Science & Technology Facilities Council

Scalability of Sparse Direct Codes

Iain Duff and NLAFET Team

STFC Rutherford Appleton Laboratory
also still at
CERFACS, Toulouse, France

University of Tennessee at Knoxville. Lunchtime talk. May 20 2016.

NLAFET

Parallel Numerical Linear Algebra for Future Extreme Scale Systems

- H2020 FET-HPC Project 671633
- Funding of around 4M Euros
- Partners are
  - University of Umeå, Sweden .. Project leader
  - University of Manchester, UK
  - INRIA, Paris, France
  - RAL-STFC, UK

NLAFET

This talk is mainly concerned with WorkPackage 3.

T3.1 Lower Bounds on Communication for Sparse Matrices

T3.2 Direct Methods for (Near-)Symmetric Systems

T3.3 Direct Methods for Highly Unsymmetric Systems

T3.4 Hybrid Direct-Iterative Methods

Outline

- NLAFET
- Direct methods
- Runtime systems
- Dense linear solution - DAGS
- Sparse linear solution - more parallelism
- Highly unsymmetric matrices
- Hybrid direct-iterative

We wish to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater and we want our codes to scale well on parallel computers.

We wish to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater and we want our codes to scale well on parallel computers.

There are two main techniques

We wish to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater and we want our codes to scale well on parallel computers.

There are two main techniques

- Direct methods (based on matrix factorization)

We wish to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater and we want our codes to scale well on parallel computers.

There are two main techniques

- Direct methods (based on matrix factorization)
- Iterative methods (with some form of preconditioning)

Direct methods

We will consider the factorizations:

$$\mathbf{P_r A P_c} \rightarrow \mathbf{LU}$$

L : Lower triangular (sparse)
U : Upper triangular (sparse)

and

$$\mathbf{P_r A P_c} \rightarrow \mathbf{QR}$$

Q : Orthogonal (sparse factors)
R : Upper triangular (sparse)

Permutations $\mathbf{P_r}$ and $\mathbf{P_c}$ chosen to preserve sparsity and maintain stability

Direct methods

- Black boxes available

Direct methods

- Black boxes available
- Complexity can be low. Almost linear storage in 2D

Direct methods

- Black boxes available
- Complexity can be low. Almost linear storage in 2D
- Routinely solving problems of order in millions

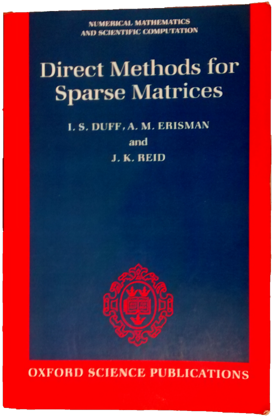
Direct methods

- Black boxes available
- Complexity can be low. Almost linear storage in 2D
- Routinely solving problems of order in millions
- Run at half asymptotic speed of machine

Direct methods

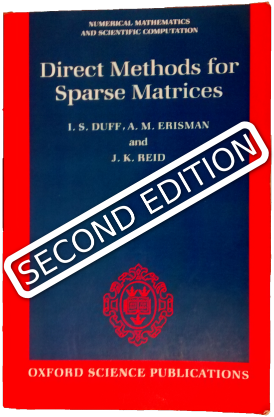
- Black boxes available
- Complexity can be low. Almost linear storage in 2D
- Routinely solving problems of order in millions
- Run at half asymptotic speed of machine
- There can be issues with storage requirement

Direct methods



Published by OUP in 1986

Direct methods



In production with OUP last month. Should appear within five months.

Iterative methods

Iterative methods

Kernel is $\mathbf{Y} \Leftarrow \mathbf{AX}$

Iterative methods

Kernel is $\mathbf{Y} \Leftarrow \mathbf{AX}$

which will not concern us here

Runtime systems

- Runtime systems are not so new.

Runtime systems

- Runtime systems are not so new.
- Here are two slides of runs I did using a system written by Dongarra and Sorensen in 1980s called Schedule.

