Science & Technology Facilities Council

# A GPU Sparse Symmetric Indefinite Solver with Pivoting

**Jonathan Hogg**,
Evgueni Ovtchinnikov,
Jennifer Scott*

STFC Rutherford Appleton Laboratory

4 June 2014
Sparse Days
CERFACS, Tolouse

* Thanks also to Jeremy Appleyard of NVIDIA

# Factorization

$$A = L D L^T$$

- Sparse
- Symmetric: $A = A^T$
- Non-singular (for purposes of talk)
- **Do it on a GPU**
- Aim to be bit-compatible

Science & Technology Facilities Council

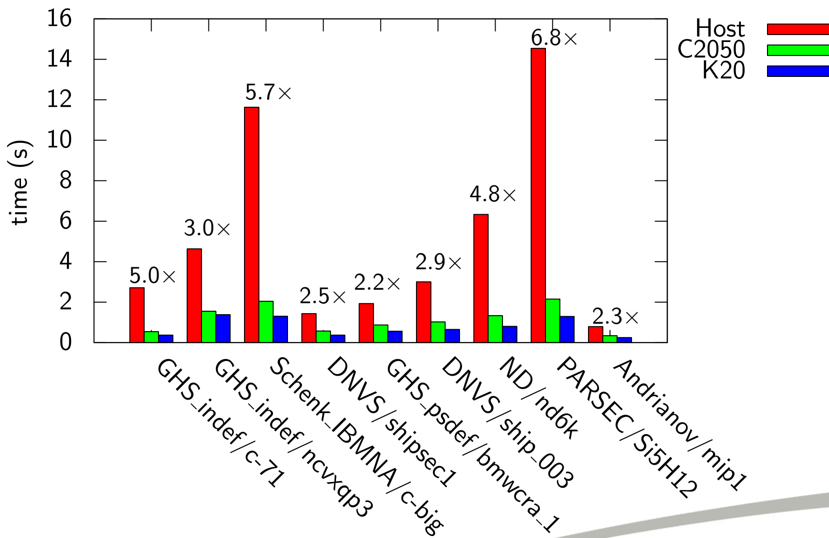
# Why GPU

| Chip | Cores | GB/sec | TFLOP/sec | GFLOPS/Watt |
|---|---|---|---|---|
| **NVIDIA K40** | $15 \times 64$ | 288 | 1.43 | 6.1 |
| **NVIDIA Titan Z** | $2 \times 15 \times 64$ | 672 | 2.66 | 7.1 |
| **AMD R9 295X2** | $2 \times 44 \times 8^*$ | 640 | 1.43 | 2.9 |
| **Intel Xeon Phi** | $60 \times 8$ | 320 | 1.00 | 4.5 |
| **Intel Desktop E5-2687W** | $16 \times 4$ | 50 | 0.40 | 1.3 |

* double precision cores. single precision is $8\times$.

Science & Technology Facilities Council

# Results preview




Science & Technology Facilities Council

# Modern direct solver design

## Four phases

**Ordering** Find fill-reducing permutation

**Analyse** Find dense submatrix structure.
Setup data representation.

**Factor** Perform factorization with pivoting.

**Solve** Use factorization to solve $Ax = b$.

Science & Technology Facilities Council

# Modern direct solver design

## Four phases

- **Ordering** Find fill-reducing permutation
- **Analyse** Find dense submatrix structure. Setup data representation.
- **Factor** Perform factorization with pivoting.
- **Solve** Use factorization to solve $Ax = b$.

