GTC 2013:

Developments in GPU-Accelerated Sparse Linear Algebra Algorithms

Kyle Spagnoli

Research Engineer @ EM Photonics

3/20/2013

INTRODUCTION

- Sparse systems
- Iterative solvers
  - High level benchmarks
- Approximate inverse preconditioner
  - GPU implementation details
  - In-depth benchmarks
- Direct solvers
- EM Photonics' *CULA Sparse* library

Previous work – CULA Dense

- Dense "LAPACK" for CUDA
  - System solvers
  - Least squares solvers
  - Eigenvalues
  - Singular value decomposition
- Multi-GPU support
- 3-7x speedup vs optimized CPU
- Presented at GTC past few years

EM Photonics

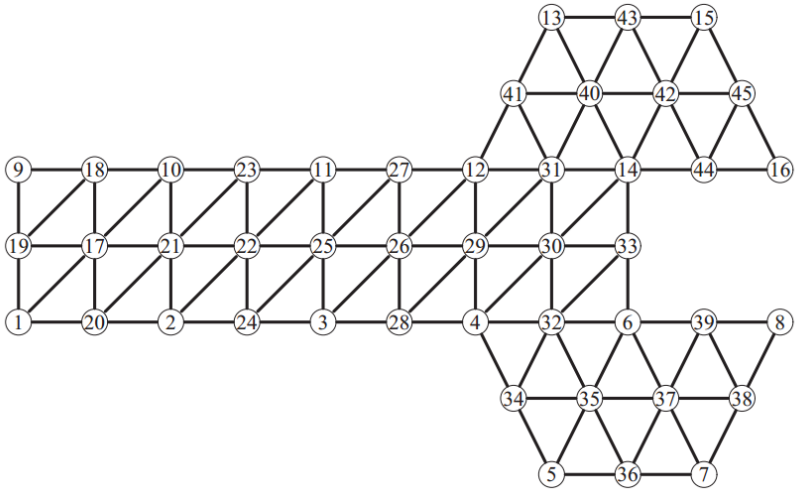
Sparse System Introduction

- Physical phenomena modeled by partial differential equations
  - $\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = -\nabla p + \nabla\cdot\mathbf{T} + \mathbf{f},$
  - Forces, momentums, velocities, energy, temperature, etc.
- Sparse systems appear in discretization methods
  - Finite element, finite volume, topology methods, etc.
  - Fluid & structural dynamics, electromagnetics, etc.

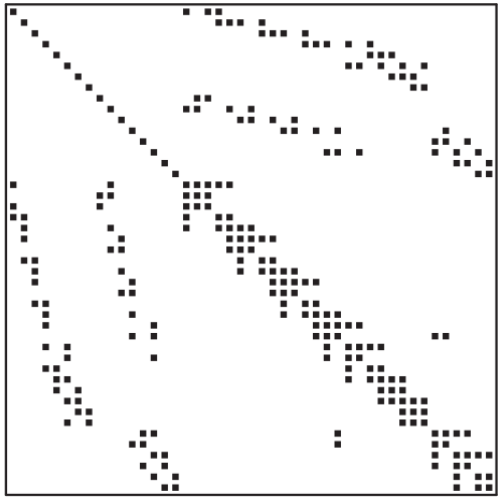
EM Photonics

Sparse System Introduction

finite element mesh



assembled matrix



Images courtesy of "Iterative Methods for Sparse Linear Systems" by Yousef Saad

EM Photonics

EM Photonics

Accelerated Computing Solutions

Iterative Solvers

Iterative Solvers - Basics

- Want to solve for x in $Ax = b$
- Each iteration works toward lowering the residual value of $r = ||Ax - b||$
- Krylov subspace methods
  - $K_m(A, v) \equiv span\{v, Av, A^2v, ..., A^{m-1}v\}$
  - Sparse matrix-vector multiplication
  - CG, GMRES, BiCGSTAB, Minres, etc.

