The MUMPS library

Jean-Yves L'Excellent* and MUMPS team

* INRIA and LIP-ENS Lyon

First Joint Laboratory for Petascale Computing Workshop
(INRIA and University of Illunois)
June 10-12, 2009

Session on Numerical Libraries

The MUMPS team

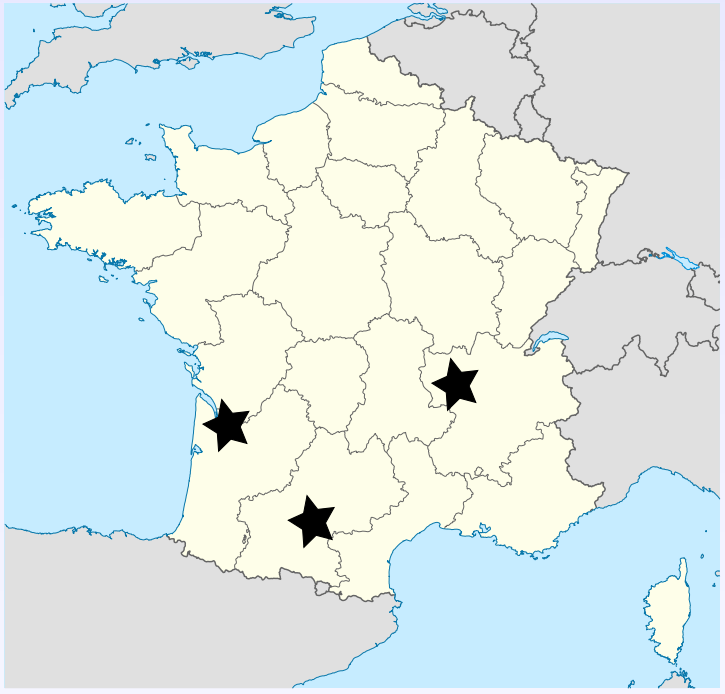

Patrick Amestoy (N7-IRIT, Toulouse)


Jean-Yves L'Excellent (INRIA-LIP, Lyon)


Abdou Guermouche (LABRI, Bordeaux)


Bora Uçar (CNRS-LIP, Lyon)


Alfredo Buttari (CNRS-IRIT, Toulouse)



Post-docs and Students : Indranil Chowdhury$^{new!}$, Emmanuel Agullo^{left}, Mila Slavova^{left}, François-Henry Rouet$^{new!}$

History

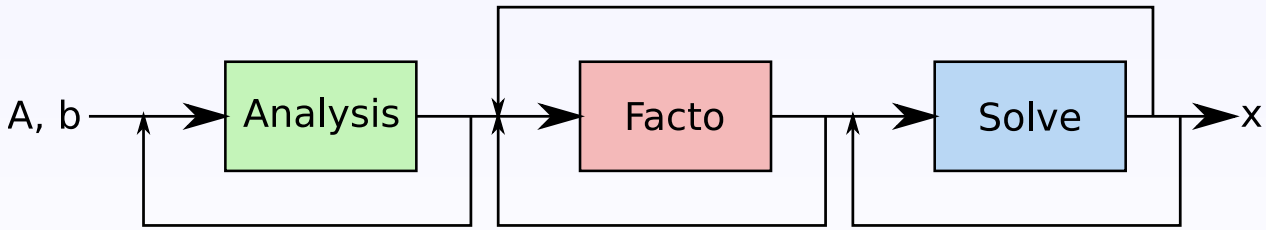
- ⋆ objective : Solve $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is large and sparse.
- ⋆ At the beginning : LTR (Long Term Research) European project, from 1996 to 1999 

- ⋆ Led to first public domain version
- ⋆ Since then, MUMPS was mainly supported by
  - CERFACS (Toulouse),
  - ENSEEIHT-IRIT (Toulouse),
  - INRIA (LIP-Lyon, LaBRI-Bordeaux)
- ⋆ Main projects and contracts around MUMPS :
  - France-Berkeley projects (1999-2000 and 2008-2009)
  - Collaboration with the SEISCOPE consortium (2006-2008)
  - Contracts with CNES (2005), with Samtech S.A. (2005-2006, 2008-2010)
  - French-Israeli project Multicomputing (2009-2010)
  - ANR Solstice project (2007-2010)
  - Starting INRIA "ADT" (2009-2012)

What is MUMPS

MUMPS (**MU**ltifrontal **M**assively **P**arallel sparse direct **S**olver) implements a direct, multifrontal method for solving large, sparse linear systems on parallel environments.
Solution of $Ax = b$ is achieved in three phases :

1. Analysis : matrix is preprocessed to improved its structural properties
2. Factorization : matrix is factorized as $A = LU, LL^T$ or LDL^T
3. Solve : the solution x is computed by means of forward and backward substitutions



Direct solver for $Ax = b$: only a black box ?

Preprocessing and postprocessing :

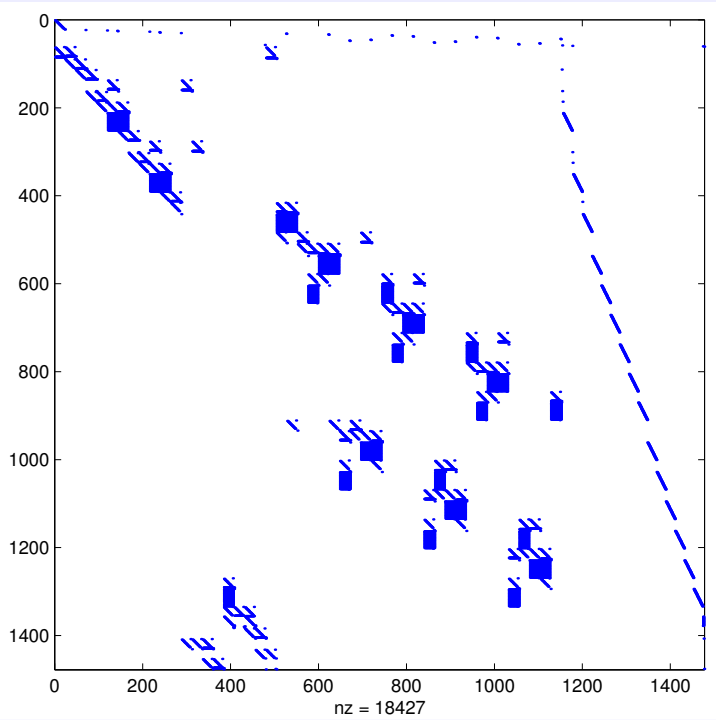
- ⋆ Symmetric permutations to reduce fill :($Ax = b \rightarrow PAP^tPx = Pb$)
- ⋆ Numerical pivoting, scaling to preserve numerical accuracy
- ⋆ Maximum transversal (set large entries on the diagonal)
- ⋆ Preprocessing for parallelism (influence of task mapping on parallelism)
- ⋆ Iterative refinement, error analysis

Default (often automatic/adaptive) setting of the options is available. However, a better knowledge of the options can help the user to further improve :

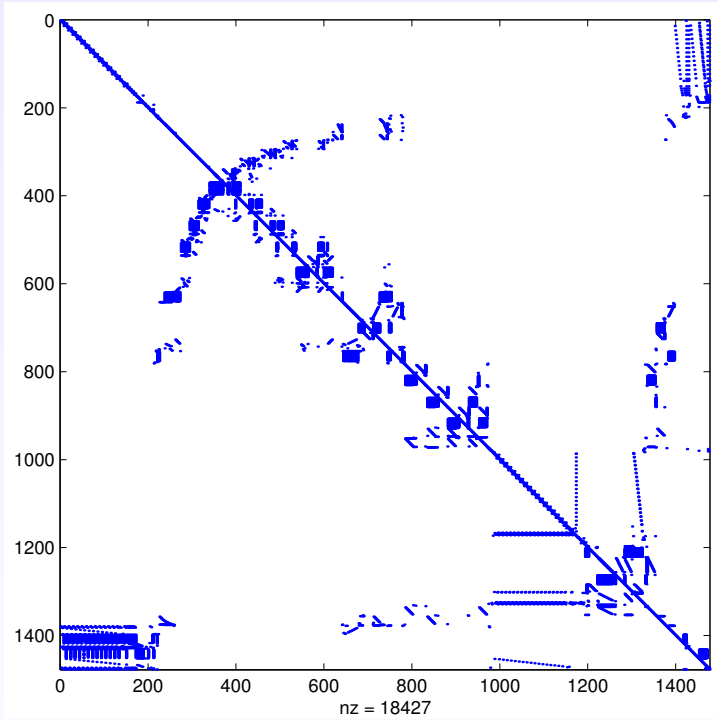
- ⋆ memory usage,
- ⋆ time for solution,
- ⋆ numerical accuracy.

Preprocessing - illustration

Original ($A =$LHR01)



Preprocessed matrix (A'(LHR01))



Modified Problem :$A'x' = b'$ with $A' = P_n P D_r A D_c Q P^t$

Impact of fill-reducing heuristics

Number of operations (millions)

	METIS	SCOTCH	PORD	AMF	AMD
GUPTA2	2757.8	4510.7	*4993.3*	2790.3	**2663.9**
SHIP_003	**83828.2**	92614.0	112519.6	96445.2	*155725.5*
TWOTONE	29120.3	**27764.7**	*37167.4*	29847.5	29552.9
WANG3	**4313.1**	5801.7	5009.9	6318.0	*10492.2*
XENON2	**99273.1**	112213.4	126349.7	237451.3	*298363.5*

Time for factorization (seconds, Power 4 processors at IDRIS)

		1p	16p	32p	64p	128p
AUDI	METIS	2640	198	108	70	42
	PORD	1599	186	146	83	54

Originality of our approach

Objectives

- ⋆ Target large problems on parallel distributed-memory machines
- ⋆ No sacrifice on numerical issues

