Factors Impacting Performance of Multithreaded Triangular Solve

VECPAR' 10

June 23, 2010

Michael Wolf, Mike Heroux, Erik Boman

Extreme-scale Algorithms and Software Institute (EASI)

Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

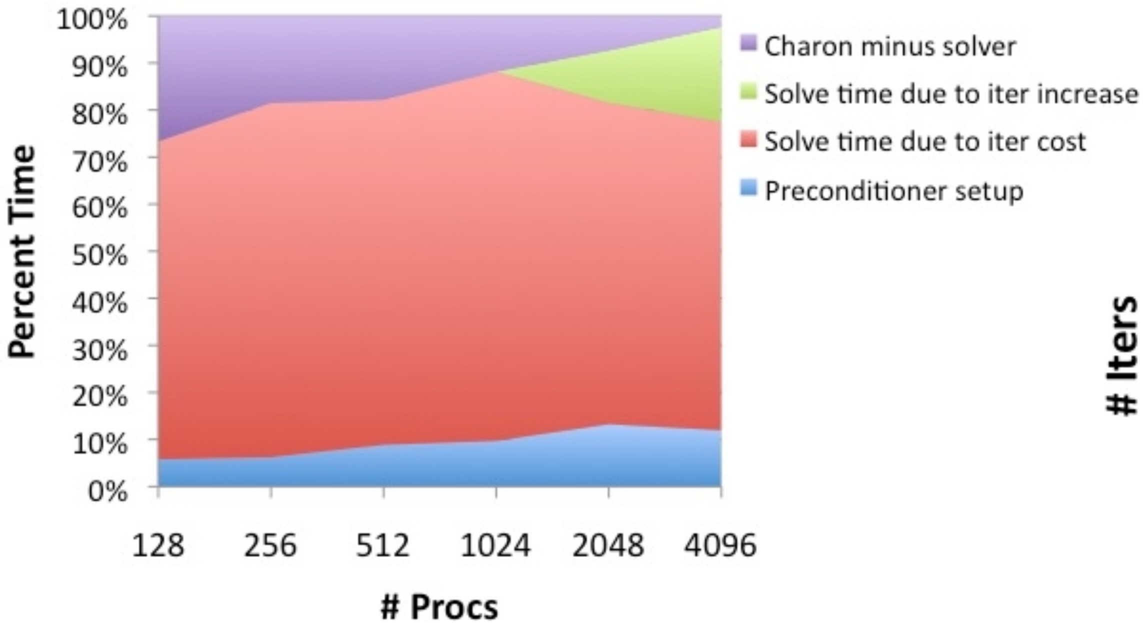
Motivation

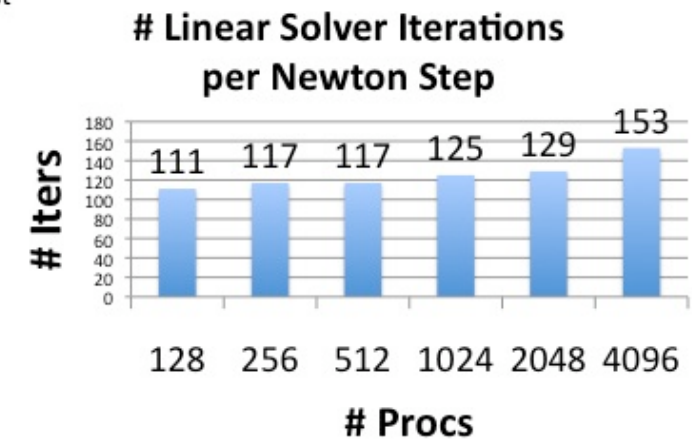
- Triangular solver is important numerical kernel
  - Essential role in preconditioning linear systems
- Difficult algorithm to parallelize

- Trend of increasing numbers of cores per socket
- Threaded or hybrid approach potentially beneficial

