Sparse Direct Solvers for Extreme-Scale Computing

Iain Duff
Joint work with Florent Lopez and Jonathan Hogg

STFC Rutherford Appleton Laboratory

SIAM Conference on Computational Science and Engineering (CSE17). Hilton Atlanta, Atlanta, Georgia, USA. 27 February - 3 March 2017.
H2020 FET-HPC Project 671633
- Funding of around 4M Euros
- Partners are
  - Umeå University, Sweden .. Coordinator: Bo Kågström
  - University of Manchester, UK .. Jack Dongarra
  - INRIA, Paris, France .. Laura Grigori
  - STFC, UK .. Iain Duff
- Started 1 November 2015 (effectively Jan 2016)
- 36 months project terminating on 31 October 2018
NLAFET Work Packages

- WP1: Management and coordination
- WP2: Dense linear systems and eigenvalue problem solvers
- WP3: Direct solution of sparse linear systems
- WP4: Communication-optimal algorithms for iterative methods
- WP5: Challenging applications— a selection
  Material science, power systems, study of energy solutions, and data analysis in astrophysics
- WP6: Cross-cutting issues
  Scheduling and runtime systems, auto-tuning, fault tolerance
- WP7: Dissemination and community outreach
T3.1 Lower Bounds on Communication for Sparse Matrices
T3.2 Direct Methods for (Near-)Symmetric Sparse Systems
T3.3 Direct Methods for Highly Unsymmetric Sparse Systems
T3.4 Hybrid Direct-Iterative Methods
Solution of sparse linear systems

We wish to solve the sparse linear system

\[ Ax = b \]

where the sparse matrix \( A \) is sparse and of large dimension, typically \( 10^6 \) or greater, and we want to solve the system on an extreme scale computer.
Direct methods

We will consider the factorization:

\[ P_r A P_c \rightarrow L U \]

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

Permutations \( P_r \) and \( P_c \) chosen to preserve sparsity and maintain stability

When \( A \) is symmetric \( U = DL^T \) and \( P_c = P_r^T \)
Sparse direct methods

- Black boxes available
Sparse direct methods

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

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

- Black boxes available
- Complexity can be low. Almost linear storage in 2D
- Routinely solving problems of order in millions
- There can be issues with storage requirement
Sparse direct methods

- Black boxes available
- Complexity can be low. Almost linear storage in 2D
- Routinely solving problems of order in millions
- There can be issues with storage requirement
- Target is half asymptotic speed of machine
Task 3.2 Direct Methods for (Near-)Symmetric Systems

**Sparse** $LL^T$, $LDL^T$, $LU$

- Tree-based solvers
- Use DAGs with runtime scheduling systems (WP6)
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
  - OpenMP Version 4.0 or above, using task features
- In all cases, we are using a task-based approach and the structure involved is a directed acyclic graph (DAG)
Sparse factorization

The kernel of most sparse direct codes is a dense factorization.

We feel it is useful to show this via a mini-tutorial.
At step 1

1. 1 2 3 6 7
2. 1 \times \times \times \times
3. 1 \times \times \times \times \times
6. 1 \times \times \times \times \times
7. 1 \times \times \times \times \times
At step 2

\[
\begin{array}{ccc}
4 & 5 & 6 \\
4 & \times & \times & \times \\
5 & \times & \times & \times \\
6 & \times & \times & \times \\
\end{array}
\]
At step 3

\[
\begin{align*}
6 & \times \times + 6 & \times \times + 6 & \times \times \rightarrow 6 & \times \times \\
7 & \times \times + 7 & \times \times + 6 & \times \times & 7 & \times \times \\
\end{align*}
\]
The computation at a node involves **dense factorization**. Pivots are chosen from the top left block in the picture below but elimination operations are performed on the whole frontal matrix. Rows and columns of the factors can be stored and the resulting **Schur complement (in bottom right) is passed up the tree** for future assemblies.
Sparse parallelism

There are several levels of parallelism in sparse systems:

- Partitioning
- Tree level parallelism
- Node parallelism (including multi-threaded BLAS)
- Inter-node parallelism
Sparse parallelism

- There are several levels of parallelism in sparse systems
There are several levels of parallelism in sparse systems

- Partitioning... block diagonal or block triangular form
- Tree level parallelism
- Node parallelism (including multi-threaded BLAS)
- Inter-node 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

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

Statistics on front sizes in assembly tree. From Duff, Erisman, Reid (2016).
Near the root there is not much tree parallelism but the nodes are large, that is the dense matrices at these nodes are of large dimension and so there is plenty of node parallelism.

