NUMA aware framework for reservoir simulation

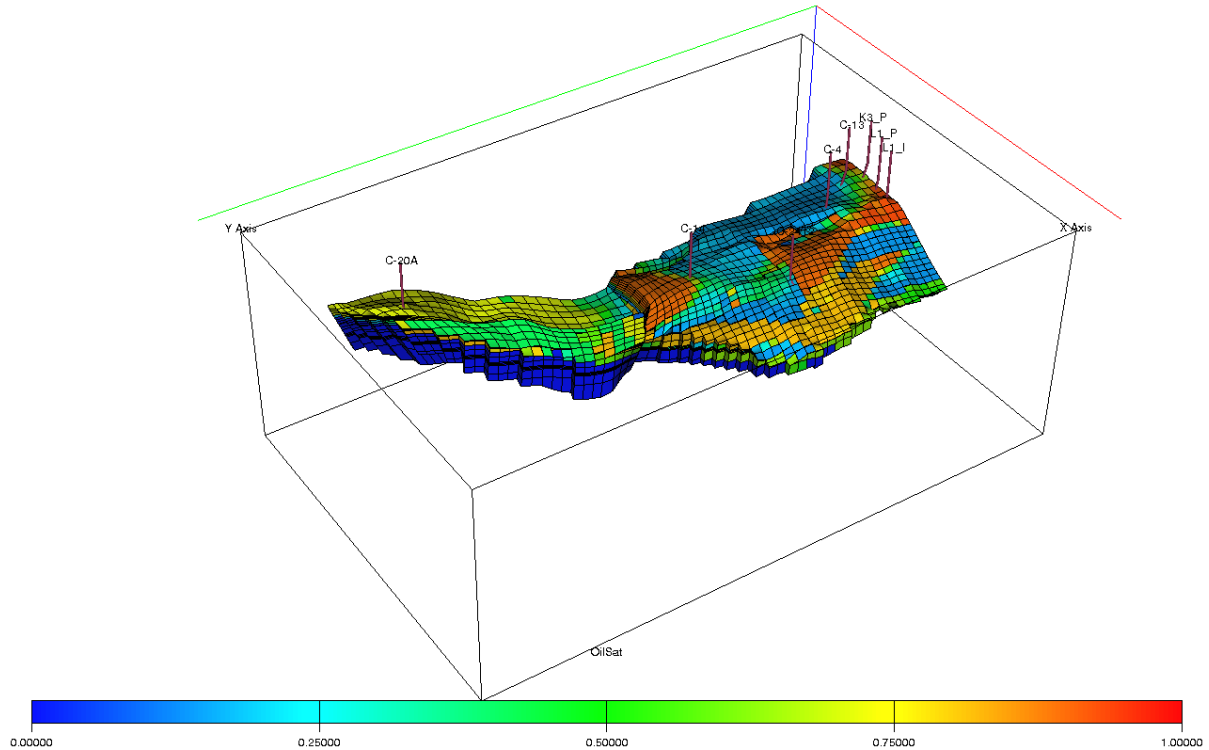
C. Rossignon, P. Hénon, O. Aumage and S. Thibault
Total S.A., Inria Bordeaux, LaBRI, Université Bordeaux I