- Focus of work: threaded triangular solve on each node/socket

Motivation



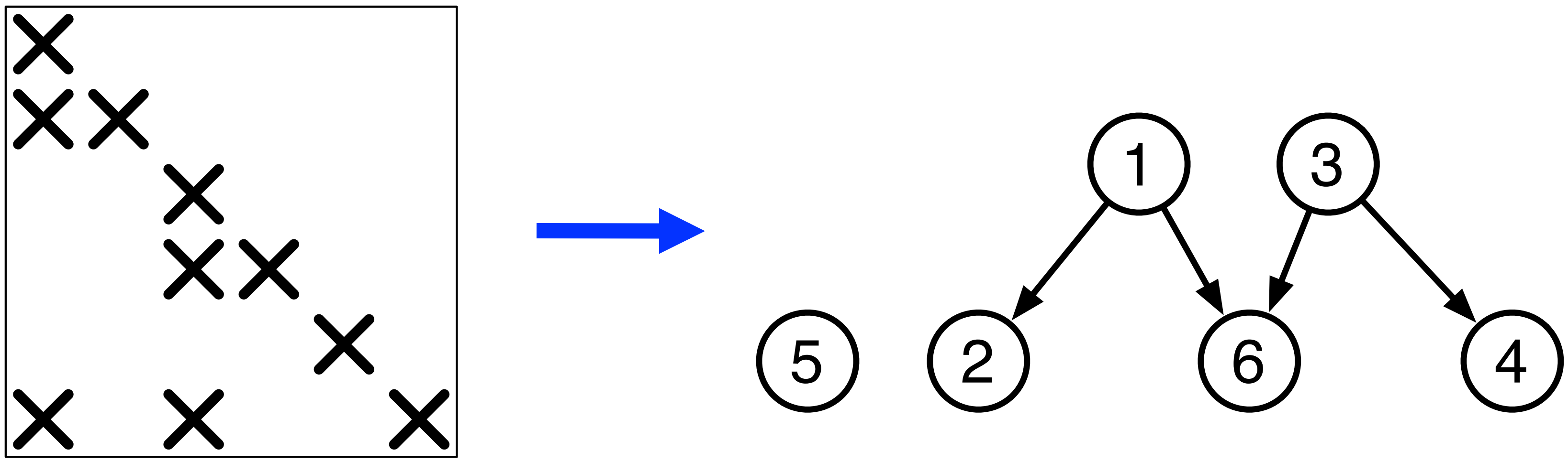


Strong scaling of Charon on TLCC (P. Lin, J. Shadid 2009)

- **Inflation in iteration count due to number of subdomains (MPI tasks)**
- **With scalable threaded triangular solves**
  - **Solve triangular system on larger subdomains**
  - **Reduce number of subdomains (MPI tasks)**

