Fine-grained Parallel Incomplete LU Factorization

Edmond Chow

School of Computational Science and Engineering
Georgia Institute of Technology

Sparse Days Meeting at CERFACS
June 5-6, 2014

Contribution

- A new method for parallelizing the construction of incomplete LU factorizations

$$A \approx LU$$

- Method designed for large amounts of fine-grained parallelism, such as in GPUs, Intel Xeon Phi, and future processors

Conventional ILU factorization

Given sparse A, compute $LU \approx A$, where L and U are sparse.

Define S to be the sparsity pattern, $(i,j) \in S$ if l_{ij} or u_{ij} can be nonzero.

for $i = 2$ **to** n **do**
    for $k = 1$ **to** $i-1$ **and** $(i,k) \in S$ **do**
        $a_{ik} = a_{ik} / a_{kk}$
        for $j = k+1$ **to** n **and** $(i,j) \in S$ **do**
            $a_{ij} = a_{ij} - a_{ik} a_{kj}$
        end
    end
end

Level scheduling ILU

At each step, find all rows that can be eliminated in parallel (rows that only depend of rows already eliminated).

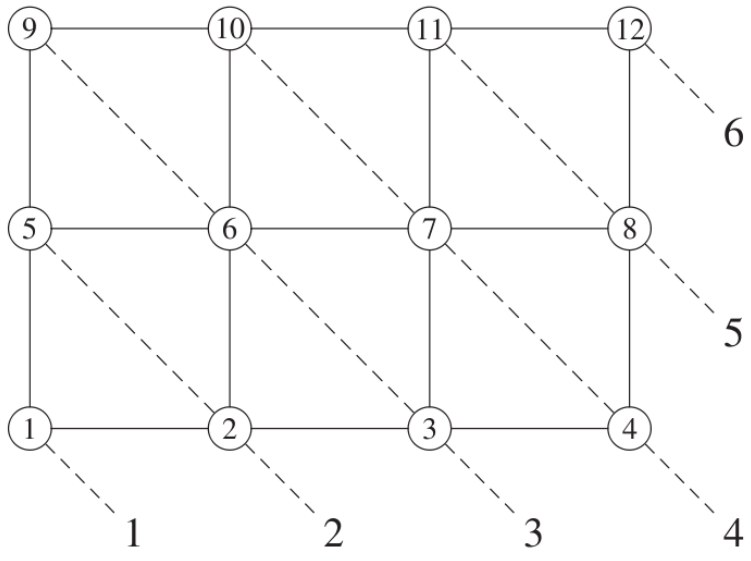


Figure from Saad 2003

Regular grids: van der Vorst 1989; Joubert & Oppe 1994
Irregular problems: Heroux, Vu, & Yang 1991; Pakzad, Lloyd, & Phillips 1997; Gonzalex, Cabaleiro, & Pena 1999; Dong & Cooperman 2011; Gibou & Min 2012; Naumov 2012

Triangular solves: Anderson & Saad 1989, Saltz 1990; Hammond & Schreiber 1992

Multicolor reordering for ILU

Multicolor reorderings can increase parallelism, but the resulting factorization is different and is a worse preconditioner.

$$\begin{bmatrix} D_1 & A_{12} & A_{13} & A_{14} \\ A_{21} & D_2 & A_{23} & A_{24} \\ A_{31} & A_{32} & D_3 & A_{34} \\ A_{41} & A_{42} & A_{43} & D_4 \end{bmatrix}$$

PCG iterations for Laplacian problem

	2D (1001^2 grid)	3D (101^3 grid)
natural	719	89
red-black	1159	142

Refs: Poole & Ortega 1987; Elman & Agron 1989; Jones & Plassmann 1994; Nakajima 2005. **GPUs:** Li & Saad 2010; Heuveline, Lukarski & Weiss 2011

It is often more important to use orderings to enhance convergence.

Domain decomposition ILU

ILU on each subdomain in parallel. Somehow eliminate the interface rows in parallel.

$$\begin{bmatrix} B_1 & & & & F_1 \\ & B_2 & & & F_2 \\ & & \ddots & & \vdots \\ & & & B_m & F_m \\ E_1 & E_2 & \cdots & E_m & C \end{bmatrix}$$

Convergence does not degrade if subdomains are large. This is a coarse-grained parallelization.