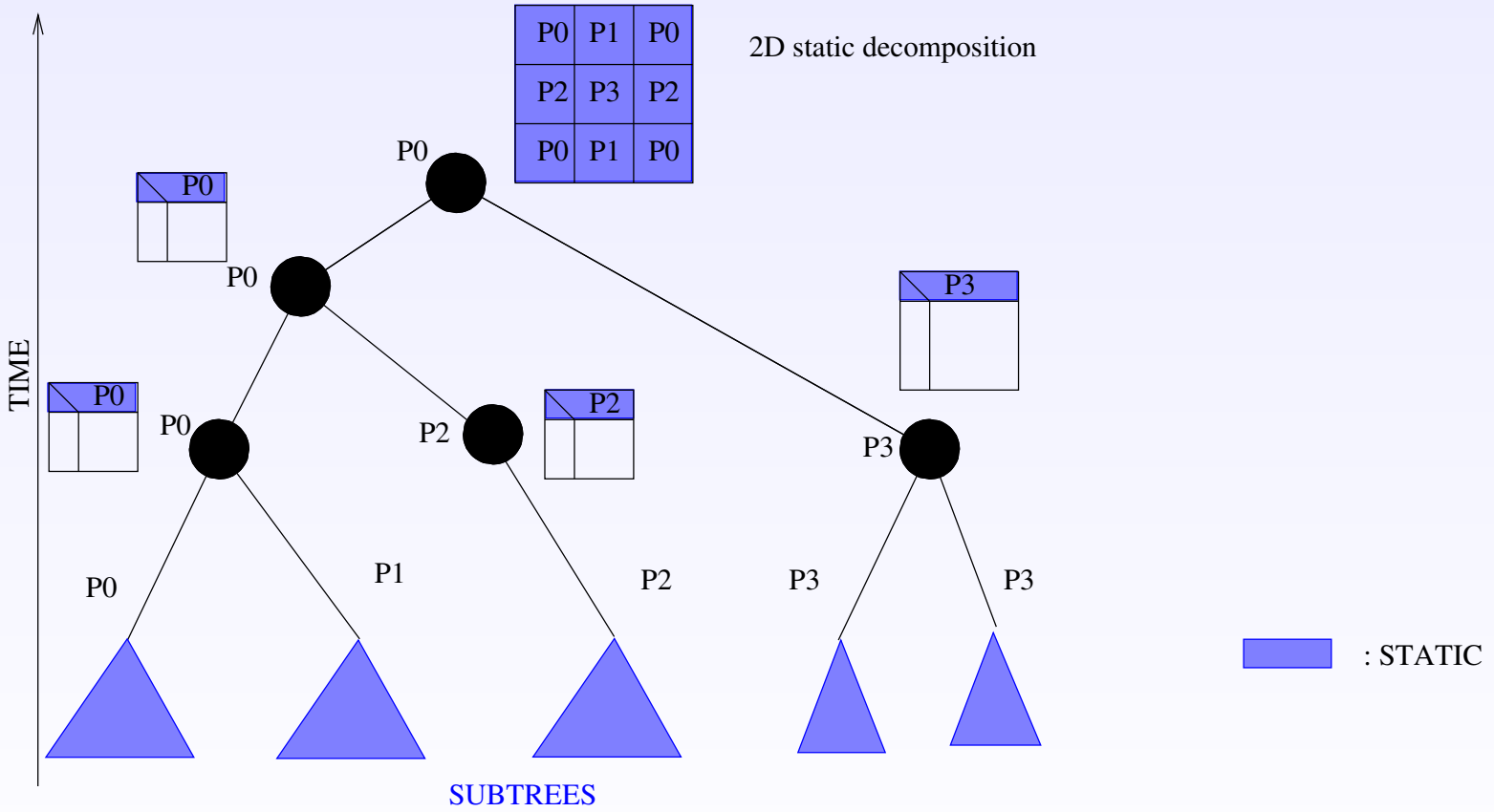
In particular, numerical pivoting $\Rightarrow$ dynamic, not fully predictable tasks graphs

Resulting choices :

- ⋆ Dynamic (runtime) scheduling and mapping of the computational tasks (static information still necessary to help dynamic decisions)
- ⋆ Can adapt to numerical pivoting and load variations
- ⋆ Each processor is a scheduler (choice of the next task to process, mapping subtasks to other processors, ...)
- ⋆ Fully asynchronous approach, guided by message receptions

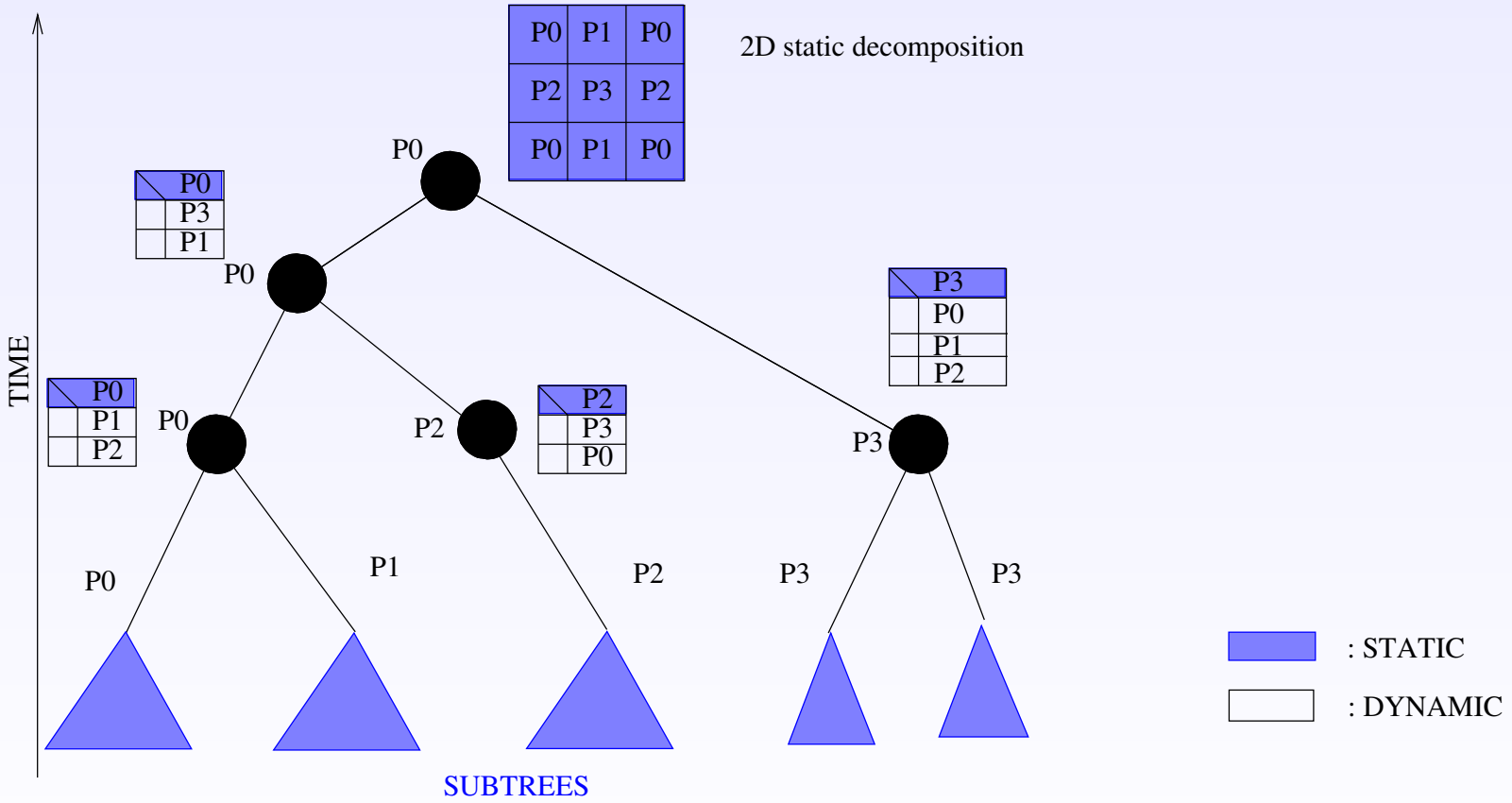
Dynamic Scheduling

- ★ Tasks graph = tree (results from matrix structure and ordering heuristic)
- ★ Each task = partial factorization of a dense matrix (multifrontal method)
- ★ Most parallel tasks mapped at runtime (typically 80 %)