## GPU Challenges

- Thousands of *small* dense subproblems (e.g. $8 \times 1$)
- Pivoting on *large* dense subproblems (e.g. $4000 \times 2000$)
- Substantial sparse scatter/gather
- Complicated kernels (register pressure)

Science & Technology Facilities Council

# Previous work

## Pre-existing work

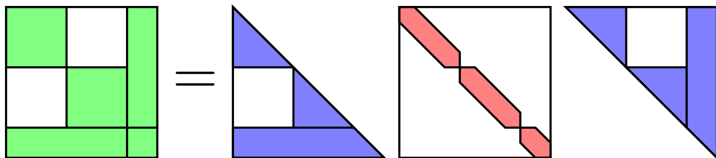
- Just offloading large BLAS 3/LAPACK operations.
  Very modest speedups on whole problem.
- A few codes go beyond this.
  None publicly available?
  No pivoting: potentially unstable
  Fairly modest speedups: CPU$\leftrightarrow$GPU bottleneck

## Our implementation

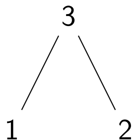
- Puts entire factorization and solve phases on GPU
- Open source, including all auxiliary codes
- Delivers over 5$\times$ speedup vs 2 CPU sockets on large problems

Science & Technology Facilities Council

# Tree parallelism

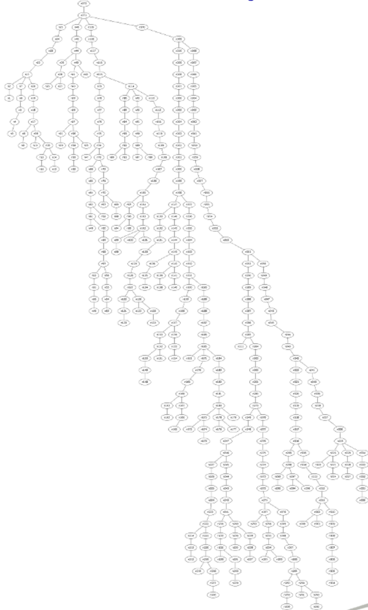



Operations in first two block columns are independent.
Data flow graph called Assembly Tree

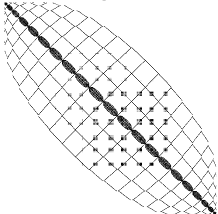



Science & Technology Facilities Council

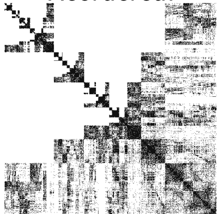
# Real world assembly tree: PARSEC/SiNa



Original:



Reordered:



Science & Technology Facilities Council

# Node parallelism

**For an individual block, in order:**

Assemble contributions from children (sparse gather)

Factor $m \times k$ matrix with threshold pivoting (partial dense $LDL^T$)

Contribution given by Schur complement (dgemm)

**Each task itself can be parallelized** (some better than others!)

Science & Technology Facilities Council

# First challenge: Exploit **both** tree and node parallelism

**Note:** CUBLAS only supports multiple BLAS on same dimensions.
⇒Have to write our own routines.

- For lots of small nodes, dominant cost is kernel setup!
- CPU populates a data structure of tasks
- Assigns an appropriate number of blocks to each task
- Launches a kernel on $\sum$ blocks

Limited registers:

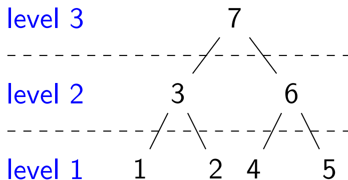
- Costs several registers to do this (can't use constant cache)

Science & Technology Facilities Council

# Enforcing task ordering

**Need to enforce assembly tree ordering**

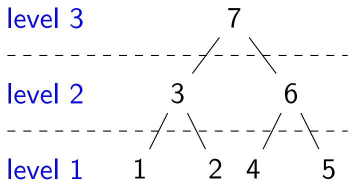
- Ideally would do so via global memory with single kernel
- Want to support Fermi, insufficient registers
- Use level based approach instead



# Enforcing task ordering

**Need to enforce assembly tree ordering**

- Ideally would do so via global memory with single kernel
- Want to support Fermi, insufficient registers
- Use level based approach instead

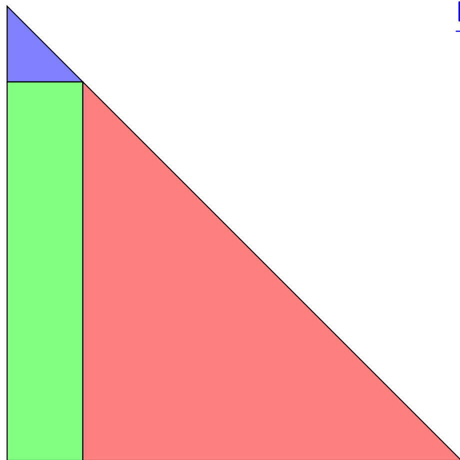



**Outstanding Issues**

Load balance:

- Disparate node sizes
- Freedom of assignment

# Factorization: basics

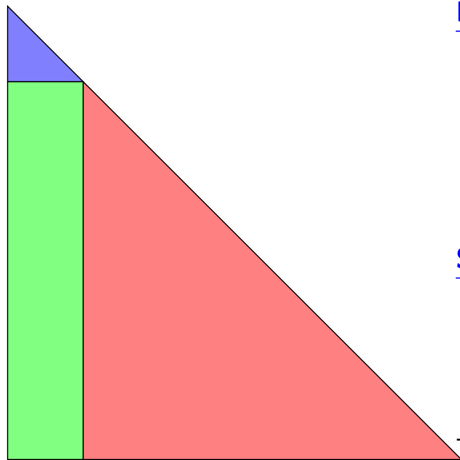


## Basic Algorithm

1. Factor $A_{11} = L_{11} D_1 L_{11}^T$
2. Divide $L_{21} = A_{21} L_{11}^{-T}$
3. Form $C = L_{21} D_1 L_{21}^T$

Science & Technology Facilities Council

# Factorization: basics



## Basic Algorithm

1. Factor $A_{11} = L_{11} D_1 L_{11}^T$
2. Divide $L_{21} = A_{21} L_{11}^{-T}$
3. Form $C = L_{21} D_1 L_{21}^T$

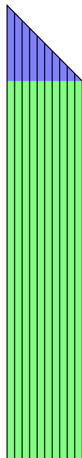
## Stability

- **All** entries in $L_{21} < u^{-1}$
- Entries of $D_1$ calculated in stable fashion

Typically $u = 0.01$.

Science & Technology Facilities Council

# Factorization: parallel pivoting I

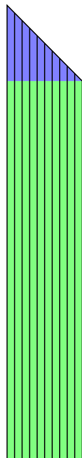


## Traditional algorithm

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

Science & Technology Facilities Council

# Factorization: parallel pivoting I



## Traditional algorithm

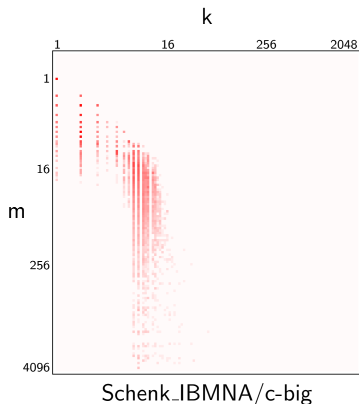
- Work column by column
- Bring column up-to-date
- Find maximum element $\alpha$ in column of $A_{21}$
- Pivot test $\alpha/a_{11} < u^{-1}$. Accept/reject pivot

## Problems

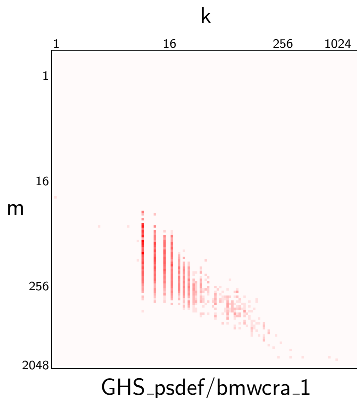
- Very stop-start (one column at a time)
- All-to-all communication for every column

Science & Technology Facilities Council

# Size distributions



Schenk_IBMNA/c-big

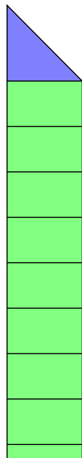


GHS_psdef/bmwcra_1

- Wide range of sizes
- Often $m \gg k$

Science & Technology Facilities Council

# Factorization: parallel pivoting II



## Solution

- Try-it-and-see pivoting (*a posteriori pivoting*)

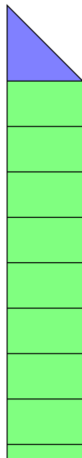
## New algorithm

- Work by blocks of $L_{21}$
- Every block factorizes copy of $A_{11}$
- Every block checks max $|l_{21}| < u^{-1}$
- All-to-all communication when all blocks are done
- Discard columns that have failed on *any* block

We use a block size of $32 \times 8$.