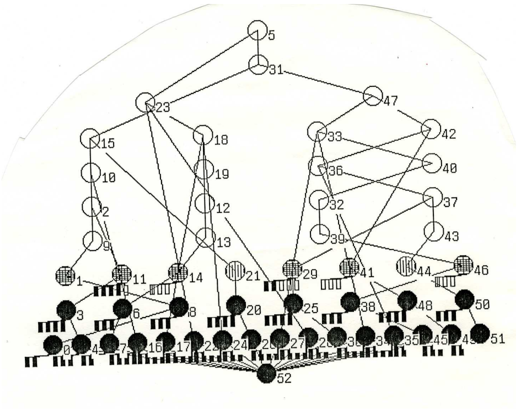
Runtime systems

- Runtime systems are not so new.
- Here are two slides of runs I did using a system written by Dongarra and Sorensen in 1980s called Schedule.
- Schedule: A Tool for Developing and Analyzing Parallel Fortran Programs, J. Dongarra and D. Sorensen, The Characteristics of Parallel Algorithms, L. H. Jamieson, D. B. Gannon, R. J. Douglass, eds., pp. 363-395, MIT Press: Cambridge, MA, 1987, ISBN 0-262-10036-3.

Runtime systems

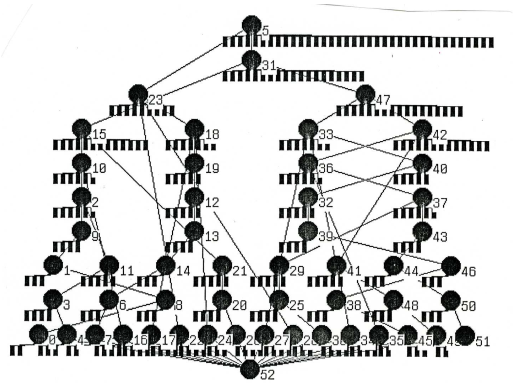
- Runtime systems are not so new.
- Here are two slides of runs I did using a system written by Dongarra and Sorensen in 1980s called Schedule.
- Schedule: A Tool for Developing and Analyzing Parallel Fortran Programs, J. Dongarra and D. Sorensen, The Characteristics of Parallel Algorithms, L. H. Jamieson, D. B. Gannon, R. J. Douglass, eds., pp. 363-395, MIT Press: Cambridge, MA, 1987, ISBN 0-262-10036-3.
- Slides were for a parallel multifrontal factorization.

Runtime systems



- Replay of Schedule run part-way through factorization

Runtime systems



- Replay of Schedule run at end of factorization

Runtime systems

- Within the framework of NLAFET, we are primarily concerned with the Runtime systems
  - StarPU using an STF (sequential task flow) model, and
  - PaRSEC using PTG (parametrized task graph) model
- In both cases, the structure involved is a directed acyclic graph (DAG)

Runtime systems

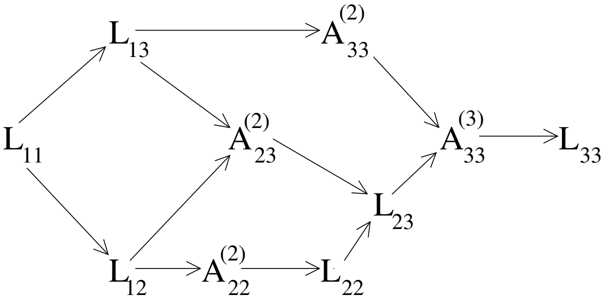
- Within the framework of NLAFET, we are primarily concerned with the Runtime systems
  - StarPU using an STF (sequential task flow) model, and
  - PaRSEC using PTG (parametrized task graph) model
- In both cases, the structure involved is a directed acyclic graph (DAG)

We could ask the question why?

DAG .. Directed Acyclic Graph

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{12}^T & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33} \end{bmatrix}$$

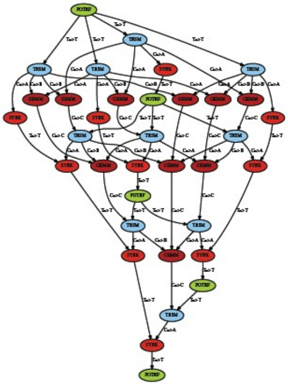
Matrix

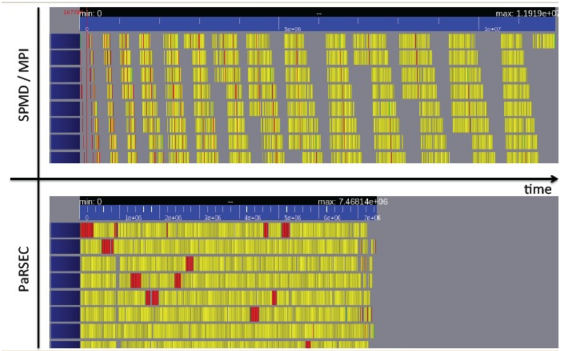


DAG for Cholesky factorization of matrix

From Duff, Erisman, and Reid (2016)

