Parallel Solution of a Sparse Linear Systems in an Out of Core Environment

Tzvetomila Slavova

PhD student, CERFACS Toulouse

in colaboration with :
Patrick Amestoy, *IRIT-Université de Toulouse*
Iain Duff, *CERFACS Toulouse*
Abdou Guermouche, *LaBRI, Université de Bordeaux*

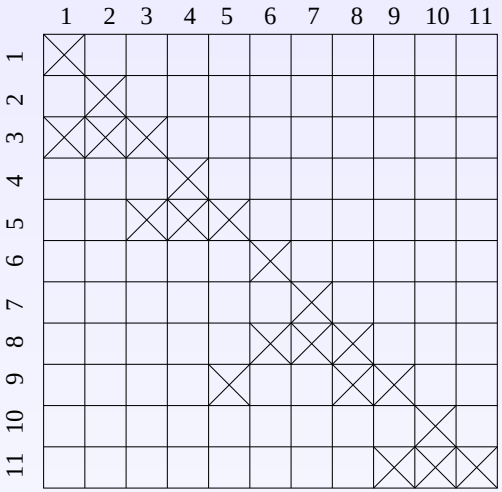
APO Project 08
Toulouse – 08 Sept, 2008

Outline

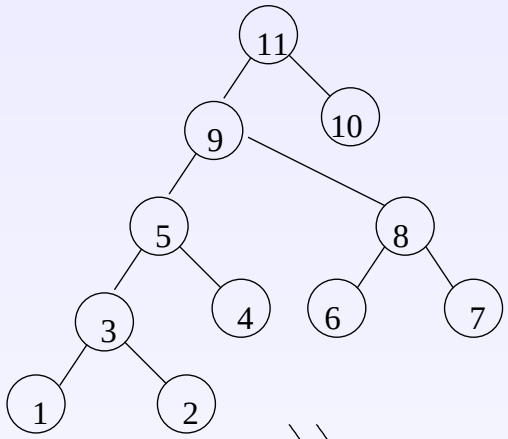
Introduction to the solution phase

For a symmetric case : $Ax = b \quad \Rightarrow \quad A = LDL^T$

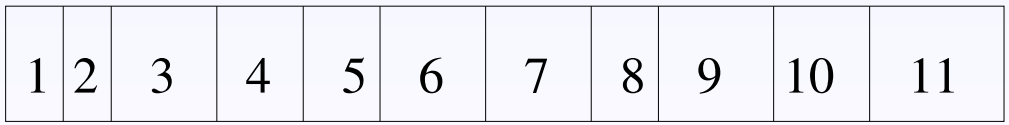
matrix pattern



elimination tree



L factors



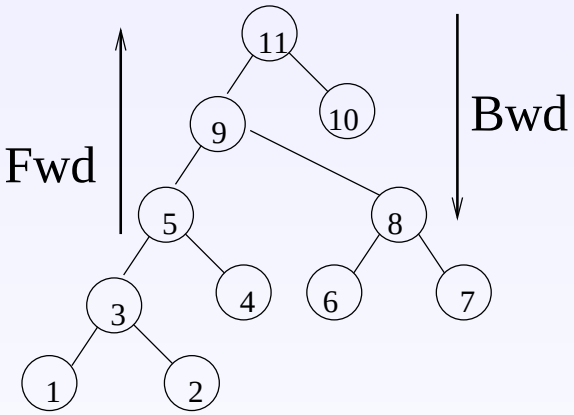
Location of factors on DISK

Solution phase $(LDL^T x = b)$ is the third and last step.

Tree traversal during solve

Solution phase is divided in two steps :

$A = LDL^T \quad \Rightarrow \quad LDy = b$ (forward - **FWD**)
$L^T x = y$ (backward - **BWD**)