EM Photonics

Iterative Solvers – CG Method

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for $i = 1, 2, \ldots$
  solve $Mz^{(i-1)} = r^{(i-1)}$
  $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
  if $i = 1$
    $p^{(1)} = z^{(0)}$
  else
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
  endif
  $q^{(i)} = Ap^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)^T} q^{(i)}$
  $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence; continue if necessary
end

Apply preconditioner M with a matrix-vector solve

Matrix-vector multiplication

EM Photonics

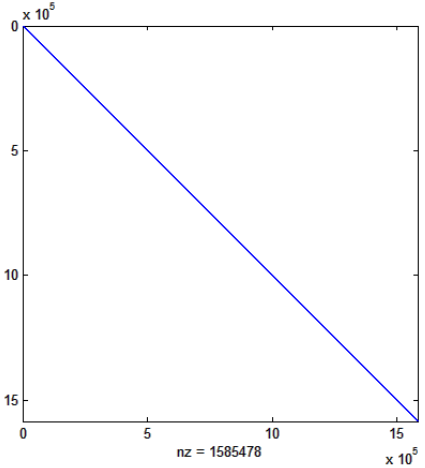
Iterative Solvers - Preconditioners

- Modify original system to make it “easier” to solve
- $Ax = b \rightarrow MAx = Mb$
- (Hopefully) reduce the number of iterations needed to solve
- One time setup & generation cost
- Per-iteration application cost

EM Photonics

Preconditioners – Jacobi

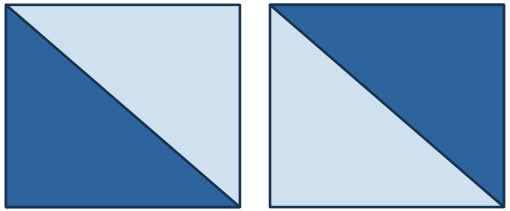
- Jacobi / Block Jacobi
  - $M = \mathrm{diag}(A)$
  - $M = \mathrm{block\_diag}(A, bs)$
- Minor reduction in iterations
- Lightweight and parallel in generation and application



EM Photonics

PRECONDITIONERS – ILU0

- Zero fill incomplete LU (ilu0)
  - $M \rightarrow LU \approx A$
  - M has same pattern as A
- Significant reduction in iterations
- Performance depends on matrix structure
  - Parallelism from graph coloring
- Applied with **sparse triangular solve**
  - Limited performance on GPU