Refs: Ma & Saad 1994, Karypis & Kumar 1997; Vuik et al 1998; Hysom and Pothen 1999; Magolu monga Made & van der Vorst 2002

For very large problems, you would likely use a multilevel method, e.g., multilevel domain decomposition, and you might use ILU in a subdomain solver on a single node.

New fine-grained parallel ILU

- Does not use level scheduling or reordering of the matrix
- Each nonzero in L and U computed in parallel (updated asynchronously)
- More parallelism than existing approaches

Theorem

The conventional ILU algorithm produces factors L and U such that

$$R = LU - A$$

in which R is the matrix of elements that are dropped during the incomplete elimination process. When $(i, j) \in S$, the entry r_{ij} of R is equal to zero. (Proposition 10.4 in Saad 2003.)

Reformulation of ILU

An ILU factorization with sparsity pattern S has the property

$$(LU)_{ij} = a_{ij}, \quad (i,j) \in S.$$

Instead of Gaussian elimination, we compute the *unknowns*

$$\begin{aligned} l_{ij}, &\qquad i > j, &\qquad (i,j) \in S \\ u_{ij}, &\qquad i \leq j, &\qquad (i,j) \in S \end{aligned}$$

using the *constraints*

$$\sum_{k=1}^{\min(i,j)} l_{ik} u_{kj} = a_{ij}, \quad (i,j) \in S.$$

If the diagonal of L is fixed, then there are $|S|$ unknowns and $|S|$ constraints.

Solution of constraint equations

The equation corresponding to (i,j) gives

$$\begin{aligned} l_{ij} &= \frac{1}{u_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}\right), \quad i > j \\ u_{ij} &= a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}, \qquad i \leq j. \end{aligned}$$

The equations have the form $x = G(x)$. It is natural to try to solve these equations via a fixed-point iteration

$$x^{(k+1)} = G(x^{(k)})$$

with an initial guess $x^{(0)}$. We update each component of $x^{(k+1)}$ in parallel and asynchronously.

Fine-grained ILU algorithm

Set unknowns l_{ij} and u_{ij} to initial values
for *sweep* $= 1, 2, \ldots$ *until convergence* **do**
  parallel for $(i, j) \in S$ **do**
    if $i > j$ **then**
      $l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right) / u_{jj}$
    else
      $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$
    end
  end
end

Arithmetic intensity can be tuned by controlling how often updates are exposed to other threads, at the cost of convergence degradation.

Actual implementation uses sparse data structures.

Will such a crazy idea work?

- More nonlinear equations than original equations
- Equations are nonlinear (bilinear)

Potential advantages

- Do not need to solve the nonlinear equations exactly (no need to compute the incomplete factorization exactly)
- Nonlinear equations may have a good initial guess (e.g., time-dependent problems)
- Nonlinear equations have surprising structure....

Convergence

Convergence is related to the Jacobian $J(x)$ of $G(x)$. The partial derivatives in the row of $J(x)$ corresponding to l_{ij} are

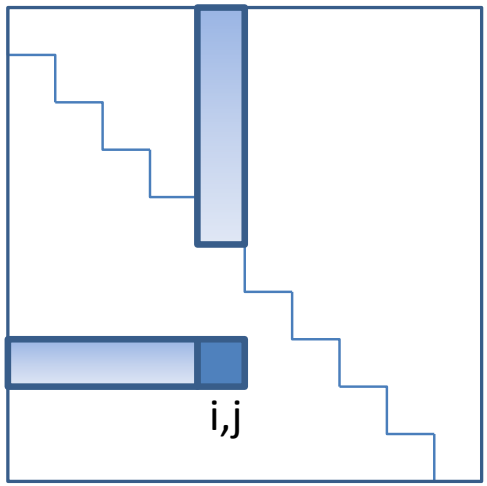
$$-\frac{l_{ik}}{u_{jj}}, \quad -\frac{u_{kj}}{u_{jj}}, \quad -\frac{1}{u_{jj}^2}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}\,u_{kj}\right), \quad k \leq j-1.$$