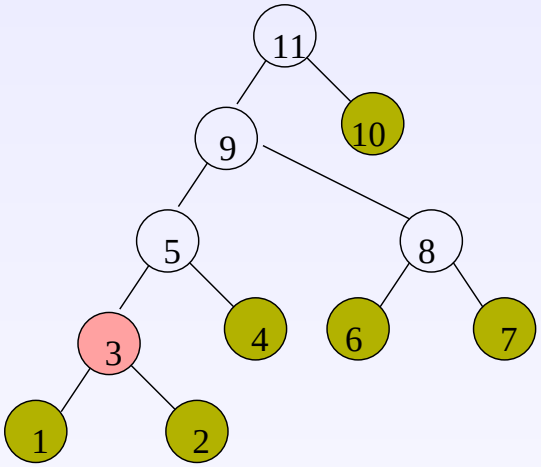
for the sequential case :
Fwd : post-ordering (idem facto. phase)
Bwd : in the reverse order

for the parallel case :
Topological ordering only : no guarantee of the order in which nodes are processed

Pool of tasks

Pool of tasks : list of all tasks ready to be executed (scheduling)

- ★ *Illustration : sequential processing of the tree*



Pool at the beginning of FWD

I - II step

10	7	6	4	2	1	

III step

10	7	6	4	3	

FWD

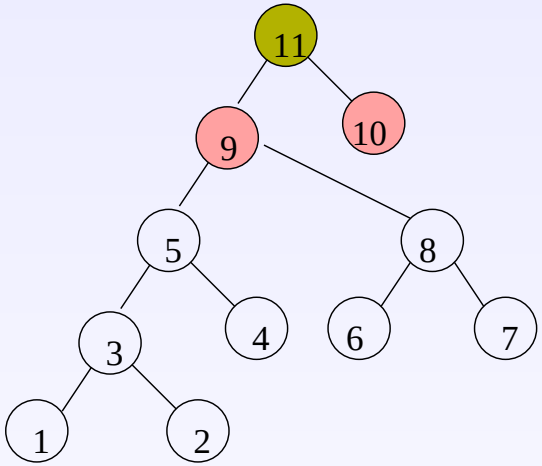
1	2	3	4	5	6	7	8	9	10	11

Factors Data on the HARD DISK

Pool of tasks

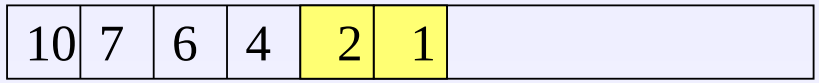
Pool of tasks : list of all tasks ready to be executed (scheduling)

- ⋆ *Illustration : sequential processing of the tree*

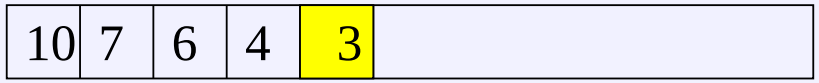


Pool at the beginning of FWD

I - II step

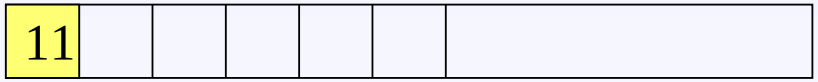


III step

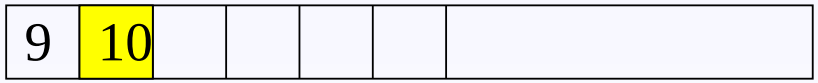


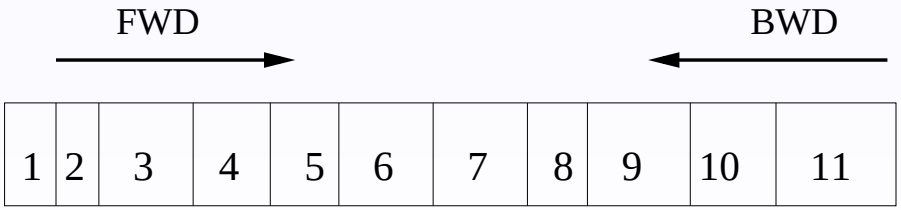
Pool at the beginning of BWD

I step



II step



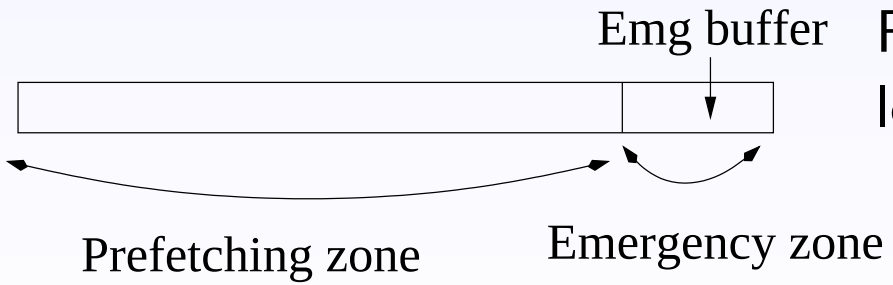


OOC Basics

Objective : Reduce time for solution of $Ax = b$
in a parallel limited memory environment (OOC)