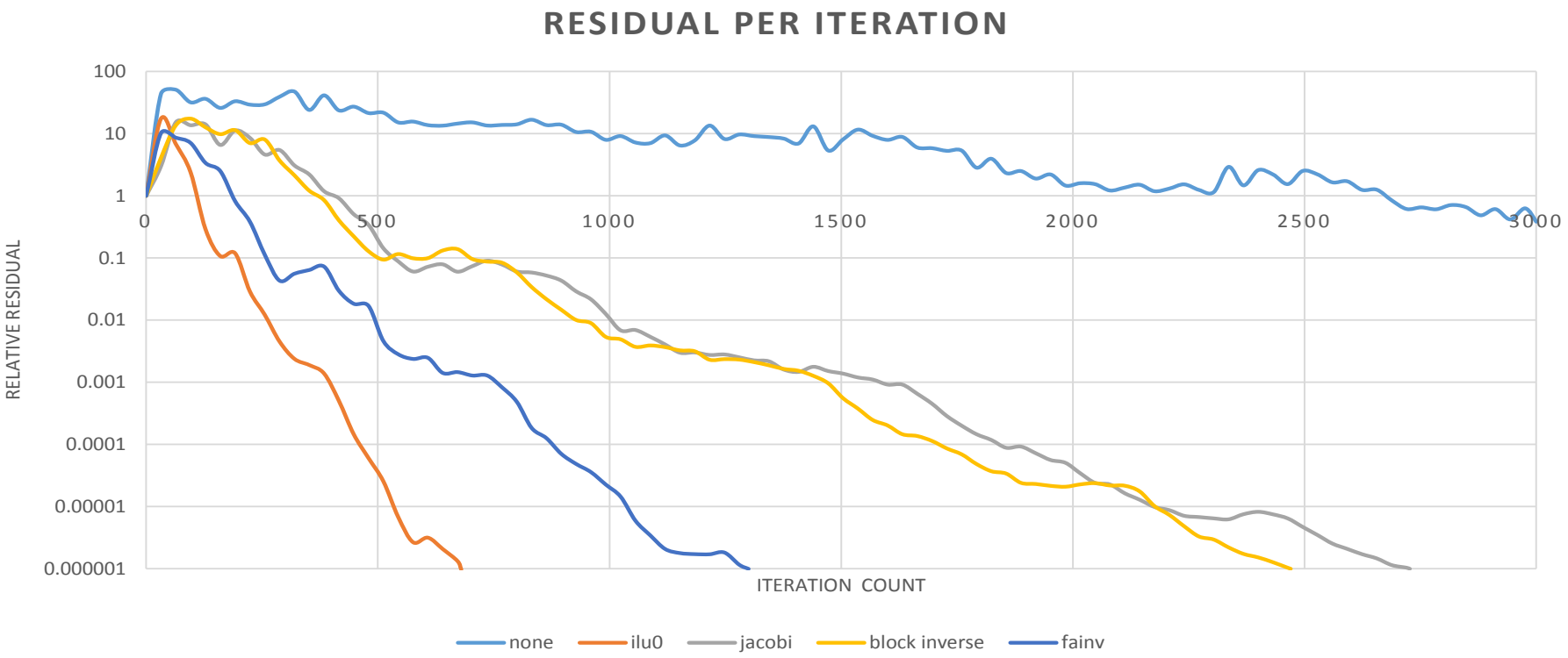
EM Photonics

Preconditioners – Approximate Inverse

- Approximate Inverse
  - $M^{-1} \rightarrow AM^{-1} \approx I$
- Factorized Approximate Inverse (for SPD systems)
  - $M^{-1} \rightarrow LL^T \approx A$
- M^{-1} has pattern of A (or A^2, A^3)
- Moderate reduction in iterations compared to ILU0
- Very expensive to generate but highly parallel
- Applied with **sparse matrix-vector multiplication**
  - Extremely high performance on GPU

