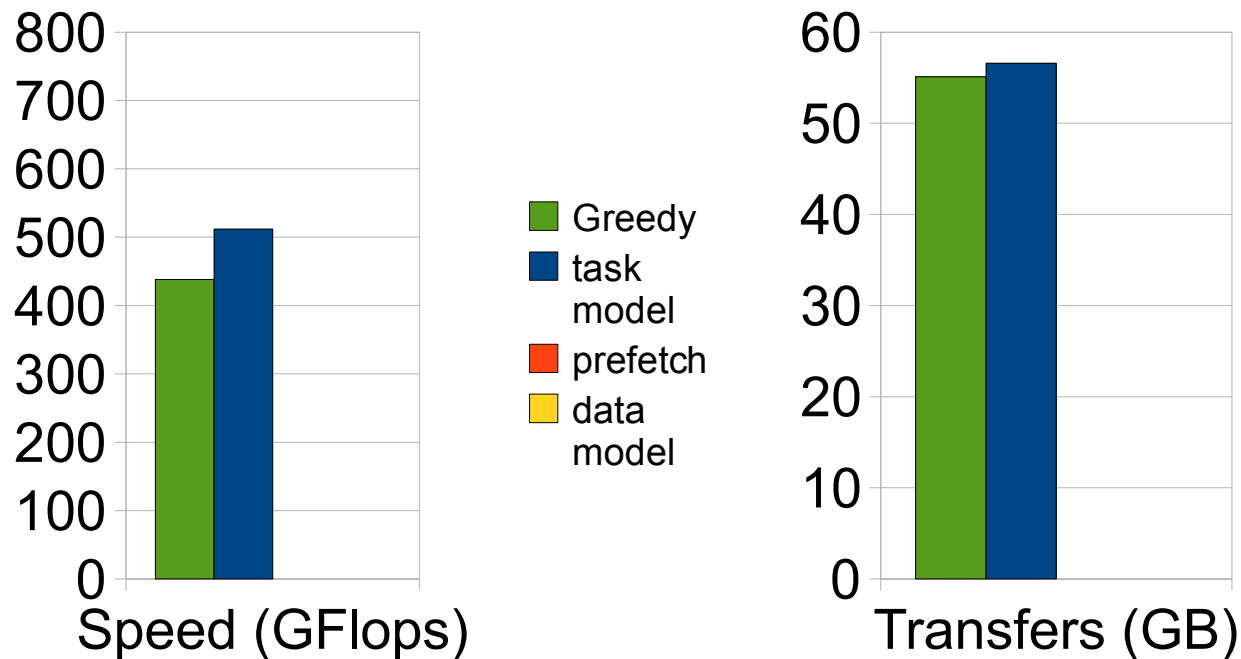
- LU decomposition
 - 8 CPUs (nehalem) + 3 GPUs (FX5800)



Evaluation

Performance models

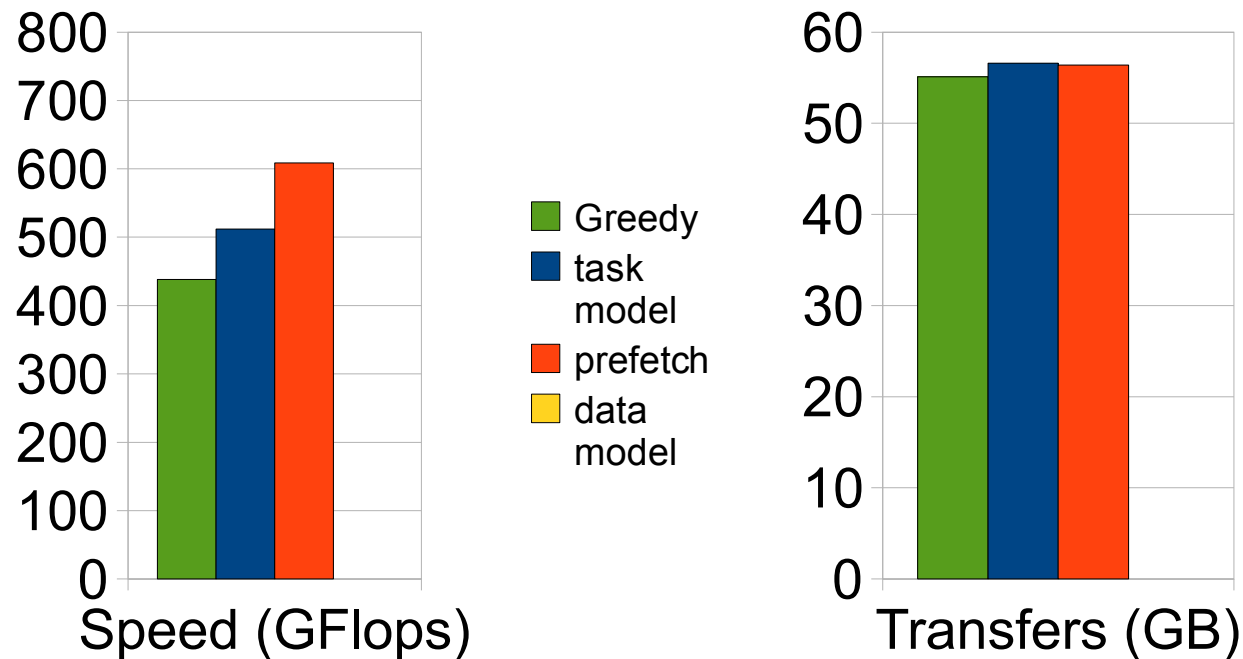
- LU decomposition
 - 8 CPUs (nehalem) + 3 GPUs (FX5800)



