Habilitation à diriger des recherches: "Multifrontal Methods: Parallelism, Memory Usage, and Numerical Aspects"

Jean-Yves L'Excellent

INRIA et LIP-ENS Lyon (Université de Lyon)
http://perso.ens-lyon.fr/jean-yves.l.excellent

Mardi 25 septembre 2012

Introduction: sparse matrices and direct solvers

Example:

$$\begin{array}{rcrcrcl} 3\,x_1 & + & 2\,x_2 & & & = & 5 \\ & & 2\,x_2 & - & 5\,x_3 & = & 1 \\ 2\,x_1 & & & + & 3\,x_3 & = & 0 \end{array}$$

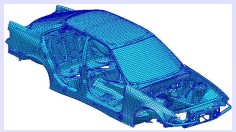
can be represented as

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

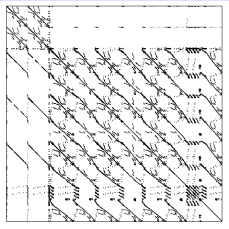
where $\mathbf{A} = \begin{pmatrix} 3 & 2 & 0 \\ 0 & 2 & -5 \\ 2 & 0 & 3 \end{pmatrix}$, $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$, and $\mathbf{b} = \begin{pmatrix} 5 \\ 1 \\ 0 \end{pmatrix}$

Sparse matrix: only nonzeros are stored
Direct solver: based on Gaussian elimination

Sparse direct solvers



Discretization of a physical problem (e.g., finite elements) $\rightarrow$ **Solution of sparse systems** $Ax = b$



Often the most expensive part in numerical simulation codes

Sparse direct methods to solve $Ax = b$:

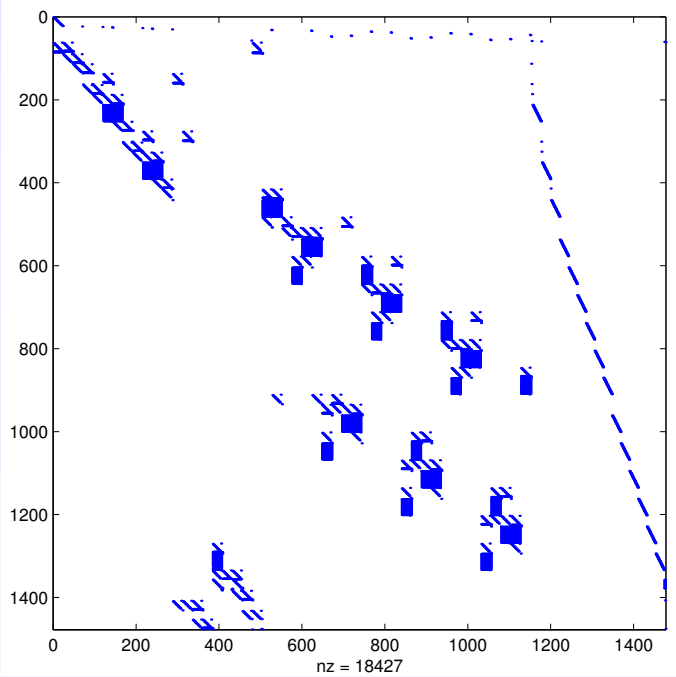
- Decompose A under the form LU,LDL^t or LL^t
- Solve the triangular systems $Ly = b$, then $Ux = y$

Black box?

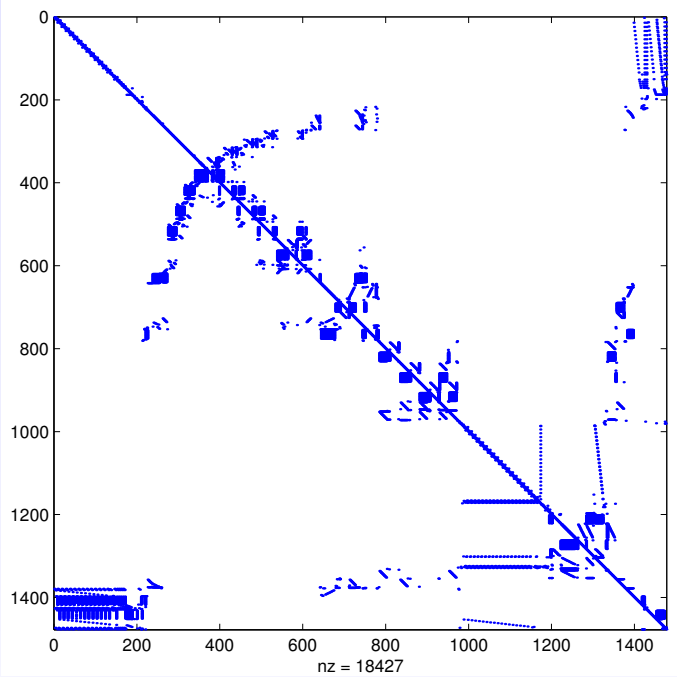
- Default (automatic/adaptive) setting of options often available
- Knowledge and setting of the preprocessing and algorithmic options can help the user to improve:
  - size of L,U factors and memory needed
  - operation count and computational time
  - numerical accuracy

Preprocessing - illustration

Original (A =lhr01)



Preprocessed matrix (A'(lhr01))



Modified problem: $A'x' = b'$ with $A' = PD_rAQD_cP^t$

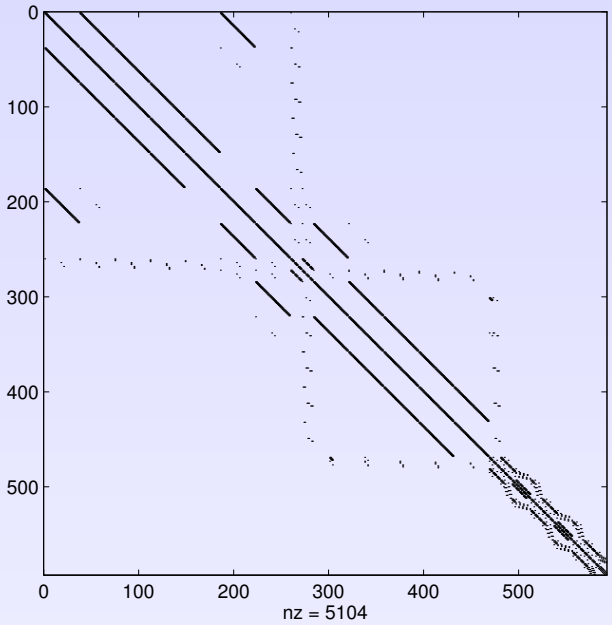
Gaussian elimination and sparsity

- Effects of permuting a matrix:

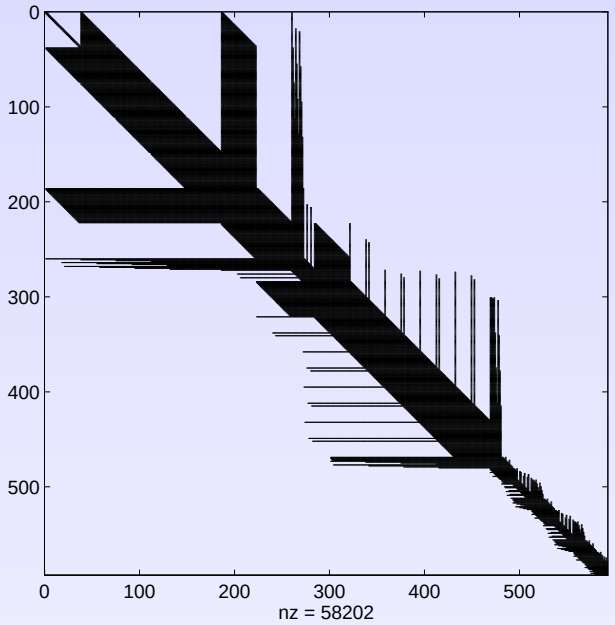
$$\begin{pmatrix} X & X & X & X & X \\ X & X & 0 & 0 & 0 \\ X & 0 & X & 0 & 0 \\ X & 0 & 0 & X & 0 \\ X & 0 & 0 & 0 & X \end{pmatrix} \longrightarrow \begin{pmatrix} X & 0 & 0 & 0 & X \\ 0 & X & 0 & 0 & X \\ 0 & 0 & X & 0 & X \\ 0 & 0 & 0 & X & X \\ X & X & X & X & X \end{pmatrix}$$

- Ordering the variables has a strong impact on
  - fill-in and memory
  - operation count and computational time
- Fill reduction is NP-complete [Yannakakis'81]
- Some heuristics: METIS, SCOTCH, PORD, AMD, AMF, RCM

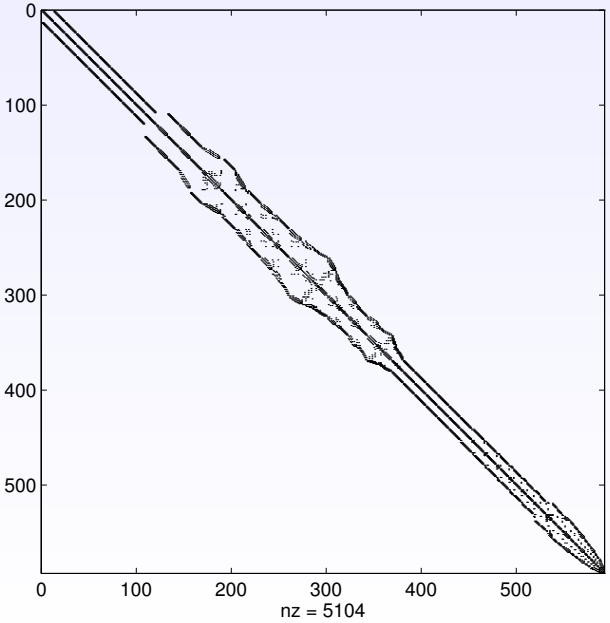
Original matrix



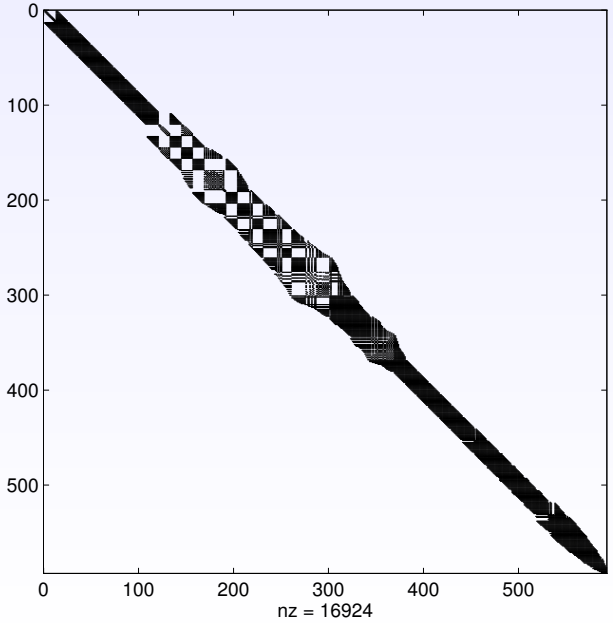
Factorized matrix (L and U factors)



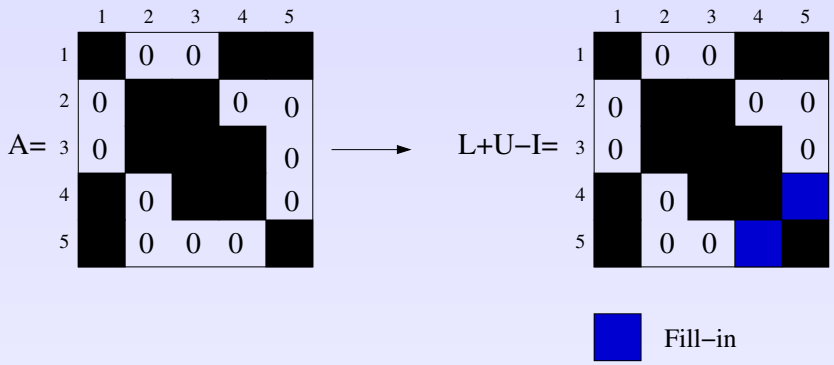
Permuted matrix (RCM)

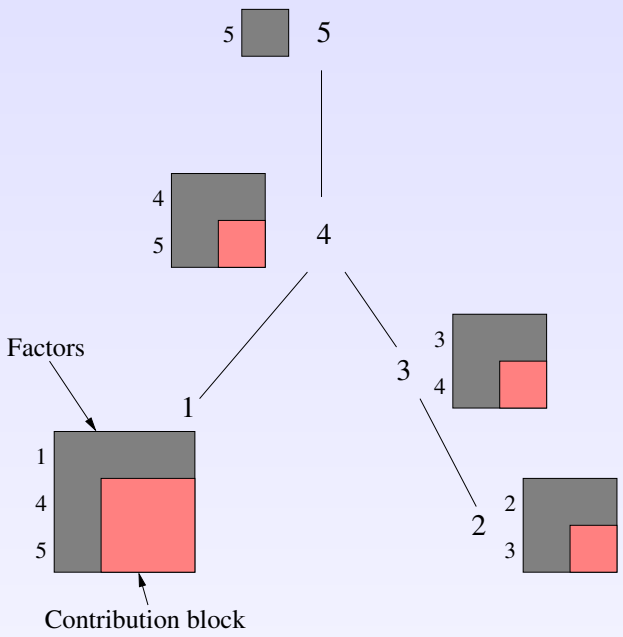


Factorized permuted matrix



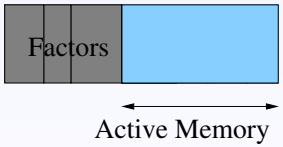
Multifrontal method [Duff Reid '83]





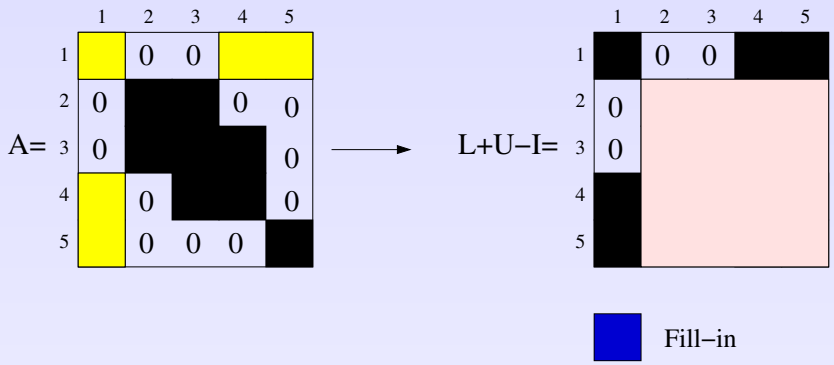
Memory is divided into two parts:

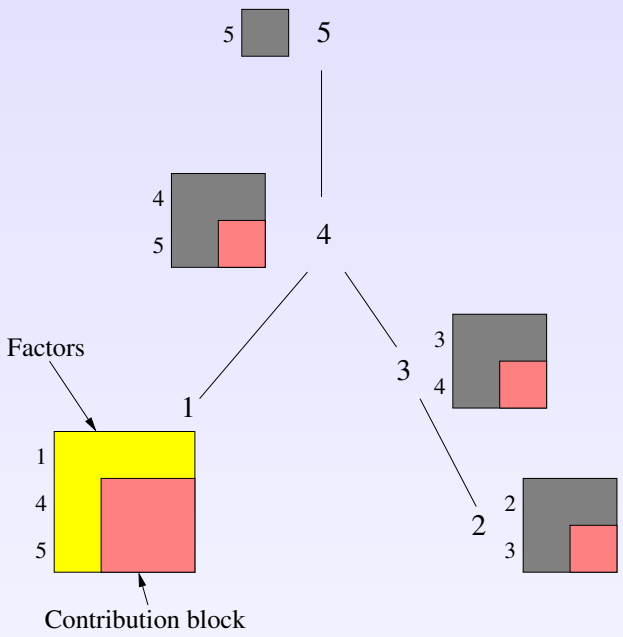
- the factors
- the active memory



Elimination tree represents dependencies between tasks

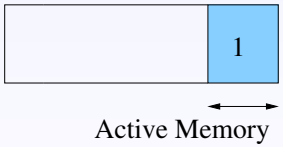
Multifrontal method [Duff Reid '83]