EM Photonics

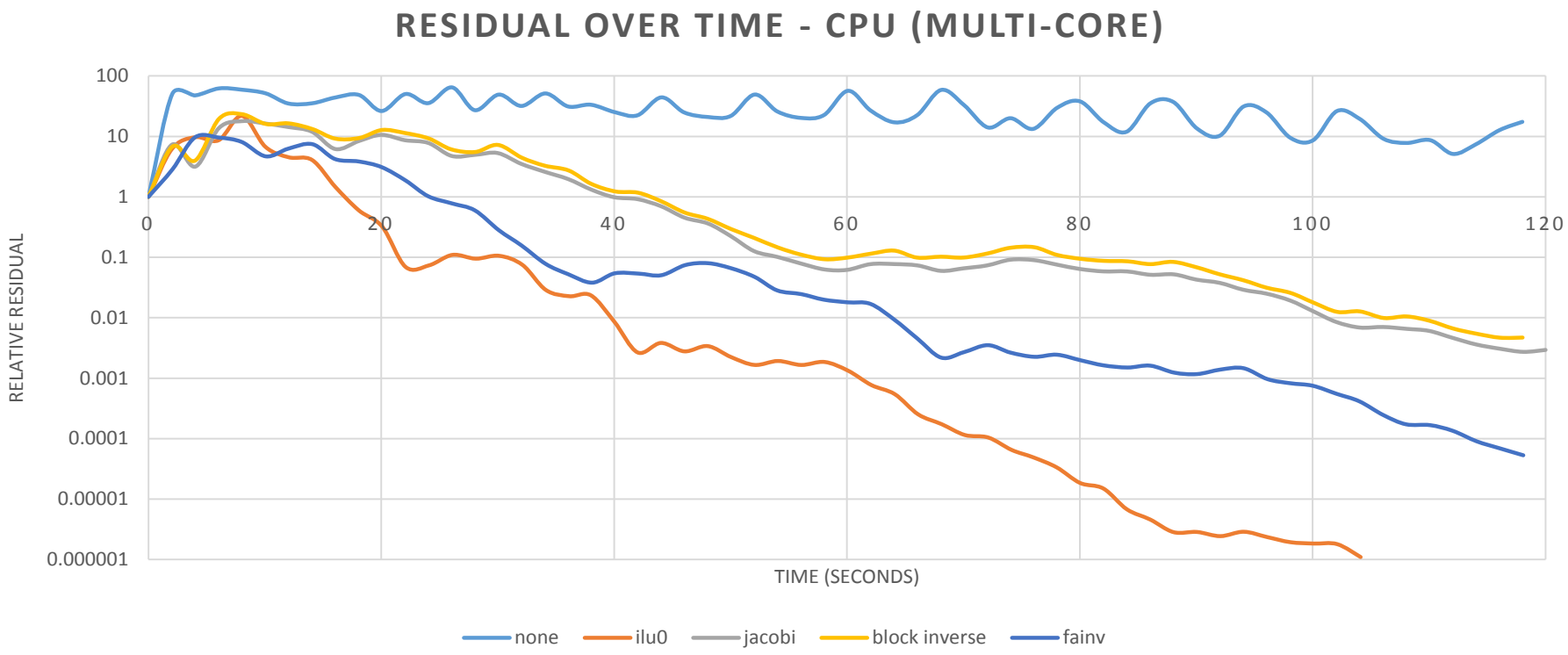
Preconditioner Performance - Iterations



circuit simulation problem with 7.5M non-zero elements

EM Photonics

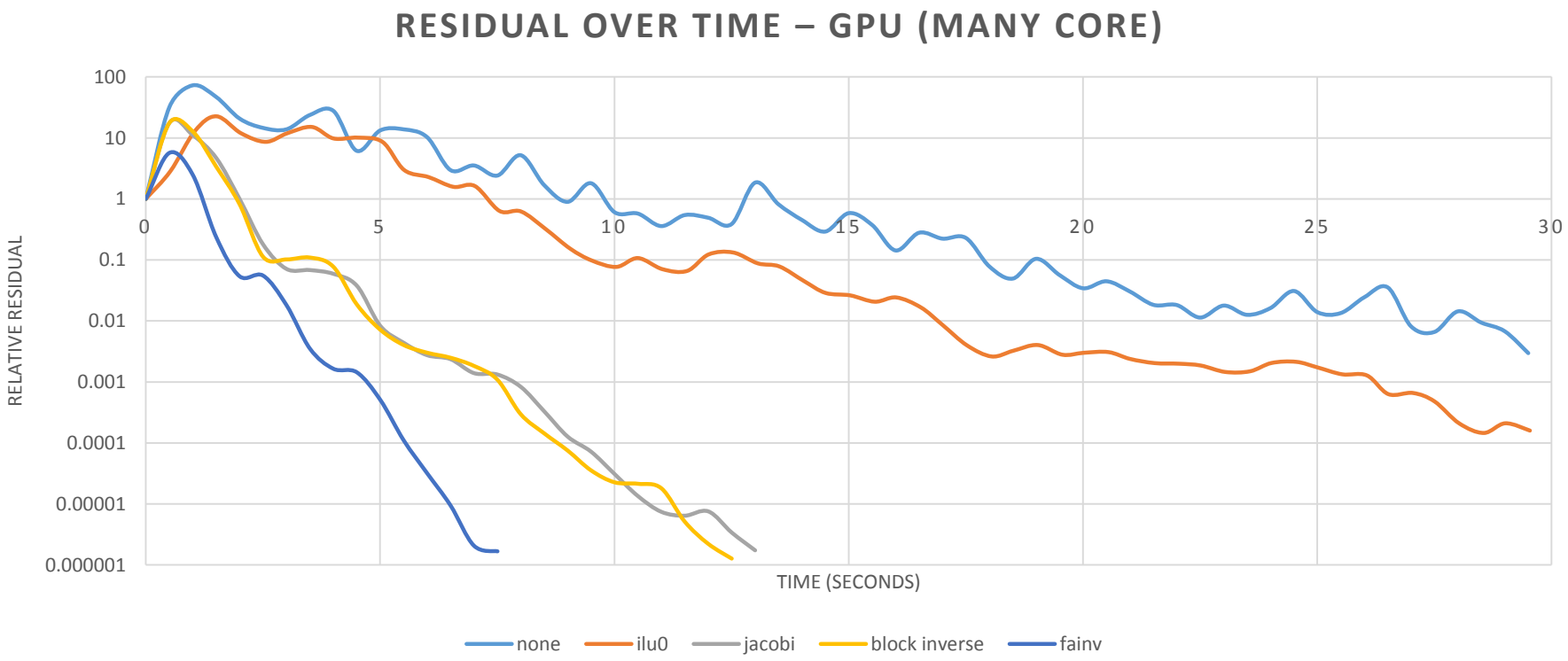
Preconditioner Performance



circuit simulation problem with 7.5M non-zero elements

EM Photonics

PRECONDITIONER PERFORMANCE



circuit simulation problem with 7.5M non-zero elements

EM Photonics

EM Photonics
Accelerated Computing Solutions

Approximate Inverse Details

Factorized Approximate Inverse – Math Lite

- Minimization of Frobenius norm:

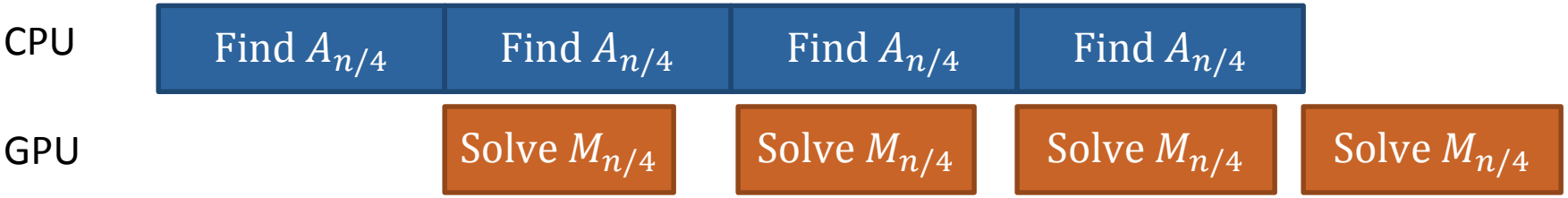
$$\|AM - I\|_F^2 \quad \rightarrow \quad \sum_{k=1}^{n} \left\| \widehat{A_k} \widehat{m}_k - e_k \right\|_2^2$$

- Can be solved with per-column parallelism!
- Finding $\widehat{A_k}$ is memory intensive…
  - extract a dense small matrix relative to the number of elements in the column
- Minimization is compute intensive…
  - positive definite solve on the small dense matrix
  - least squares solve for non-SPD inputs

EM Photonics

Factorized Approximate Inverse - Implementation

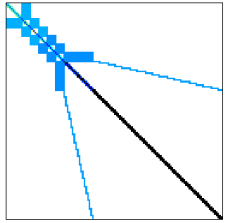
- High performance *hybrid* CPU/GPU solution…
  - Find *chunk* of successive $\widehat{A_k}$ matrices using CPU
    - batched with OpenMP
  - Perform positive definite solves on the GPU
    - batched with custom CUDA kernels
  - Overlap *chunks* using asynchronous operations



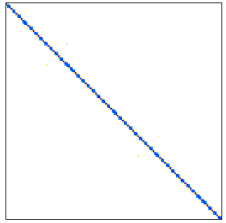
EM Photonics

Factorized Approximate Inverse – Test Matrices

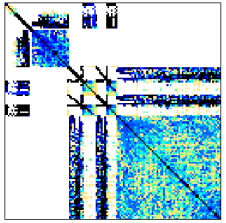
G3_circuit
circuit simulation
n = 1.5M nnz = 7.6M



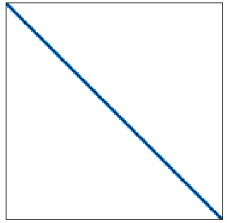
apache2
structure
n = 700K nnz = 4.8M



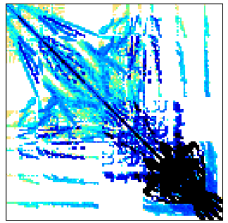
msdoor
structure
n = 400K nnz = 19M



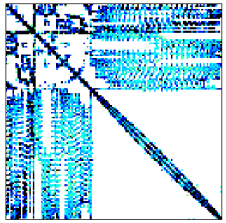
ecology2
2d/3d problem
n = 1M nnz = 5M



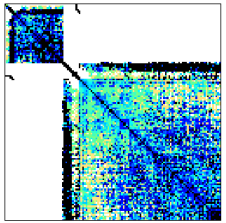
offshore
electromagnetics
n = 250k nnz = 4.2M