May 18, 2013

Outline

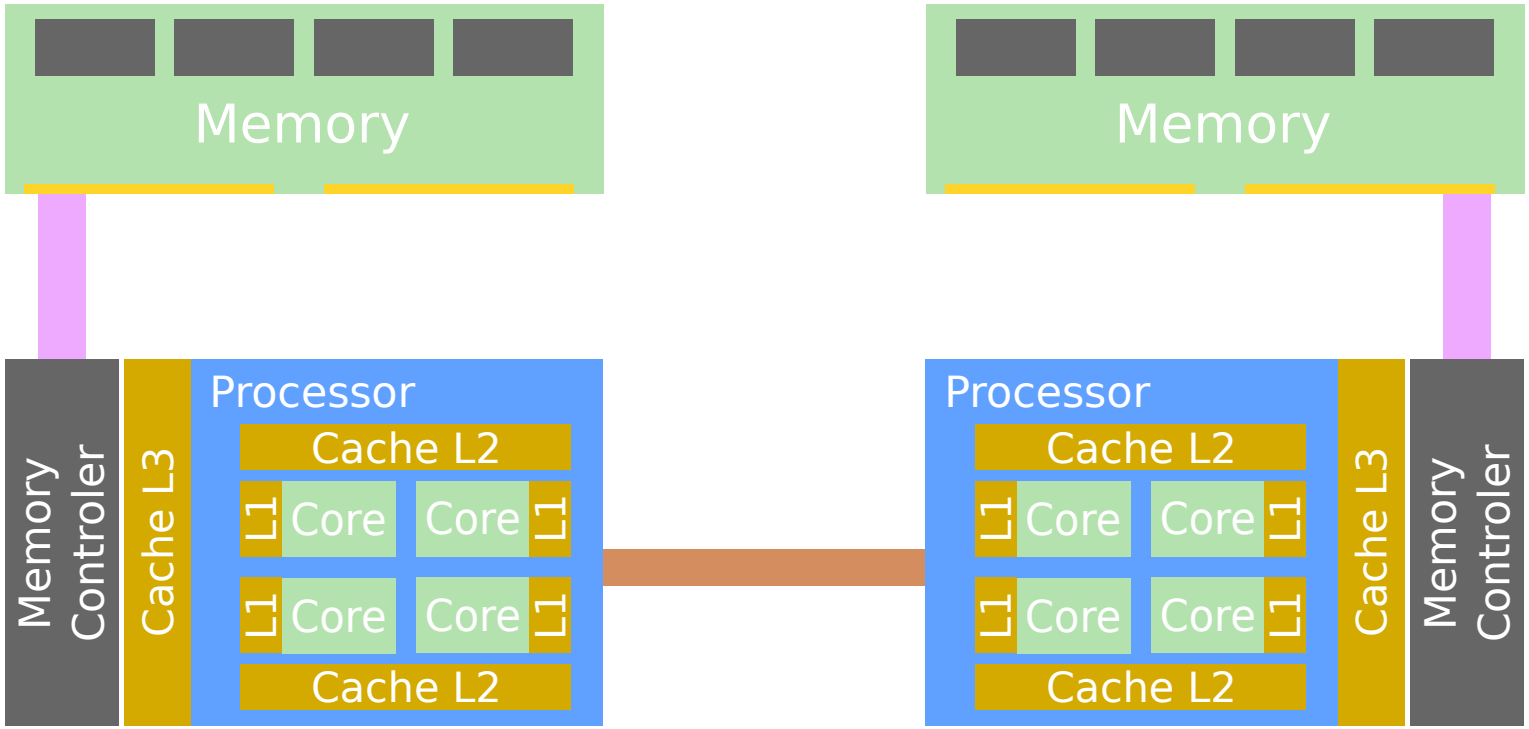
Outline

What do we need for reservoir simulation ?



- Simulation over a period of decades :
  - multi-day time steps
  - must resolve several sparse linear systems
  - septadiagonal matrices with millions of non-zeros
- Use of preconditioned FGMRES

Hierarchical architectures



- NUMA[1] effect concerns most modern clusters
- Need two levels of parallelism :
  - MPI : parallelism over nodes (domain decomposition)
  - Thread : parallelism inside nodes (task based solution)

[1] Non Uniform Memory Access

Task scheduling

Task parallelism

A paradigm to parallelize code under a DAG[2] form where nodes are computation task and edges are dependencies between nodes.

- Good paradigm for problems with data dependencies
- Programmer must define an appropriate grain size

[2] Direct Acyclic Graph

Outline

Problems of ILU(k)[3] factorization

Natural description of task parallelism, but:

- Over 1 million tasks
- 1 task duration is around 100 *ns*
- 1 task management is approximately 500 *ns*

Problem 1

The grain size is too small.

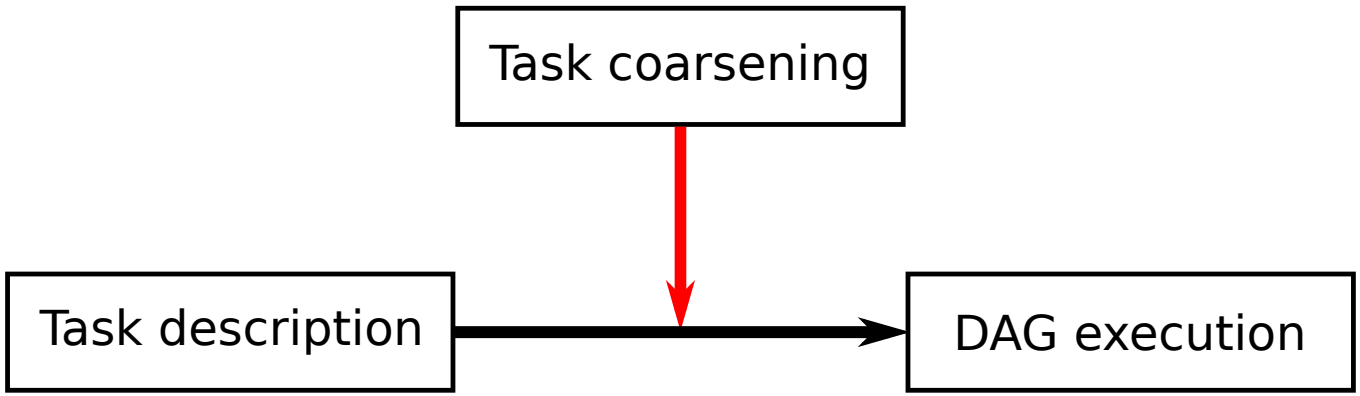
Problem 2

How to take advantage of machine topology.

[3] Incomplete LU, mathematical method

Outline

Task coarsening

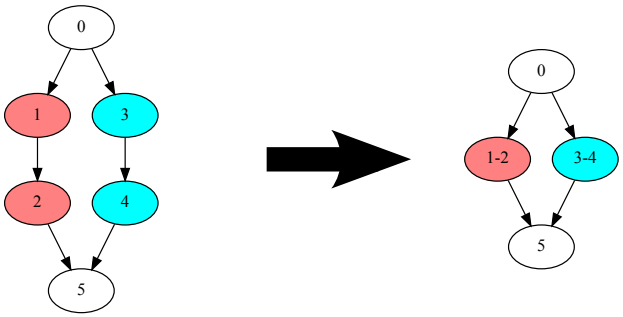


- New step between the task description and DAG execution
- Involves increasing grain size by grouping tasks

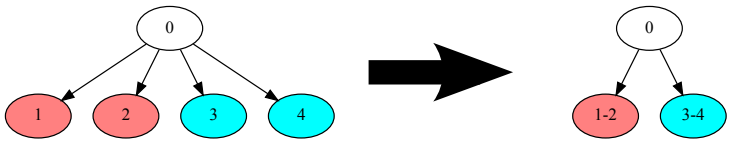
Task coarsening

From a finer DAG, we compute a coarser DAG by applying some generalist coarsening algorithms.