Dense systems .. Cholesky factorization



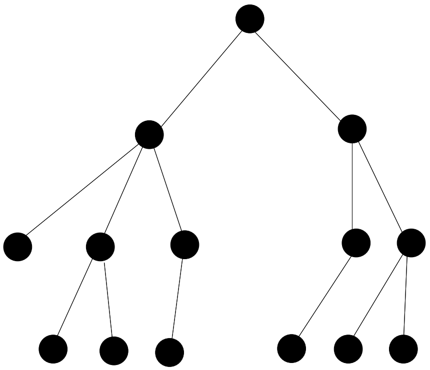


- DAG expressing algorithmic data flow
- PTG representation

- Using PaRSEC runtime system

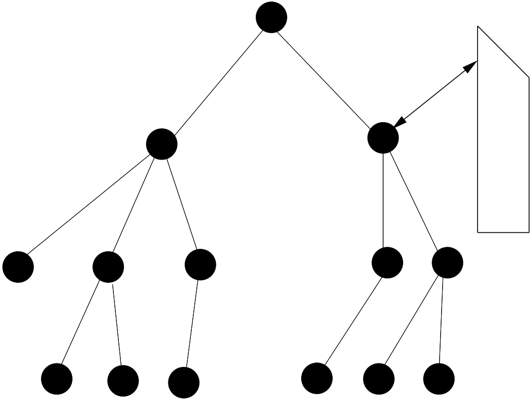
From NLAFET poster

Dense kernels in sparse factorization



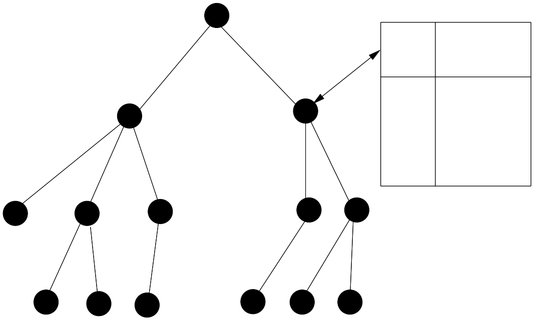
Assembly tree

Dense kernels in sparse factorization



Typical supernodal kernel

Dense kernels in sparse factorization



Typical multifrontal kernel

Sparse factorization

So we have all the parallelism and tricks that are used for dense systems.

Sparse factorization

So we have all the parallelism and tricks that are used for dense systems.

But we also have ....

Sparse factorization

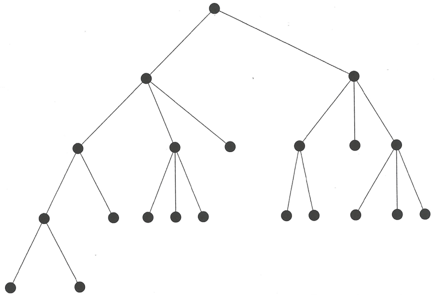
So we have all the parallelism and tricks that are used for dense systems.

But we also have ....

Tree parallelism

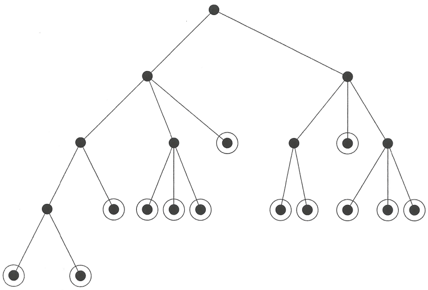
Tree parallelism

Tree parallelism



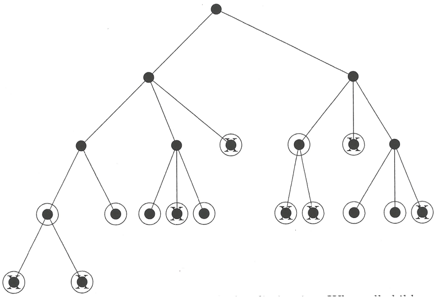
Assembly tree.