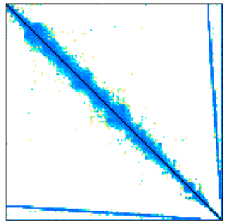
audikw_1
structure
n = 950k nnz = 77M



ldoor
structure
n = 900k nnz = 42M



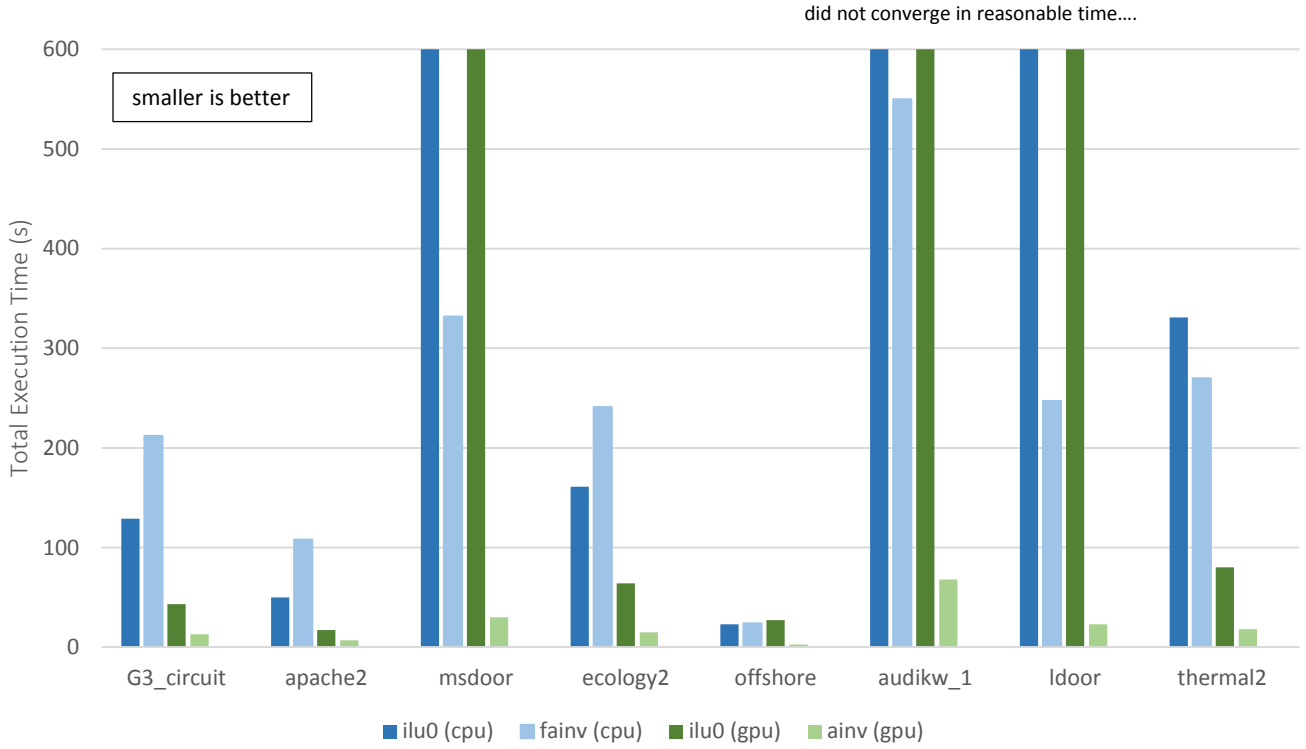
thermal2
heat transfer
n = 1.2M nnz = 8.5M