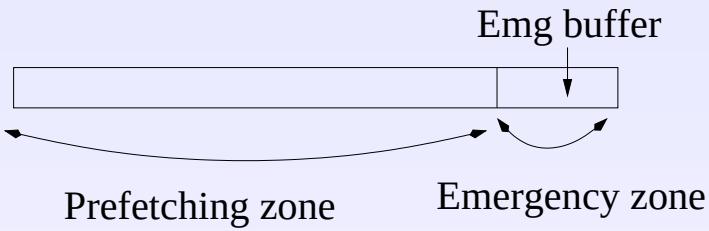
Time performance of the solution phase is related to :

- ⋆ the bandwidth for reading data (Cray XD1 : 16 MB/s)
- ⋆ the number of processors and volume of factors per processor
- ⋆ the regularity in the disk accesses
- ⋆ the number of emergency calls (irregular demand driven access)



For loading data from disk,
local user buffers are implemented

OOC Methods



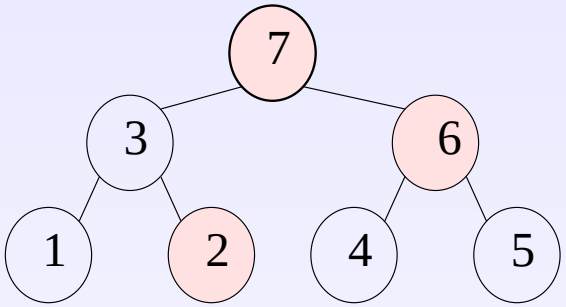
- ⋆ demand driven - disk access on request
- ⋆ look-ahead - prefetch data as soon as "enough" memory available
- ⋆ using system cache for loading data outside of the user space (SC)
- ⋆ using user buffers - with user defined prefetch mechanism (UB)

Comparison between SB and Direct IO methods on Coneshl matrix

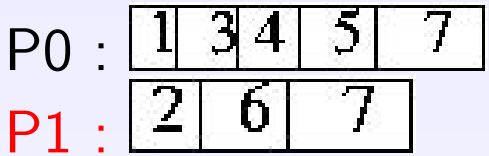
Method	Nb procs	Factor Size (MB)	Solve space per proc (MB)	Fwd (sec)	Bwd (sec)
SB	1	5 908	∗	446.1	448.1
UB	1	5 908	709	375.2	378.3