Conversely, near the leaves, there is little node parallelism but plenty of tree parallelism.
Inter-node parallelism

Part of tree

Directed acyclic graph
Sparse Cholesky using runtime systems

Factorization GFlop/s - 28 cores

- MA87
- SpLLT-STF (OpenMP)
- SpLLT-STF (StarPU)
The main problem with using the runtime systems (for example matrix 15) is that when the tasks are small, the overhead in setting them up in the runtime system predominates.

We can avoid some of this by grouping together into a single task nodes near the leaves of the tree. We call this tree pruning.
Tree pruning strategy
Effect of tree pruning on OpenMP version

Factorization GFlop/s - 28 cores

GFlop/s

Matrix #

Factorization GFlop/s - 28 cores

MA87

SpLLT-STF (OpenMP)

SpLLT-STF (OpenMP) \w pruning
Runs on large three-dimensional problems

Poisson 3D - Factor. GFlop/s - 28 cores

Mesh size

GFlop/s

Poisson 3D - Factor. GFlop/s - 28 cores

MA87
SpLLT-STF (OpenMP)
SpLLT-STF w/ pruning (OpenMP)
SpLLT-STF (StarPU)
SpLLT-STF w/ pruning (StarPU)
Scalability

Poisson 3D - Factor. GFlop/s

- SpLLT-STF (OpenMP), N=60
- SpLLT-STF (StarPU), N=60
- SpLLT-STF (OpenMP), N=100
- SpLLT-STF (StarPU), N=100
- SpLLT-STF (OpenMP), N=160
- SpLLT-STF (StarPU), N=160

GFlop/s vs # Cores for different parallel approaches and problem sizes.
Symmetric indefinite matrices

If the matrix is indefinite then numerical pivoting is needed.

A simple example is the matrix

\[
\begin{bmatrix}
0 & \times \\
\times & 0
\end{bmatrix}
\]
Numerical pivoting in indefinite case

Good news is that we can stably factorize an indefinite matrix using only $1 \times 1$ and $2 \times 2$ pivots (Bunch and Kaufmann).

As is standard in sparse factorization, we use threshold rather than partial pivoting so we want ...

$$\|Pivot\| \geq u \times \|\text{Largest in column}\|$$

where $u$ is the threshold parameter ($0 < u \leq 1$).
Numerical pivoting in indefinite case

Pivots can only be chosen from $A_{11}$.

Can restrict pivoting to $A_{11}$ or only choose pivots from $A_{11}$ but then fail if large entry in $A_{21}$ causes pivot to fail test.

Is a real problem when implementing algorithm on parallel machine.
Threshold Partial Pivoting

**TPP Algorithm**
- Work column by column
- Bring column up-to-date
- Find maximum element $\alpha$ in column of $A_{21}$
- Pivot test $\frac{\alpha}{a_{11}} < u^{-1}$. Accept/reject pivot

**Problems**
- Very stop-start (one column at a time)
- Communication for every column
A Posteriori Pivoting

- Work by blocks of $A_{21}$
- Every block uses factors of $A_{11}$
- Every block checks $\max |l_{21}| < u^{-1}$
- Communication when all blocks are done
- Discard all columns to right of failed entry
- Scaling and ordering $\Rightarrow$ failed pivots are rare
Flop rate

Machine achieves 794 Gflop/s on LINPACK
Compare new code **SSIDS** with standard TPP code **HSL_MAA97** and with **PARDISO** as implemented in MKL.
Hard indefinite on 28-core machine
Numerical pivoting in indefinite case

- **HSL_MA97** Threshold partial pivoting
- **PARDISO** Only chooses pivots from $A_{11}$
- **SSIDS** Uses a posteriori pivoting
Numerical pivoting in indefinite case

- **HSL_MA97** Threshold partial pivoting
- **PARDISO** Only chooses pivots from $A_{11}$
- **SSIDS** Uses a posteriori pivoting