Science & Technology Facilities Council

# Factorization: parallel pivoting III

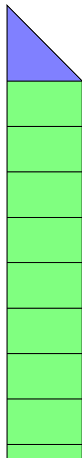


## Implementation Issues

- Inefficient if lots of rejected pivots
- Still quite stop-start
- High register pressure (especially on Fermi)

Science & Technology Facilities Council

# Factorization: parallel pivoting III



**Implementation Issues**

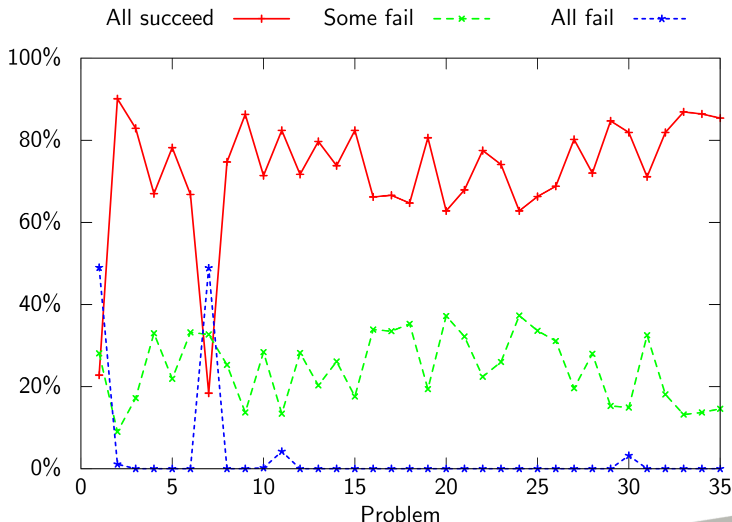
- Inefficient if lots of rejected pivots
- Still quite stop-start
- High register pressure (especially on Fermi)

**Future work**

- Implement Subset pivoting or other CA technique as fall back
- Move to DAG-based implementation (Kepler only) (Significant performance improvement expected)

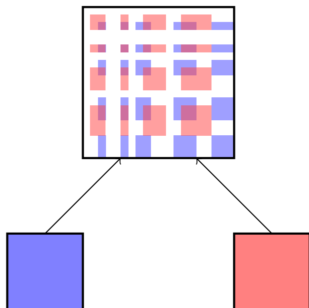
Science & Technology Facilities Council

# A posteriori pivoting samples




Science & Technology Facilities Council

# Assembly: Sparse gather/scatter



Can be framed as *either* sparse gather *or* sparse scatter.

- Need to enforce ordering: prefer sparse gather
- Launch one kernel per child
  (i.e. all first children, then all second, ...)

Science & Technology Facilities Council

# Auxiliary codes

Many auxiliary routines are required that are still CPU-based:

- Ordering (Nested Dissection)
- Analyse (Assorted Graph Algorithms)
- Scaling (MC64 or SpMv)

... but only run once for a sequence of problems

**Auction-based scaling: alternative to MC64**

For some problems, serial MC64 scaling takes $> 75\%$ of time

- 95% of the quality
- 10% of the time
- Parallelizable

Science & Technology Facilities Council

# Results

## Test Problems

- $3\times$ Optimization (IPM)
- $3\times$ Finite Element
- $3\times$ Other assorted

Science & Technology Facilities Council

# Times(s) and Speedup: Factor+Solve

| Problem | CPU | GPU | Speedup |
|---|---|---|---|
| GHS_indef/c-71 | 2.76 | 0.64 | 4.3 |
| GHS_indef/ncvxqp3 | 4.75 | 1.61 | 2.9 |
| Schenk_IBMNA/c-big | 11.81 | 2.35 | 5.0 |
| DNVS/shipsec1 | 1.51 | 0.61 | 2.5 |
| GHS_psdef/bmwcra_1 | 2.09 | 0.93 | 2.3 |
| DNVS/ship_003 | 3.12 | 1.08 | 2.9 |
| ND/nd6k | 6.42 | 1.36 | 4.7 |
| PARSEC/Si5H12 | 14.65 | 2.20 | 6.7 |
| Andrianov/mip1 | 0.82 | 0.38 | 2.2 |