∗ Cannot be estimated

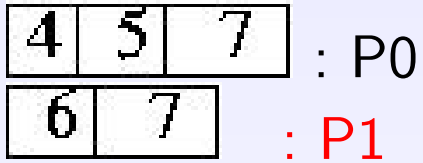
Schedulings and Parallelism in OOC



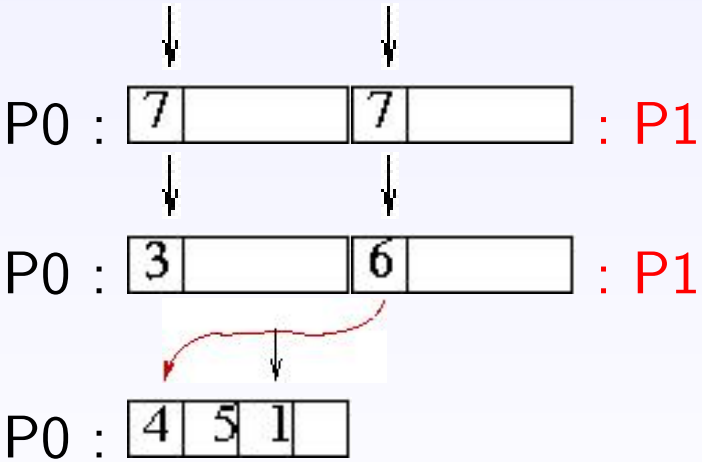
Hard Disk



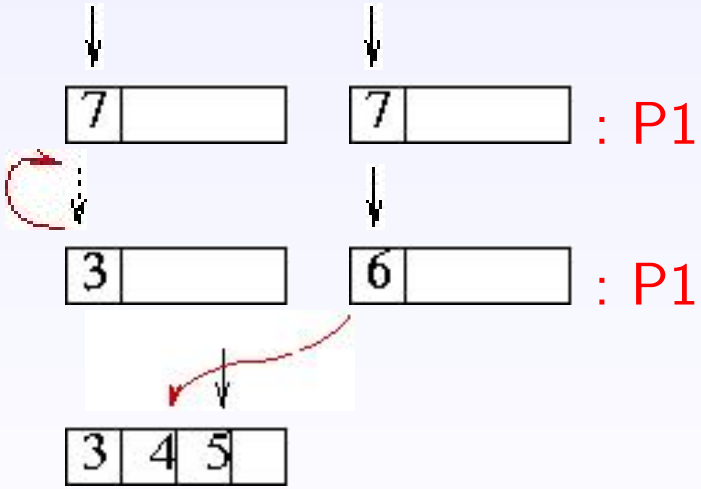
User Buffer



POOL : LIFO strategy



New NNS strategy



Scheduling and Parallelism in OOC

Properties for NNS scheduler :

- ⋆ each processor – its own write sequence to follow
- ⋆ no deadlocks (see TechRep IRIT RT/APO/07/3)
- ⋆ more regular access to the disk (less emergency calls)
- ⋆ more contiguous free space in user buffer
- ⋆ introduces synchronisation (wait for NNS node)

Testing environment

- Multiprocessor Cray XD1 (CERFACS)
  - 58 nodes (2 processors/node)
  - 4 GB memory/ node
  - IDE disk managing for each node
  - Reiserfs file system
  - max bandwidth for read operation : 16 MB/sec
- analysis phase – sequential in-core
- factorization phase – parallel in out-of-core
- solution phase – parallel in out-of-core
- 1 MPI process per node

Testing matrices

Set of test matrices : sorted by factor size

Matrix name	Order	Nb entries (Millions)	Factor size (MBytes)	Nb Nodes in the tree	Origin
QIMONDA07	8 613 291	66.9	2 534	3 083 998	Qimonda AG
CAS4R-L15	2 423 135	195.8	4 832	864 447	EADS
CONESHL	1 262 212	43.0	5 908	113 513	SAMTECH
NICE20MC	715 923	28.1	9 263	68 134	BRGM Lab.
AUDI	943 695	39.3	12 202	113 119	Parasol collection
GRID3.5M	3 500 000	37.8	15 720	1 535 044	11pt-disc.
COR5HZ	2 233 031	90.2	21 622	268 798	BRGM Lab.
AMANDE	6 994 683	584.8	55 295	871 621	CEA-CESTA
NICE9HZ	5 140 838	215.5	64 848	603 495	BRGM Lab.

Matrices reordered with Metis.

All matrices are real symmetric, except CAS4R-L15 and AMANDE which are complex symmetric. Publicly available matrices can be obtained on `gridtlse.org` .

Testing matrices

Set of test matrices : sorted by factor size

Matrix name	Order	Nb entries (Millions)	Factor size (MBytes)	Nb Nodes in the tree	Origin
QIMONDA07	8 613 291	66.9	2 534	3 083 998	Qimonda AG
CAS4R-L15	2 423 135	195.8	4 832	864 447	EADS
CONESHL	1 262 212	43.0	5 908	113 513	SAMTECH
NICE20MC	715 923	28.1	9 263	68 134	BRGM Lab.
AUDI	943 695	39.3	12 202	113 119	Parasol collection
GRID3.5M	3 500 000	37.8	15 720	1 535 044	11pt-disc.
COR5HZ	2 233 031	90.2	21 622	268 798	BRGM Lab.
AMANDE	6 994 683	584.8	55 295	871 621	CEA-CESTA
NICE9HZ	5 140 838	215.5	64 848	603 495	BRGM Lab.

Matrices reordered with Metis.

All matrices are real symmetric, except CAS4R-L15 and AMANDE which are complex symmetric. Publicly available matrices can be obtained on `gridtlse.org` .