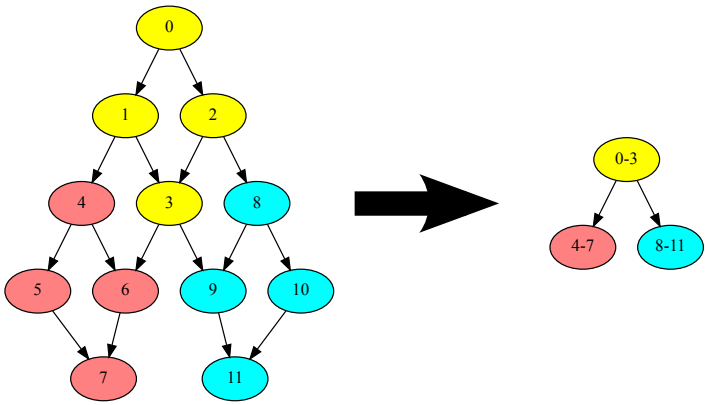
Task coarsening - Algorithms



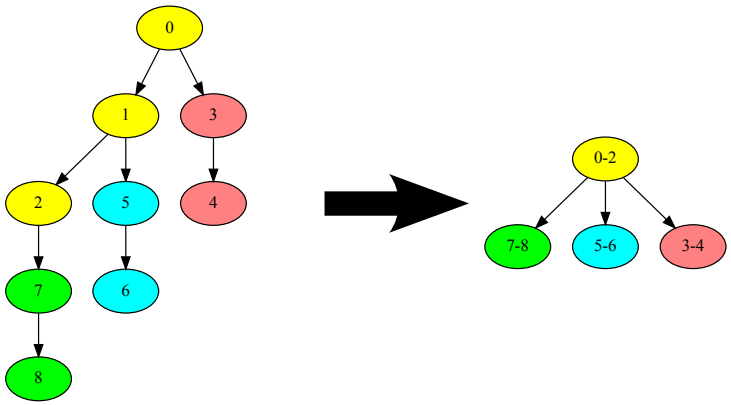
Sequential - S



Reduced Front - F (2)



De-zoomed - D (4)