Tree parallelism



Available nodes at start of factorization. Work corresponding to leaf nodes can proceed immediately and independently.

Tree parallelism



Situation part way through the elimination. When all children of a node complete then work can commence at parent node.

Node and tree parallelism

		Tree	Leaf nodes		Top 3 levels		
Matrix	Order	nodes	No.	Av. size	No.	Av. size	% ops
bratu3d	27 792	12 663	11 132	8	296	37	56
cont-300	180 895	90 429	74 673	6	10	846	41
cvxqp3	17 500	8 336	6 967	4	48	194	70
mario001	38 434	15 480	8 520	4	10	131	25
ncvxqp7	87 500	41 714	34 847	4	91	323	61
bmw3_2	227 362	14 095	5 758	50	11	1 919	44

Statistics on front sizes in assembly tree. From Duff, Erisman, Reid (2016).

Node and tree parallelism

- Although there are many tasks near the leaf nodes, we note that the dimensions of the matrices are small.

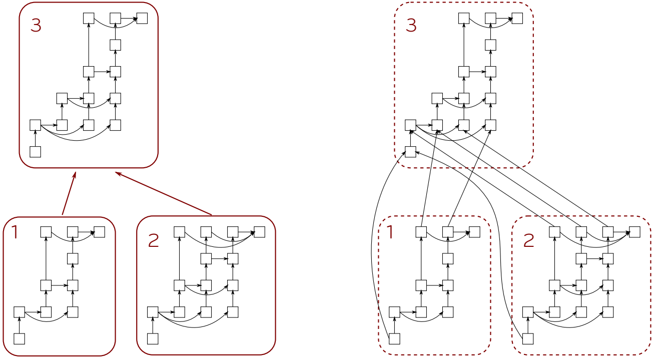
Node and tree parallelism

- Although there are many tasks near the leaf nodes, we note that the dimensions of the matrices are small.
- We are thus looking at the use of batched BLAS where BLAS operations on small matrices are combined.

However, our DAG-based algorithms can extract more parallelism between children and parent nodes.

We illustrate this in the following slide from Florent Lopez' thesis.

Inter-node parallelism



Extra edges in DAG to show inter-node parallelism on the right

Sparse parallelism

Sparse parallelism

- There are several levels of parallelism in sparse systems

Sparse parallelism

- There are several levels of parallelism in sparse systems
  - Partitioning ... block diagonal or triangular form ... see later

Sparse parallelism

- There are several levels of parallelism in sparse systems
  - Partitioning ... block diagonal or triangular form ... see later
  - Tree level parallelism

Sparse parallelism

- There are several levels of parallelism in sparse systems
  - Partitioning ... block diagonal or triangular form ... see later
  - Tree level parallelism
  - Node parallelism (including multi-threaded BLAS)

Sparse parallelism

- There are several levels of parallelism in sparse systems
  - Partitioning ... block diagonal or triangular form ... see later
  - Tree level parallelism
  - Node parallelism (including multi-threaded BLAS)
  - Inter-node parallelism

Task 3.2 Direct Methods for (Near-)Symmetric Systems

Sparse LL^T, LDL^T, LU

- Tree-based solvers
- Targeting scalability
- Reduce communication
- Use DAGs with runtime scheduling systems from WP6

Goals:

- Design energy-efficient, low-communication dense kernels for use within multifrontal or supernodal sparse factorizations both for positive definite and indefinite systems
- Design novel fine-grained parallel algorithms based on DAGs rather than trees
- Develop interactive and novel ways of using mixed precision within the sparse factorization process

Use of runtime system

Matrix	spLLT	MA87
	Factorization time (secs)	
GHS_psdef/apache2	1.848	**0.717**
Koutsovasilis/F1	0.920	**0.786**
Oberwolfach/boneS10	1.599	**1.111**
ND/nd12k	**1.405**	1.498
JGD_Trefethen/Trefethen_20000	**2.406**	3.829
ND/nd24k	**5.076**	5.498
Oberwolfach/bone010	7.392	**7.195**
GHS_psdef/audikw_1	10.680	**10.642**

Factorization of sparse symmetric positive-definite matrices.
Runs comparing StarPU code (spLLT) with hand-tuned HSL code MA87.
Runs on scarf .... Haswell Intel Xeon E5-2695 v3, 2.3 GHz, 2 $\times$ 14-core.