Performance study - I

QIMONDA07 (Qimonda AG company)

Strategy	Nb of Procs	Factor Size (MB)	Workspace (MB)	Fwd (sec)	Bwd (sec)
LIFO				171.5	177.2
NNS	1	2 534	12	170.6	176.8
LIFO				25.2	137.6
NNS	8	317	12	29.0	45.2
LIFO				11.0	53.1
NNS	32	79	8.2	10.2	10.7

AMANDE (CEA-CESTA)

Strategy	Nb of Procs	Factor Size (MB)	Workspace (MB)	Fwd (sec)	Bwd (sec)
LIFO				725.9	964.8
NNS	20	1625	425	678.0	866.1
LIFO				679.8	1071.6
NNS	24	1364	366	475.5	629.5
LIFO				358.9	814.6
NNS	32	1028	261	350.9	564.6

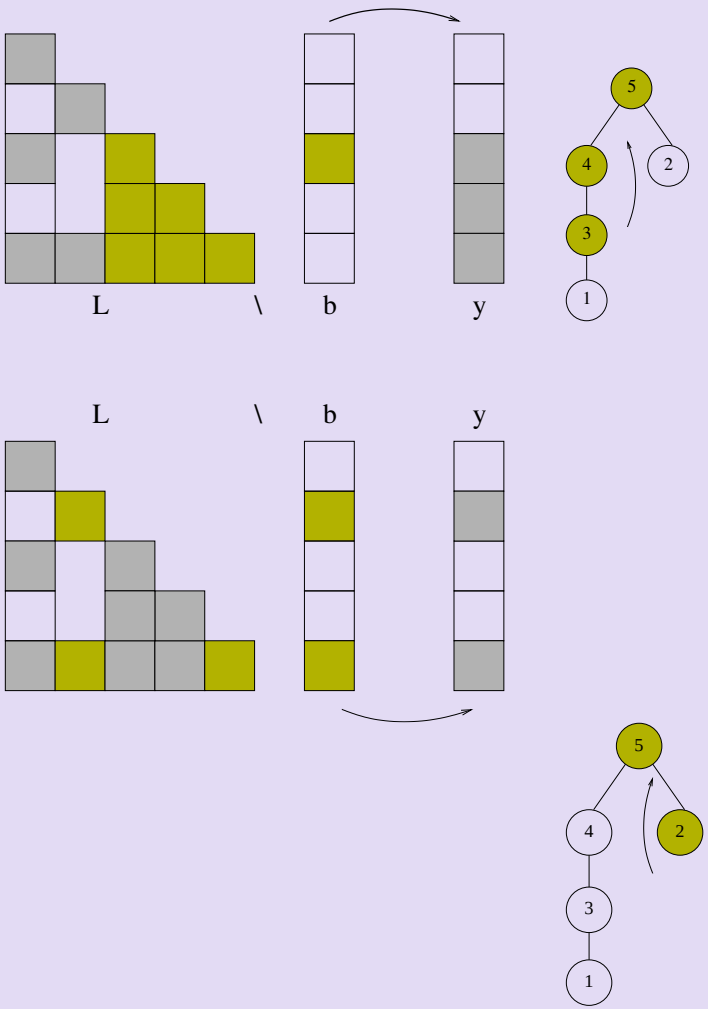
Conclusion

- ★ System cache based approach
  - memory is not stressed
    - easy to use/implement and provides good performance (prefetch and disk access rate)
  - memory is stressed
    - Performance issue
  - No control of memory used (risk of swapping user's data)
- ★ Direct IO approach with user buffers
  - User buffers needed
  - Performance depends on :
    - factor distribution onto the processors
    - number and regularity of disk accesses
  - Parallel execution : scheduling is important
  - *on all matrices, forcing* **NNS ordering** *improves performance*

Outline

Motivations

solve $y \leftarrow L \setminus b$



- ⋆ Objectives
  - Efficient use of the rhs sparsity
  - Characterize LU factors to be loaded from disk (pruning)
  - Efficiently load only needed factors from disk
- ⋆ In all application cases, only part of factors need to be loaded