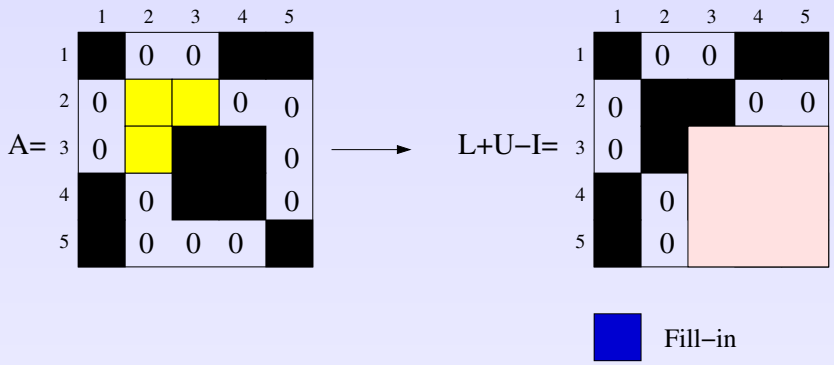
Memory is divided into two parts:

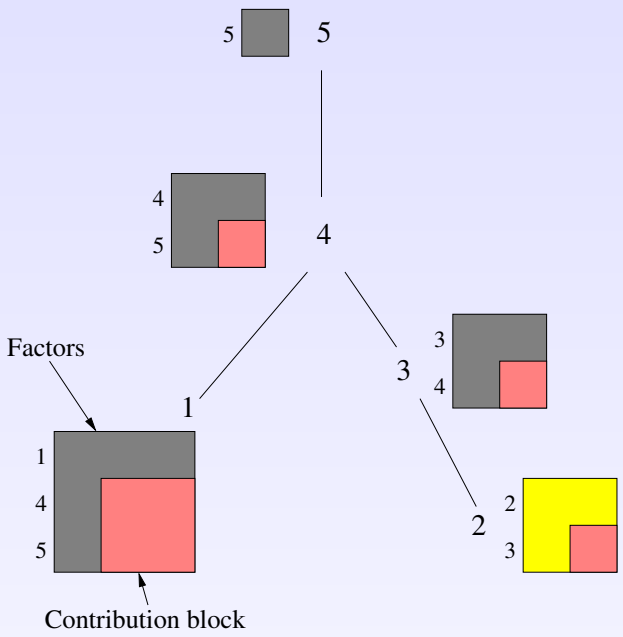
- the factors
- the active memory



Elimination tree represents dependencies between tasks

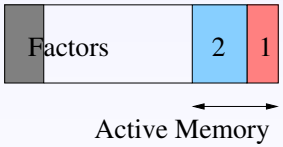
Multifrontal method [Duff Reid '83]





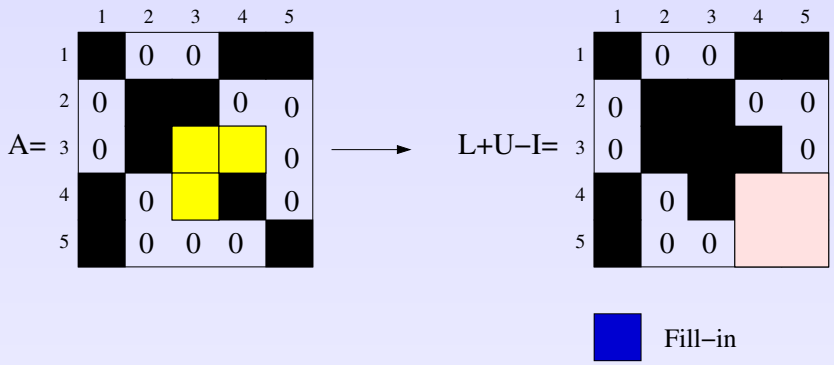
Memory is divided into two parts:

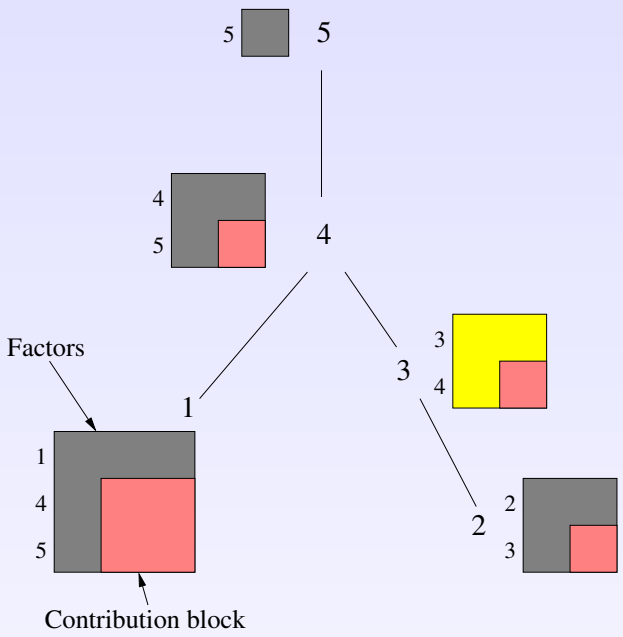
- the factors
- the active memory



Elimination tree represents dependencies between tasks

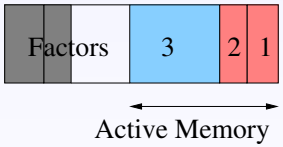
Multifrontal method [Duff Reid '83]





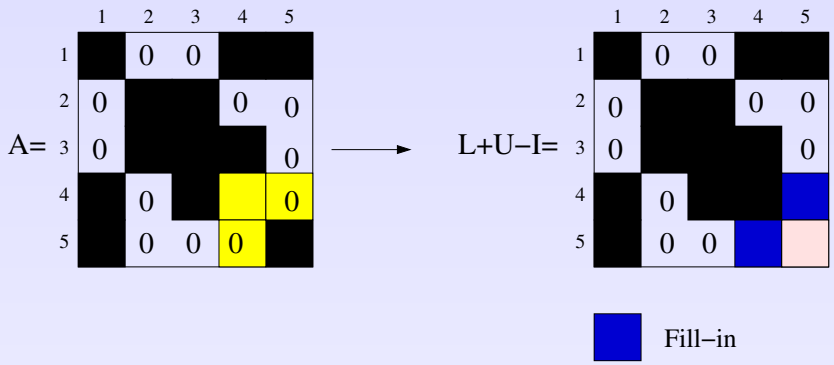
Memory is divided into two parts:

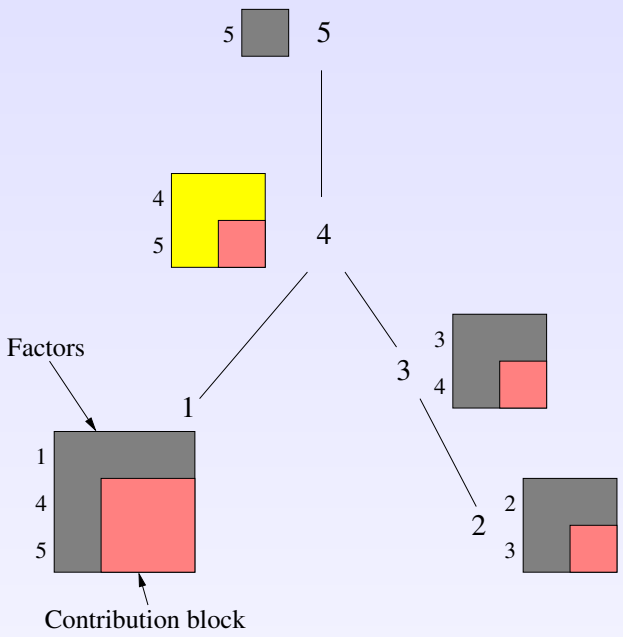
- the factors
- the active memory



Elimination tree represents dependencies between tasks

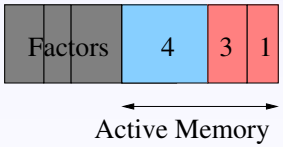
Multifrontal method [Duff Reid '83]





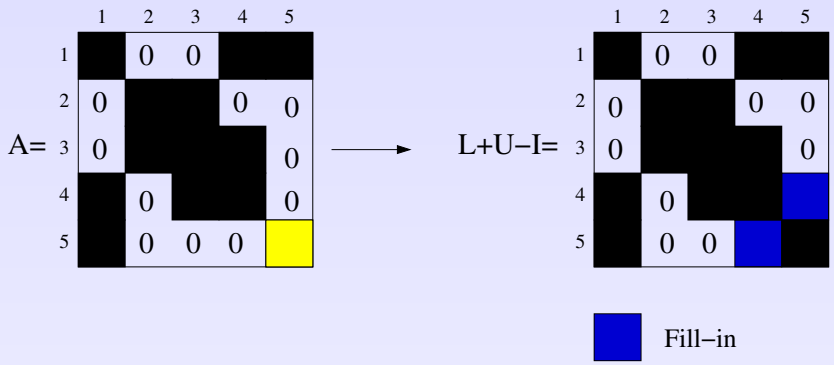
Memory is divided into two parts:

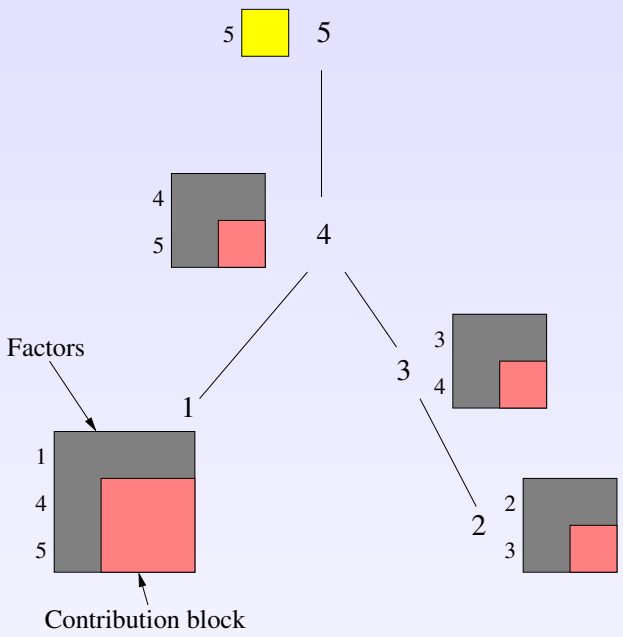
- the factors
- the active memory



Elimination tree represents dependencies between tasks

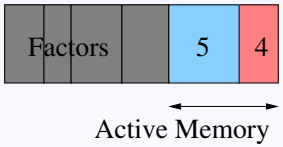
Multifrontal method [Duff Reid '83]





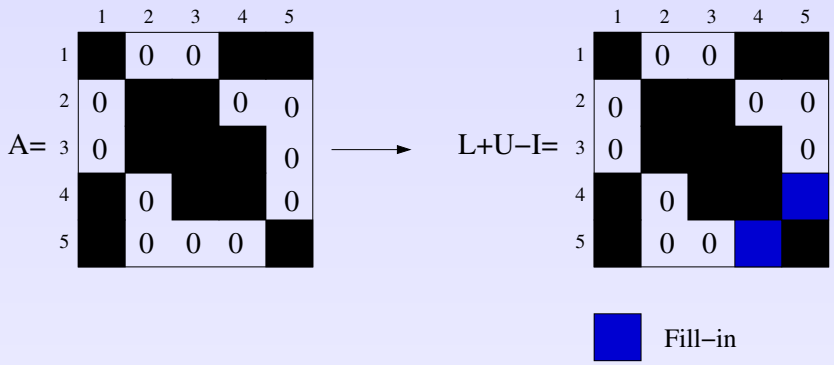
Memory is divided into two parts:

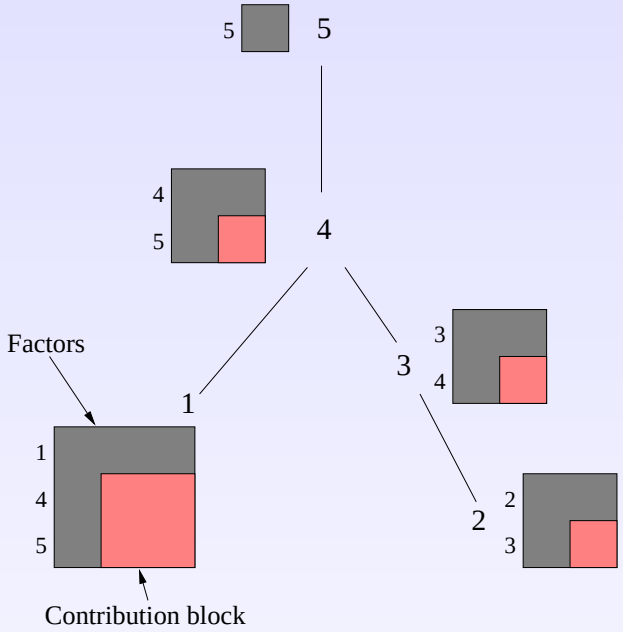
- the factors
- the active memory



Elimination tree represents dependencies between tasks

Multifrontal method [Duff Reid '83]





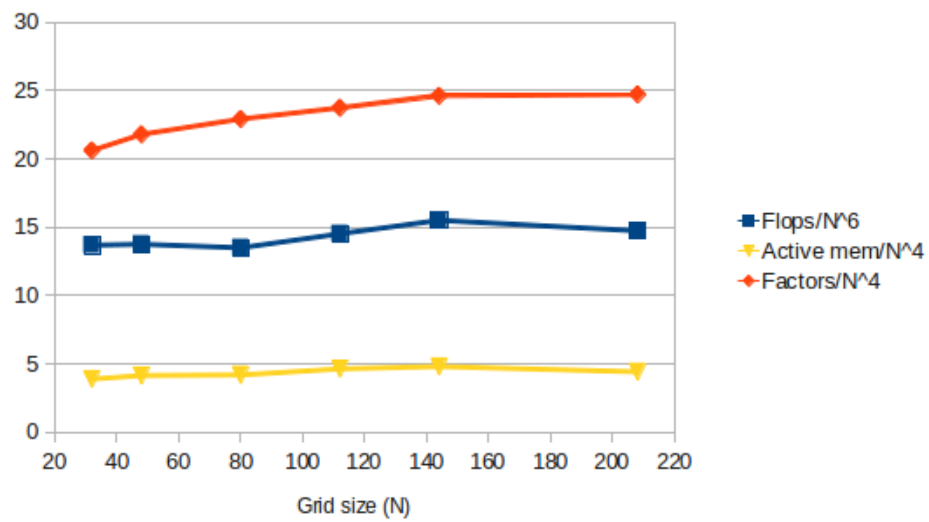
Memory is divided into two parts:

- the factors
- the active memory

Elimination tree represents dependencies between tasks

Complexity of sparse direct methods

Regular problems (nested dissections)	2D $N \times N$ grid	3D $N \times N \times N$ grid
Nonzeros in original matrix	$\Theta(N^2)$	$\Theta(N^3)$
Nonzeros in factors	$\Theta(N^2 log\ n)$	$\Theta(N^4)$
Floating-point ops	$\Theta(N^3)$	$\Theta(N^6)$



3D example in earth science:
acoustic wave propagation,
27-point finite difference grid

Current goal [Operto Virieux]:
LU on complete earth

Extrapolation on a $1000 \times 1000 \times 1000$ grid:

15 exaflops, 200 Tbytes for factors, 32 TBytes for active memory!

Too big (but we are far from exploiting all cores of exascale computers)

Complexity of sparse direct methods

Regular problems (nested dissections)	2D $N \times N$ grid	3D $N \times N \times N$ grid
Nonzeros in original matrix	$\Theta(N^2)$	$\Theta(N^3)$
Nonzeros in factors	$\Theta(N^2 log\ n)$	$\Theta(N^4)$
Floating-point ops	$\Theta(N^3)$	$\Theta(N^6)$

Evolution of computers and applications needs $\Rightarrow$ considerable efforts on algorithms and code development: parallelism, avoid synchronizations, mapping irregular data structures, scheduling for memory/for performance, memory scalability, out-of-core storage, exploit multicore architectures, low-rank representations of dense submatrices, ...