**CPU:** Westmere-EP

- 2× E5620 = 8 cores [76.8GFlops, 160W TDP]

**GPU:** Fermi

- C2050 GPU [515GFlops, 238 TDP]

Flops ratio about 7×

Science & Technology Facilities Council

# Times(s) and Speedup: Factor+Solve

| Problem | CPU | GPU | Speedup |
|---|---|---|---|
| GHS_indef/c-71 | 1.67 | 0.43 | 3.9 |
| GHS_indef/ncvxqp3 | 2.31 | 1.42 | 1.6 |
| Schenk_IBMNA/c-big | 6.63 | 1.49 | 4.4 |
| DNVS/shipsec1 | 0.63 | 0.40 | 1.6 |
| GHS_psdef/bmwcra_1 | 0.77 | 0.60 | 1.3 |
| DNVS/ship_003 | 1.24 | 0.68 | 1.8 |
| ND/nd6k | 2.84 | 0.82 | 3.5 |
| PARSEC/Si5H12 | 7.20 | 1.32 | 5.4 |
| Andrianov/mip1 | 0.40 | 0.28 | 1.5 |

**CPU:** Sandybridge-EP

- 2×E5-2687W = 16 cores [397GFlops, 300W TDP]

**GPU:** Kepler

- K20 GPU [1170Flops, 225 TDP]

Flops ratio about 3×

# Factorization phase hot-spots

| | c-72 | c-big | shipsec1 | Lin |
|---|---|---|---|---|
| Speedup | 1.4 | 6.2 | 1.9 | 4.4 |
| Contrib | 19 | 780 | 1607 | 1568 |
| Assembly | 27 | 446 | 38 | 302 |
| Factor | 82 | 481 | 850 | 666 |
| Waiting | 143 | 525 | 405 | 352 |

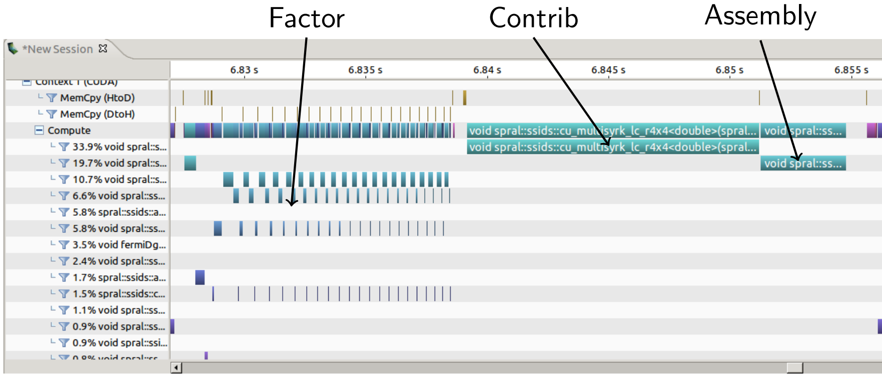
Times are in ms.

Waiting = time not in kernels.



Science & Technology Facilities Council

# Dense factor is poor




Science & Technology Facilities Council

# Conclusions and Future Work

## Story so far

- New open source sparse direct solver in CUDA
  - http://www.numerical.rl.ac.uk/spral/
  - Report forthcoming
- Speedups over host of around 5 on large problems
- Needed to both:
  - Handle peculiarities of device
  - Use new algorithms for massive parallelism

## Long-term

- DAG-based factor
- GPU-based scaling
- Auto-generation from stencil?

# Thanks for listening!

Questions?

# A Supplementary slide

Some supplementary text.
(Note numbering of supplementary slides is outside that of normal slides.)

Science & Technology Facilities Council