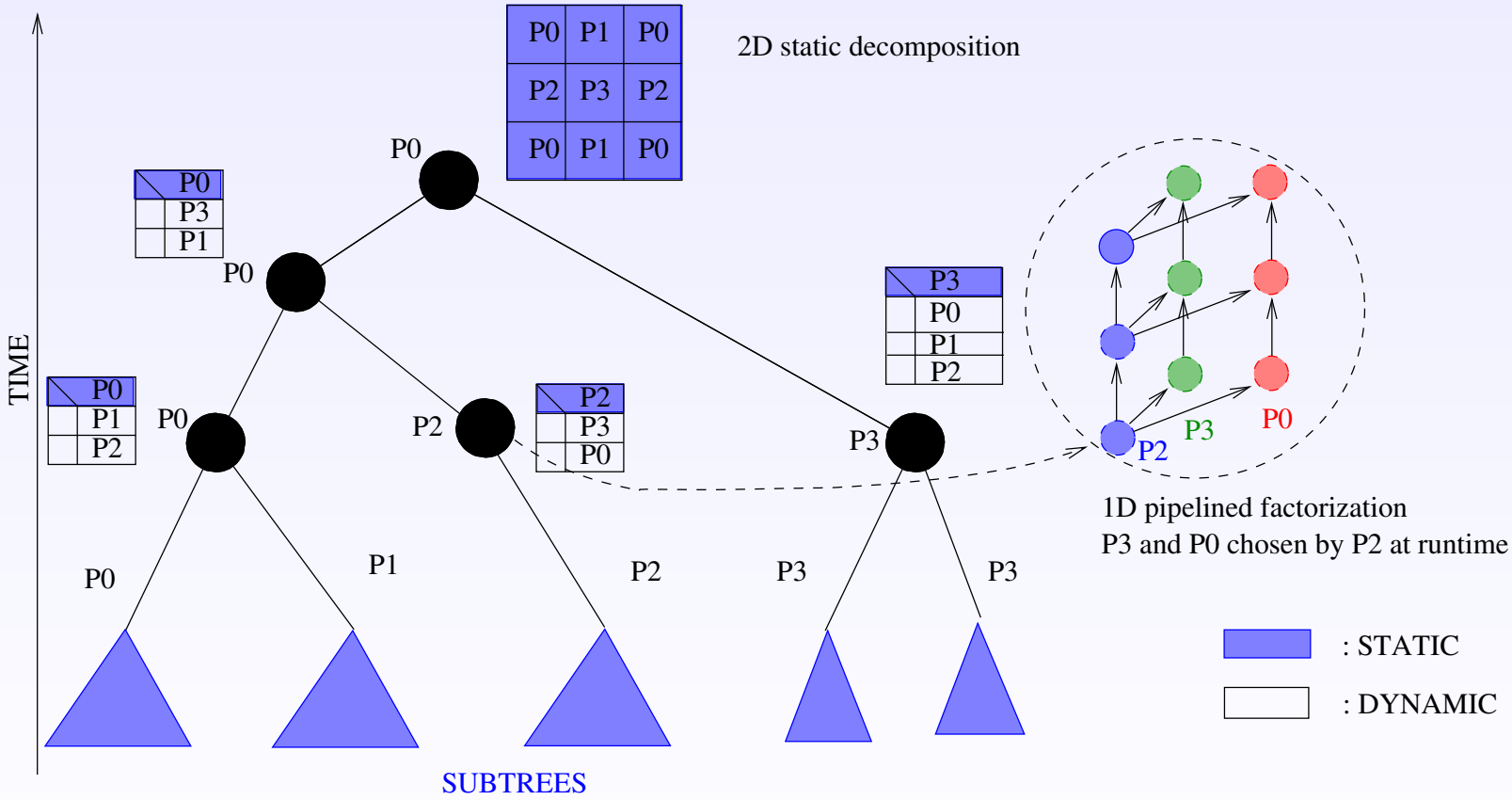
Dynamic Scheduling

- ★ Tasks graph = tree (results from matrix structure and ordering heuristic)
- ★ Each task = partial factorization of a dense matrix (multifrontal method)
- ★ Most parallel tasks mapped at runtime (typically 80 %)

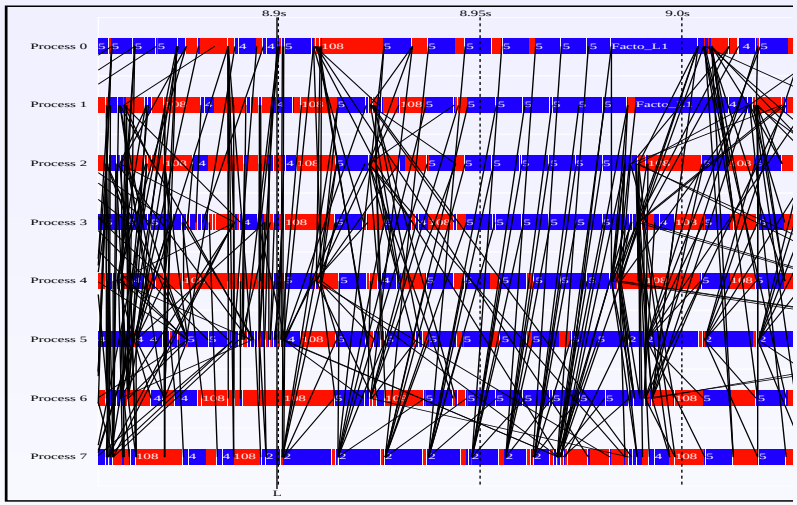


Dynamic Scheduling

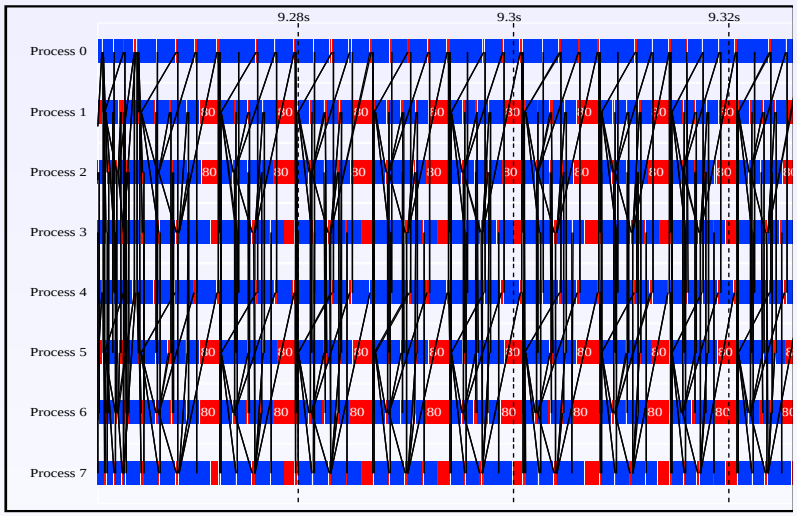
- ★ Tasks graph = tree (results from matrix structure and ordering heuristic)
- ★ Each task = partial factorization of a dense matrix (multifrontal method)
- ★ Most parallel tasks mapped at runtime (typically 80 %)



Trace of execution(BBMAT, 8 processors)



MUMPS



SuperLU_DIST

MUMPS vs other sparse direct solvers

- ⋆ Address wide classes of problems
- ⋆ Very good numerical stability (dynamic pivoting)
- ⋆ Wide range of numerical features

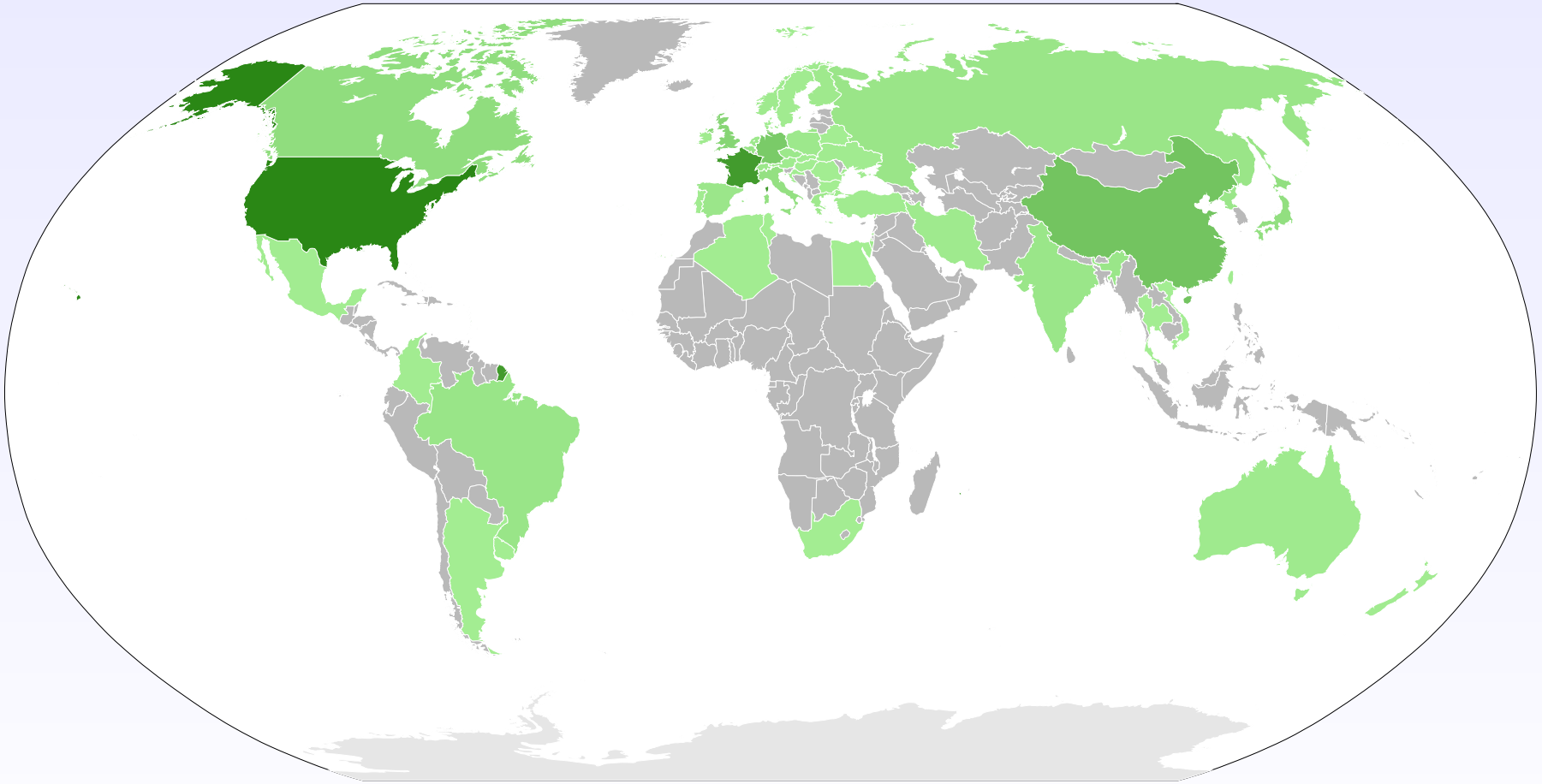
- ⋆ Parallelism and asynchronism harder to manage than in static approaches (e.g. Pastix or `SuperLU_DIST`)
- ⋆ Current version is MPI-based, no explicit management of threads inside MUMPS

Users

- Academics and industrials (Boeing, BRGM, CEA, CNES, EADS, EDF, ESI Group, Free Field Technologies, Samtech, ...),
- Types of applications :
  - Structural mechanics, Fluid dynamics
  - Astrophysics, Magnetohydrodynamics, Physical chemistry
  - Seismic imaging, Ocean modelling
  - Econometric models
  - Oil reservoir simulation
  - Acoustics and electromagnetics wave propagation
  - Biomechanics, Medical image processing
  - Modeling of organs,
  - Heat transfer analysis
  - Research in domain decomposition and hybrid direct-iterative solvers
  - ...
- More and more groups rely on MUMPS in products that they redistribute (Petsc, Samcef, Ipopt, Trilinos, Matlab*P, ...)