test matrices from the UF Sparse Matrix Collection

EM Photonics

Factorized Approximate Inverse – Benchmarks



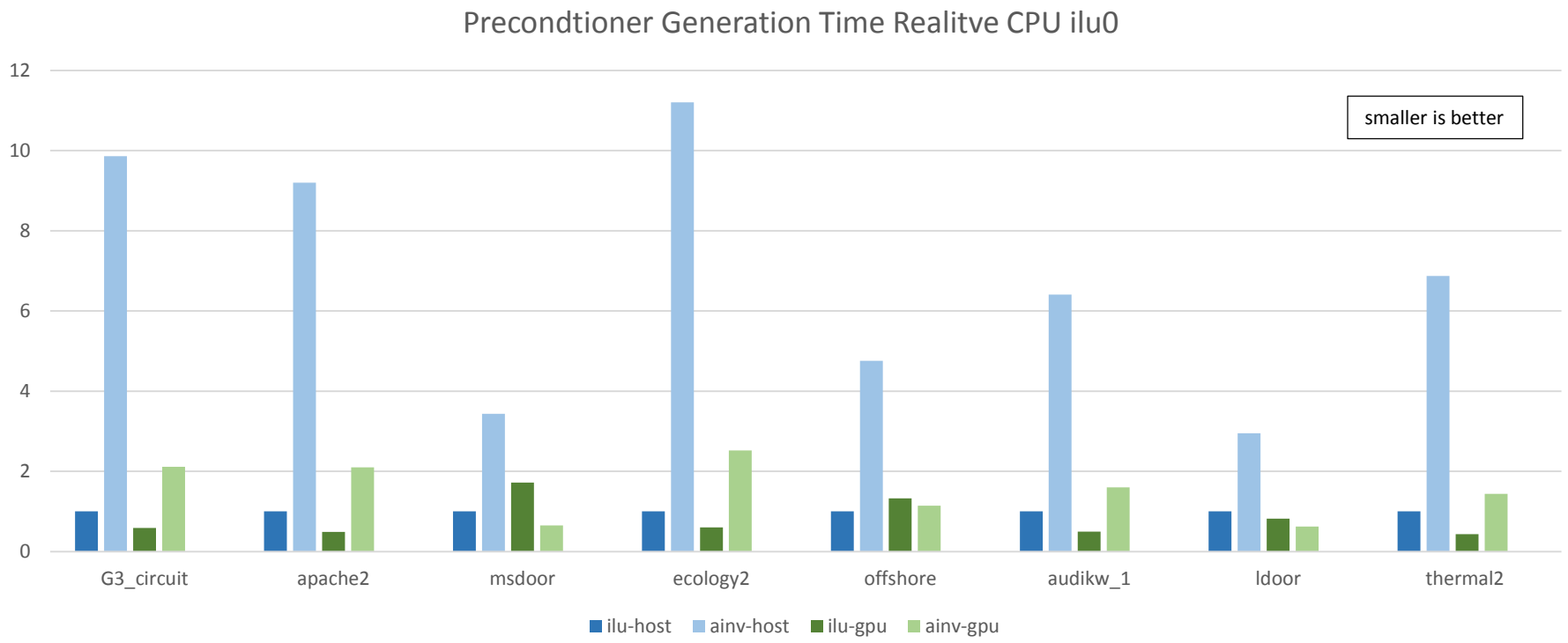
GPU: NVIDIA C2070
CPU: Dual Intel Xeon 5660

name	speed up*
G3_circuit	9.92x
apache2	7.14x
msdoor	11.06x
ecology2	10.73x
offshore	8.21x
audikw_1	8.08x
ldoor	10.73x
thermal2	15.00x

* Best CUDA algorithm relative to best CPU algorithm. Includes data transfers, preconditioner generation, and iterations.

EM Photonics

Factorized Approximate Inverse – Benchmarks



EM Photonics

Factorized Approximate Inverse - Summary

- Successful in reducing number of iterations
- Very expensive to generate
- **Massively parallel**
- Excellent preconditioners for GPU
- Chose an algorithm that fits your platform

EM Photonics

EM Photonics

Accelerated Computing Solutions

Direct Sparse Linear System Solvers

Direct Sparse Solvers

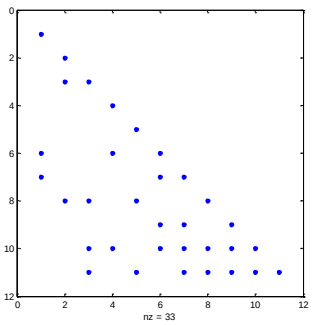
- LU, Cholesky, or QR factorization of matrix
- Dense methods require full *MxN* matrix
- Express problem as a number of relatively small dense problems with task and data parallelism
- Factorize stages:
  1. Symbolic – Analyze graph structure to determine a "plan of attack" that best suits platform
  2. Numeric – Perform factorization calculations

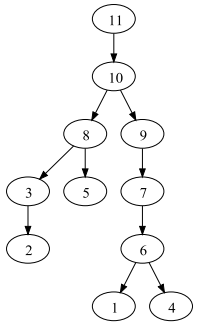
EM Photonics

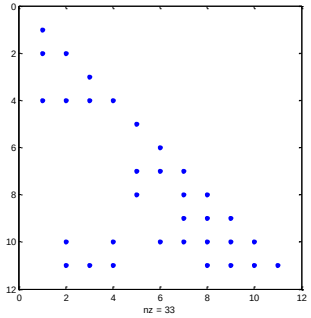
Direct Sparse Solvers - Symbolic Factorization