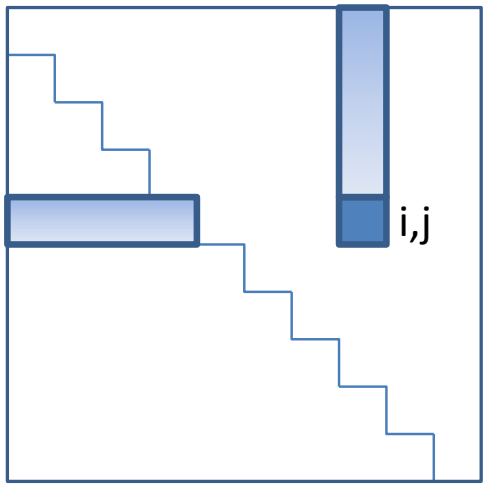
The partial derivatives corresponding to u_{ij} are

$$-l_{ik}, \quad -u_{kj}, \quad k \leq i-1.$$

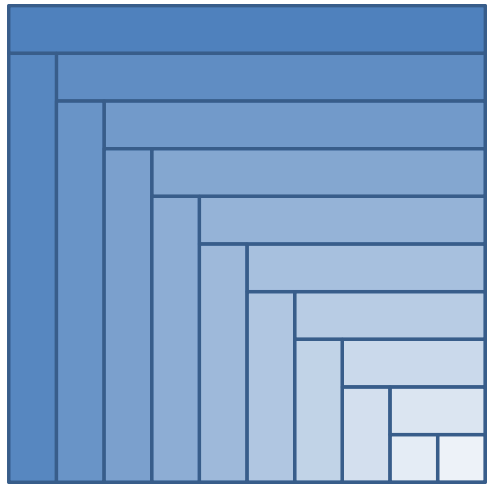
- $G(x)$ is F-differentiable on its domain because its partial derivatives are well defined and continuous.
- $J(x)$ is sparse and has all zeros on the diagonal.
- $J(x)$ can be symmetrically reordered to be strictly lower triangular (Gaussian elimination ordering).

Data dependencies





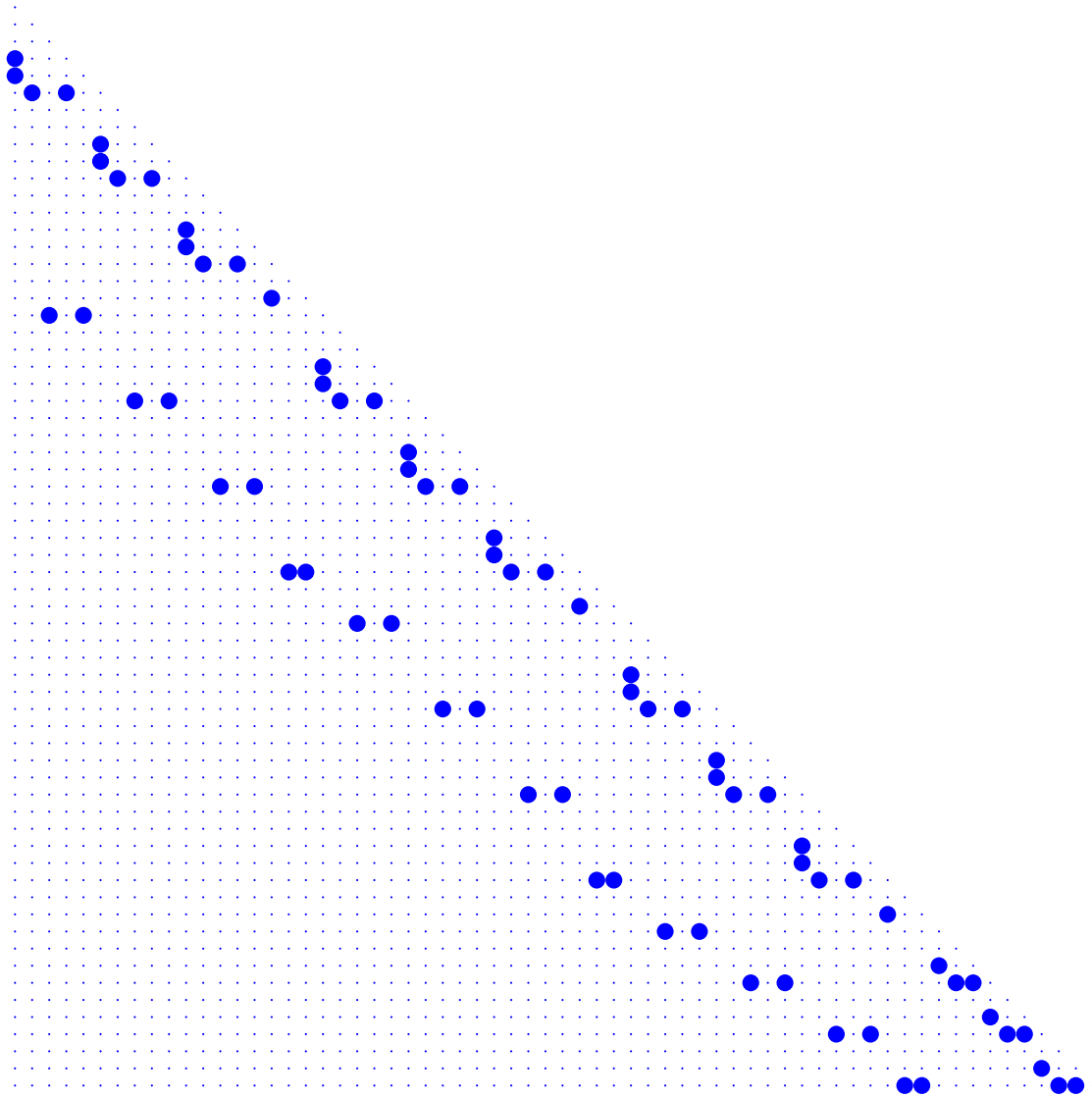
Formula for unknown at (i,j) (dark square) depends on other unknowns left of (i,j) in L and above (i,j) in U (shaded regions).