User's distribution map

1000+ download requests per year



MUMPS at UIUC

Downloads from UIUC users from :

- ⋆ Center for Computational Electromagnetics
- ⋆ Siebel Center for Computer Science
- ⋆ Center for Simulation of Advanced Rockets
- ⋆ Department of Civil and Environmental Engineering
- ⋆ Department of Electrical and Computer Engineering
- ⋆ **National Center for Supercomputing Applications**

Success story (NCSA news, April 2002) : *"A taste of Teraflop"*

`http://www.ncsa.uiuc.edu/News/Access/Stories/MUMPS/`

MUMPS : Wide range of features

- ★ Symmetric or unsymmetric, definite or indefinite matrices
- ★ Real or complex, single or double precision
- ★ Assembled (distributed) or elemental matrices
- ★ Sparse, multiple right hand sides
- ★ Null pivots detection, estimate kernel
- ★ 64-bit integers support
- ★ Iterative refinement and backward error analysis
- ★ Partial factorization and Schur complement calculation
- ★ Leverages the high efficiency of Level-3 BLAS libraries
- ★ Interface for AMD, AMF, PORD, Metis/ParMetis, SCOTCH/PT-SCOTCH ordering methods
- ★ Fortran, C, MATLAB and SciLAB interfaces
- ★ out-of-core version
- ★ free of charge

Recently added features

Parallel scaling (B. Uçar et.al.)

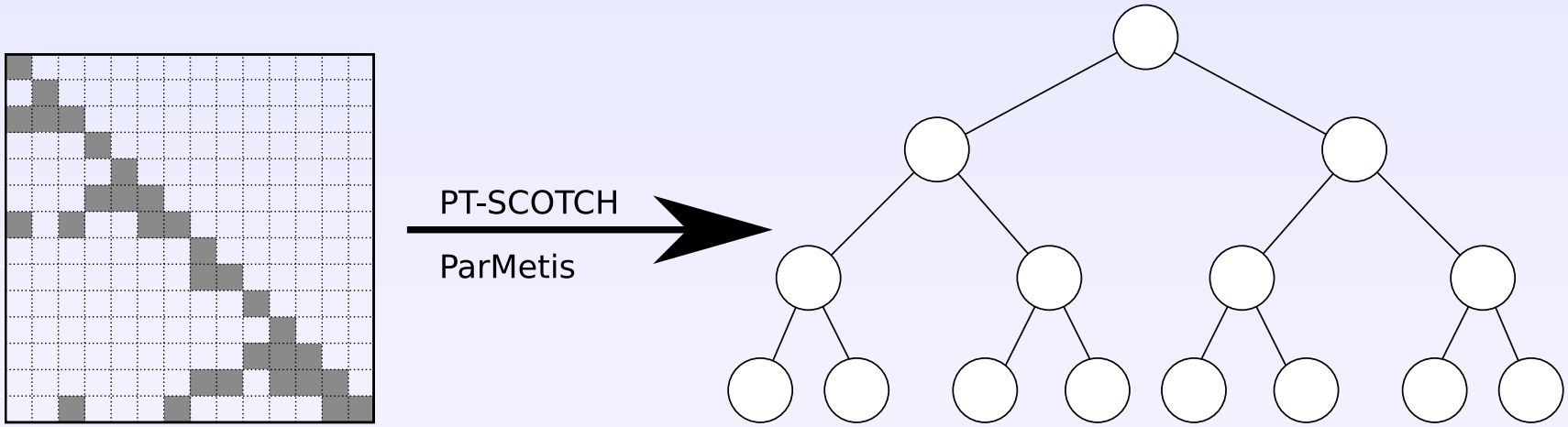
MUMPS includes a parallel implementation of an iterative scaling operation of the type

$$A^{(k+1)} = D_r^{(k)} A^{(k)} D_c^{(k)}$$

$$where \quad D_r^{(k)} = diag(\sqrt{||A_{i*}^{(k)}||_p})^{-1}, D_c^{(k)} = diag(\sqrt{||A_{*i}^{(k)}||_p})^{-1},$$

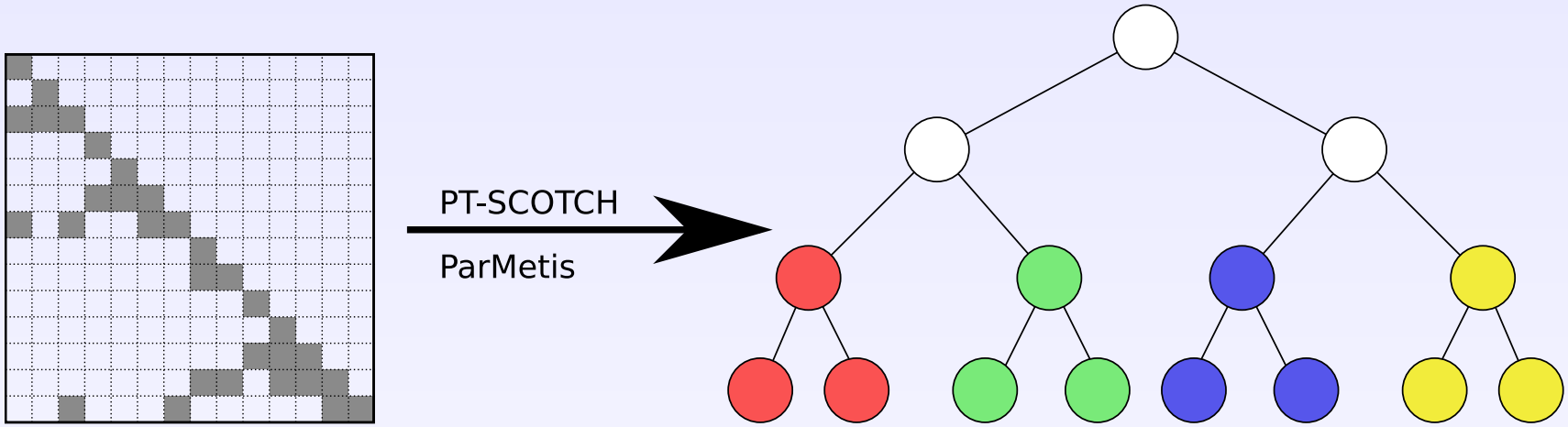
	Flops ($\times 10^6$)		# of entries in factors ($\times 10^6$) estimated		effective	
scaling	OFF	ON	OFF	ON	OFF	ON
a0nsdsil	7.7	2.5	0.42	0.42	0.57	0.42
C-54	281	209	1.42	1.42	1.76	1.58

Parallel ordering and symbolic facto (A. Buttari et.al.)



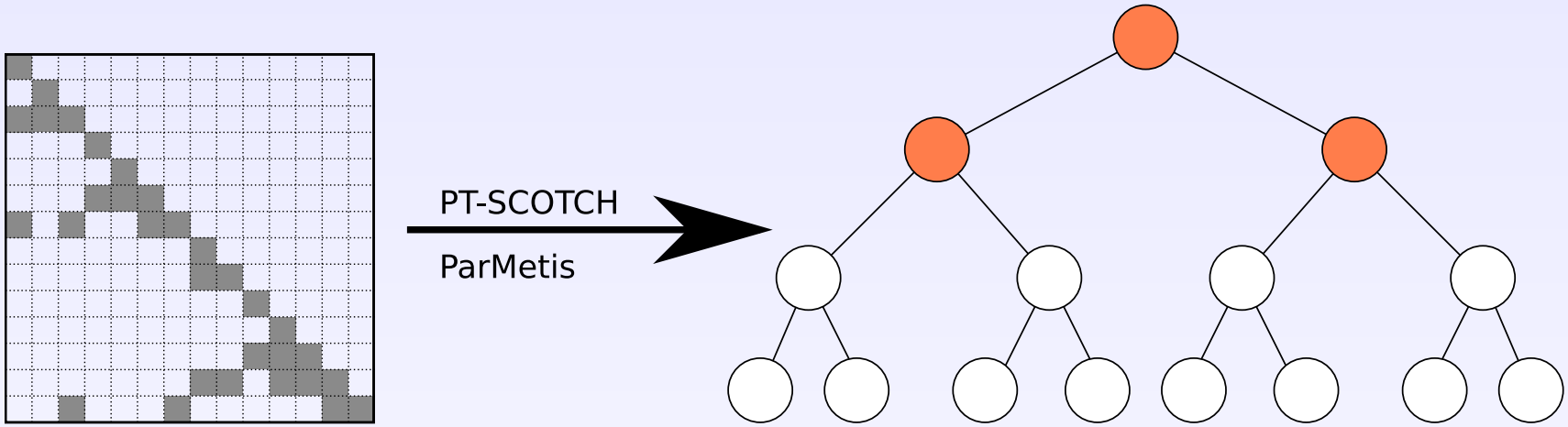
1 – First pass adjacency graph of the matrix to a parallel ordering tool (PT-SCOTCH or ParMetis). As a result, a pivotal order and a binary separators tree are returned

Parallel ordering and symbolic facto (A. Buttari et.al.)



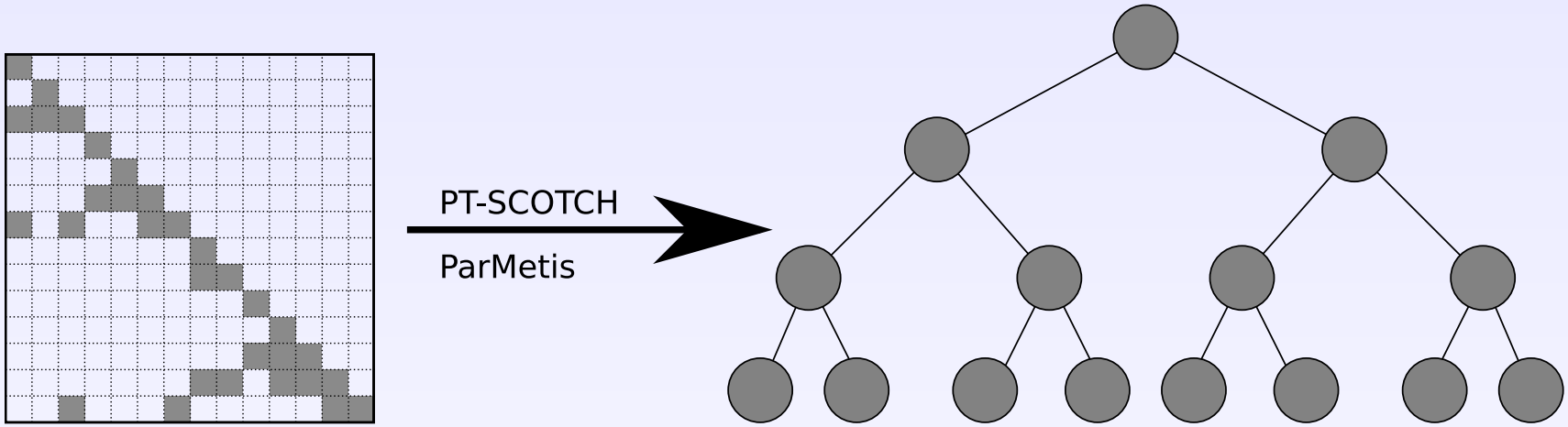
2 – Then each processor separately performs the symbolic elimination of the variables contained in a subtree. This symbolic factorization is based on the usage of quotient graphs with a restarting technique that mixes left and right looking factorization methods

Parallel ordering and symbolic facto (A. Buttari et.al.)