Goal of the talk: present some facets of this work in the context of the MUMPS project/solver

Outline

A parallel asynchronous multifrontal approach

Overview of MUMPS project, 1996-2012

Memory usage and task scheduling

Shared-memory parallelism (work-in-progress)

Experience in distributing a parallel sparse direct solver

Concluding remarks

Outline

A parallel asynchronous multifrontal approach

Overview of MUMPS project, 1996-2012

Memory usage and task scheduling

Shared-memory parallelism (work-in-progress)

Experience in distributing a parallel sparse direct solver

Concluding remarks

A parallel asynchronous multifrontal method

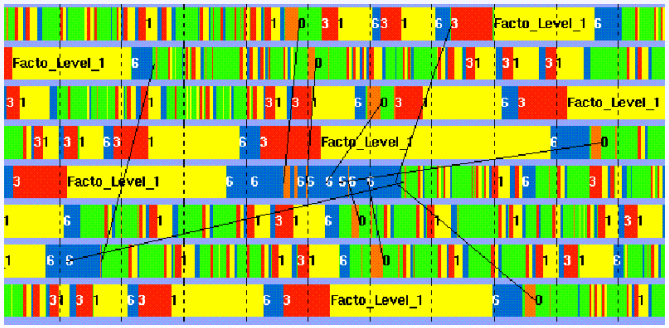
First developed in the framework of European project PARASOL (1996-1999)

Objectives:

- Provide direct methods to industrial partners
- Address distributed-memory clusters
- Allow for numerical pivoting → dynamic data structures

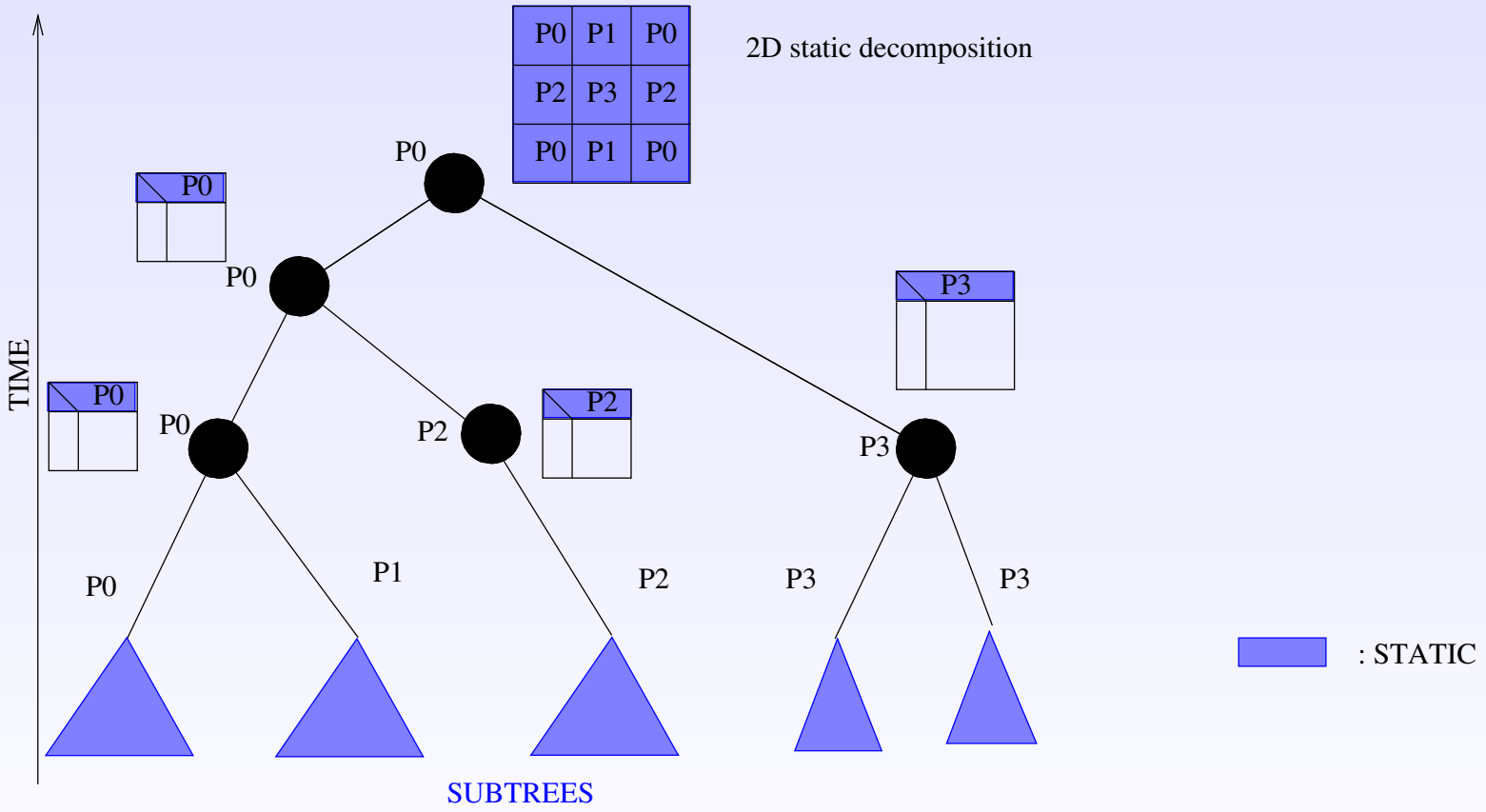
Management of parallelism

- MPI-based (distributed-memory clusters)
- Dynamic and asynchronous approach, with distributed, dynamic schedulers



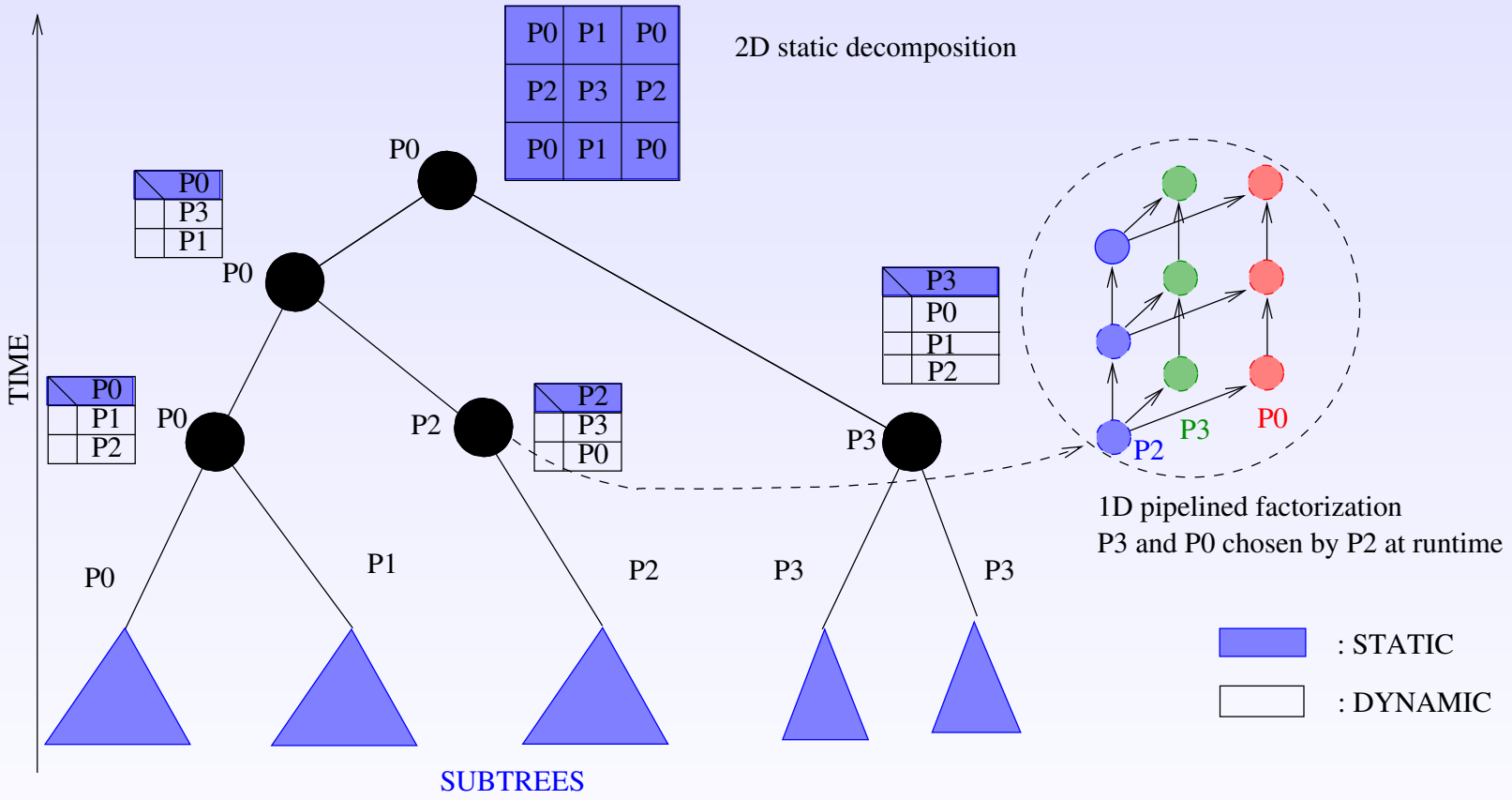
Dynamic scheduling

- Task graph = tree
- Each task = partial factorization and update of a dense matrix
- Most parallel tasks defined and mapped at runtime (typically 80 %)



Dynamic scheduling

- Task graph = tree
- Each task = partial factorization and update of a dense matrix
- Most parallel tasks defined and mapped at runtime (typically 80 %)



Distributed dynamic scheduling

Main algorithm (on each processor):

while (! global termination) **do**
  if load information is ready-to-be-received **then**
    Receive it and update load estimates
  else if another message is ready-to-be-received **then**
    Receive and process it (new ready task, contributions, ...)
  else
    Select a new local ready task (if any)
    If task is large, select workers to help
  end if
end while

- priority to message receptions
- estimate processors load
- Two schedulers, on each processor: task selection, worker selection

Outline

Birth: the PARASOL project

Previously-described method [Amestoy Duff L'Excellent, CMAME'99] [Amestoy Duff Koster L'Excellent, TOMS'01]

- funded by European project PARASOL (1996-1999), 12 partners from 5 countries
- started from PVM code from [Amestoy Espirat'96]
- itself inspired by MUPS code [Amestoy Duff'94]

Tight roadmap (MUMPS MUltifrontal Massively Parallel Solver)

- 22 internal releases, for example:

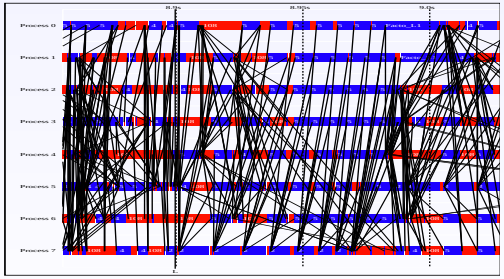
[Feb'98] v. 2.0: asynchronous code with tree and node parallelism
[Jun'98] v. 2.1.0: first symmetric solver (SPD matrices)
[Sep'98] v. 2.2.0: general symmetric solver
[Feb'99] v. 3.1.0: unassembled/element-entry (sym. + unsym.)
[Mar'99] v. 3.2.0: distributed-entry
[May'99] v. 4.0: Schur complement
[Sep'99] MUMPS 4.0.4: first public domain version

1999-2000: collaboration with Berkeley (Sherry Li)

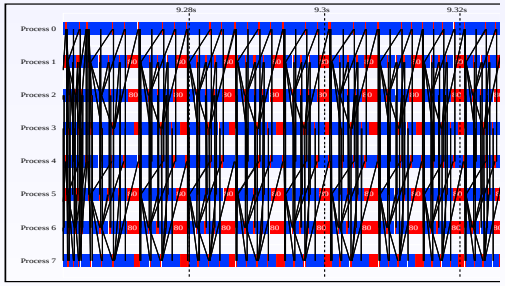
- July 1999: first experiments on Cray T3E from Lawrence Berkeley National Laboratory (France-Berkeley fund)
- 1999-2000: sabbatical of P. Amestoy in Berkeley; J.-Y. L'Excellent at NAG Ltd, UK

Comparison and improvements of SuperLU (Supernodal) and MUMPS (Multifrontal) approaches [Amestoy Duff L'Excellent Li TOMS'01]

- static vs. numerical pivoting
- effects of weighted matching algorithms ([Duff Koster'01])
- performance and scalability
- *Impact of the Implementation of MPI Point-to-Point Communications on the Performance of Two General Sparse Solvers* [Amestoy Duff L'Excellent Li PARCO'03]



MUMPS

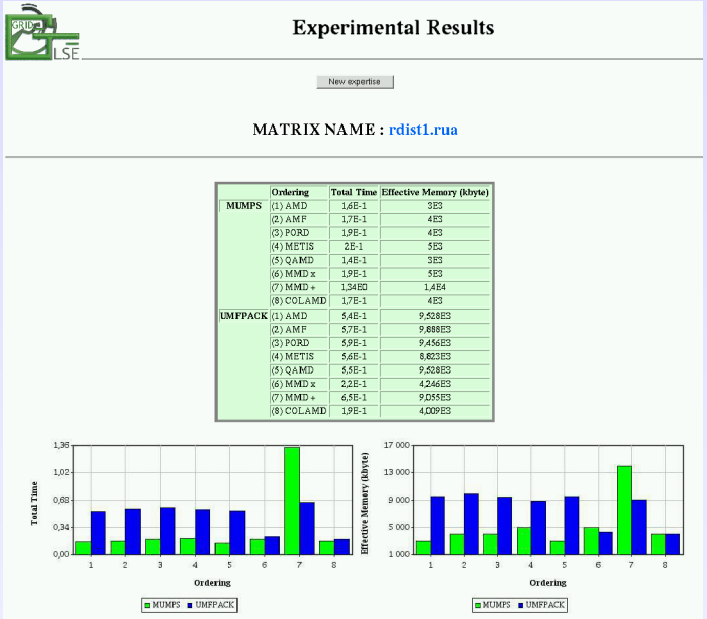


SuperLU_DIST

2001: Inria recruitment in F. Desprez's team

First experiments of Grid Computing

- client-server prototype, using DIET middleware
- GRID TLSE project [Puglisi et al., 2002-], an expertise site for sparse linear algebra (`http://gridtlse.org`)

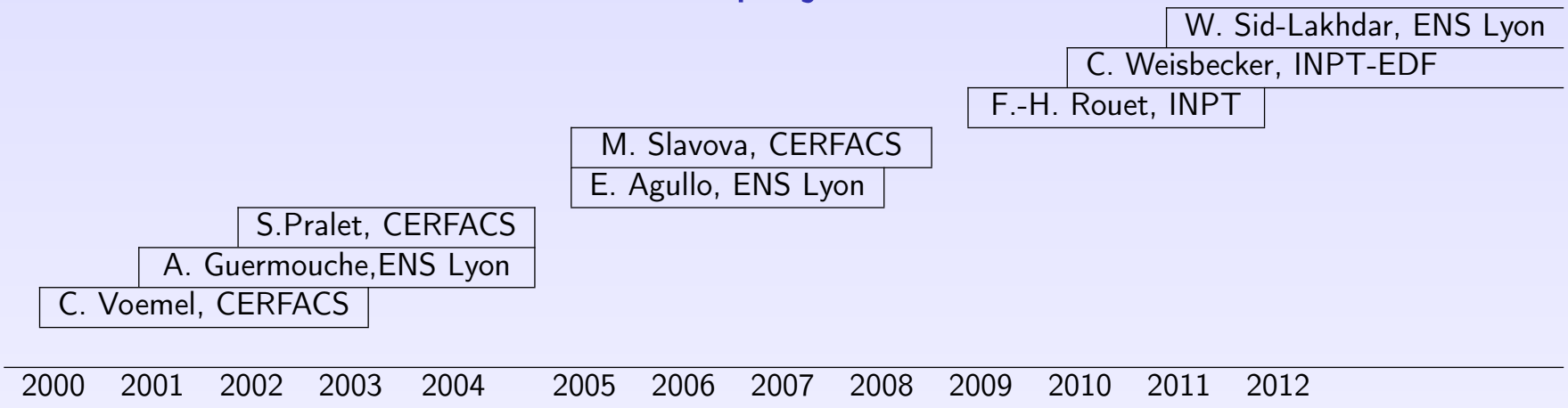


Some research projects with industry (2002-now):

- Samtech [2005-2006,2009-2010], CNES [2006], EDF [2010-2013]
- ACI Grid-TLSE [2002-2005], ANR Solstice project [2007-2010]
- Example of realizations for very large, distributed matrices (ANR Solstice project):
  - parallel scaling algorithms [Amestoy Duff Ruiz Uçar '08]
  - parallel symbolic factorization [Amestoy Buttari L'Excellent '10]

2000-2012: Research through PhD's

Ph.D. students connected to the project:

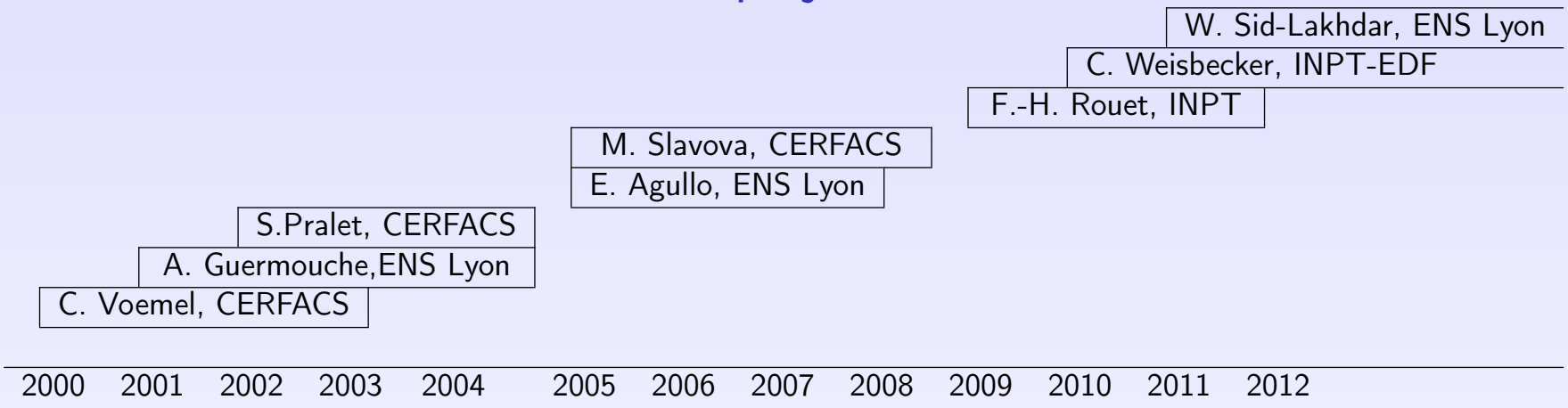


Some research themes: Preprocessing and orderings, Numerical pivoting and accuracy, Numerical functionalities, Memory usage and task scheduling, Shared-memory parallelism

Closely related to MUMPS: Experiment and validate research

2000-2012: Research through PhD's

Ph.D. students connected to the project:



Some research themes: Preprocessing and orderings, Numerical pivoting and accuracy, Numerical functionalities, Memory usage and task scheduling, Shared-memory parallelism

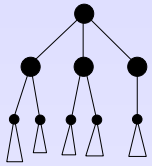
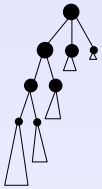
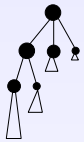
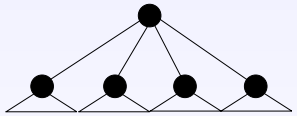
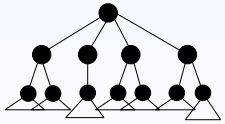
Closely related to MUMPS: Experiment and validate research

Outline

Memory usage: impact of ordering heuristic

Peak of active memory ($\times 10^6$ entries)

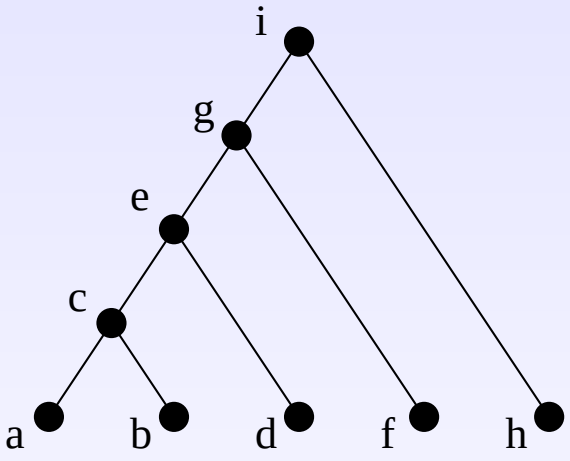
	METIS	SCOTCH	PORD	AMF	AMD
gupta2	58.33	*289.67*	78.13	**33.61**	52.09
ship_003	*25.09*	23.06	20.86	**20.77**	32.02
twotone	13.24	13.54	11.80	**11.63**	*17.59*
wang3	3.28	3.84	**2.75**	3.62	*6.14*
xenon2	14.89	15.21	**13.14**	23.82	*37.82*

Ordering heuristic	Shape of the tree	Observations
AMD		▸ Deep well-balanced ▸ Large frontal matrices on top
AMF		▸ Very deep unbalanced ▸ Small frontal matrices
PORD		▸ Deep unbalanced ▸ Small frontal matrices
SCOTCH		▸ Very wide well-balanced ▸ Large frontal matrices
METIS		▸ Wide well-balanced ▸ Smaller frontal matrices (than SCOTCH)

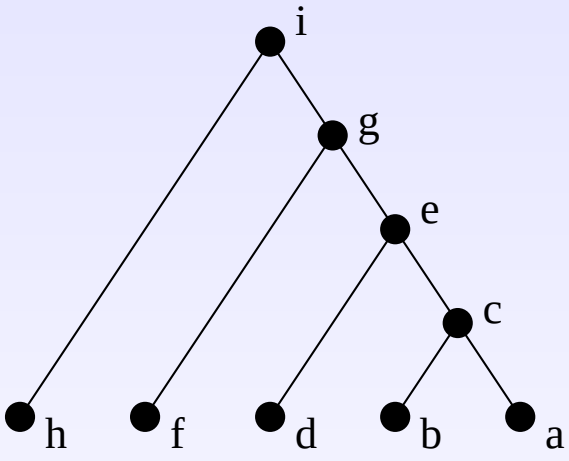
[Guermouche L'Excellent Utard'03]

Memory: impact of tree traversal

Best (abcdefghi)

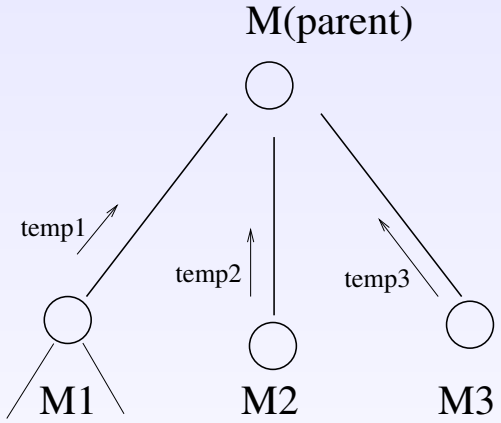


Worst (hfdbacegi)



Memory-minimizing tree traversals [Liu'87]

- Assumption: parents are processed as soon as all children have completed (postorder)
- M_i: memory peak for complete subtree rooted at i
- $temp_i$: temporary memory produced by node i
- m: memory for storing the frontal matrix of the parent



$$M_{parent} = \max\left(\max_{i=1}^{nbchildren}\left(M_i + \sum_{k=1}^{i-1} temp_k\right), m + \sum_{j=1}^{nbchildren} temp_j\right)$$

Memory-minimizing tree traversals

$$M_{parent} = \max(\max_{i=1}^{nbchildren}(M_i + \sum_{k=1}^{i-1} temp_k), m + \sum_{j=1}^{nbchildren} temp_j)$$

Theorem (Liu'87).

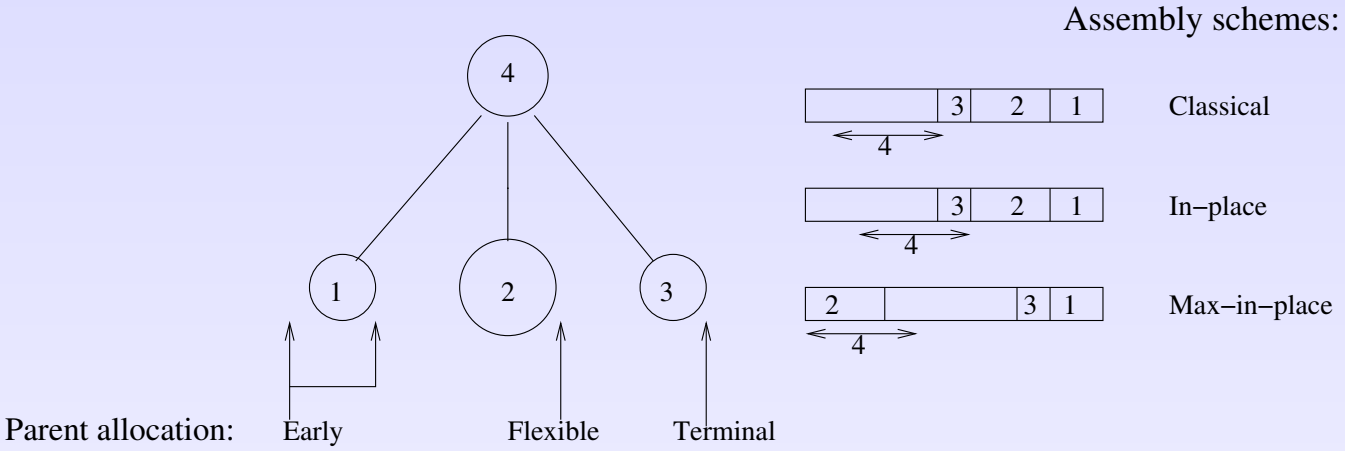
The minimum of $\max_j(x_j + \sum_{i=1}^{j-1} y_i)$ is obtained when the sequence (x_i, y_i) is sorted in decreasing order of $x_i - y_i$.

Corollary

An optimal child sequence is obtained by rearranging the children nodes in decreasing order of $M_i - temp_i$.

Interpretation: At each level of the tree, child with relatively large peak of memory in its subtree (M_i large with respect to $temp_i$) should be processed first.

Postorders and multifrontal variants



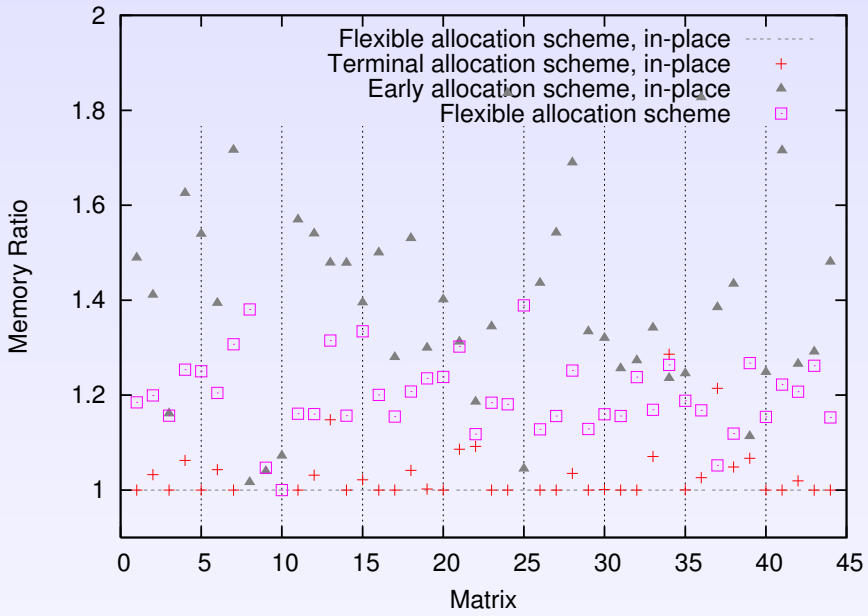
Minimize an objective function:

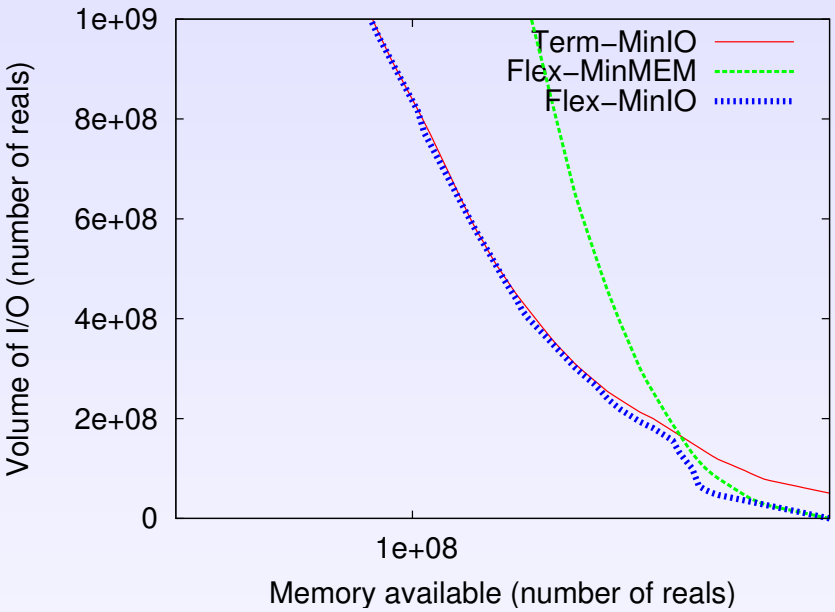
- Total memory: factors + active memory
- Active memory: current front + contribution blocks
- Volume of I/O (disk traffic), depending on available memory

[Phd Guermouche'01-'04, Phd Agullo'05-'08]:

- New variants, with associated memory-management
- Study combinations (assembly/parent allocation/objective)
- Propose optimal postorders or (in NP-hard cases) heuristics

Illustration for flexible allocation scheme (allocate parent after arbitrary number of children)





Flexible parent allocation, in-place
Objective: minimize active memory
Set of 45 matrices, METIS

Flexible allocation scheme, in-place
Objective: minimize I/O
Matrix MHD1, PORD

What about the parallel case?

In our scheme, two schedulers, on each processor:

- Task selection: which tree node should I process next?
- Workers selection: who should I choose to help me?

Objectives: performance, memory usage, accuracy of memory predictions

Object of many studies:

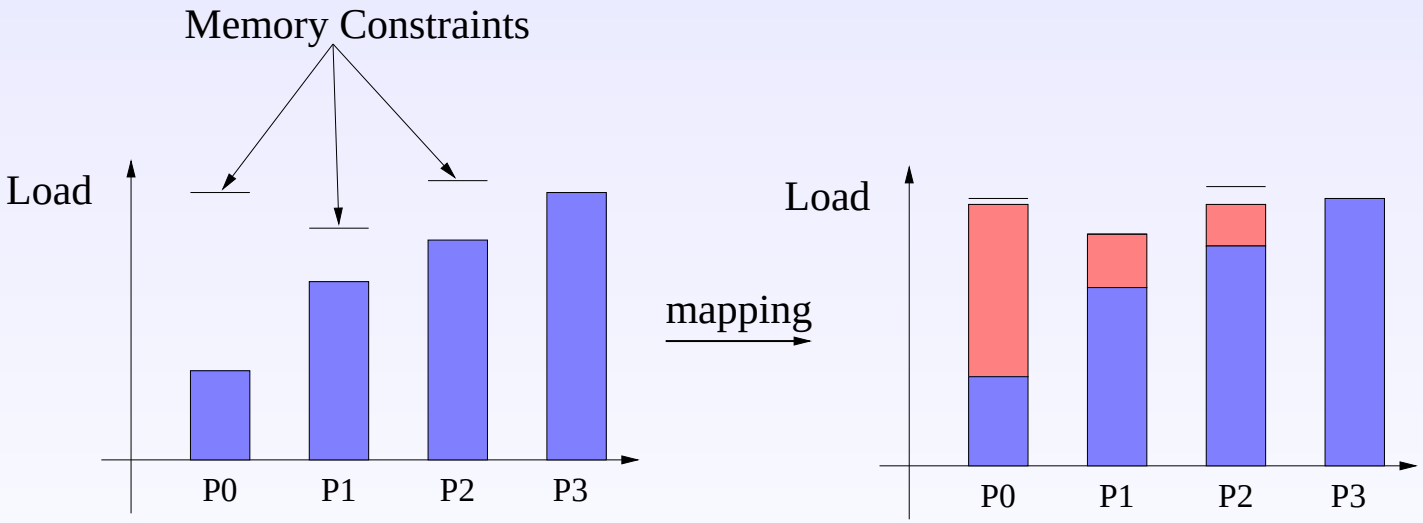
- Candidate processors [Amestoy Duff Voemel SIMAX'05]
- Clusters of SMP's [Amestoy Duff Pralet Voemel PARCO'03]
- Memory-based scheduling [Guermouche L'Excellent IPDPS'04]
- Accuracy of load estimates [Guermouche L'Excellent IPDPS'05]
- Hybrid scheduling (memory+performance) [Amestoy Guermouche L'Excellent Pralet PARCO'06] $\rightarrow$ default scheduling strategy since MUMPS 4.6 [Jan'06]