Sandia National Laboratories

Level Set Triangular Solver

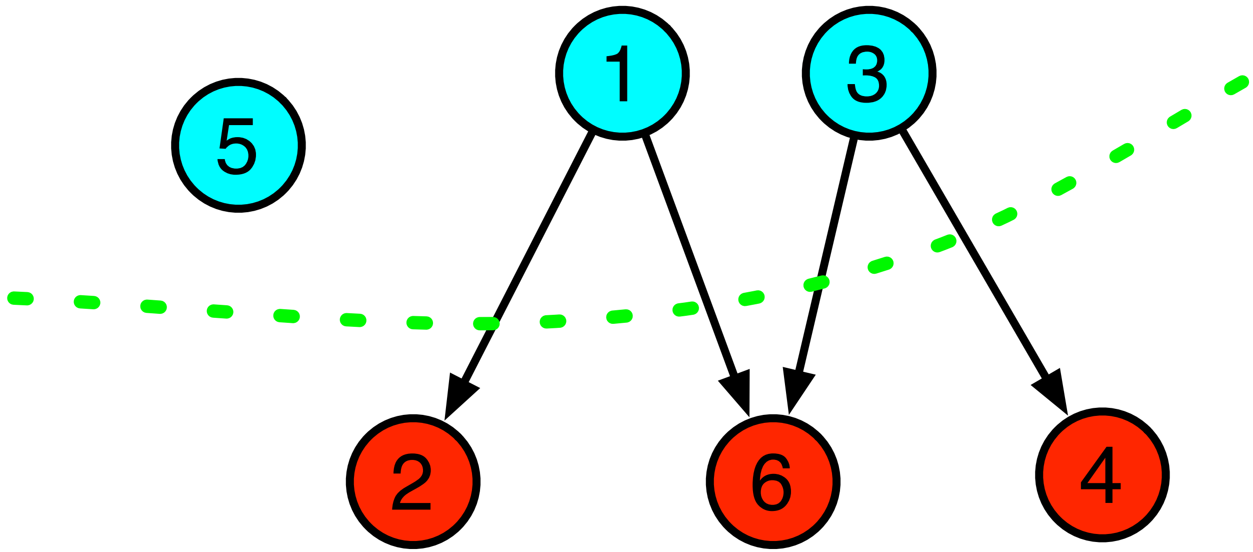


L　　　　　　　　　　DAG

- **Initially, focus attention on level set triangular solver (J. Saltz, 1990)**
  - **Level set approach exposes parallelism**
- **First, express data dependencies for triangular solve with a directed acyclic graph (DAG)**

Sandia National Laboratories

Level Set Triangular Solver



- **Determine level sets of this DAG**
  - **Represent sets of row operations that can be performed independently**

Level Set Triangular Solver

$$\tilde{L} = PLP^T = \begin{bmatrix} D_1 & & & & \\ A_{2,1} & D_2 & & & \\ A_{3,1} & A_{3,2} & D_3 & & \\ \vdots & \vdots & \vdots & \ddots & \\ A_{l,1} & A_{l,2} & A_{l,3} & \dots & D_l \end{bmatrix}$$

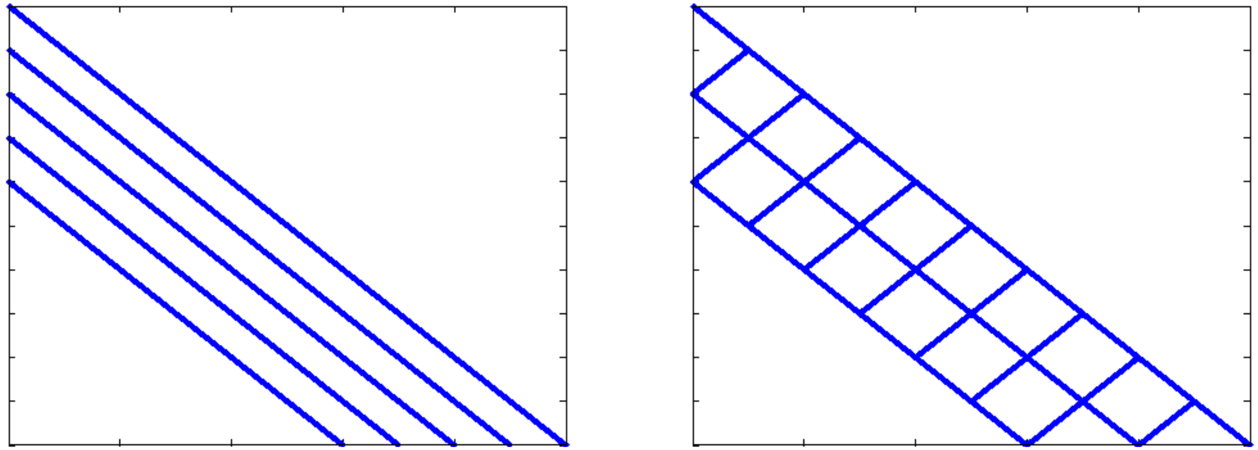
- **Permuting matrix so that rows in a level set are contiguous**
  - **D_i are diagonal matrices**
  - **Row operations in each level set can be performed independently**

Level Set Triangular Solver

$$
\begin{aligned}
\tilde{x}_1 &= D_1^{-1}\tilde{y}_1 \\
\tilde{x}_2 &= D_2^{-1}\left(\tilde{y}_2 - A_{2,1}\tilde{x}_1\right) \\
\vdots \quad &\vdots \quad \vdots \\
\tilde{x}_l &= D_l^{-1}\left(\tilde{y}_l - A_{l,1}\tilde{x}_1 - \ldots - A_{l,l-1}\tilde{x}_{l-1}\right)
\end{aligned}
$$