3 – The host processor eliminates the variables in the top part of the tree using the same technique

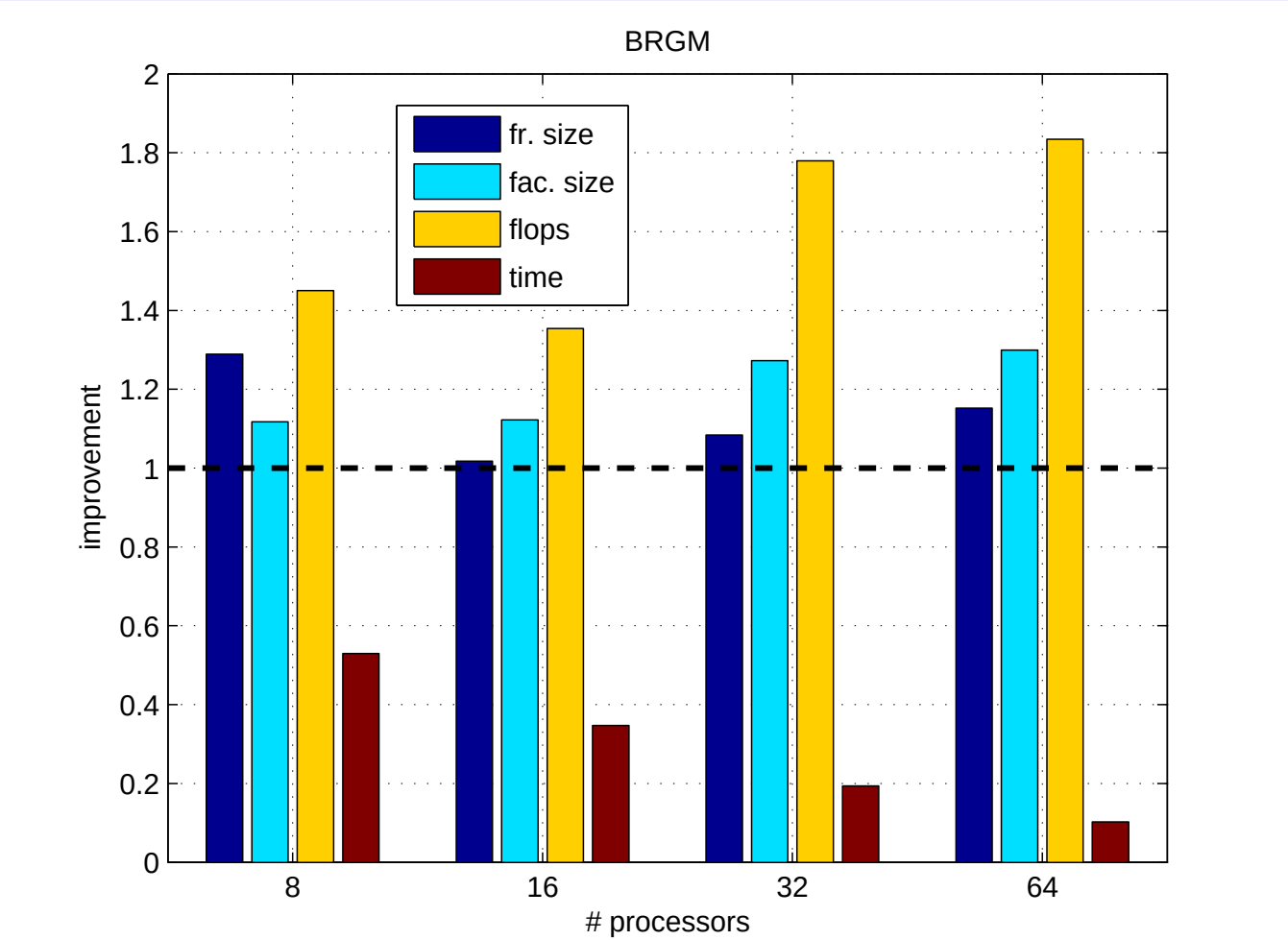
Parallel ordering and symbolic facto (A. Buttari et.al.)



4 – The distributed data are merged into a centralized data structure that is used in subsequent steps of the analysis phase like *amalgamation*, *mapping* etc.

Parallel ordering and symbolic facto (A. Buttari et.al.)

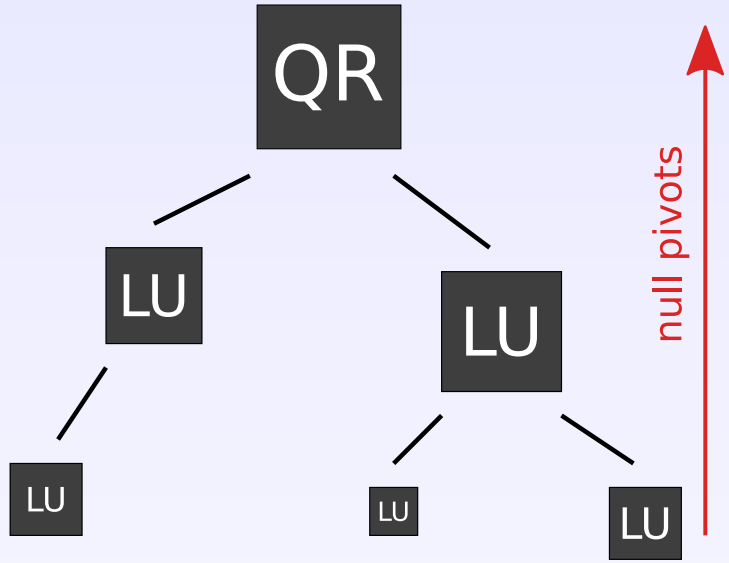
- Either PT-SCOTCH or ParMetis can be used
- Typical improvements of PT-SCOTCH vs ParMetis (3.7 million by 3.7 million matrix from BRGM)



Kernel computation

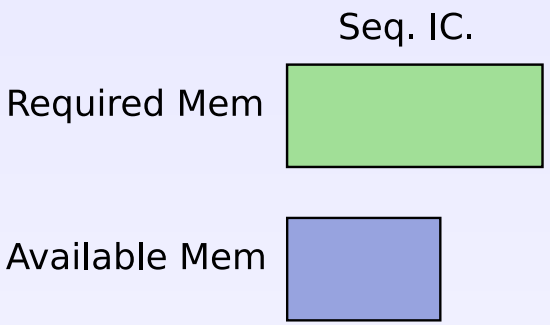
Finding the null-space X such that $AX = 0$ is equivalent to solving, for each null pivot j detected at facto time the equation

$$Ux = e_j$$

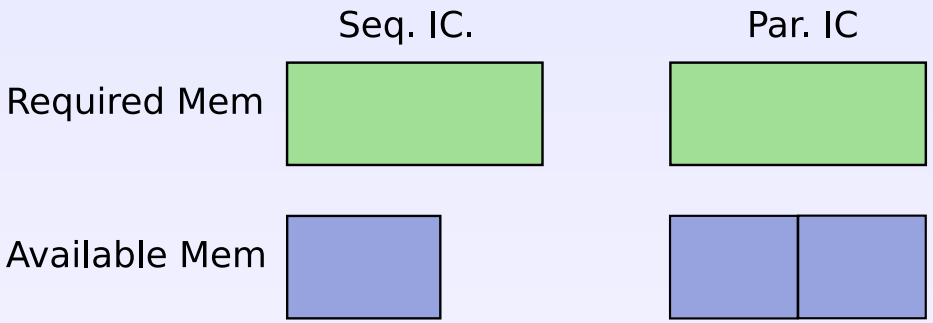


- only the nodes in the subtree rooted at node j are involved in the computation
- the accuracy of the method can be improved by delaying some pivots to the root of tree which is then factorized by means of a rank-revealing QR operation

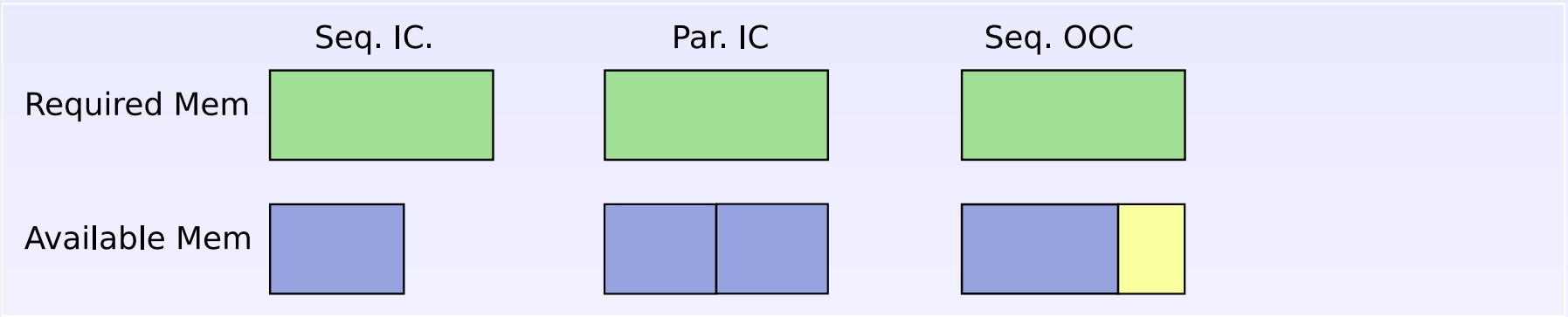
Out-Of-Core (PhD thesis of E. Agullo & M. Slavova)