Main components of hybrid scheduling

- Static mapping and candidates: limit scope of dynamic decisions
- Maintain memory gap with prediction from analysis

  gap=0 → no freedom to dynamic schedulers

  gap>0 → freedom to balance workload on the processors
- Initial gap: memory relaxation → enables irregular partitioning of frontal matrices:

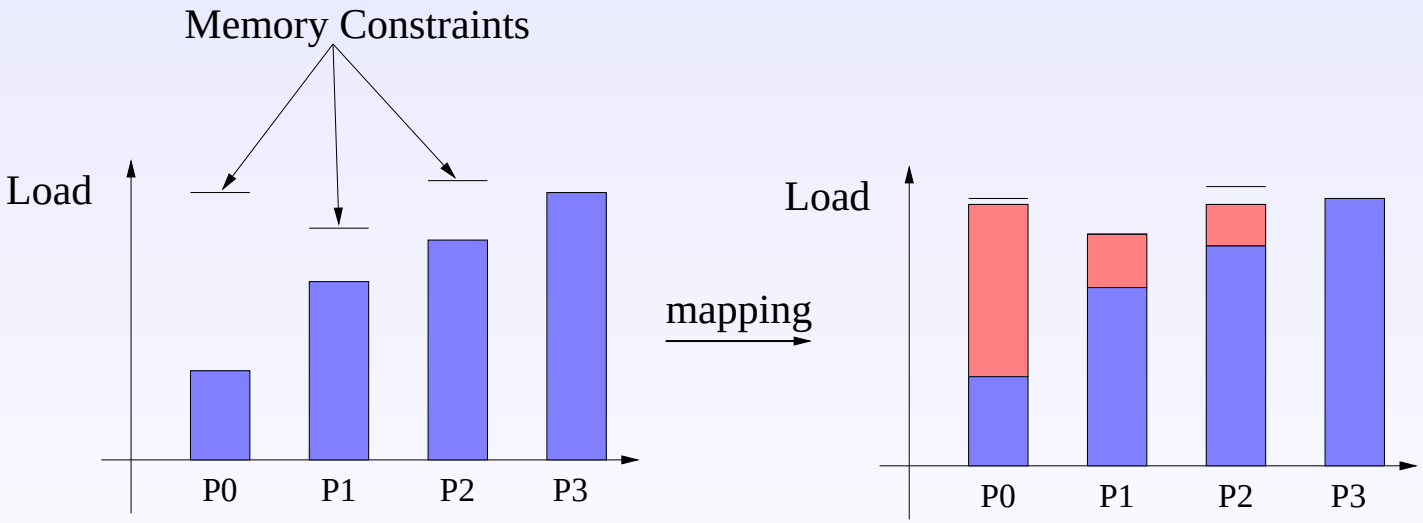


Main components of hybrid scheduling

- Static mapping and candidates: limit scope of dynamic decisions
- Maintain memory gap with prediction from analysis

  gap=0 $\rightarrow$ no freedom to dynamic schedulers

  gap>0 $\rightarrow$ freedom to balance workload on the processors
- Initial gap: memory relaxation $\rightarrow$ enables irregular partitioning of frontal matrices:

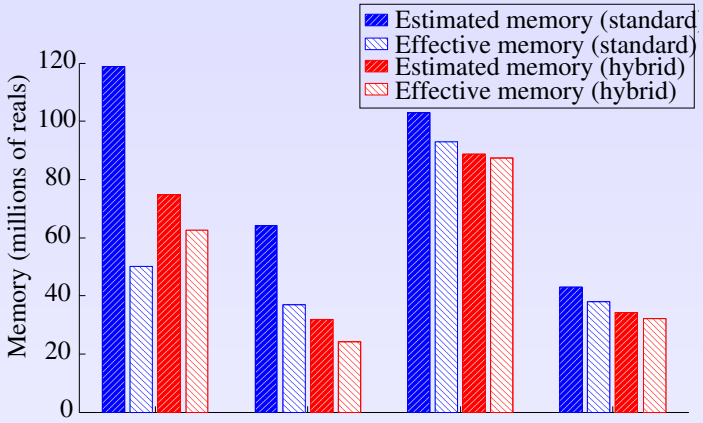


Consequence on Memory Estimates from Analysis

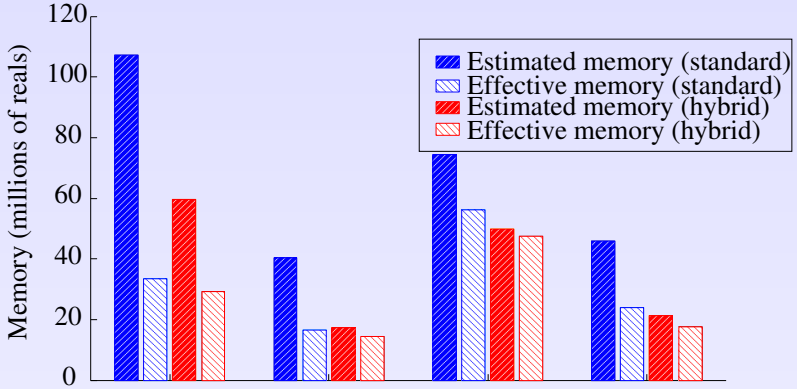
Worst-case estimate $\longrightarrow$ average-case estimate

Experiments on IBM Power4+ processors

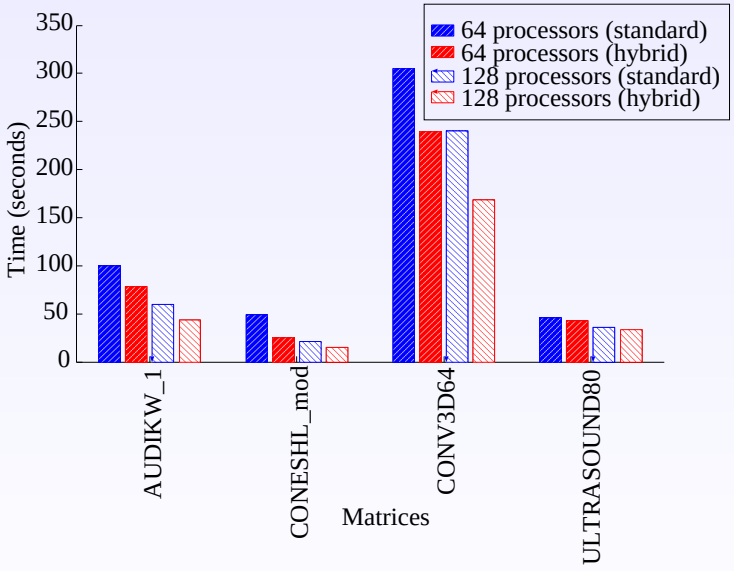
Memory (64 procs)



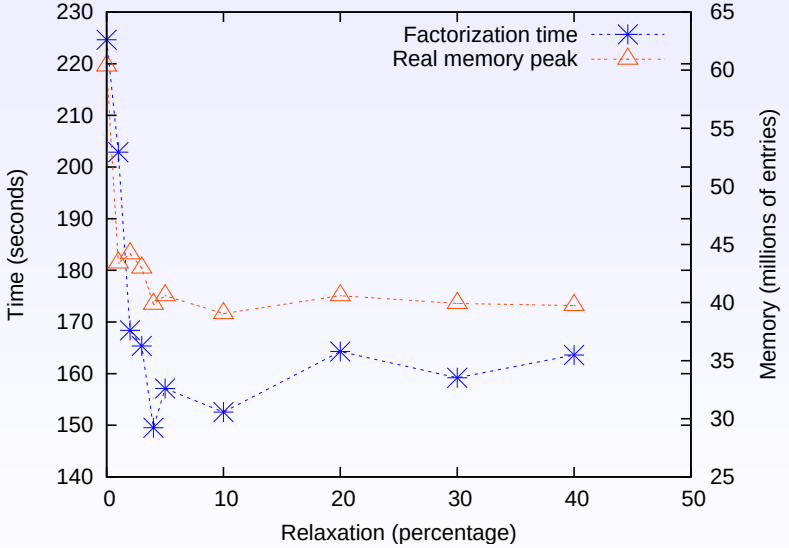
Memory (128 procs)



Factorization time



Sensitivity to memory relaxation



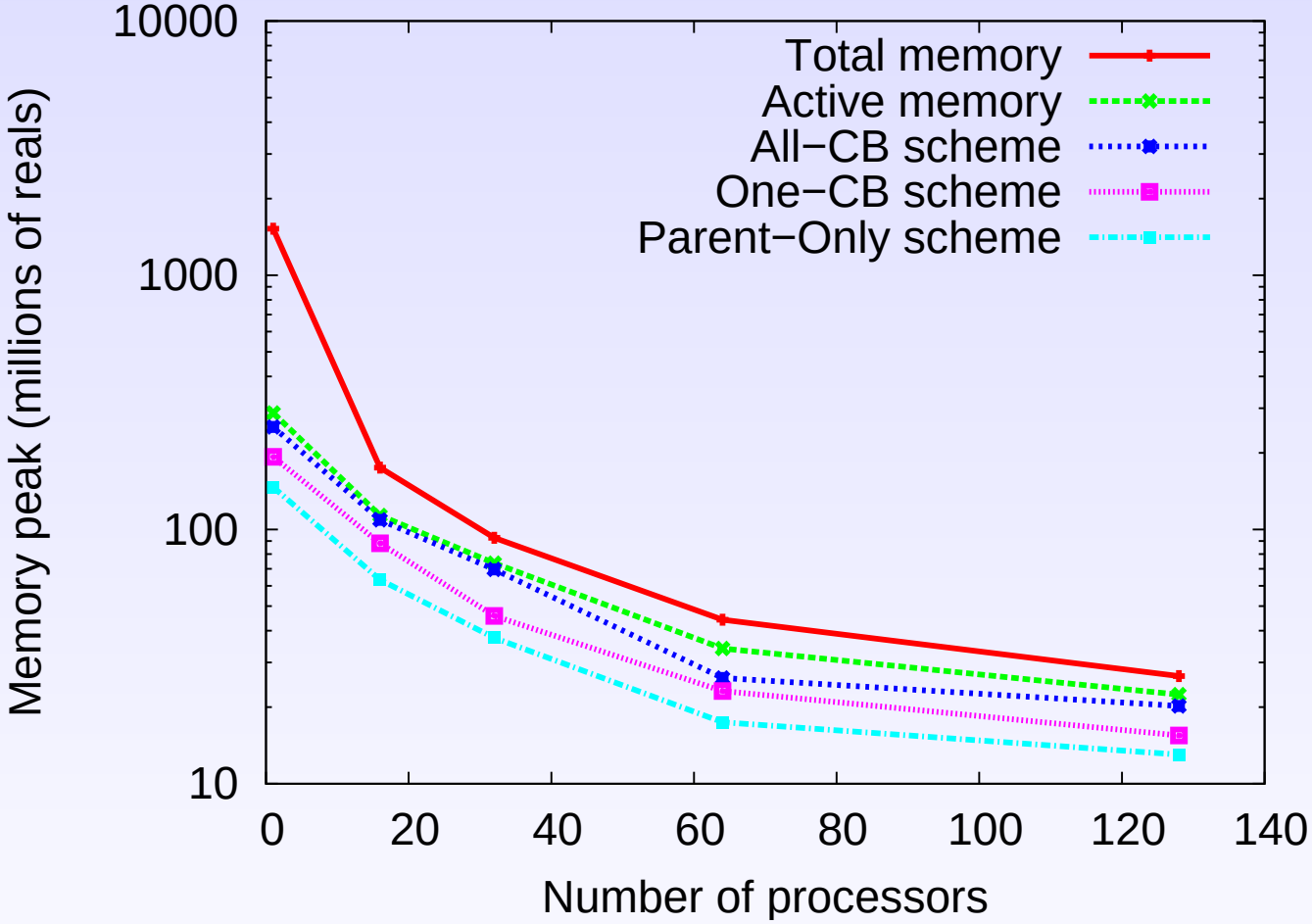
Still problems with memory: out-of-core storage

- First version developed in the context of a contract with Samtech (2005-2006)

- Two Ph.D. completed
  - E. Agullo (ENS Lyon, 2005-2008): Out-of-core Factorization
  - M. Slavova (CERFACS, 2005-2009): Out-of-core Triangular Solution

- Out-of-core storage of factors with asynchronous approach:
  - write factors to disk as soon as they are computed
  - importance of low-level I/O mechanisms

Simulation of an out-of-core active memory



Matrix AUDIKW_1 - METIS ordering
Minimum core memory

Measure of active memory scalability

Problem:

Memory consumption is often a bottleneck for sparse direct solvers

Definition: *Memory Efficiency* on p processors

$$e(p) = \frac{M_{seq}}{p \times M_{max}},$$

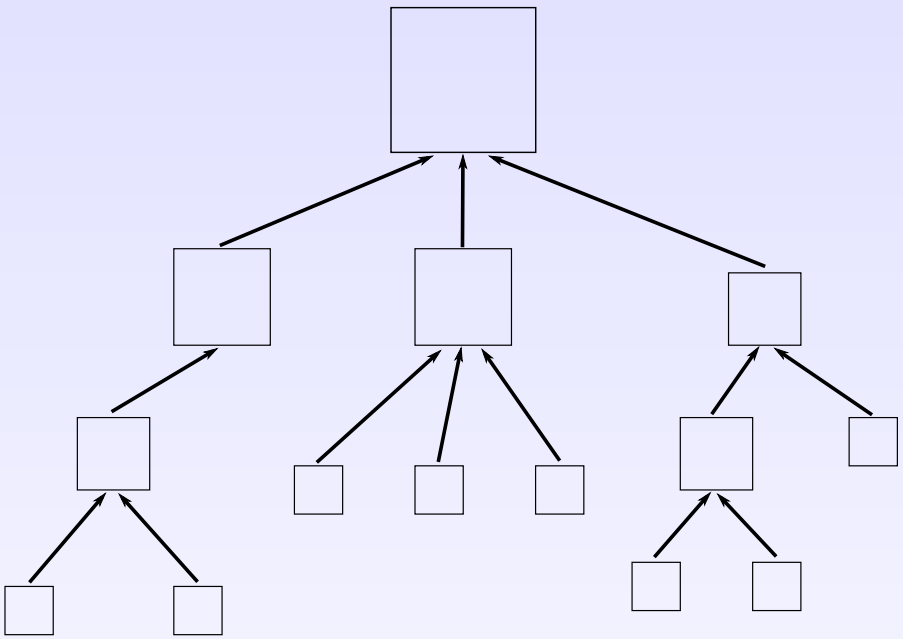
M_{seq}: serial storage, M_{max}: parallel storage

Results: Memory Efficiency (with factors on disk)

Number p 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

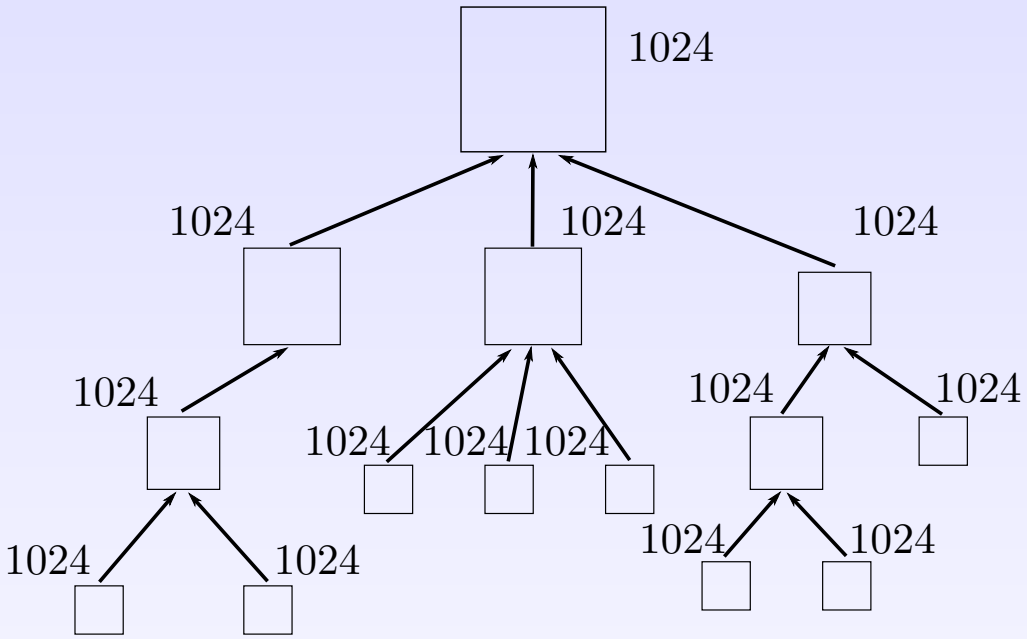
Mapping techniques

Processor-to-node mapping:



Mapping techniques

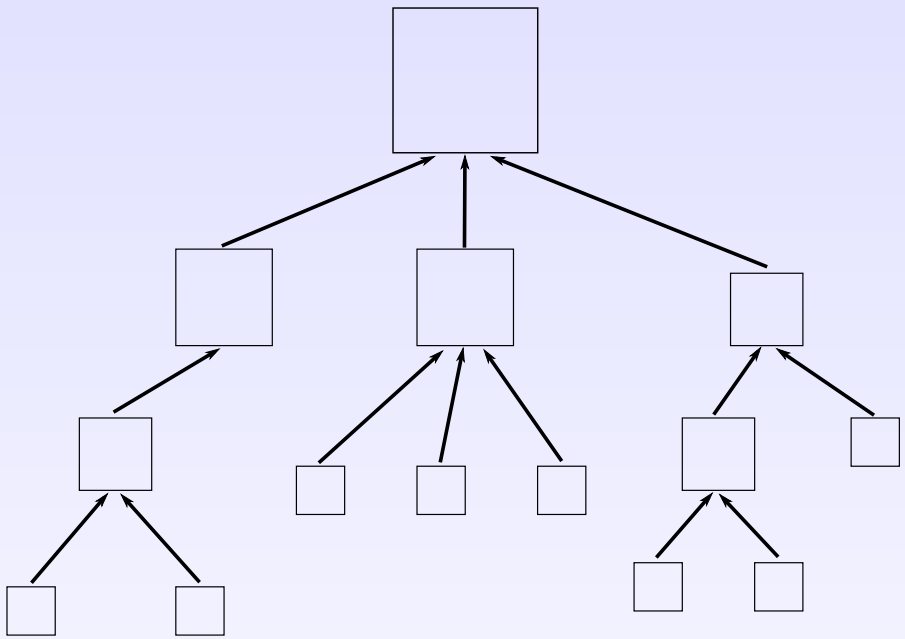
Processor-to-node mapping: all-to-one mapping (postorder traversal)



- Optimal memory scalability: $M_{max} = M_{seq}/p$
- Poor parallelism: only node parallelism is exploited

Mapping techniques

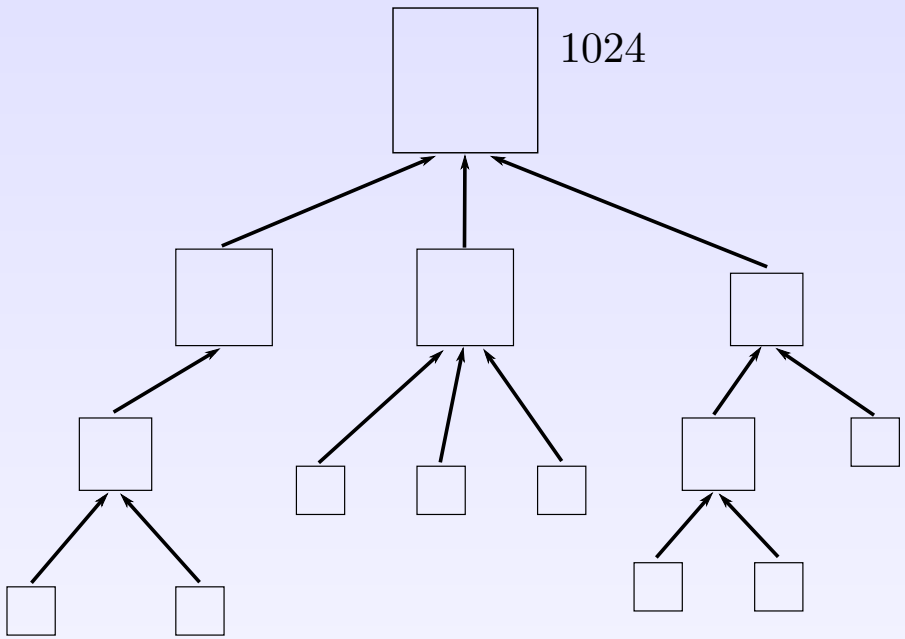
Processor-to-node mapping: proportional mapping [Pothen Sun'93]



- Good properties for parallelism:
  node and tree parallelism, locality of communications
- Not so good for memory

Mapping techniques

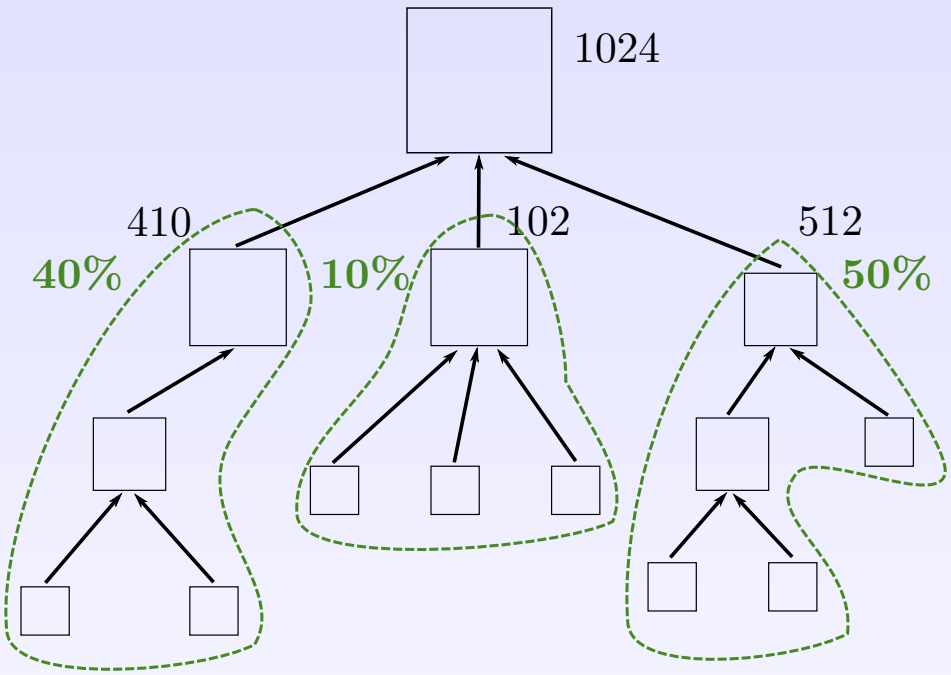
Processor-to-node mapping: proportional mapping [Pothen Sun'93]



- Good properties for parallelism:
  node and tree parallelism, locality of communications
- Not so good for memory

Mapping techniques

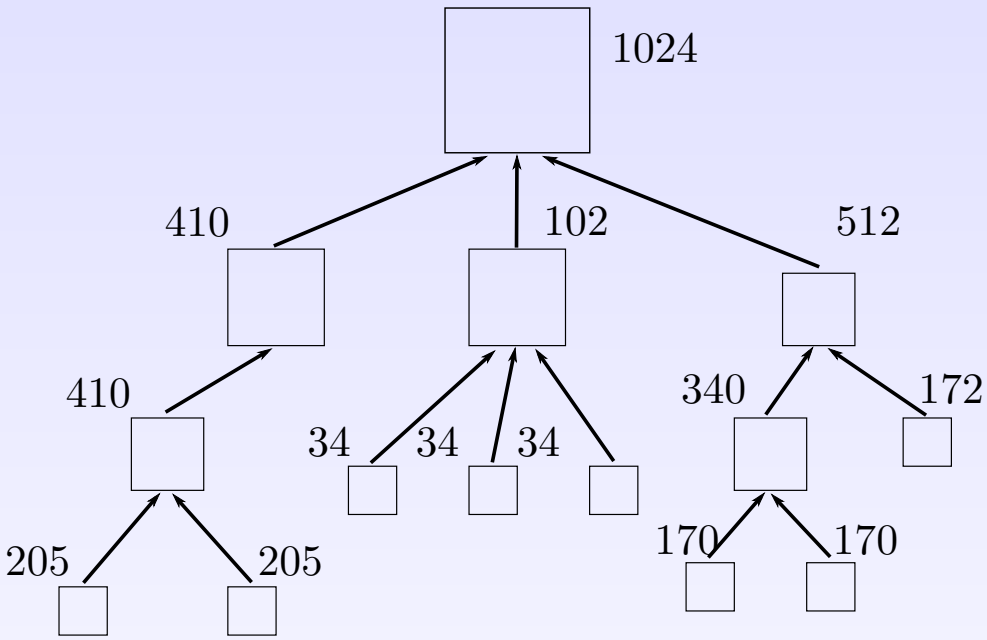
Processor-to-node mapping: proportional mapping [Pothen Sun'93]



- Good properties for parallelism:
  node and tree parallelism, locality of communications
- Not so good for memory

Mapping techniques

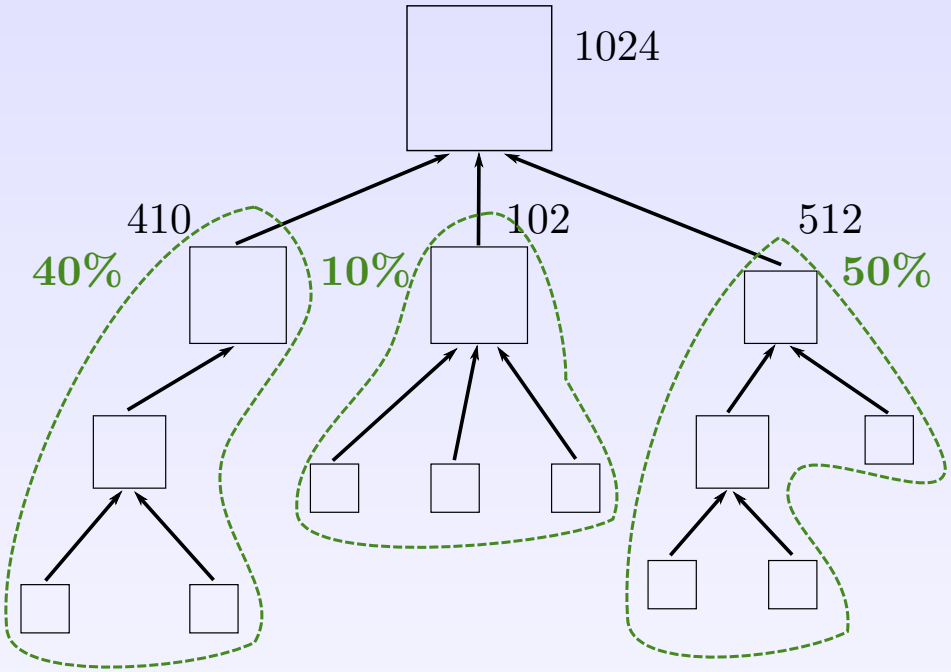
Processor-to-node mapping: proportional mapping [Pothen Sun'93]



- Good properties for parallelism:
  node and tree parallelism, locality of communications
- Not so good for memory

Mapping techniques

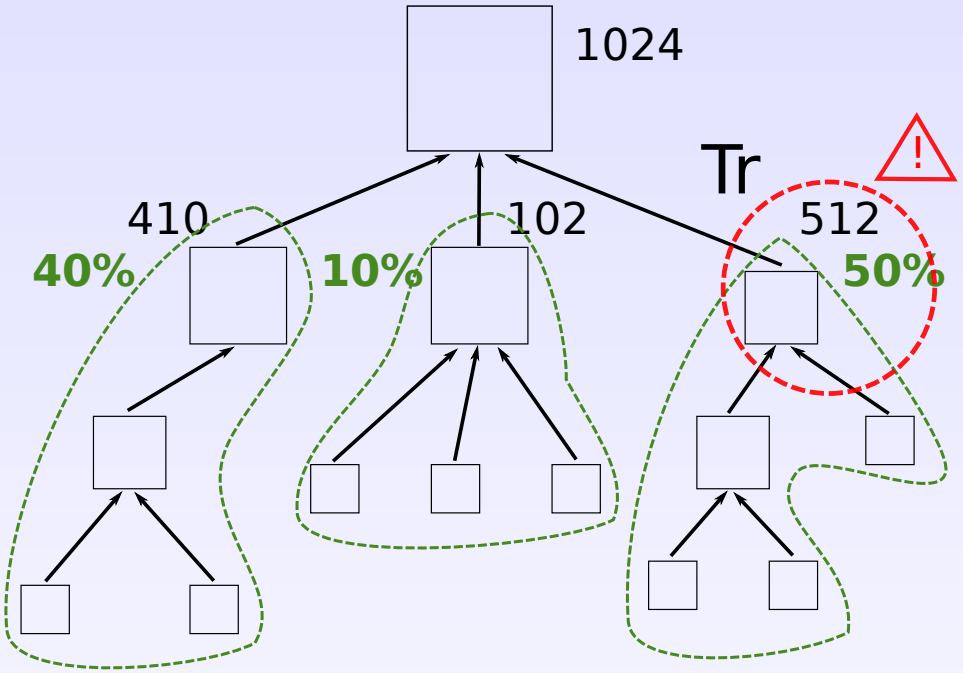
Processor-to-node mapping: "memory-aware" mapping



1. Try to apply proportional mapping
2. Check constraint for each subtree: is there enough memory? If not, serialize subtrees

Mapping techniques

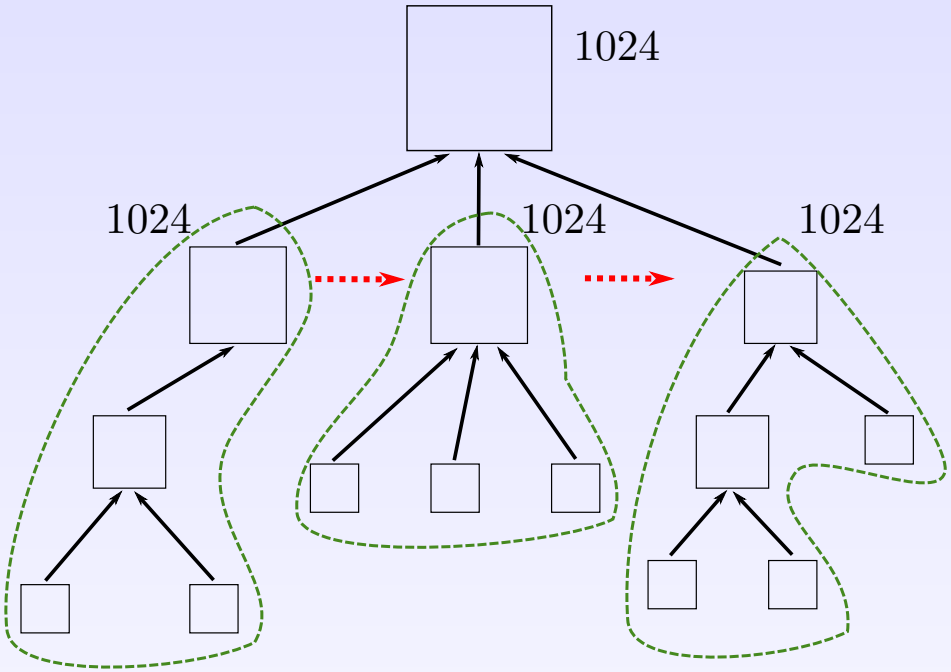
Processor-to-node mapping: "memory-aware" mapping



1. Try to apply proportional mapping
2. Check constraint for each subtree: is there enough memory?
   If not, serialize subtrees

Mapping techniques

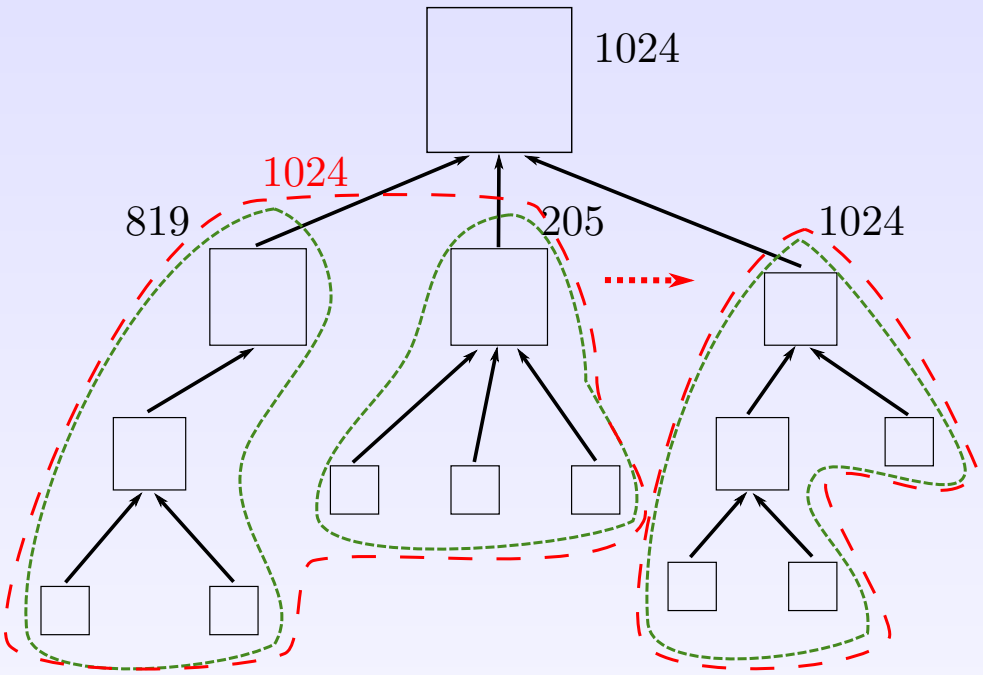
Processor-to-node mapping: "memory-aware" mapping



1. Try to apply proportional mapping
2. Check constraint for each subtree: is there enough memory?
   If not, serialize subtrees

Mapping techniques

Processor-to-node mapping: improved "memory-aware" mapping



1. Find groups of subtrees on which proportional mapping works
2. Serialize these groups

Preliminary results: scheduling influences memory

- Modify static tree mapping to reduce the memory requirement during parallel executions
- Core memory (MB) - `AUDIKW_1`, 16 procs:

Factors		Current (MUMPS 4.10.0)	Memory-oriented mapping
In-Core	Max	4038	2587
	Avg	3345	2446
Out-Of-Core	Max	3028	968
	Avg	2251	827

- Full details and up-to-date results in PhD thesis of F.-H. Rouet, to be defended in Toulouse on October 17, 2012
- Another critical issue to address is to reintroduce dynamism (deadlock avoidance algorithms?)