Gaussian elimination order: Entries $(i,j) \in S$ in darker rows and columns are ordered before those in lighter rows and columns.

If the unknowns are computed in Gaussian elimination order, then they are computed exactly in a single sweep.

Structure of the Jacobian



5-point matrix on 4x4 grid

Local convergence

Local convergence of the synchronous iteration

- Spectral radius of $J(x)$ is zero for all x
- $G(x)$ is F-differentiable

From Ostrowski's theorem, if $G(x)$ has a fixed point x^*, then x^* is a point of attraction of the synchronous fixed point iteration.

Local convergence

Local convergence of the synchronous iteration

- Spectral radius of $J(x)$ is zero for all x
- $G(x)$ is F-differentiable

From Ostrowski's theorem, if $G(x)$ has a fixed point x^*, then x^* is a point of attraction of the synchronous fixed point iteration.

Local convergence of the asynchronous iteration

- Assume all variables are eventually updated, etc.
- Spectral radius of $|J(x)|$ is zero for all x
- $G(x)$ is F-differentiable

From Bertsekas (1983), if $G(x)$ has a fixed point x^*, then x^* is a point of attraction of the asynchronous fixed point iteration.

Transient convergence

- Possible for nonlinear iterations to diverge before converging.
- Want norm of $J(x)$ to be small, and thus the partial derivatives of $J(x)$ to be small (want u_{jj} to be large).
- This heuristic guides the choice to scale A symmetrically such that its diagonal is all ones.

Contraction mapping property

Theorem. If A is a 5-point or 7-point finite difference matrix, and if $\tilde{L}$ ($\tilde{U}$) has sparsity pattern equal to the strictly lower (upper) triangular corresponding to the unknowns, then

$$\|J(\tilde{L},\tilde{U})\|_1 = \max(\|\tilde{L}\|_{\max},\ \|\tilde{U}\|_{\max},\ \|A_{L^*}\|_1)$$

where A_{L^*} is the strictly lower triangular part of A and $\|\cdot\|_{\max}$ denotes the maximum absolute value among the entries in the matrix.

Contraction mapping property

Theorem. If A is a 5-point or 7-point finite difference matrix, and if $\tilde{L}$ ($\tilde{U}$) has sparsity pattern equal to the strictly lower (upper) triangular corresponding to the unknowns, then

$$\|J(\tilde{L},\tilde{U})\|_1 = \max(\|\tilde{L}\|_{\max},\ \|\tilde{U}\|_{\max},\ \|A_{L^*}\|_1)$$

where A_{L^*} is the strictly lower triangular part of A and $\|\cdot\|_{\max}$ denotes the maximum absolute value among the entries in the matrix.

Example. Any diagonally dominant 5-point or 7-point finite difference matrix has $\|J(x^{(0)})\|_1 < 1$ where $x^{(0)}$ is the standard initial guess.

Example. For a 5-point or 7-point centered-difference approximation of the Laplacian, $\|J(x^{(0)})\|_1 = 0.5$. In the 10×10 mesh case, $\|J(x^*)\|_1 \approx 0.686$.

Measuring convergence of the nonlinear iterations

Nonlinear residual

$$\|(A - LU)_S\|_F = \sum_{(i,j)\in S} \left| a_{ij} - \sum_{k=1}^{\min(i,j)} l_{ik} u_{kj} \right|$$

This differs from what we actually want to minimize:

ILU residual

$$\|A - LU\|_F$$

Convergence of the preconditioned linear iterations is known to be strongly related to the ILU residual (Duff & Meurant 1989).