Continuation Oriented - C

Example of task coarsening

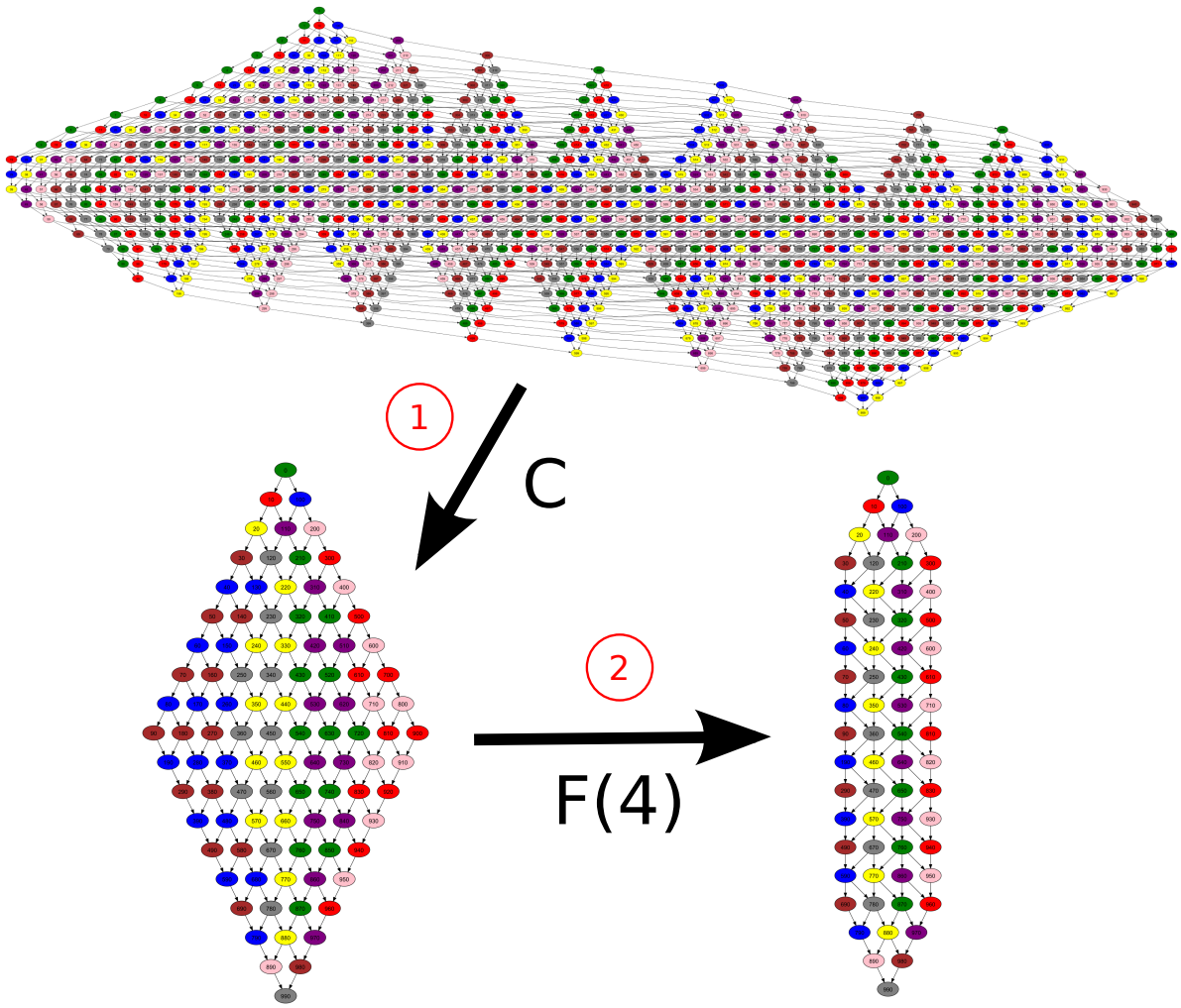
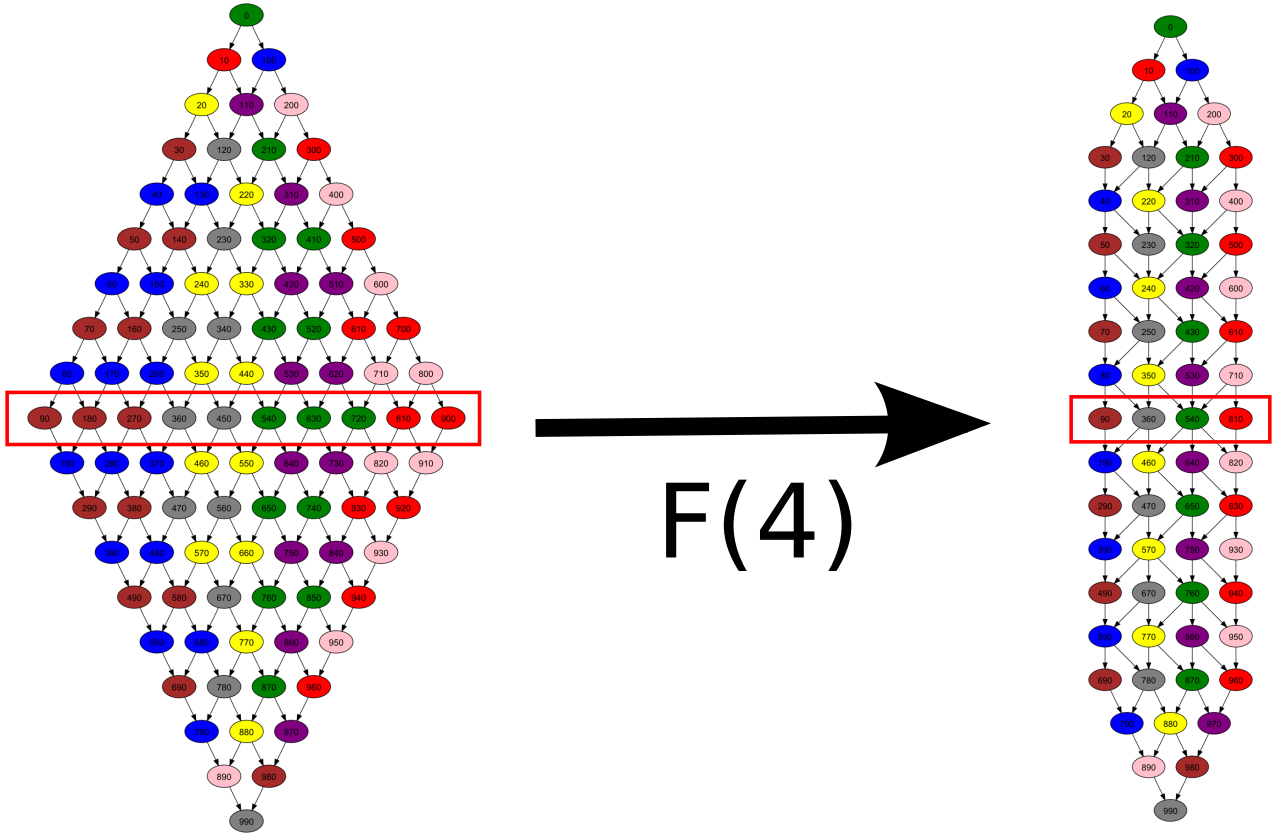


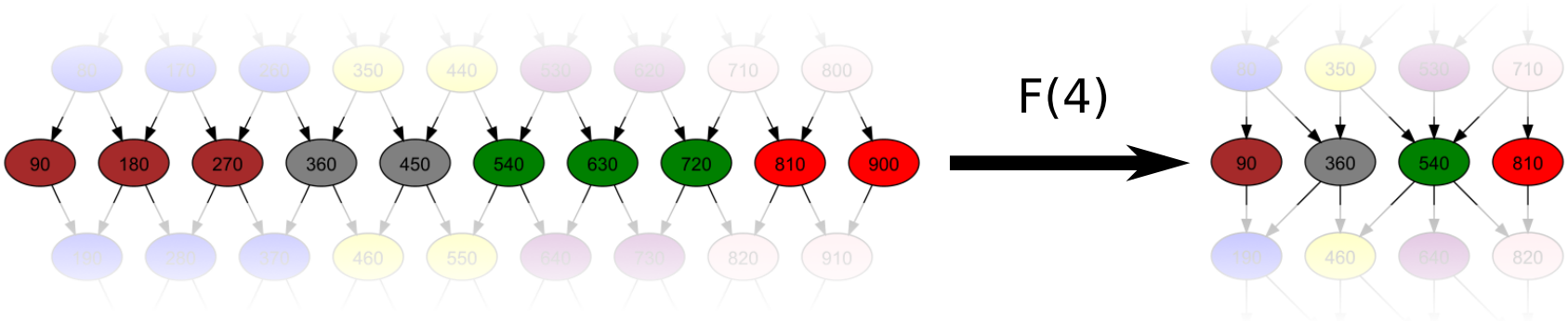
Figure : Aggregation CF(4) of a regular cube of 10 elements per side.