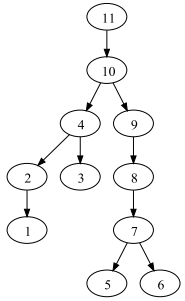
» The nonzero pattern forms a elimination tree describing an order of column (vector) operations

» Reordering allows for better parallelism and data locality





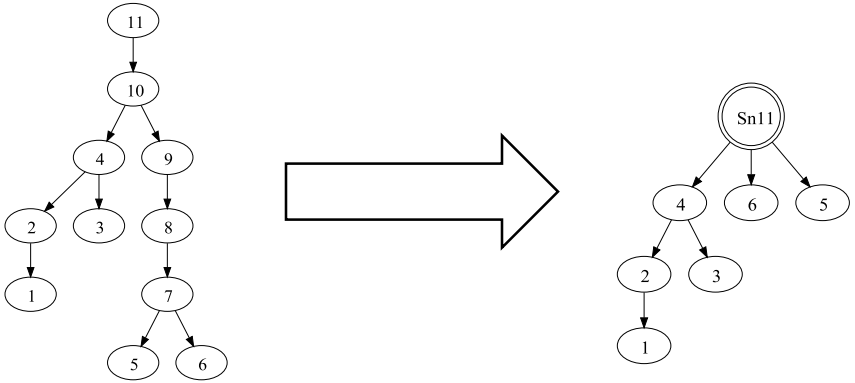


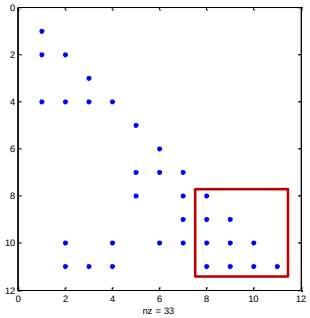


EM Photonics

Direct Sparse Solvers - Symbolic Factorization

- Merge adjacent nodes into *supernodes* that can now utilize high-performance **matrix** operations
- Fill in with explicit zeros if needed
- Make choices based on the hardware available
  - Zero fill thresholds
  - Number of processors/GPUs





EM Photonics

Direct Sparse Solvers

- Tree can now expressed as a *task dependency graph* of *dense matrices* operations
  - matrix multiply, triangular solve, small dense factorization
- Traverse graph queuing operations on the GPU
- Process smaller problems on the host
- Manage memory locality to prevent ping-ponging
- Kepler's **Hyper-Q** improves performance of processing different sized matrices performing different tasks

EM Photonics

Accelerated Computing Solutions

CULA Sparse

CULA Sparse

- Software package available from www.culatools.com
  - Free for personal academic usage
  - Version “S4” available now
  - Version “S5” beta starting in early April

- Platforms
  - Multi-threaded host, CUDA, CUDA (device pointers)
- Solvers
  - CG, BiCG, BiCGStab, BiCGStab(L), GMRES, Minres, …
- Preconditioners:
  - Jacobi, Block Jacobi, SSOR, ILU0, Ainv, Fainv, …
- Formats
  - Coo, Csr, Csc, Eli, Dense, …

EM Photonics

CULA Sparse – Pseudo-code Usage example

```
// create plan
culaSparsePlan plan;
culSparseCreatePlan(&plan);

// configure plan
culaSparseSetCudaPlatform(plan, &cudaOptions);
culaSparseSetDCooFormat(plan, &cooOptions, n, nnz, a, b, x);
culaSparseSetFainvPreconditioner(plan, &fainvOptions);
culaSparseSetCgSolver(plan, &cgOptions);

// execute plan
culaSparseExecute(plan, &config, &results);
```


EM Photonics

CULA Sparse – Psudedo Usage example

```
// change preconditioner to ilu0
culaSparseSetIlu0Solver(plan, &ilu0Options);

// re-execute plan with cached data (avoids data transfer)
culaSparseExecute(plan, &config, &results);
```

```
// change solver to bicg
culaSparseSetBicgSolver(plan, &bicgOptions);

// re-execute plan with cached data (avoid preconditioner regeneration)
culaSparseExecute(plan, &config, &results);
```


EM Photonics

Future Work

- “Matrix free” methods
- Multi-GPU support
- More solvers, preconditioners, data formats, and performance
- PETSc integration
- Intel Xeon Phi platform
- Direct solver realeases

EM Photonics

THANKS!

Thanks!
Questions?

» Convention hall @ booth 311

» More information @ www.culatools.com

EM Photonics