Use of runtime system

Matrix	spLLT				MA87	
	OpenMP (gnu)		StarPU		MA87	
	nb	facto (s)	nb	facto (s)	nb	facto (s)
Schmid/thermal2	512	1.801	1024	2.123	256	**0.376**
Rothberg/gearbox	256	**0.220**	384	0.318	256	0.252
DNVS/m_t1	256	0.205	384	0.262	256	**0.194**
DNVS/thread	256	**0.203**	384	0.240	256	0.213
DNVS/shipsec1	256	**0.247**	384	0.363	256	0.259
GHS_psdef/crankseg_2	256	0.267	384	0.310	256	**0.257**
AMD/G3_circuit	512	2.631	512	3.345	256	**0.586**
Koutsovasilis/F1	384	0.812	512	0.920	256	**0.786**
Oberwolfach/boneS10	384	1.186	384	1.599	256	**1.111**
ND/nd12k	384	1.478	384	**1.405**	384	1.498
JGD_Trefethen/Trefethen_20000	512	3.692	384	**2.406**	512	3.829
ND/nd24k	384	5.379	384	**5.076**	384	5.498
Oberwolfach/bone010	384	7.416	768	7.392	384	**7.195**
GHS_psdef/audikw_1	768	10.650	768	10.680	384	**10.642**

Factorization of sparse symmetric positive-definite matrices.

Runs with block sizes nb = (256, 384, 512, 768, 1024) on 28 cores and nemin=32.

Illustration using sparse QR code

We now illustrate some of the points discussed earlier using results from the thesis work of our NLAFET postdoc Florent Lopez.

Illustration using sparse QR code

We now illustrate some of the points discussed earlier using results from the thesis work of our NLAFET postdoc Florent Lopez.

Task-based multifrontal QR solver for heterogeneous architectures.
PhD Thesis, UPS, Toulouse, France.
December 2015.

Illustration using sparse QR code

We now illustrate some of the points discussed earlier using results from the thesis work of our NLAFET postdoc Florent Lopez.

Task-based multifrontal QR solver for heterogeneous architectures.
PhD Thesis, UPS, Toulouse, France.
December 2015.

Implementing multifrontal sparse solvers for multicore architectures with Sequential Task Flow runtime systems.
Emmanuel Agullo, Alfredo Buttari, Abdou Guermouche, and Florent Lopez.
To appear in ACM TOMS
`http://buttari.perso.enseeiht.fr/stuff/IRI-RT--2014-03--FR.pdf`

Illustration using sparse QR code

We now illustrate some of the points discussed earlier using results from the thesis work of our NLAFET postdoc Florent Lopez.

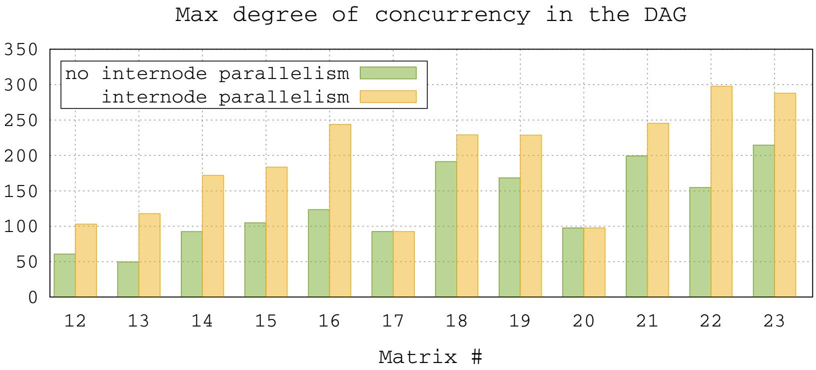
Task-based multifrontal QR solver for heterogeneous architectures.
PhD Thesis, UPS, Toulouse, France.
December 2015.