I parodize these as
Numerical pivoting in indefinite case

- **HSL_MAA97** Threshold partial pivoting
- **PARDISO** Only chooses pivots from $A_{11}$
- **SSIDS** Uses a posteriori pivoting

I parodize these as

- **HSL_MAA97 PAY**
Numerical pivoting in indefinite case

- **HSL_MA97** Threshold partial pivoting
- **PARDISO** Only chooses pivots from $A_{11}$
- **SSIDS** Uses a posteriori pivoting

I parodize these as

- **HSL_MA97** PAY
- **PARDISO** PRAY
Numerical pivoting in indefinite case

- **HSL_MAA97** Threshold partial pivoting
- **PARDISO** Only chooses pivots from $A_{11}$
- **SSIDS** Uses a posteriori pivoting

I parodize these as

- **HSL_MAA97** PAY
- **PARDISO** PRAY
- **SSIDS** PLAY
## Hard indefinite on 28-core machine

<table>
<thead>
<tr>
<th>Matrix</th>
<th>stokes128</th>
<th>cvxqp3</th>
<th>ncvxqp7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order $\times 10^3$</td>
<td>49.7</td>
<td>17.5</td>
<td>87.5</td>
</tr>
<tr>
<td>Entries $\times 10^6$</td>
<td>0.30</td>
<td>0.07</td>
<td>0.31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor time</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSL_MA97</td>
</tr>
<tr>
<td>PARDISO</td>
</tr>
<tr>
<td>SSIDS V2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Backward error</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSL_MA97</td>
</tr>
<tr>
<td>PARDISO</td>
</tr>
<tr>
<td>SSIDS V2</td>
</tr>
</tbody>
</table>
Summary

- There is lots of parallelism in sparse direct solvers
Summary

- There is lots of parallelism in sparse direct solvers
- Using a runtime system can compete with hand-coded versions
Summary

- There is lots of parallelism in sparse direct solvers
- Using a runtime system can compete with hand-coded versions
- Pivoting can be accommodated with little overhead in performance
Summary

- There is lots of parallelism in sparse direct solvers
- Using a runtime system can compete with hand-coded versions
- Pivoting can be accommodated with little overhead in performance
- We meet our goal of running at half the peak performance
Summary

- There is lots of parallelism in sparse direct solvers
- Using a runtime system can compete with hand-coded versions
- Pivoting can be accommodated with little overhead in performance
- We meet our goal of running at half the peak performance
- Programming this is tough
THANK YOU FOR YOUR ATTENTION
Direct methods

Published by OUP in 1986
Direct methods

Numerical Mathematics and Scientific Computation is a series designed to provide texts and monographs for graduate students and researchers in a wide range of mathematical topics at the interface of computational science and numerical analysis.

Direct Methods for Sparse Matrices
I. S. Duff, A. M. Erisman, and J. K. Reid

This book is concerned with solving very large sets of linear equations, where each equation involves only a small number of variables. Many applications involve equations of this kind and they often need to be solved repeatedly, as their entries change. Very special methods are needed to make these calculations feasible. The authors have been involved in designing special algorithms and writing codes to implement them for over 40 years. This book aims to describe those algorithms that have stood the test of time, as well as those that have been developed recently to enable the efficient solution of far larger systems and to take advantage of modern hardware.

Direct Methods for Sparse Matrices, second edition, is a complete rewrite of the first edition, which was published 30 years ago. Much has changed since that time. Problems have grown greatly in size and complexity, and computer architectures are now much more complex, requiring new ways of adapting algorithms to parallel environments with memory hierarchies. Because the area is such an important one to computational science and engineering, a huge amount of research has been done since the first edition. This new research is integrated into the text with clear explanations of the underlying mathematics and algorithms.

New research described here includes new techniques for scaling and error control, new techniques for preconditioning symmetric and unsymmetric problems, and detailed descriptions of the multifrontal approach to solving systems that was pioneered by the research of the authors and colleagues. This includes a discussion of techniques for exploiting parallel architectures and new work for indefinite and unsymmetric systems.

Written to be as accessible as possible, this book will be a useful resource to a wide range of readers: from researchers and practitioners, to applications scientists, to graduate students, and their teachers.