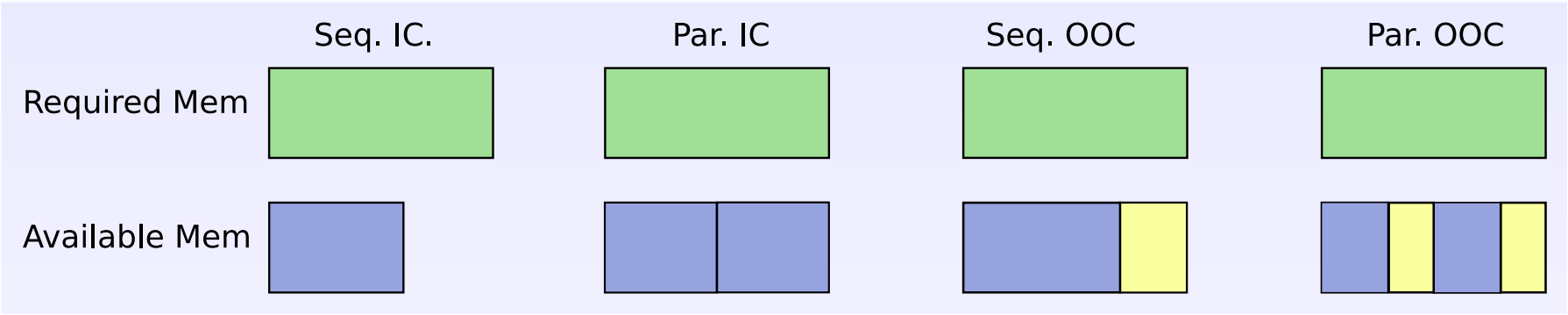
Out-Of-Core (PhD thesis of E. Agullo & M. Slavova)



Out-Of-Core (PhD thesis of E. Agullo & M. Slavova)



Out-Of-Core (PhD thesis of E. Agullo & M. Slavova)



The performance of an OOC solver strongly depends on :

- ⋆ number and speed of disk accesses
- ⋆ number of processors and volume of data per processor
- ⋆ regularity of disk accesses and scheduling of computational tasks
- ⋆ low-level implementation of I/O features

Out-Of-Core (PhD thesis of E. Agullo & M. Slavova)

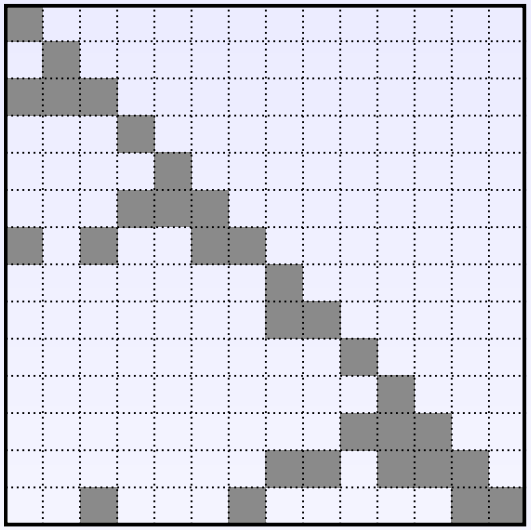
- ★ Factorization time on AMD platform

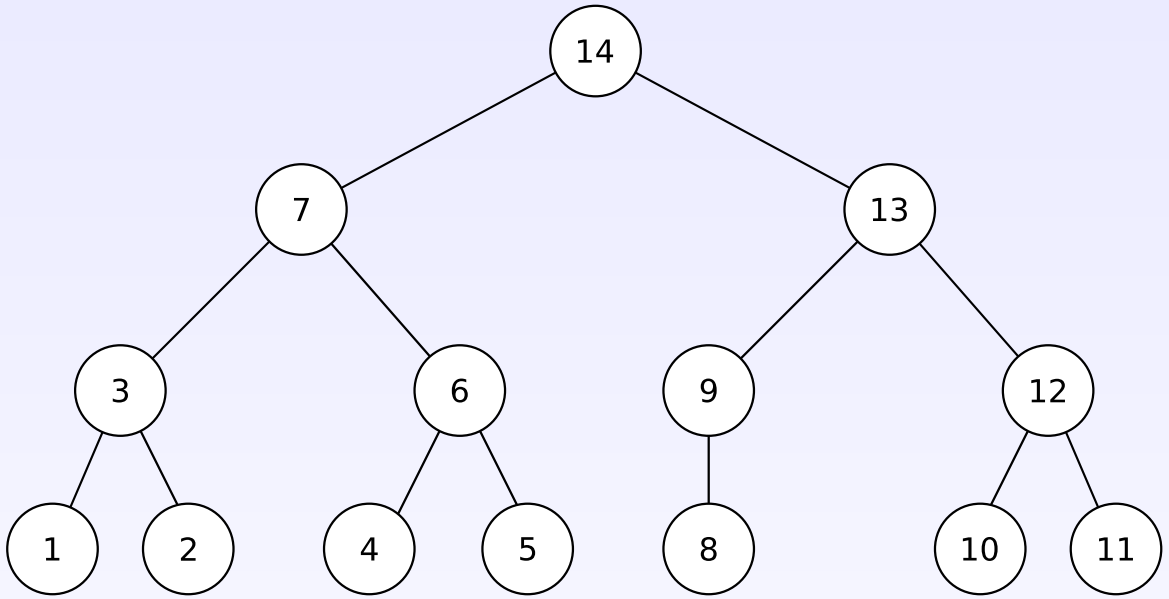
	Direct I/O		Pagecache		In Core
Matrix	Synch.	Asynch.	Synch.	Asynch.	
SHIP003	43.6	36.4	37.7	35.0	33.2
XENON2	45.4	33.8	42.1	33.0	31.9
CONESHL2	158.7	123.7	144.1	125.1	out-of-mem
QIMONDA07	159.2	89.6	190.1	171.1	out-of-mem

- ★ Solution time on CRAY XD-1

	procs	Factors size (MB)	Workspace size (MB)	Solution time
FIFO	2	2547	559	606
NNS	2	2547	559	603
FIFO	8	1512	183	368
NNS	8	1512	183	217
FIFO	32	340	42	149
NNS	32	340	42	114

OOC : Computing entries in A^{-1} (F.-H. Rouet and M. Slavova)

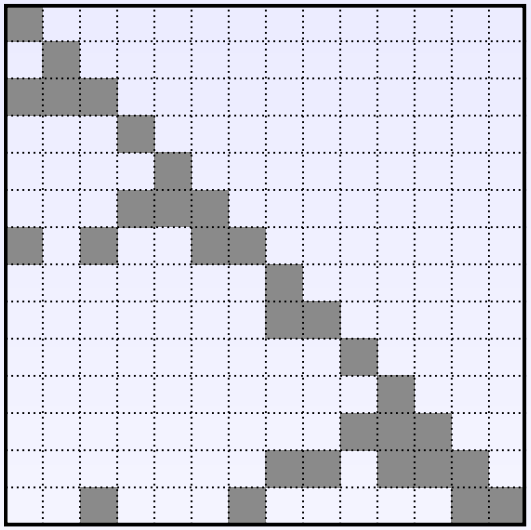


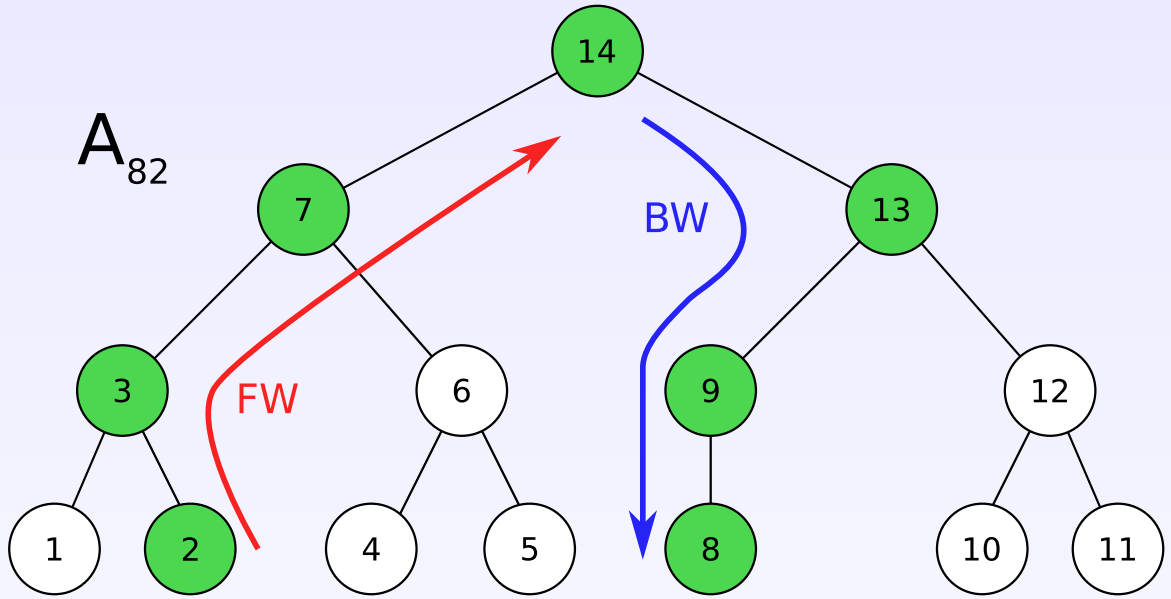


Computing columns of A^{-1} amounts to solving systems $Ax_j = e_j$. If only a small amount of elements must be computed, the computational cost can be significantly reduced by exploiting the sparsity of RHSs. Example : on an application from astrophysics (CESR, Toulouse) ; computational time for $diag(A^{-1})$ (size = 148K) :

- ★ without exploiting the sparsity : **11000 seconds**
- ★ exploiting the sparsisty : **760 seconds**

OOC : Computing entries in A^{-1} (F.-H. Rouet and M. Slavova)





Computing columns of A^{-1} amounts to solving systems $Ax_j = e_j$. If only a small amount of elements must be computed, the computational cost can be significantly reduced by exploiting the sparsity of RHSs. Example : on an application from astrophysics (CESR, Toulouse) ; computational time for $diag(A^{-1})$ (size = 148K) :

- ★ without exploiting the sparsity : **11000 seconds**
- ★ exploiting the sparsisty : **760 seconds**