Example of coarsening with the Front algorithm I



- Width of DAG $\approx$ maximum number of tasks at same time
- Reduce width $\Longrightarrow$ reduce contention in task scheduler

Example of coarsening with the Front algorithm II



- For each level :
  - limit group number
  - similar group size
- Useful when too many tasks compared to available CPUs

Results

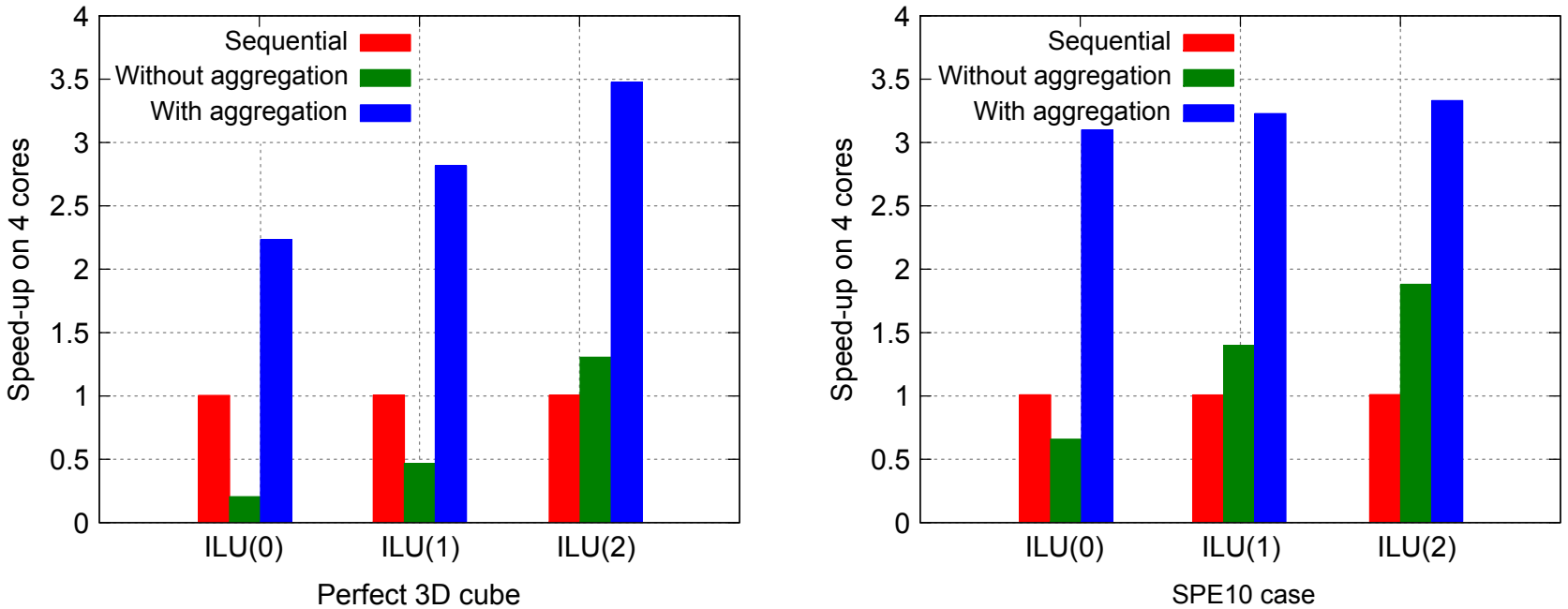
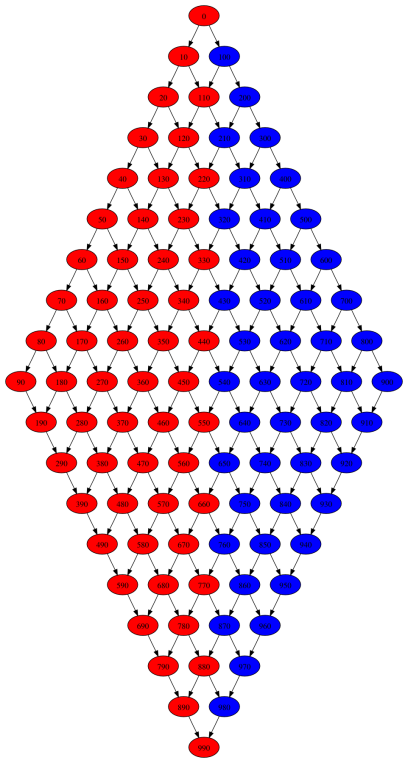


Figure : Speed-up with and without CD(4)[4] aggregation.