Numerical tests

- Do the asynchronous iterations converge?
- How fast is convergence with the number of threads?
- How good are the approximate factorizations as preconditioners?

Measure performance in terms of solver iteration count.

Tests on Intel Xeon Phi.

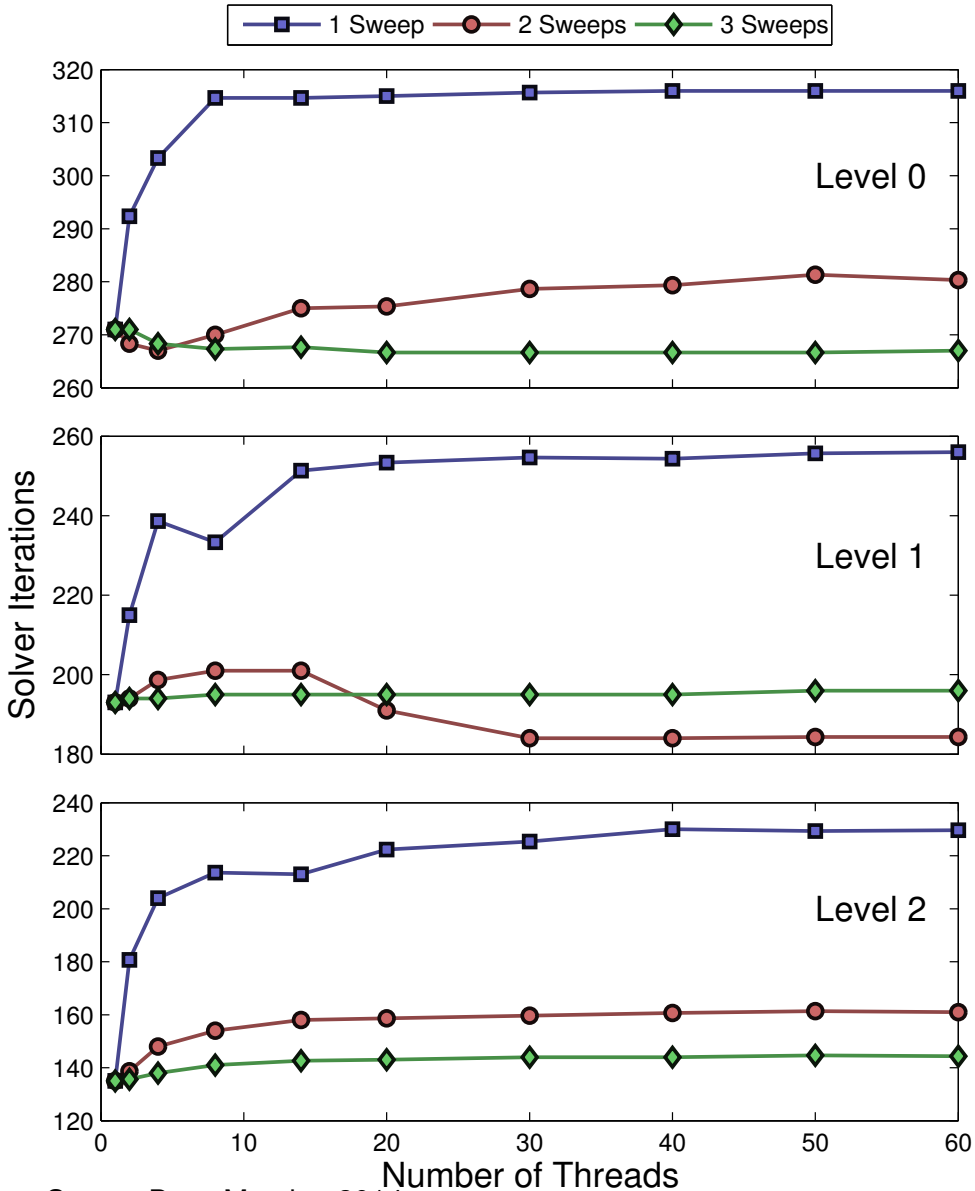
Initial L and U are the lower and upper triangular parts of A.

Use Gaussian elimination ordering for the nonlinear equations.