Applications

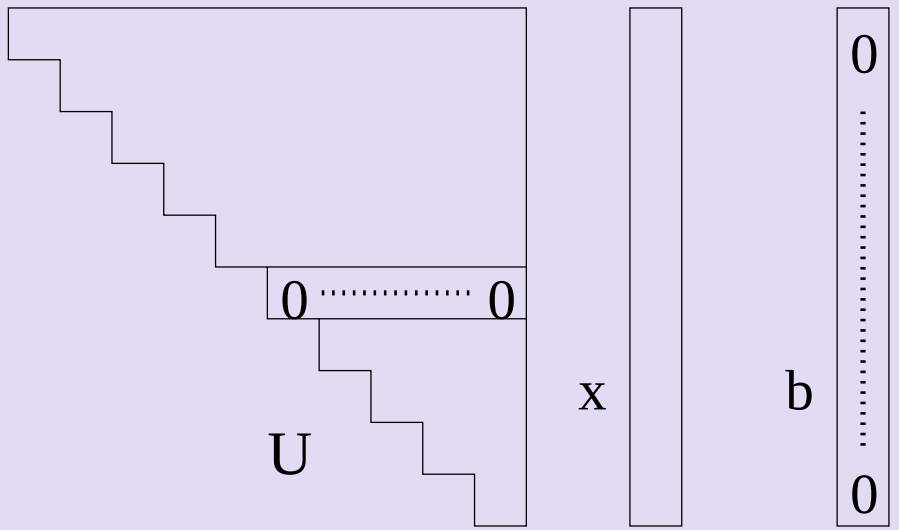
- Applications in :
  - Null space computations (quasi-null rows/pivots)
  - Sparse rhs on reducible matrices (linear programming)
  - Computing elements in A^{-1} (covariant matrices)
- Background
  - Predicting structure of the solution vector (Gilbert-Liu 93)
  - Computing elements of the inverse of a matrix (Erisman-Tinney 75)
  - Computing sparse inverse subset (Campbell-Davis 95)

Application I : Null Space Compuation

Compute x such that :

$$\begin{aligned} Ax &= 0 \\ A = LU &\quad , \quad LUx = 0 \\ det(L) \neq 0 &\quad , \quad Ux = 0 \end{aligned}$$

Null Space Computations



Application I : Null Space Compuation

Compute x such that :

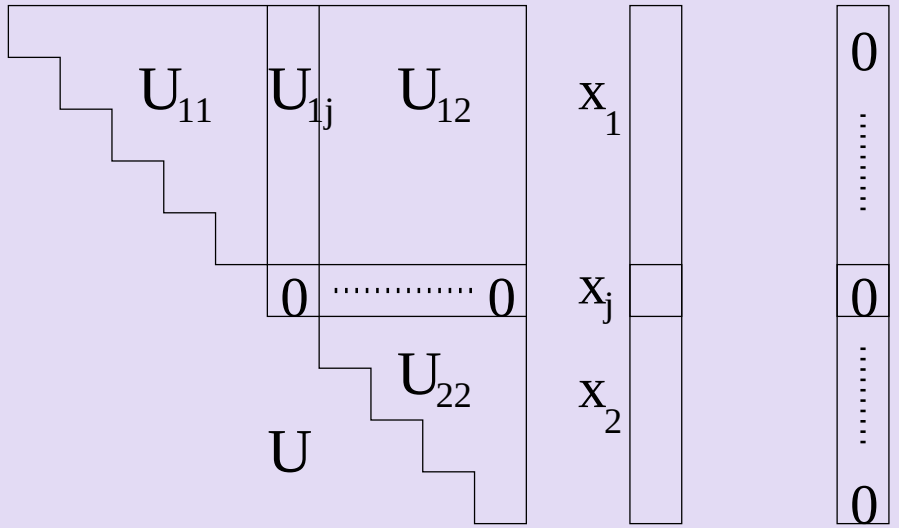
$$\begin{array}{rcl} Ax & = & 0 \\ A = LU & , & LUx = 0 \\ det(L) \neq 0 & , & Ux = 0 \end{array}$$

$$\begin{cases} U_{11}x_1 + U_{1j}x_j + U_{22}x_2 = 0 \\ 0x_j + 0x_2 = 0 \\ U_{22}x_2 = 0 \end{cases}$$