Implementing multifrontal sparse solvers for multicore architectures with Sequential Task Flow runtime systems.
Emmanuel Agullo, Alfredo Buttari, Abdou Guermouche, and Florent Lopez.
To appear in ACM TOMS
`http://buttari.perso.enseeiht.fr/stuff/IRI-RT--2014-03--FR.pdf`

Code called `qr_mumps`

Inter-node parallelism

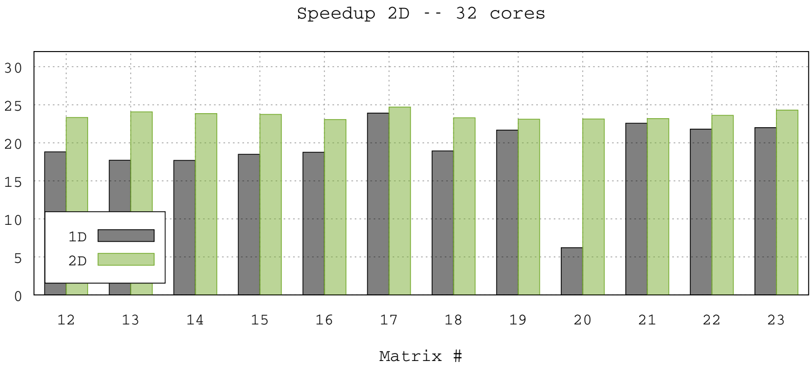
Inter-node parallelism



- Maximum speedup is given by $\frac{\sum_{i \in DAG} w_i}{\sum_{i \in CP} w_i}$ where CP is critical path and weight is execution time on one thread.
- In yellow, DAG can schedule tasks at parent node before completion of children.

Use of CA algorithms

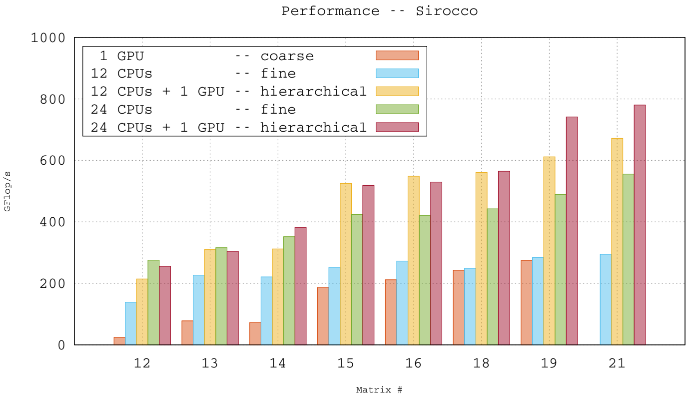
Use of CA algorithms



- Runs on Ada at IDRIS supercomputing centre (IBM x3750-M4)
- Intel Sandy Bridge E5-4650, 2.7 GHz, 4 8-core nodes, 128 GB NUMA memory
- 2D distribution at each node to use communication avoiding algorithms

Using StarPU in heterogeneous environment

Using StarPU in heterogeneous environment



- Shows performance in heterogeneous environment
- Runs on Sirocco at Plafrim supercomputer centre, Bordeaux.
- Haswell Intel Xeon E5-2680, 2.5 GHz, 2 12-core nodes, 128 GB NUMA memory, 4 NVIDIA K40 GPUs

Task 3.3 Direct Methods for Highly Unsymmetric Systems

Sparse *LU*

- Non-tree-based method
- Parallel orderings based on threshold Markowitz
- Extensive use of blocking
- Reduction of communications

Goals:

- Develop parallel versions of algorithms and prototype software for the factorization of highly unsymmetric sparse matrices

Highly unsymmetric systems

- These do occur in, for example ...

Highly unsymmetric systems

- These do occur in, for example ...
  - chemical engineering

Highly unsymmetric systems

- These do occur in, for example ...
  - chemical engineering
  - linear programming

Highly unsymmetric systems

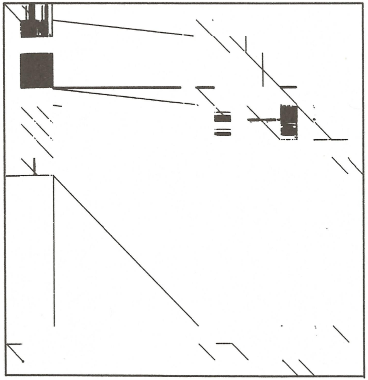
- These do occur in, for example ...
  - chemical engineering
  - linear programming
  - economic modelling

Highly unsymmetric systems

- These do occur in, for example ...
  - chemical engineering
  - linear programming
  - economic modelling

Both code and matrices can be very evil

Highly unsymmetric systems



Matrix from econometric model of SE Asia

Highly unsymmetric systems

```
    DO 590 JJ = J1, J2
       J = ICN (JJ)
       IF (IQ(J). GT.0) GO TO 590
       IOP = IOP + 1
       PIVROW = IJPOS - IQ (J)
       A(JJ) = A(JJ) + AU x A (PIVROW)
       IF (LBIG) BIG = DMAXI (DABS(A(JJ)), BIG)
       IF (DABS(A(JJ)). LT. TOL) IDROP = IDROP + 1
       ICN (PIVROW) = ICN (PIVROW)
590 CONTINUE
```

Innermost loop of early version of MA48

Highly unsymmetric systems

Let me know if you have good ideas for these systems!!