OOC and Memory

Peak of active memory for multifrontal approach (factors on disk) ($\times 10^6$ entries)

	METIS	SCOTCH	PORD	AMF	AMD
GUPTA2	58.33	*289.67*	78.13	**33.61**	52.09
SHIP_003	*25.09*	23.06	20.86	**20.77**	32.02
TWOTONE	13.24	13.54	11.80	**11.63**	*17.59*
WANG3	3.28	3.84	**2.75**	3.62	*6.14*
XENON2	14.89	15.21	**13.14**	23.82	*37.82*

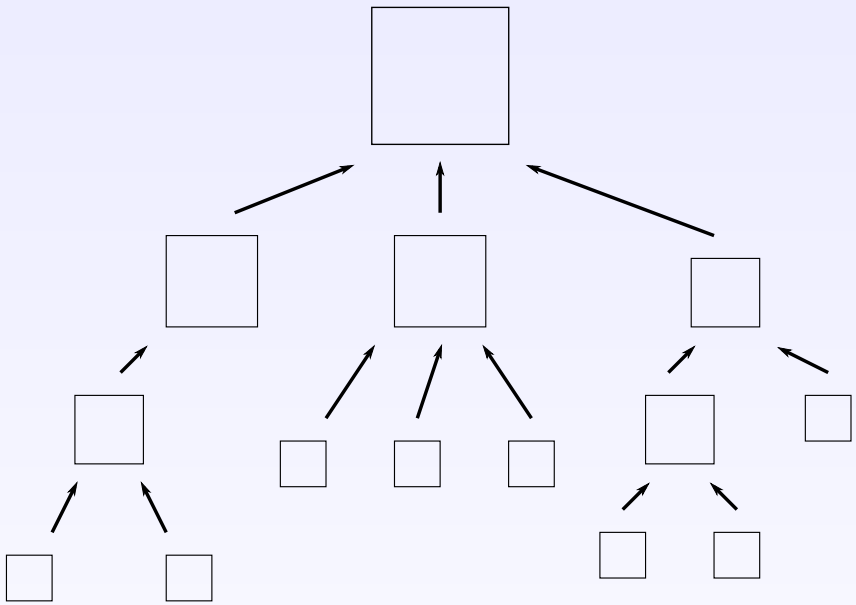
Memory Efficiency of MUMPS (with factors on disk)

Number of processors	16	32	64	128
AUDI_KW_1	0.16	0.12	0.13	0.10
CONESHL_MOD	0.28	0.28	0.22	0.19
CONV3D64	0.42	0.40	0.41	0.37
QIMONDA07	0.30	0.18	0.11	-
ULTRASOUND80	0.32	0.31	0.30	0.26

$$e(p) = \frac{\text{Sequential storage}}{p \times \max_{i=1,\ldots,p}(\text{Parallel storage(i)})}$$

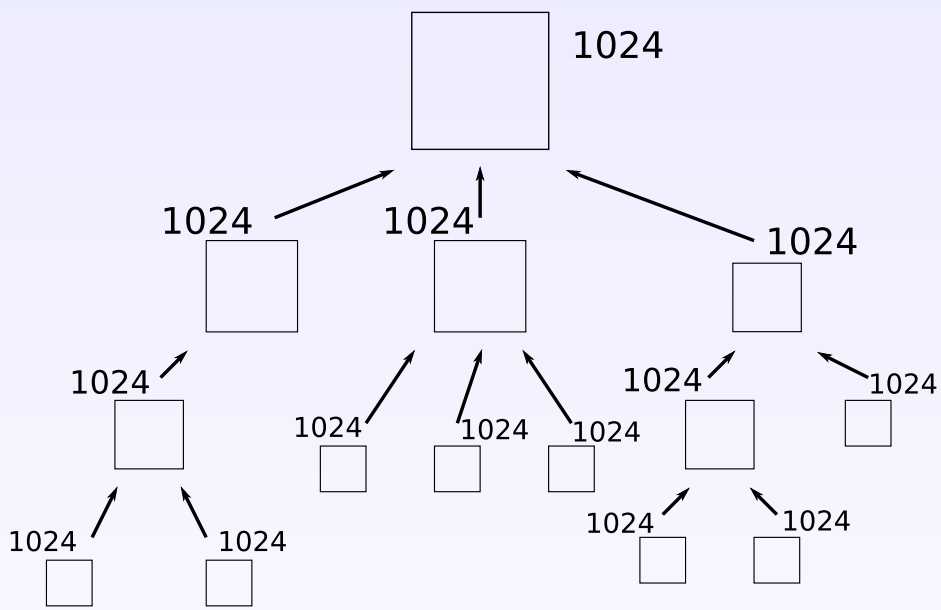
OOC and Memory Scalability

Processor-to-node mapping



OOC and Memory Scalability

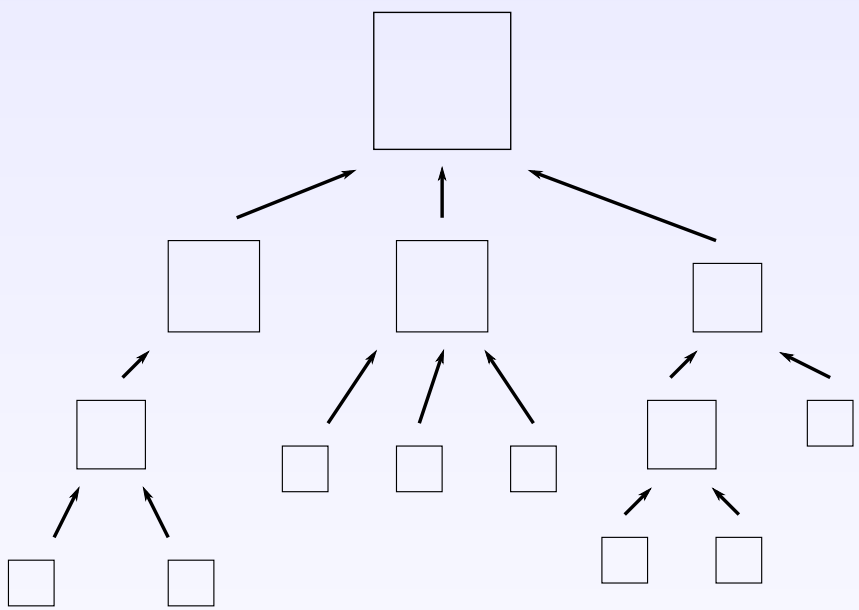
Processor-to-node mapping : All-to-one mapping



- ⋆ optimal memory scalability : $M_{seq}/nprocs$
- ⋆ poor parallelism : only intra-node parallelism is exploited