Evaluation

Performance models

- LU decomposition
 - 8 CPUs (nehalem) + 3 GPUs (FX5800)

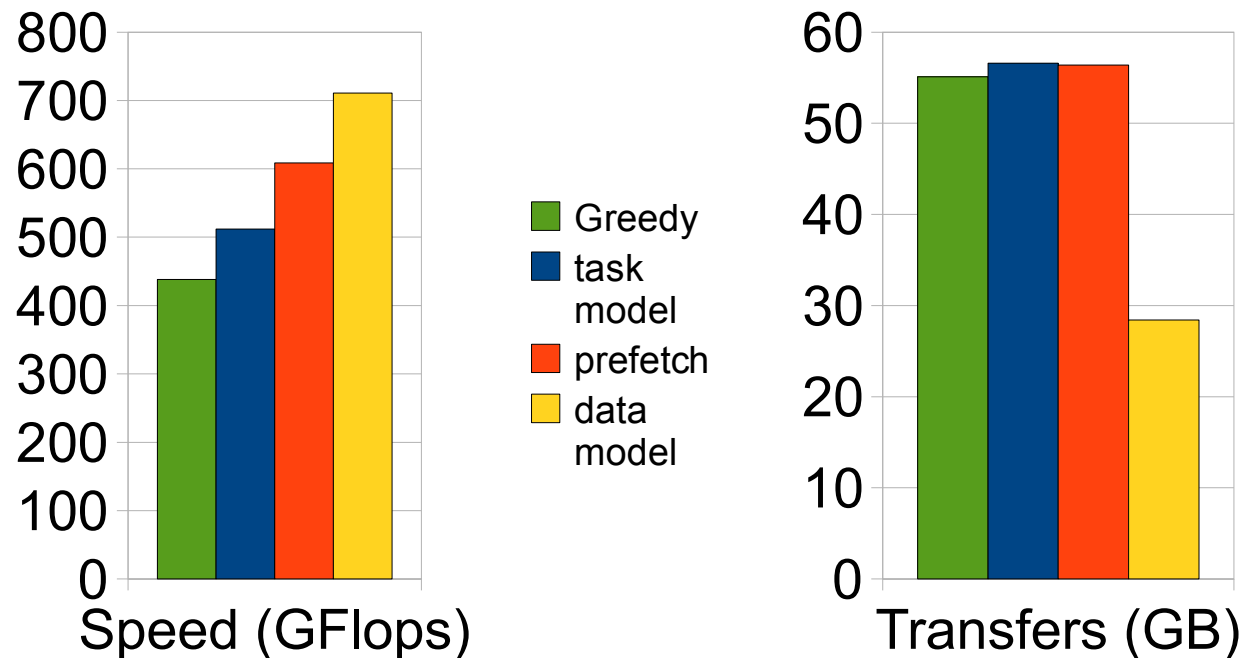


« It's all about the data »

Evaluation

Performance models

- LU decomposition
 - 8 CPUs (nehalem) + 3 GPUs (FX5800)



« It's all about the data »

Conclusion

Summary

- StarPU supports heterogeneous multicore architectures
 - eg. multicore CPUs + heterogeneous multi-GPU
 - Uniform execution model
- Flexible scheduling engine
 - Portable scheduling policies
 - Portability of performance
 - High-level abstractions
 - Playground for scheduling algorithms
- Implemented various policies
 - Significant performance improvements

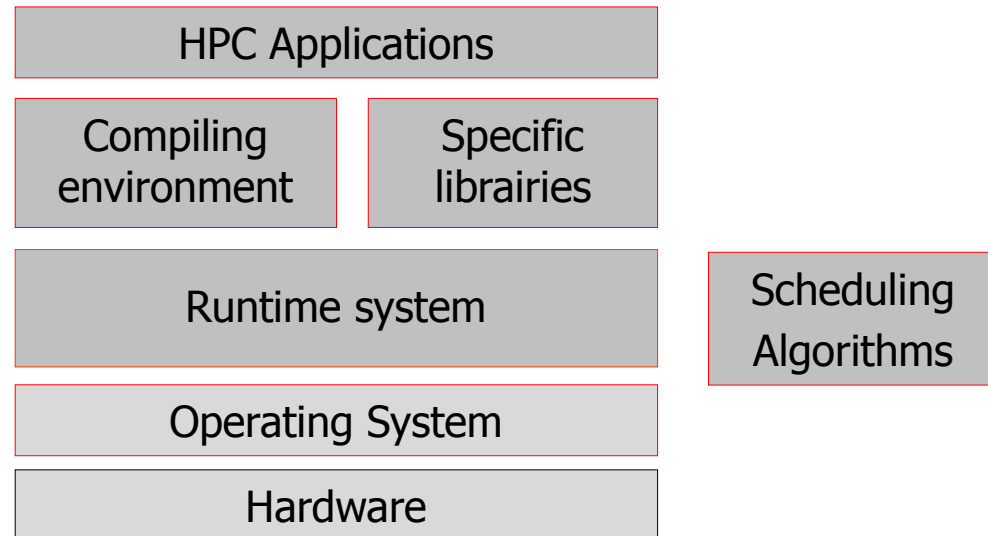


Conclusion

Future works and Perspectives

- Future works

- Support more platforms
 - (eg. OpenCL)
- Implement more policies
- Adaptive granularity



- Perspectives

- Port HPC libs
 - eg. MAGMA
- High-level language
- **Meeting point for various research domains !**

<http://starpu.αforce.inria.fr/>

Thanks for your attention!