Task 3.4 Hybrid Direct-Iterative Methods

Block projection methods

- Targeting scalability
- Extension to overdetermined systems
- Intelligent partitionings
- Links to WP 4.3

Goals:

- To determine bottlenecks to extreme scalability of block iterative methods and to redesign the constituent algorithms to achieve high scalability in prototype software
- To extend this work to include solution of saddle-point problems and overdetermined systems

Block Cimmino method

Block Cimmino method

- A stationary iterative method that solves linear systems using a row projection technique

Block Cimmino method

- A stationary iterative method that solves linear systems using a row projection technique
- Wonderfully parallel

Block Cimmino method

- A stationary iterative method that solves linear systems using a row projection technique
- Wonderfully parallel
- We will use it to enhance parallelism of direct method in hierarchical fashion

Block Cimmino method

- A stationary iterative method that solves linear systems using a row projection technique
- Wonderfully parallel
- We will use it to enhance parallelism of direct method in hierarchical fashion

Early work on this by Elfving (1980) and Sameh, Kamath, and Bramley (1988-1992). We worked on this at CERFACS in the early 1990s with Mario Arioli, Tony Drummond, Joseph Noailles, Daniel Ruiz, and Miloud Sadkane.

More recently, further work was done by Mohamed Zenadi for his thesis at ENSEEIHT.

Block Cimmino method

$$Ax = b \text{ is partitioned as } \begin{pmatrix} A^1 \\ A^2 \\ . \\ . \\ . \\ A^P \end{pmatrix} x = \begin{pmatrix} b^1 \\ b^2 \\ . \\ . \\ . \\ b^P \end{pmatrix}$$

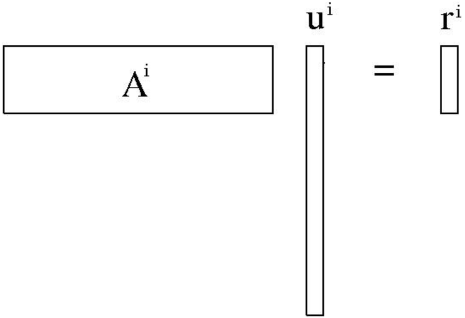
and then the algorithm computes a solution iteratively from an initial estimate $x^{(0)}$ according to:

$$\begin{aligned} u^i &= A^{i^+}\left(b^i - A^i x^{(k)}\right) \quad i = 1, \ldots .P \\ x^{(k+1)} &= x^{(k)} + \omega \sum_{i=1}^{P} u^i \end{aligned}$$

Note independence of set of P equations

Underdetermined systems

The shape of these subproblems is:



Block Cimmino method

We choose to solve

$$A^i u^i \; = \; r^i, \qquad (r^i = b^i - A^i x^{(k)})$$

using the augmented system

$$\begin{pmatrix} I & {A^i}^T \\ A^i & 0 \end{pmatrix} \begin{pmatrix} u^i \\ v^i \end{pmatrix} = \begin{pmatrix} 0 \\ r^i \end{pmatrix}$$

We will solve these augmented systems using a direct method (MUMPS).

The partitioning can be used to make the approach direct (one partition), iterative one (single row partition), or hybrid.

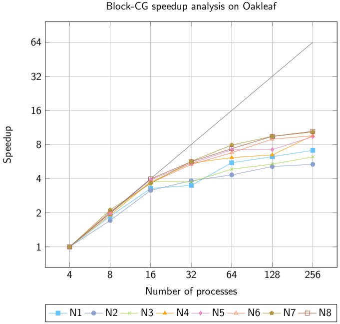
Hence my view of thinking of hybrid methods as extensions to direct methods rather than as a preconditioning of an iterative method. In particular it adds another possibility for parallelism.