Remark on MPI communication schemes

Effects of Memory-Oriented Mapping

- Memory-constraints $\Rightarrow$ serialize branches in the tree
- Each branch gets more processors

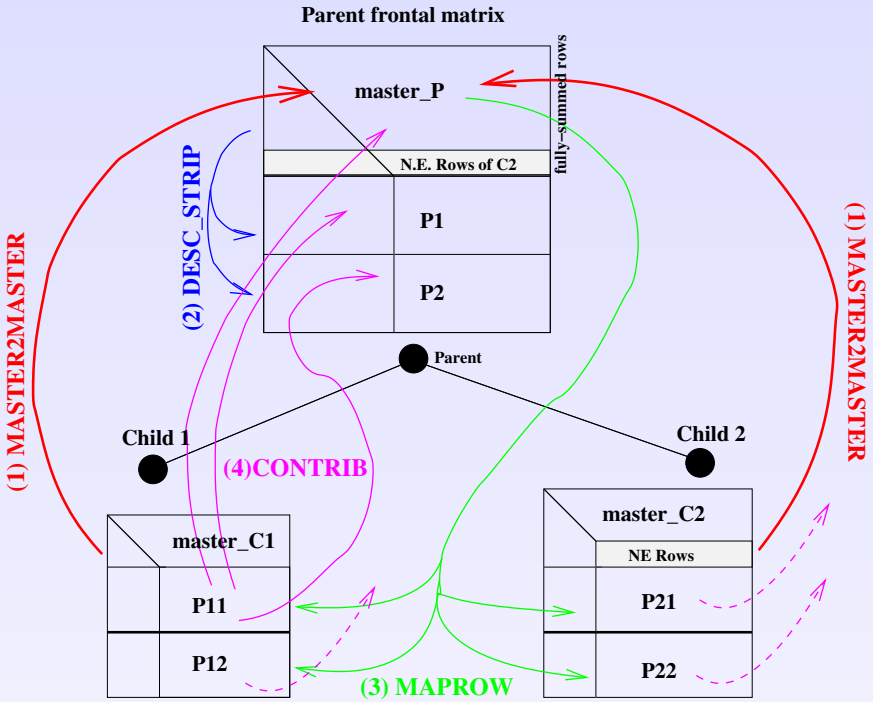
Observation, on one medium-size regular 3D problem

- On one branch, 128 procs showed same performance as 64
- Overall significant speed-down by serializing branches

Analysis

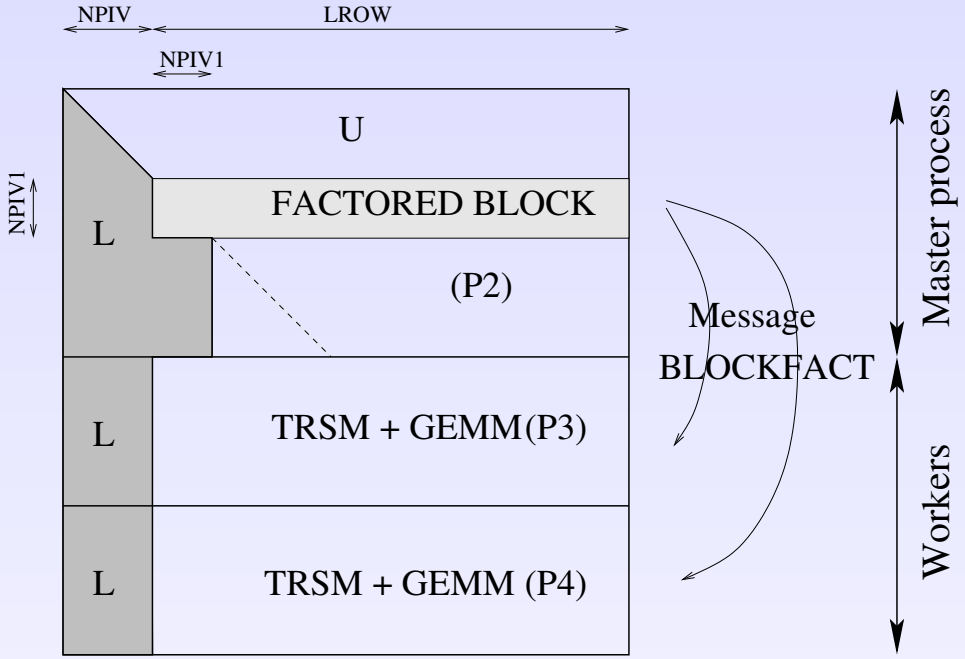
- Traces, Gannt charts, new experiments, models, ...
- Some communication schemes must be revisited

1. Assembly of distributed frontal matrices



- Objective: reduce quantity of assembly messages (4)
- How: force global order for indices in frontal matrices
- Results:
  - p procs per node, $p^2 \rightarrow kp$ messages, $k \approx 2$ or 3
  - time reduction with 64 processors $\approx$ 10%

2. Pipelined 1D factorization



At one stage of the factorization

- One *master* process eliminates variables in first block
- Other processes perform the updates
- (new) bottleneck: send bandwidth of *master* process

Asynchronous broadcast currently revisited

- 16× improvement in communication speed (128 processors)

Memory for triangular solves

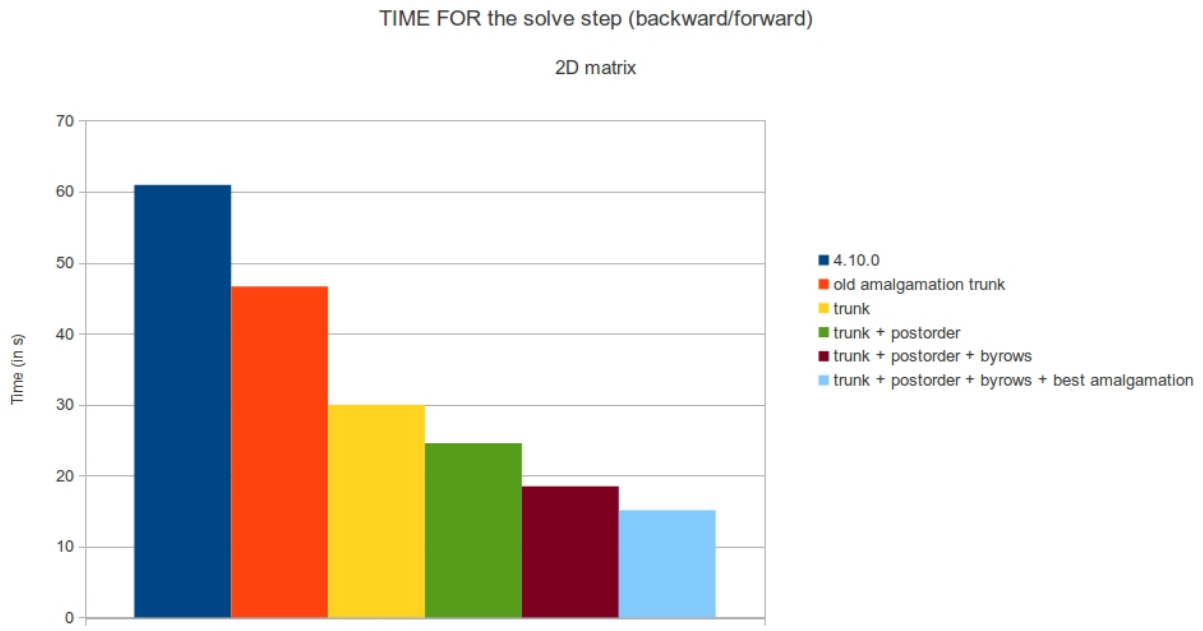
Context:

- MUMPS 4.10.0: Workspace for solution phase increases linearly with the number of processors
- Critical with multiple right-hand sides (RHS)
- Even more critical with sparse multiple RHS

Redesign solve algorithm:

- Specific work on memory $\rightarrow$ scalable workspace and better locality
- Temporary right-hand sides stored by rows (instead of by columns).
- Push node amalgamation (collapse tasks)

Performance of solution phase (128 right-hand sides)



4.10.0: *Latest public release*

- Initial objective was memory reduction

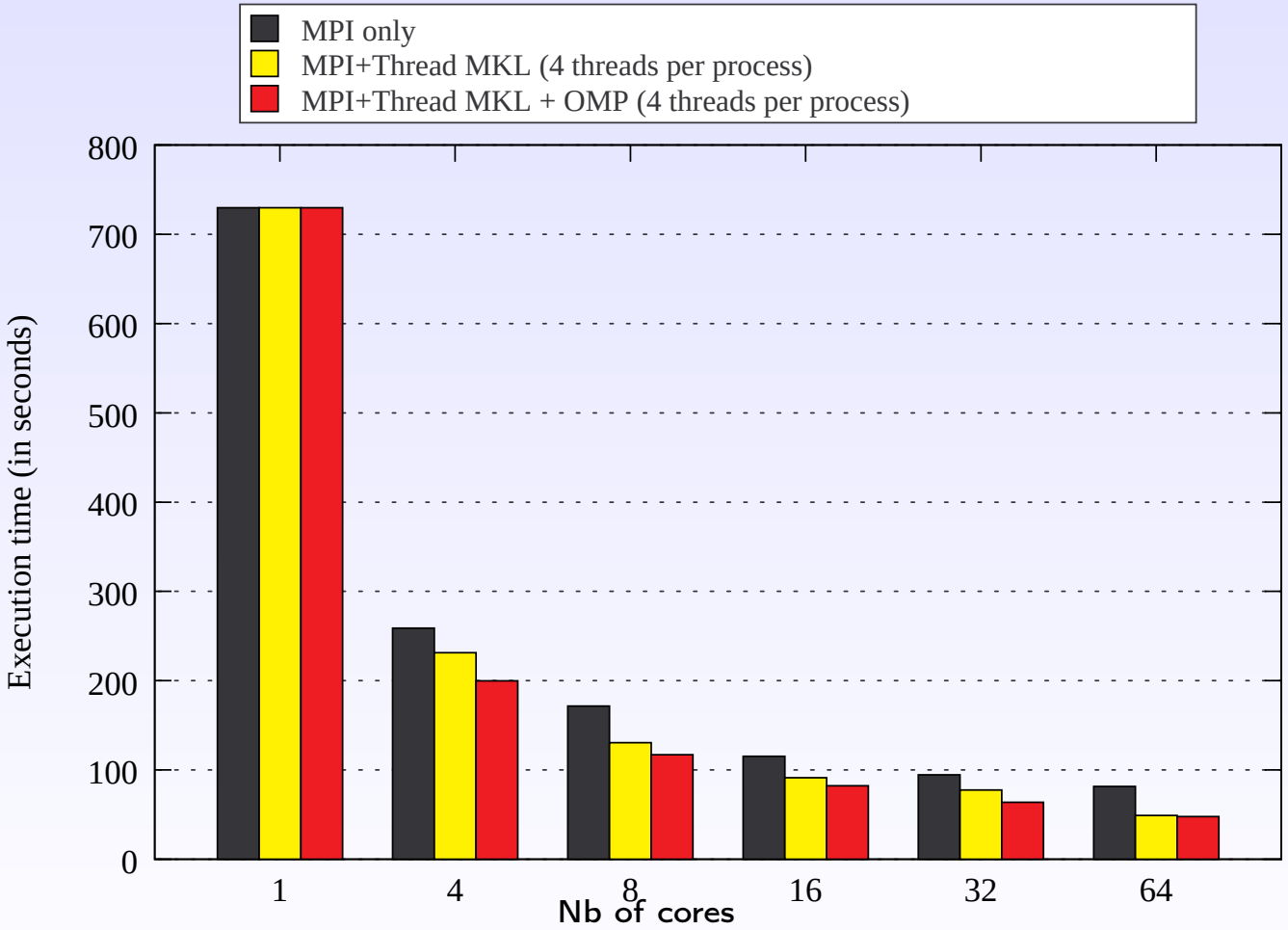
→ more potential to exploit algorithms with better locality

→ significant time reduction

Outline

Shared-memory parallelism: mix MPI + threads

- threaded BLAS (4 threads per MPI) better than all-MPI
- OpenMP directives outside BLAS calls help



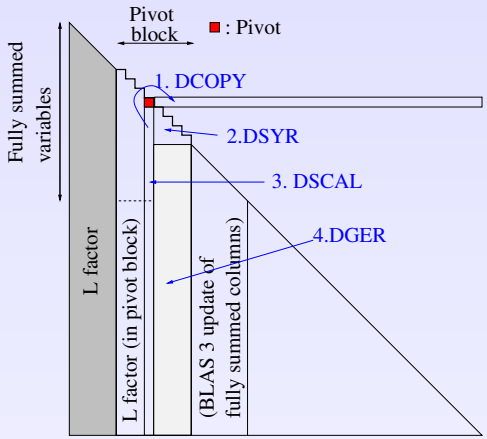
8 MPI $\times$ 1 thread $\rightarrow$ 16 GigaBytes
2 MPI $\times$ 4 threads $\rightarrow$ 11 GigaBytes

Shared-memory parallelism (work-in-progress)

Effect of rewriting panel factorization (LDL^T, HALTERE)

#threads	Panel factorization: Threaded BLAS	Panel factorization: Hand-written (OpenMP)
1	115	110
8	66	34

Total factorization time (seconds)



On-going work

→ similar work needed for other kernels (symmetric updates)

→ exploit multithreaded tree parallelism near the leaves

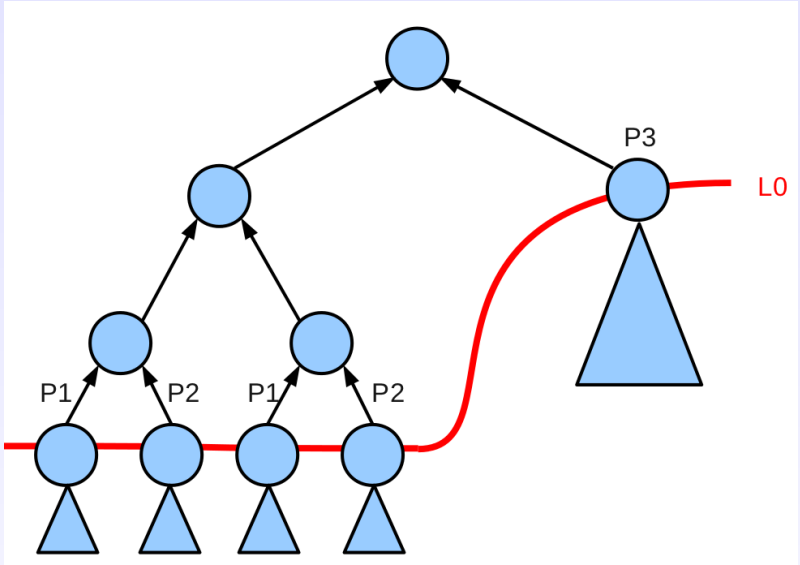
→ study memory allocation policies (NUMA machines)