2D FEM Laplacian, $n = 203841$, RCM ordering, 240 threads

	Level 0			Level 1			Level 2		
Sweeps	PCG iter	nonlin resid	ILU resid	PCG iter	nonlin resid	ILU resid	PCG iter	nonlin resid	ILU resid
0	404	1.7e+04	41.1350	404	2.3e+04	41.1350	404	2.3e+04	41.1350
1	318	3.8e+03	32.7491	256	5.7e+03	18.7110	206	7.0e+03	17.3239
2	301	9.7e+02	32.1707	207	8.6e+02	12.4703	158	1.5e+03	6.7618
3	298	1.7e+02	32.1117	193	1.8e+02	12.3845	132	4.8e+02	5.8985
4	297	2.8e+01	32.1524	187	4.6e+01	12.4139	127	1.6e+02	5.8555
5	297	4.4e+00	32.1613	186	1.4e+01	12.4230	126	6.5e+01	5.8706
IC	297	0	32.1629	185	0	12.4272	126	0	5.8894

2D FEM Laplacian, $n = 203841$, RCM ordering



- very small number of sweeps required
- possible to use a large number of threads
- speedup is approx. 50 with 60 threads
- 5x faster (3 sweeps) than level-scheduled ILU with 60 threads

Non-diagonally-dominant problems

Finite difference discretization of the convection-diffusion equation

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\beta\left(\frac{\partial e^{xy}u}{\partial x}+\frac{\partial e^{-xy}u}{\partial y}\right)=g$$

on a square domain $[0,1]\times[0,1]$ with Dirichlet boundary conditions, 450×450 mesh. Large values of β of 1500 and 3000 used to generate non-diagonally-dominant (and nonsymmetric) problems.

Non-diagonally-dominant problem, $\beta = 1500$

	Solver Iterations			Nonlinear Residual		
No. of	Number of Sweeps			Number of Sweeps		
Threads	1	2	3	1	2	3
1	30.0	30.0	30.0	1.3e-14	9.6e-15	9.6e-15
2	23.3	30.0	30.0	5.70	0.184	0.0035
4	22.3	30.0	30.0	10.48	0.746	0.0388
8	23.7	30.7	30.0	12.43	0.966	0.0821
14	24.0	31.0	30.0	13.03	1.114	0.1047
20	24.0	31.0	30.0	13.31	1.159	0.1110
30	24.0	31.0	30.0	13.32	1.161	0.1094
40	24.0	31.0	30.0	13.34	1.166	0.1137
50	24.0	31.0	30.0	13.45	1.178	0.1147
60	24.0	31.0	30.0	13.41	1.172	0.1145

Solver iteration counts and nonlinear residuals (in units of 10^3).

Univ. Florida sparse matrices (SPD cases)

	Sweeps	PCG iter	Rel. nonlin. resid.
af_shell3	0	1187	1
	1	710	1.22e+00
	2	685	7.60e−02
	3	685	5.17e−03
	IC	643	0
apache2	0	1674	1
	1	1199	2.04e+00
	2	894	9.92e−01
	3	870	1.53e−01
	IC	896	0
ecology2	0	1519	1
	1	1775	1.39e+00
	2	1711	3.99e−01
	3	1707	1.98e−02
	IC	1711	0
G3_circuit	0	1515	1
	1	947	1.68e+00
	2	870	3.33e−01
	3	871	1.04e−02
	IC	780	0

	Sweeps	PCG iter	Rel. nonlin. resid.
offshore	0	2000+	1
	1	348	1.44e+00
	2	297	7.86e−02
	3	297	7.38e−05
	IC	300	0
parabolic_fem	0	1260	1
	1	502	1.65e+00
	2	412	5.45e−01
	3	405	4.71e−02
	IC	406	0
thermal2	0	1984	1
	1	1418	1.61e+00
	2	1314	4.08e−01
	3	1308	2.41e−02
	IC	1257	0

Conclusion

- ILU factorization computed approximately, leading to large amounts of fine-grained parallelism
- No matrix reordering is necessary (but nonlinear equation reordering can enhance convergence)
- ILU factorization can be improved iteratively from an initial guess

How extensible is this approach?

- Can potentially be applied to other matrix factorizations
  - low-rank matrix completions
  - sparse approximate inverses, $GAG^T \approx I$
- When there are more nonlinear equations than unknowns, use the nonlinear Kaczmarz algorithm or stochastic gradient descent
- Can trade accuracy for parallelism in triangular solves

Acknowlegment: NSF ACI-1306573 and Intel Corporation.