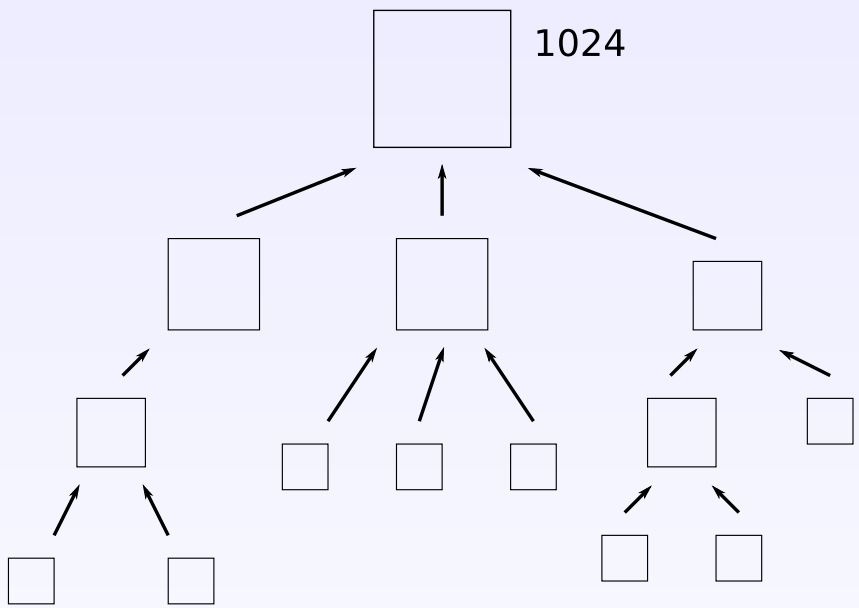
OOC and Memory Scalability

Processor-to-node mapping : Proportional Mapping

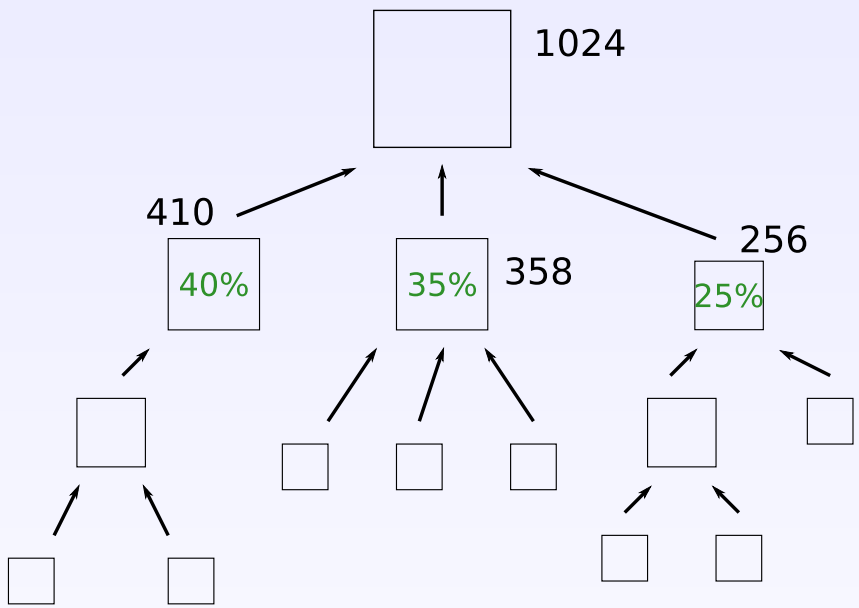


OOC and Memory Scalability

Processor-to-node mapping : Proportional Mapping

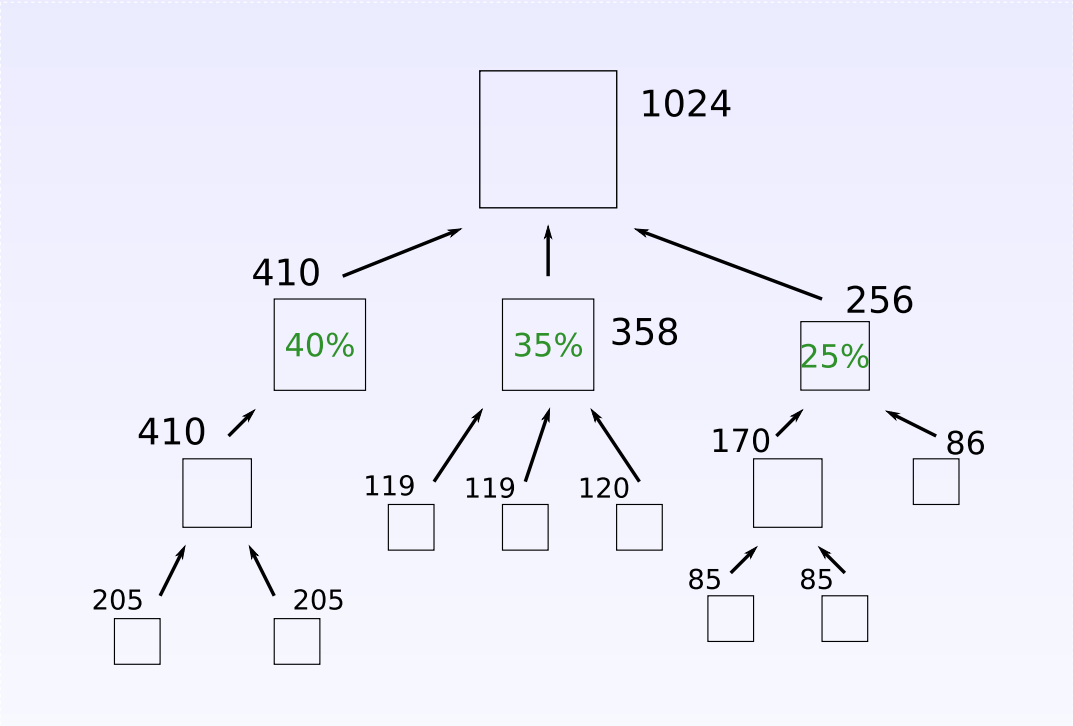


Processor-to-node mapping : Proportional Mapping



OOC and Memory Scalability

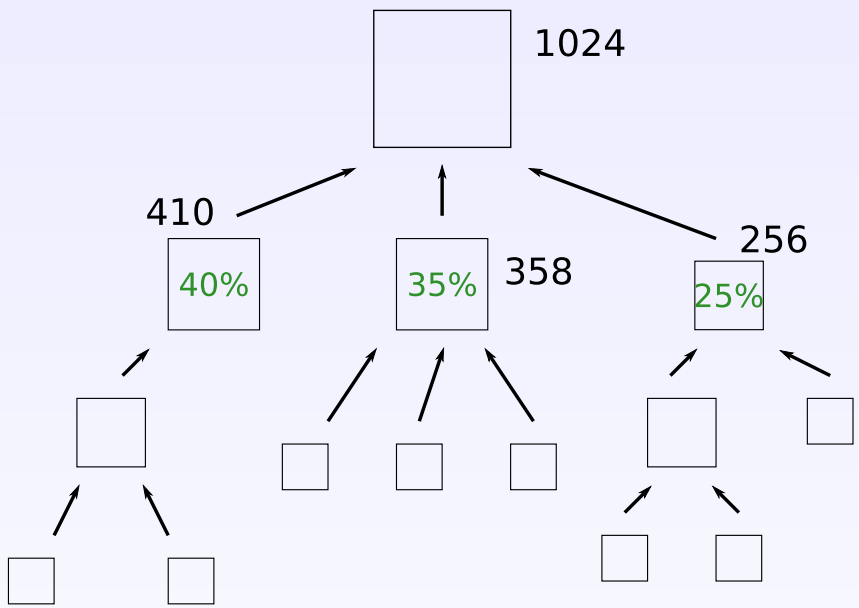
Processor-to-node mapping : Proportional Mapping



- ⋆ suboptimal memory scalability due to tree traversal order
- ⋆ good parallelism : inter-node and intra-node parallelism exploited
- ⋆ flops aware

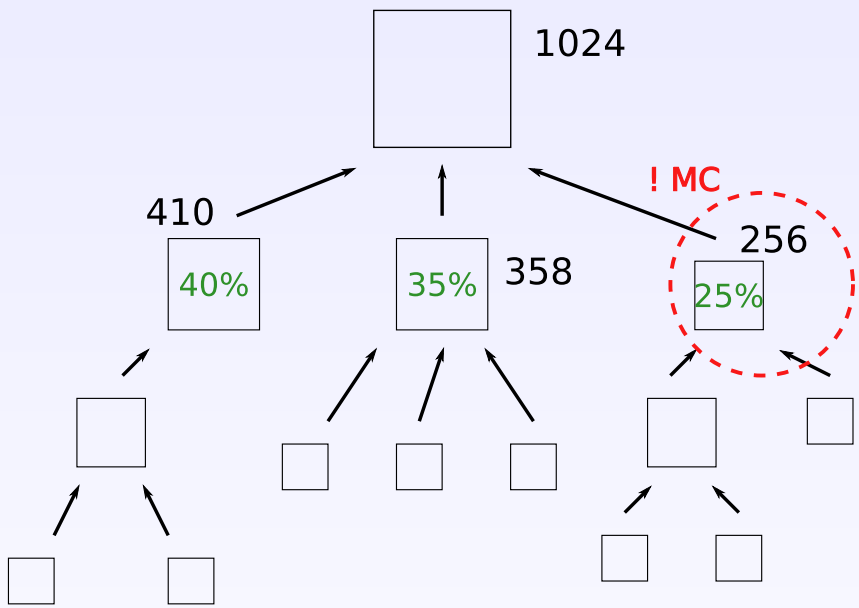
OOC and Memory Scalability

Processor-to-node mapping : Memory-aware mapping



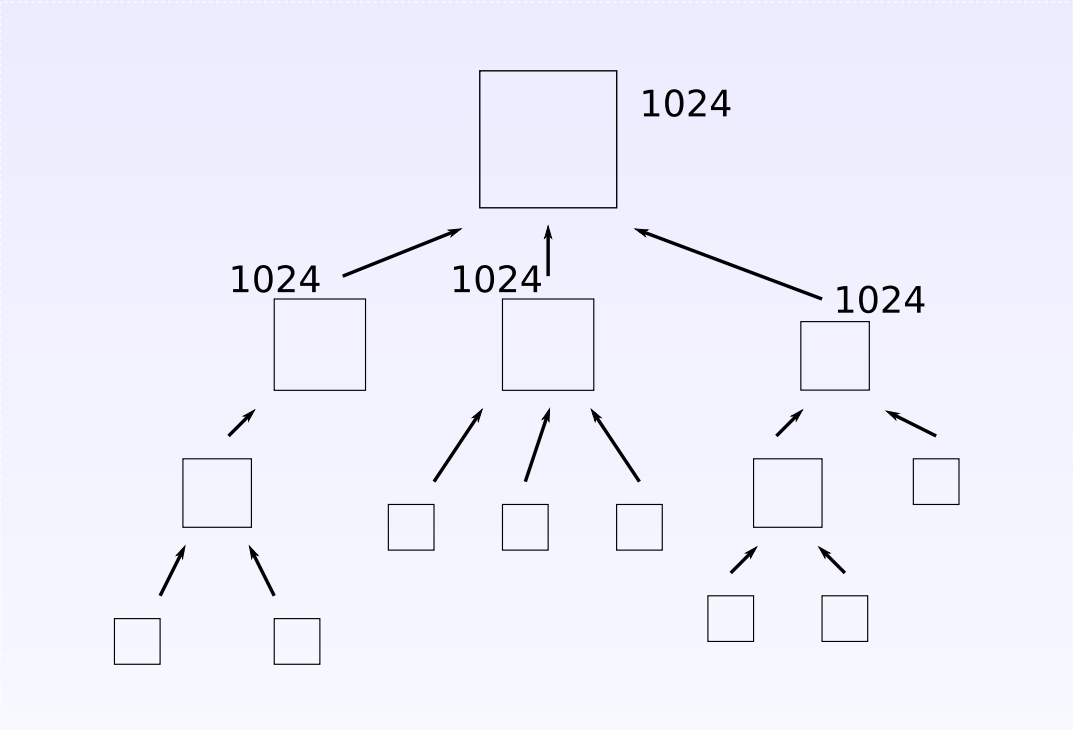
OOC and Memory Scalability

Processor-to-node mapping : Memory-aware mapping



OOC and Memory Scalability

Processor-to-node mapping : Memory-aware mapping



- ⋆ node factorizations are serialized if memory constrains are not satisfied
- ⋆ under development

Future plans

- ⋆ analyze and produce methods to optimize computation of A^{-1}
- ⋆ pursue developments around out-of-core features
- ⋆ multi-core parallelism
  - exploit mc in dense frontal matrix factorizations (not just by relying on threaded BLAS)
  - exploit mc outside dense frontal matrix factorizations (assembly operations, analysis phase)
  - keep an eye on the arrival of standards/abstractions related to gp-gpu (we are not ready yet !)
- ⋆ work on memory scalability for large numbers of processors
- ⋆ experiment with larger and larger problems

Future plans

- ⋆ analyze and produce methods to optimize computation of A^{-1}
- ⋆ pursue developments around out-of-core features
- ⋆ multi-core parallelism
  - exploit mc in dense frontal matrix factorizations (not just by relying on threaded BLAS)
  - exploit mc outside dense frontal matrix factorizations (assembly operations, analysis phase)
  - keep an eye on the arrival of standards/abstractions related to gp-gpu (we are not ready yet !)
- ⋆ work on memory scalability for large numbers of processors
- ⋆ experiment with larger and larger problems

- ⋆ research directions motivated by application needs (solving larger problems efficiently and accurately is one of the main needs)
- ⋆ progress in sparse direct solver technology should naturally benefit to hybrid direct-iterative solvers

Thanks / Merci