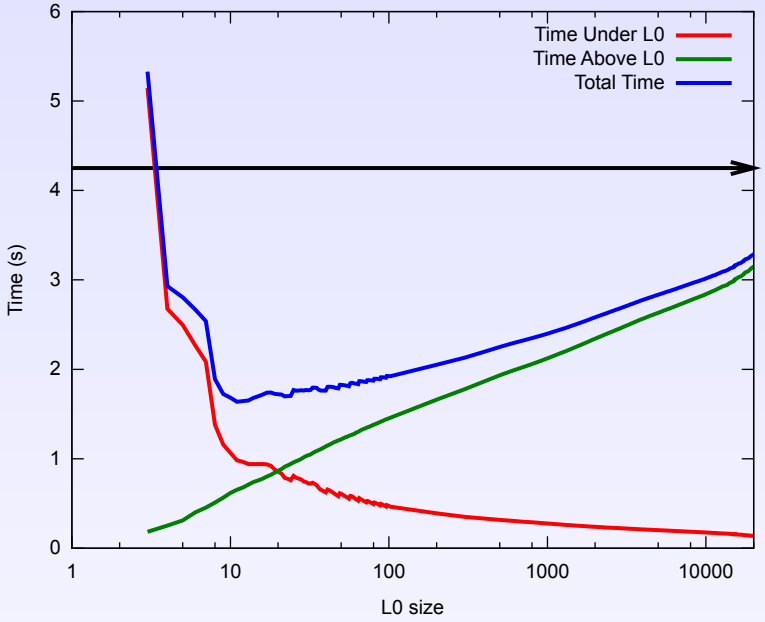
Exploiting tree parallelism at multithread level

Define layer in the tree (L0) [Geist Ng'89]

Under L0: tree parallelism + serial BLAS

Above L0: multithreaded BLAS





$L0 \leftarrow$ root

repeat

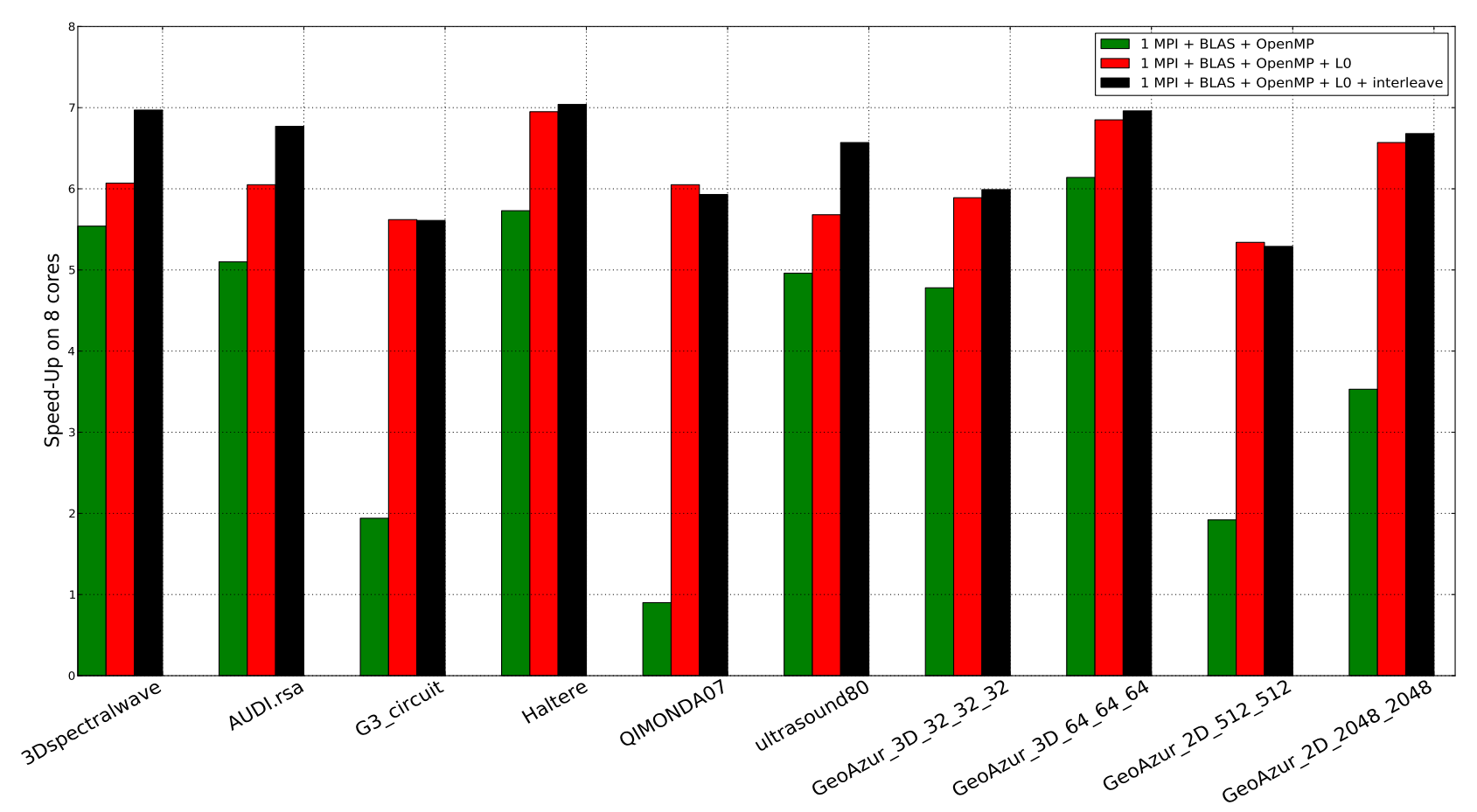
Identify most time-consuming subtree $\mathcal{S}$ (perf. model)

$L0 \leftarrow L0 \setminus \mathcal{S} \bigcup \{\text{children of } \mathcal{S}\}$

Tentative mapping

until reaching minimum estimated total time (perf. model)

Impact of L0 algorithm



On-going work (PhD M. Sid-Lakhdar, ENS Lyon, 2011-2014):

Multitreaded solve algorithm, MPI+OpenMP scaling on large numbers of threads/of MPI processes, Dynamically reuse idle threads, Adapt/tune dense kernels to our context, Combine with memory-aware approaches

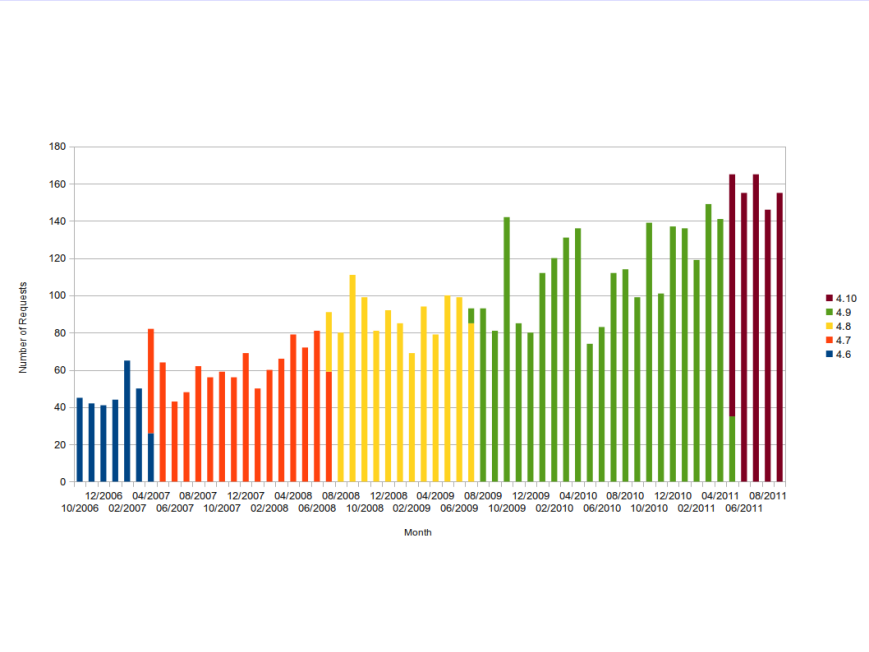
Outline

MUMPS

1. Platform for research
   - Research projects, PhD thesis
   - Hybrid direct-iterative methods

2. Competitive and original software package used worldwide
   - co-developed by INPT, INRIA, CERFACS, Univ. Bordeaux, CNRS, ENS Lyon (Toulouse, Lyon-Grenoble, Bordeaux)
   - latest release: MUMPS 4.10.0, May'11, 250 000 lines of code
   - integrated within various commercial and academic packages (Samcef from Samtech, Actran from Free Field Technologies, *Code_Aster* and *Telemac* from EDF, IPOPT optimization package, Petsc, Trilinos, debian packages, ...)

Users and application fields



1000+ downloads per year from our website

- finite elements
- finite differences
- numerical optimization

Many interactions with users: mumps-dev & mumps-users

Many application fields: Structural mechanical engineering, Biomechanics, Astrophysics, Fluid dynamics, Magnetohydrodynamics, Physical chemistry, Seismic imaging, Ocean modelling, Econometric models, Oil reservoir simulation, Acoustic and electromagnetic wave propagation, Medical image processing, Modeling of organs, Heat transfer analysis, ...

Software engineering is critical

- Many software-related tasks
  version management, trunk and release branches, non-regression tests, code coverage, night builds, website, warning corrections, coding rules, improve modularity, automatic testing on distant platforms, experimentation tools for developers, addition of usecases automatically executed on a daily/weekly/monthly basis, performance evolution between releases, expertise and experiments for industrial users, post-mortem scheduling analysis tool

- Some short-term funding for engineers (1 year, 2 years)
  Maurice Brémond, Philippe Combes, Aurélia Fèvre, Guillaume Joslin, Chiara Puglisi

Summary

Research work transferred in MUMPS software

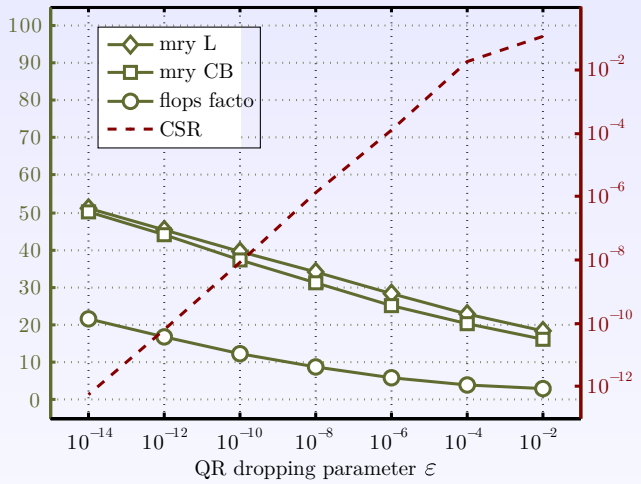
- Validation of research by a large community
- Research guided by application needs/feedback
  (also constrained by complexity of software)
- Several groups depend on the software and on its future and are willing to support it (e.g., EDF, ESI Group, Samtech)

Looking for long-term solutions for development, maintenance, support, organization of industrial relations, etc.

Outline

Project activity: on-going research

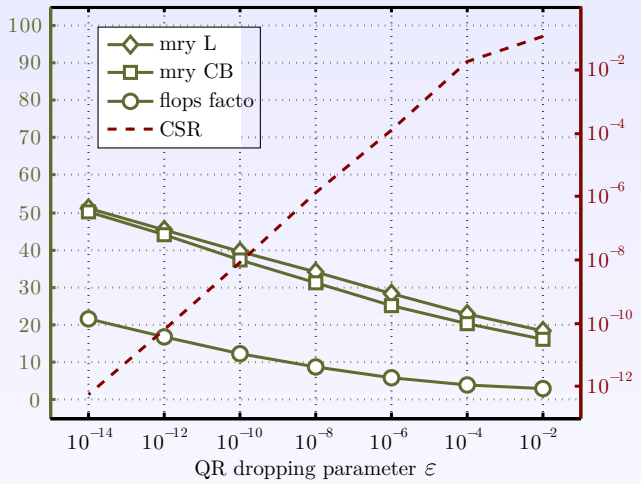
- PhD thesis of F.-H. Rouet (INPT-IRIT, Toulouse, 2009-2012)
  - Memory scalability and performance under memory constraints
  - Exploit sparsity in solution, compute selected entries of A^{-1}
- PhD thesis of M. Sid Lakhdar (ENS-Lyon, 2011-2014)
  - Shared-memory parallelism for multicore processors
  - Hybridize OpenMP + MPI to address large-scale systems
- PhD thesis of C. Weisbecker (INPT, Toulouse, 2010-2013)
  - Low-Rank multifrontal solver
  - Collaboration with EDF (O. Boiteau) and C. Ashcraft (Livermore, USA)



- Rank-revealing factorizations G. Joslin (CERFACS) et al.

Project activity: on-going research

- PhD thesis of F.-H. Rouet (INPT-IRIT, Toulouse, 2009-2012)
  - Memory scalability and performance under memory constraints
  - Exploit sparsity in solution, compute selected entries of A^{-1}
- PhD thesis of M. Sid Lakhdar (ENS-Lyon, 2011-2014)
  - Shared-memory parallelism for multicore processors
  - Hybridize OpenMP + MPI to address large-scale systems
- PhD thesis of C. Weisbecker (INPT, Toulouse, 2010-2013)
  - Low-Rank multifrontal solver
  - Collaboration with EDF (O. Boiteau) and C. Ashcraft (Livermore, USA)



- Rank-revealing factorizations G. Joslin (CERFACS) et al.

Despite all these gains, we have not factorized yet
$1000 \times 1000 \times 1000$ *geophysics earth models from Jean Virieux*

Some perspectives for direct solvers (i)

- More parallelism
  - Many efforts towards multicores (Pastix, WSMP, HSL MA87)
  - MPI on 5000+ nodes (performance and memory scalability)
  - Fully parallel preprocessing on distributed matrices/graphs?
  - Fault tolerance? Energy-aware algorithms?
- GPGPU, accelerators? (High potential – large blocks)
  - D. Pierce et al. (multifrontal solver in BCSLIB-GPU), B. Lucas et al. (also multifrontal)
  - T. Davis (multifrontal *QR*)
  - Use runtimes to automatically schedule tasks: cores + GPUs
    - Buttari, Guermouche, Lopez et al.: QR-MUMPS
    - Faverge, Ramet et al.: Pastix
- Memory aspects and memory locality are central
  - Memory per core will decrease
  - Importance of memory-aware algorithms in a dynamic context
  - Adapt memory-demanding strategies such as numerical pivoting?

Some perspectives for direct solvers (ii)

- Efficient solution phase (forward and backward)
  - Take into account multiple right-hand sides
  - Exploit sparsity of right-hand sides and/or solution
  - Preprocessing and factorization (distribution of factors) guided by performance of solve?

- Numerical evolutions
  - Specific classes of applications (e.g., augmented systems)
  - Tool for hybrid direct-iterative methods: domain decomposition, Schur-complement methods, block-Cimmino approaches
  - Exploit low-rank properties of off-diagonal blocks in frontal matrices (Xia, Li et al., Purdue + Berkeley; PhD C. Weisbecker, Toulouse)

Some collaborators/contributors to the MUMPS project:

Emmanuel Agullo, Patrick Amestoy, Caroline Bousquet, Maurice Brémond, Alfredo Buttari, Indranil Chowdhury, Philippe Combes, Iain Duff, Vincent Espirat, Aurélia Fèvre, Serge Gratton, Abdou Guermouche, Guillaume Joslin, Jacko Koster, Anne-Monique Philippon, Stéphane Pralet, Chiara Puglisi, Grégoire Richard, François-Henry Rouet, Alexis Salzmann, Mohamed Sid-Lakhdar, Tzvetomila Slavova, Miroslav Tuma, Bora Uçar, Gil Utard, Xavier Vasseur, Christof Voemel, Clément Weisbecker

THANK YOU!