[4] Coarser string : Continuation Oriented + De-zoomed with parameter 4

Outline

NUMA aware



Data and execution on NUMA node 0

Data and execution on NUMA node 1

Requirement to use NUMA properly :

- Distribute tasks evenly over NUMA nodes
- Move memory pages close to the task execution core

Nas - a NUMA Aware Scheduler

- Currently, no NUMA aware scheduler exists

Nas - a NUMA Aware Scheduler

- Currently, no NUMA aware scheduler exists
- We have written a very simple NUMA aware scheduler (Nas)

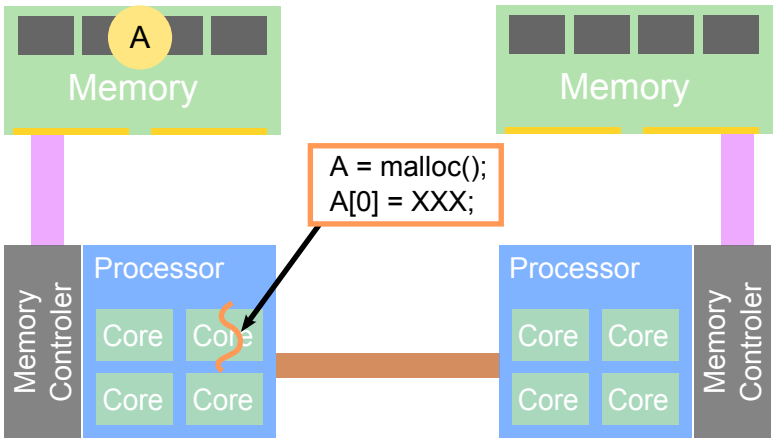
Nas - a NUMA Aware Scheduler

- Currently, no NUMA aware scheduler exists
- We have written a very simple NUMA aware scheduler (Nas)
- Only one task queue per NUMA node

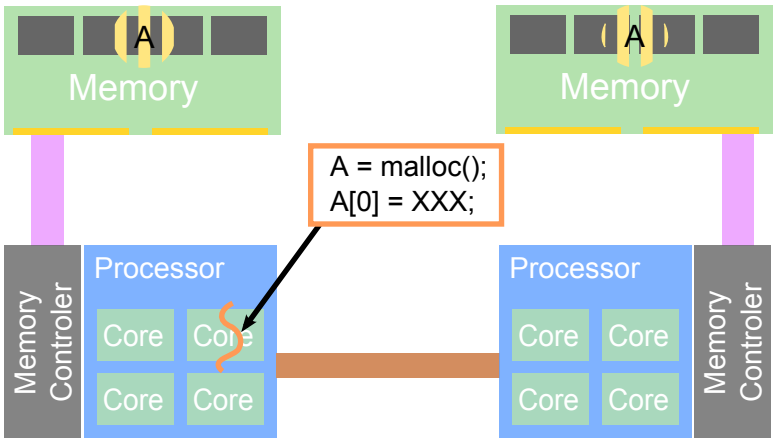
Nas - a NUMA Aware Scheduler

- Currently, no NUMA aware scheduler exists
- We have written a very simple NUMA aware scheduler (Nas)
- Only one task queue per NUMA node
- Without NUMA effect, performance close to Intel TBB