$$x_2 = 0 \quad , \quad x_j = cnst$$
$$U_{11}x_1 + U_{1j}x_j = 0$$

$$\forall x_j, \quad x_1 \ / \quad U_{11}x_1 = -U_{1j}x_j$$

Null Space Computations



Application I : Null Space Compuation

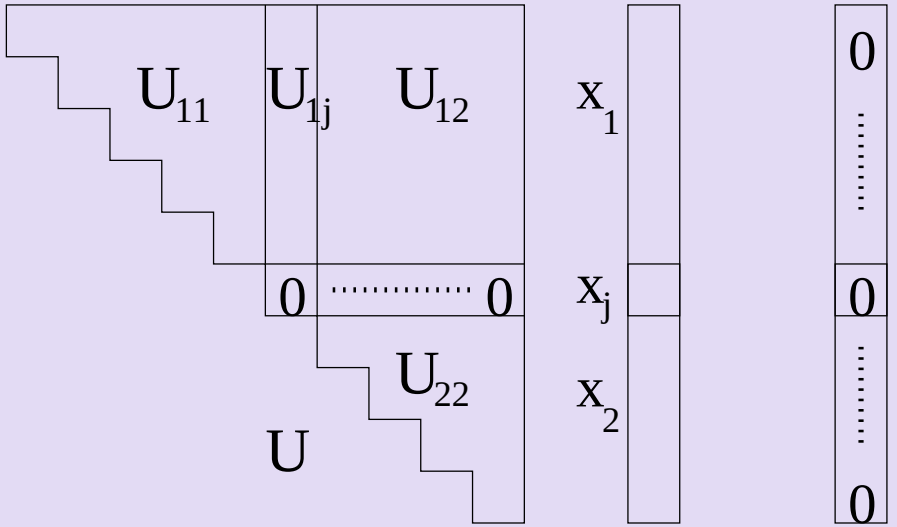
$$\begin{aligned} Ax &= 0 \\ A = LU &\quad , \quad LUx = 0 \\ det(L) \neq 0 &\quad , \quad Ux = 0 \end{aligned}$$

$$\begin{cases} U_{11}x_1 + U_{1j}x_j + U_{22}x_2 = 0 \\ 0x_j + 0x_2 = 0 \\ U_{22}x_2 = 0 \end{cases}$$

$x_2 = 0 \quad , \quad x_j = cnst = 1$
$U_{11}x_1 + U_{1j}x_j = 0$

$\forall x_j, \quad x_1 \; / \quad U_{11}x_1 = -U_{1j}x_j$

Null Space Computations



$Ax = 0$ such that $x = \begin{pmatrix} x_1 \\ 1 \\ 0 \end{pmatrix}$

$U_{11}x_1 = -U_{1j}$

Application I : Null Space Compuation

$$\begin{array}{rcl} Ax & = & 0 \\ A = LU & , & LUx = 0 \\ det(L) \neq 0 & , & Ux = 0 \end{array}$$

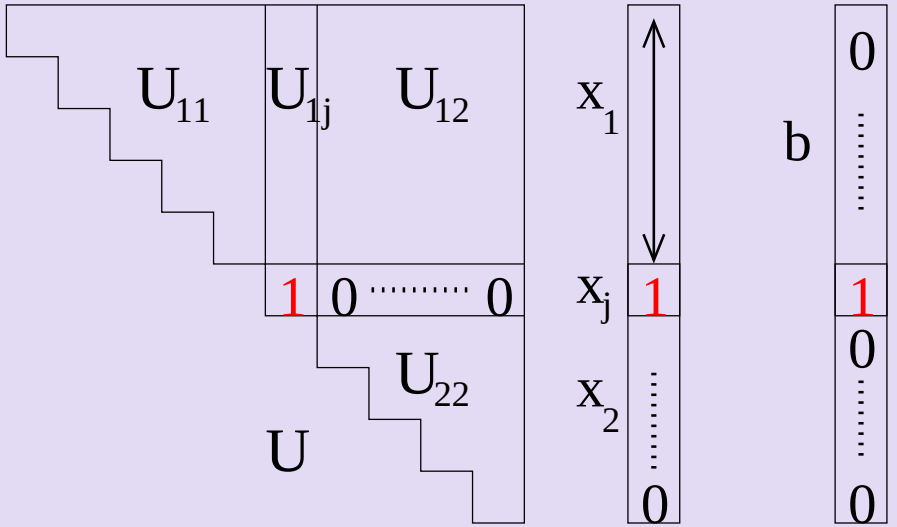
$$\begin{cases} U_{11}x_1 + U_{1j}x_j + U_{22}x_2 = 0 \\ 0x_j + 0x_2 = 0 \\ U_{22}x_2 = 0 \end{cases}$$