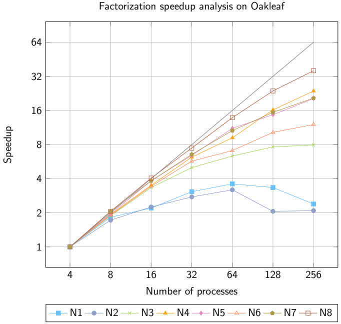
Block Cimmino: distributed results

Problems (from Tim Davis' University of Florida collection except EDF/R6):

Problem	Order	Nonzeros	Parts	BS	Nb.Iter
N1:EDF/R6	132,106	2,103,332	16	16	846
N2:lhr71c	70,304	1,528,092	16	32	257
N3:torso3	259,156	4,429,042	32	1	22
N4:Hamrle3	1,447,360	5,514,242	64	4	745
N5:Hamrle3	1,447,360	5,514,242	128	4	868
N6:cage13	445,315	7,479,343	256	1	14
N7:cage14	1,505,785	27,130,349	1024	1	14
N8:nlpkkt80	1,062,400	28,192,672	256	4	1473

Hybrid Block Cimmino: Distributed results





Fujitsu FX10 configuration at Tokyo University with each node having one Fujitsu Sparc64-IXfx (16 cores) processor and 32 GBytes memory

Block Cimmino vs MUMPS

For most problems MUMPS is fastest

Block Cimmino vs MUMPS

For most problems MUMPS is fastest

BUT ...

Block Cimmino vs MUMPS

64 mpi-processes and 16 thread per mpi-process

Problem	Factorization times (seconds)	
	MUMPS	**BC**
Cage13	76	2.8
Cage14	F	5.8
Hamrle3	208.4	4.7

Block Cimmino vs MUMPS

64 mpi-processes and 16 thread per mpi-process

Problem	Factorization times (seconds)	
	MUMPS	**BC**
Cage13	76	2.8
Cage14	F	5.8
Hamrle3	208.4	4.7

But the main gain of block Cimmino is in memory requirements

Block Cimmino vs MUMPS

64 mpi-processes and 16 thread per mpi-process

Problem	Factorization times (seconds)	
	MUMPS	**BC**
Cage13	76	2.8
Cage14	F	5.8
Hamrle3	208.4	4.7

Problem	Memory per node	
	MUMPS	**BC**
Cage13	2.8 GB	
Cage14	30 GB	
Hamrle3	462 MB	

Block Cimmino vs MUMPS

64 mpi-processes and 16 thread per mpi-process

Problem	Factorization times (seconds)	
	MUMPS	**BC**
Cage13	76	2.8
Cage14	F	5.8
Hamrle3	208.4	4.7

Problem	Memory per node	
	MUMPS	**BC**
Cage13	2.8 GB	37 MB
Cage14	30 GB	102 MB
Hamrle3	462 MB	53 MB

Towards a new unit of performance

Towards a new unit of performance

- elapsed time

Towards a new unit of performance

- elapsed time
- real time

Towards a new unit of performance

- elapsed time
- real time
- real elapsed time

Towards a new unit of performance

- elapsed time
- real time
- real elapsed time
- wall-clock time

Towards a new unit of performance

- elapsed time
- real time
- real elapsed time
- wall-clock time
- execution time

Towards a new unit of performance

- elapsed time
- real time
- real elapsed time
- wall-clock time
- execution time
- makespan

Towards a new unit of performance

- elapsed time
- real time
- real elapsed time
- wall-clock time
- execution time
- makespan
- a new measure ...

Towards a new unit of performance

- elapsed time
- real time
- real elapsed time
- wall-clock time
- execution time
- makespan
- a new measure ...
  - surreal time

Towards a new unit of performance

- elapsed time
- real time
- real elapsed time
- wall-clock time
- execution time
- makespan
- a new measure ...
  - surreal time
  - Surreal (OED)
    Represents and interprets the phenomena of dreams and similar experiences

THANK YOU FOR YOUR ATTENTION

Sparse Days at CERFACS

Deadline for (free) registration is 31 May 2016.

At CERFACS in Meteopole in Toulouse, France.

Thursday 30 June and Friday 1 July. Banquet on Thursday.

http://cerfacs.fr/en/sparse-days-meeting-2016-at-cerfacs-toulouse-5/
or (in French) on:
http://cerfacs.fr/colloque-sparse-days-2016-cerfacs-toulouse/