NUMA allocator



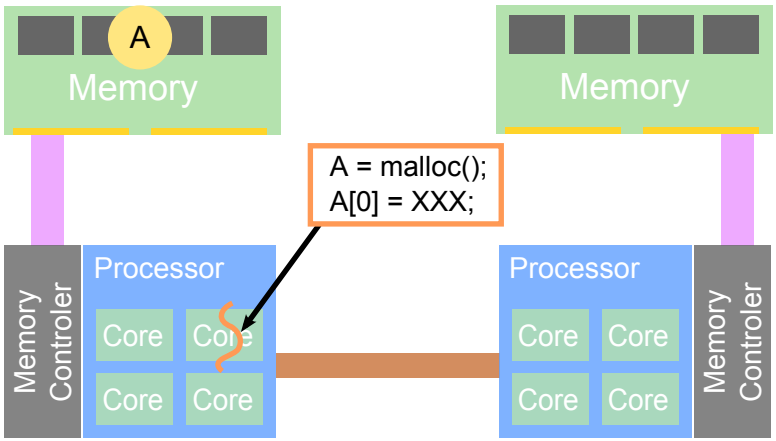
First Touch



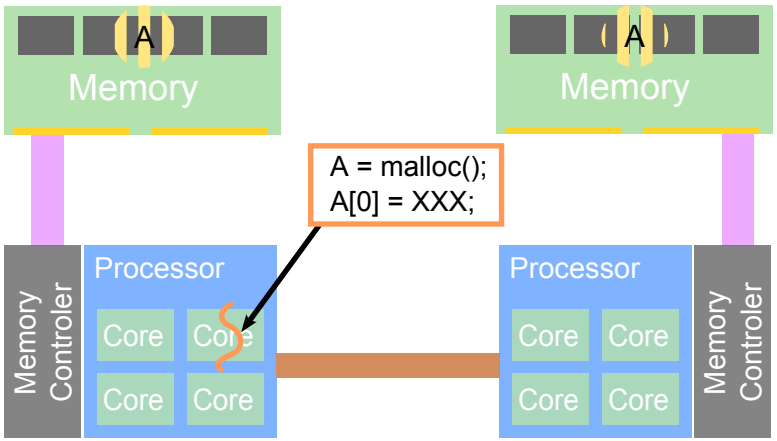
Interleaved

- Linux default behavior is First Touch

NUMA allocator



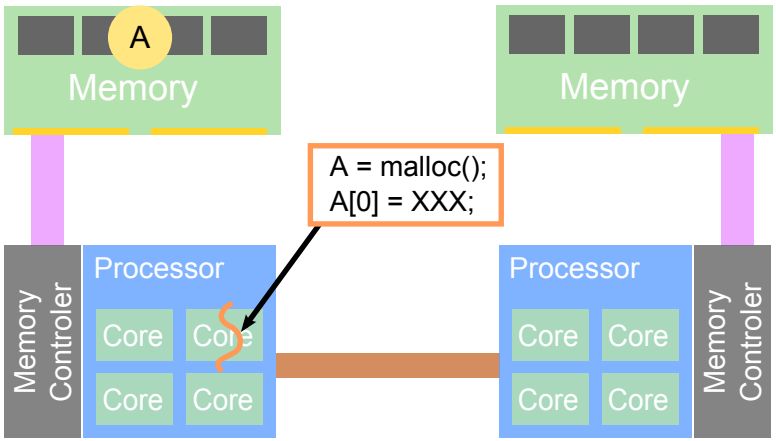
First Touch



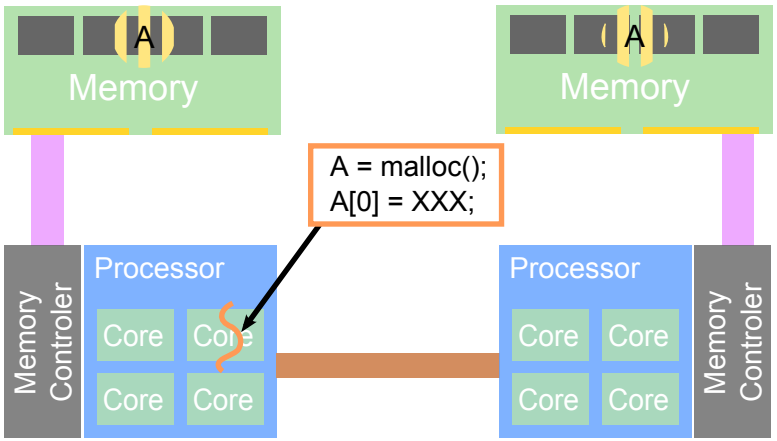
Interleaved

- Linux default behavior is First Touch
- Interleaved allocation improves bandwidth

NUMA allocator



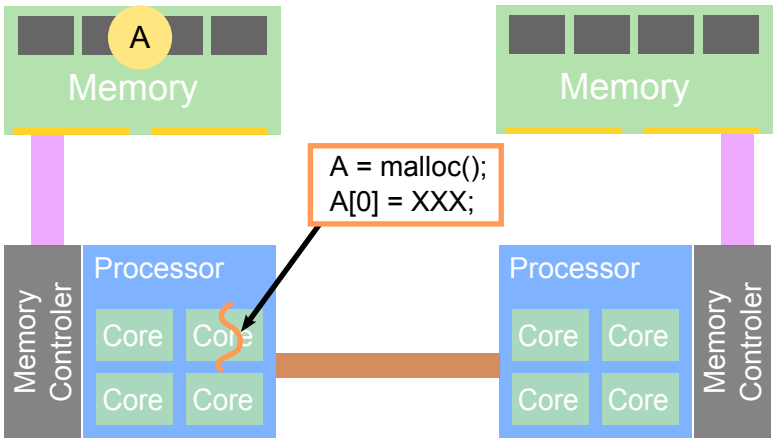
First Touch



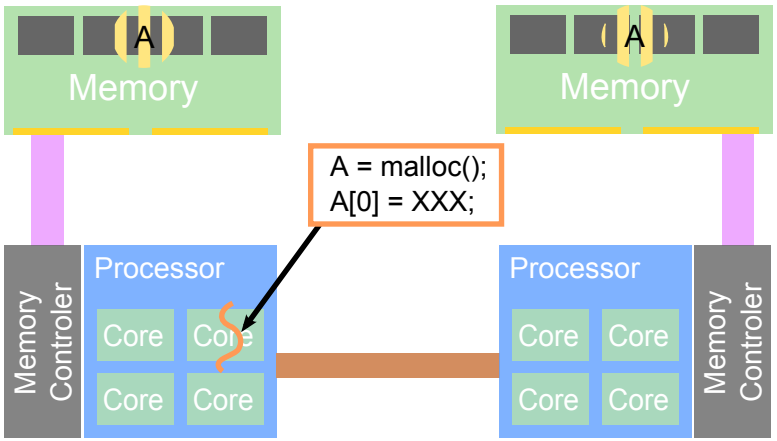
Interleaved

- Linux default behavior is First Touch
- Interleaved allocation improves bandwidth
- Can be changed with “numactl” program