In MUMPS :

$$\begin{array}{lll} U_{jj} \leftarrow 1 & , & x_2 = 0 \\ b \leftarrow e_j & , & x_j = 1 \end{array}$$

Solve : $Ux = e_j$

Null Space Computations



$Ax = 0$ such that $x = \begin{pmatrix} x_1 \\ 1 \\ 0 \end{pmatrix}$

$U_{11}x_1 = -U_{1j}$

Application I : Null Space Computation

In practice we solve

$$AX = 0$$

where NbColumns(X) is the deficiency of A (≥ 1)

When deficiency is large (ex : electromagnetism $N = 33000$, deficiency=4400) we need to compute X by blocks (working memory limitation)
$X = \{X_1, \ ... \ X_b\}$ – partition of b blocks of size Nb

Property 1 :

factors loaded $(X_j) = \cup_{x \in X_j}$ {factors loaded for $Ax = 0$}

Question :

How to partition X to minimize the total size of factors loaded ?

Null Space Computations : Simple Test

Solve $AX = 0$, $X = \{x_1, \ldots, x_b\}$ – partition on b blocks of size Nb

Property 2 :

$Nb =$ Deficiency , factors loaded $= |U|$

$Nb = 1$, factors loaded $= |U| * k$ (if sparsity not exploited)

Box-cave=8x5x3 Order=619 NZ=3 471 X – order of 56

Total Factors Loaded	Solve a block of N columns at once			
	Nb= 1	Nb = 10	Nb = 16	Nb = 56
with ES	[MB] : 4, 276	0,554	0,409	0,144
with no es	[MB] : 8, 088	0,866	0,577	0,144

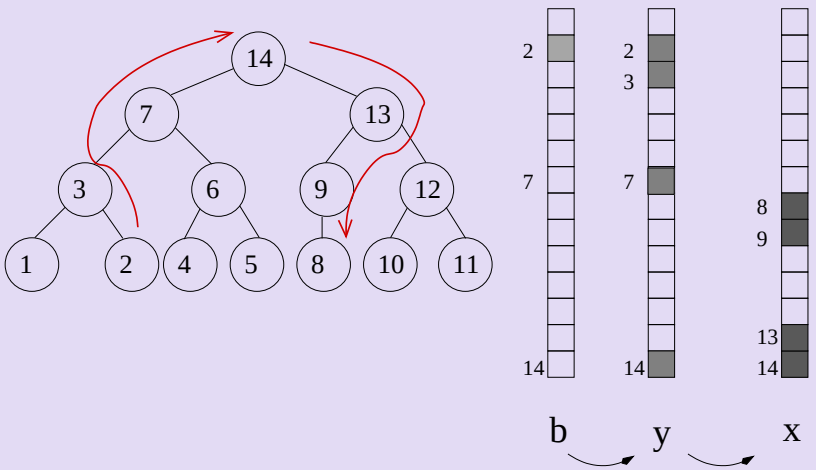
Null Space in OOC : with/without Exploit Sparsity, $\mathbf{Nb = 16}$

Matrix name	Order	Non-zero	Defic.	Time	
				no es [s]	ES [s]
16x10x3	2 675	15 953	270	1.90	0.80
30x20x4	14 454	89 185	1 653	95.98	16.21
40x27x5	33 627	212 883	4 056	818.66	181.89

Application II : compute elements in A^{-1}

- ⋆ Modelization of the problem is similar

some elements in A^{-1}



$$a_{28}^{-1} = ?$$

We need to load :
- L factors associated with nodes $2, 3, 7, 14$
- U factors 1associated to nodes $14, 13, 9, 8$.

- ⋆ Case of Multiple RHS

Conclusions

- On going works : **Multiple RHS**
  - How to partition the RHS efficiently (hypergraphs)

    joined work with Bora Ucar, post-doctoral researcher at CERFACS and futur CR CNRS at ENS-Lyon-LIP
  - Parallel scheduler to optimize loading data