NUMA allocator



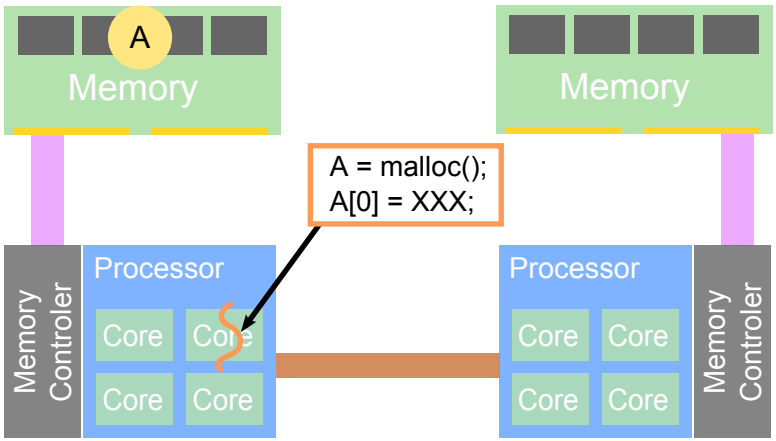
First Touch



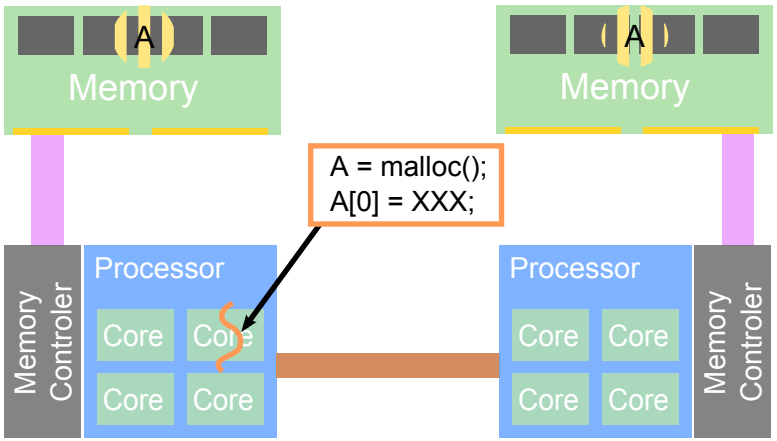
Interleaved

- Linux default behavior is First Touch
- Interleaved allocation improves bandwidth
- Can be changed with “numactl” program
- NUMA allocators in Nas to match data locality with execution

NUMA allocator



First Touch



Interleaved

- Linux default behavior is First Touch
- Interleaved allocation improves bandwidth
- Can be changed with “numactl” program
- NUMA allocators in Nas to match data locality with execution
- Beneficial effects on BLAS

Results

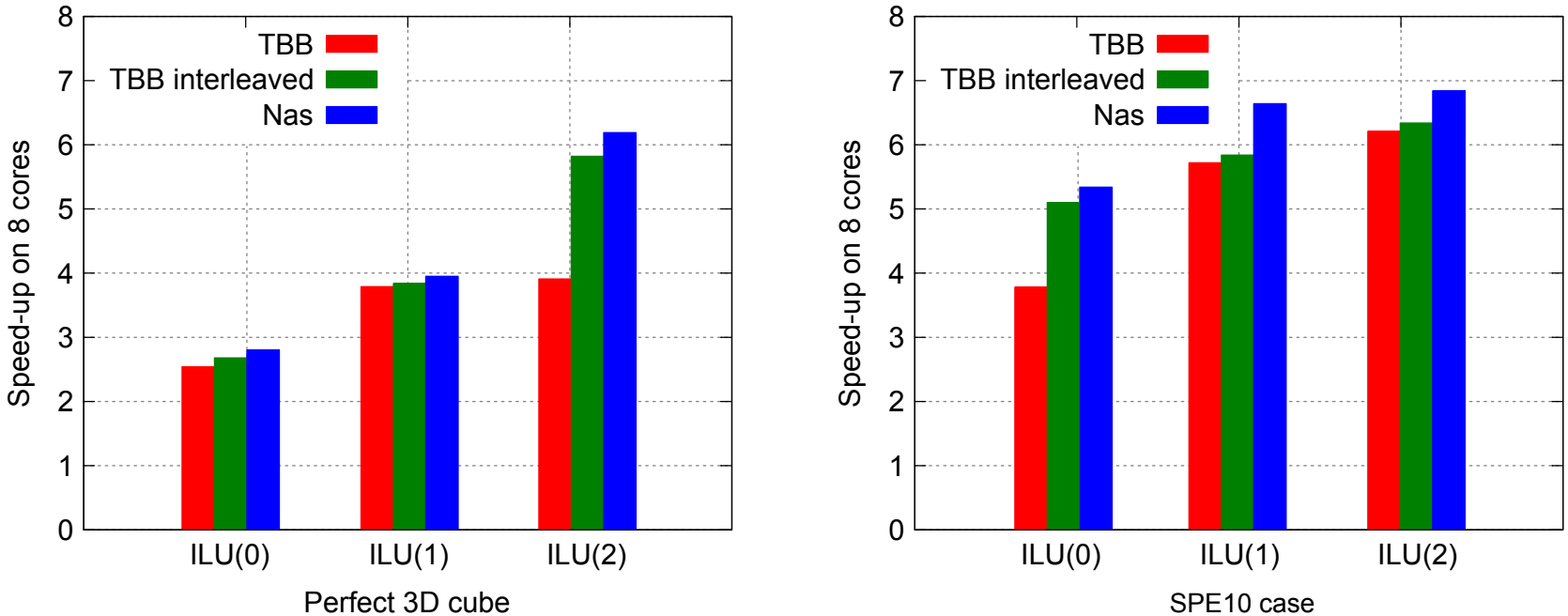


Figure : Speed-up comparing Intel TBB and our NUMA aware scheduler.

Outline

Conclusion and perspectives

Conclusion :

- Major improvement of the ILU(k) thanks to task aggregation

Conclusion and perspectives

Conclusion :

- Major improvement of the ILU(k) thanks to task aggregation
- The basic NUMA scheduler already increases performance

Conclusion and perspectives

Conclusion :

- Major improvement of the ILU(k) thanks to task aggregation
- The basic NUMA scheduler already increases performance
- Almost all BLAS can take advantage of NUMA allocator

Conclusion and perspectives

Conclusion :

- Major improvement of the ILU(k) thanks to task aggregation
- The basic NUMA scheduler already increases performance
- Almost all BLAS can take advantage of NUMA allocator

Perspectives :

- Improve NUMA integration with a cache-aware hierarchical scheduler

Conclusion and perspectives

Conclusion :

- Major improvement of the ILU(k) thanks to task aggregation
- The basic NUMA scheduler already increases performance
- Almost all BLAS can take advantage of NUMA allocator

Perspectives :

- Improve NUMA integration with a cache-aware hierarchical scheduler
- Try aggregation on other problems

Thank you for your attention ...