- **Resulting operations for triangle solve**
  - **Row operations in each level can be performed independently (parallel for)**

Simple Prototype



- **Simple prototype of level set threaded triangular solve**
  - **Assumes fixed number of rows per level**
  - **Assumes matrices preordered by level**
  - **Pthreads**
- **Allowed us to explore factors affecting performance**
- **Run experiments on two platforms**
  - **Intel Nehalem: two 2.93 GHz quad-core Intel Xeon processors**
  - **AMD Istanbul: two 2.6 GHz six-core AMD Opteron processors**

Factor 1: Type of Barrier

Algorithm 1 Passive Barrier.

```
void passiveBarrier()
{
   pthread_mutex_lock(&mutex);
   numArrived++;
   if(numArrived < NUM_THREADS) {
     pthread_cond_wait(&barrCond,&mutex);
   }
   else {
     pthread_cond_broadcast(&barrCond);
     numArrived = 0;
   }
   pthread_mutex_unlock(&mutex);
}
```

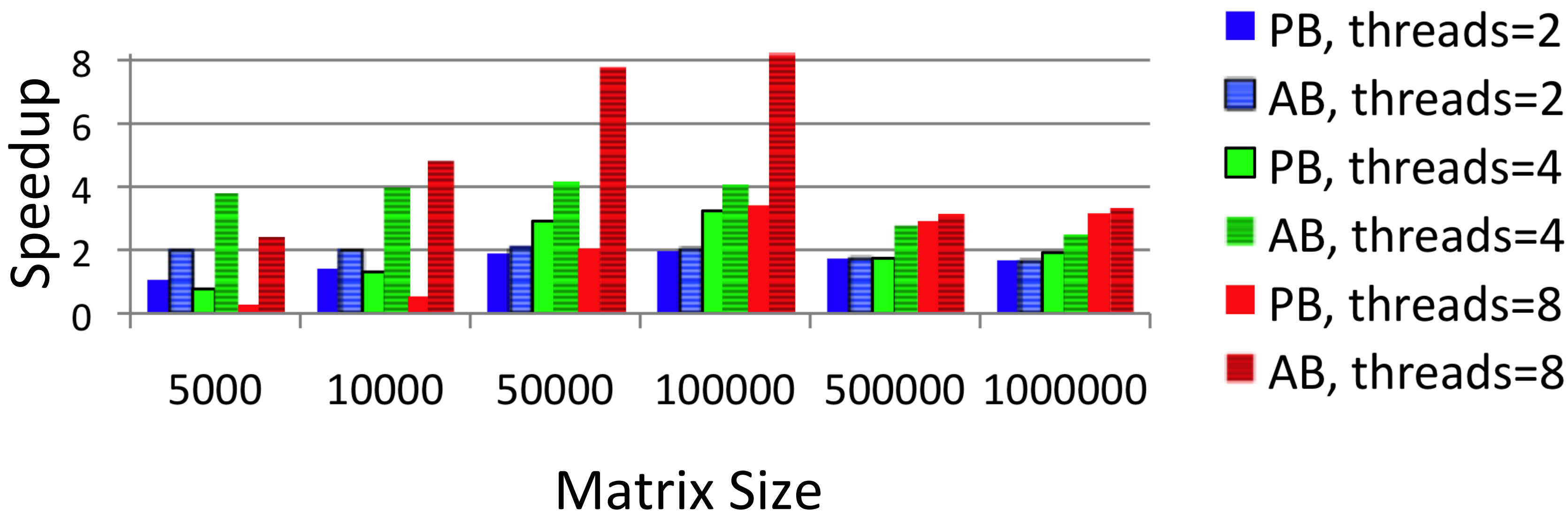
Algorithm 2 Active Barrier.

```
void activeBarrier()
{
   pthread_spin_lock(&lock);
   actNumArrived++;
   if(actNumArrived==NUM_THREADS) {
     actLoopFlag = false;
   }
   pthread_spin_unlock(&lock);

   while(actLoopFlag) {}
}
```

- **Implemented two different barriers**
  - **"Passive" barrier**
    - **Mutexes and conditional wait statements**
  - **"Active" barrier**
    - **Spin locks and active polling**

Barriers

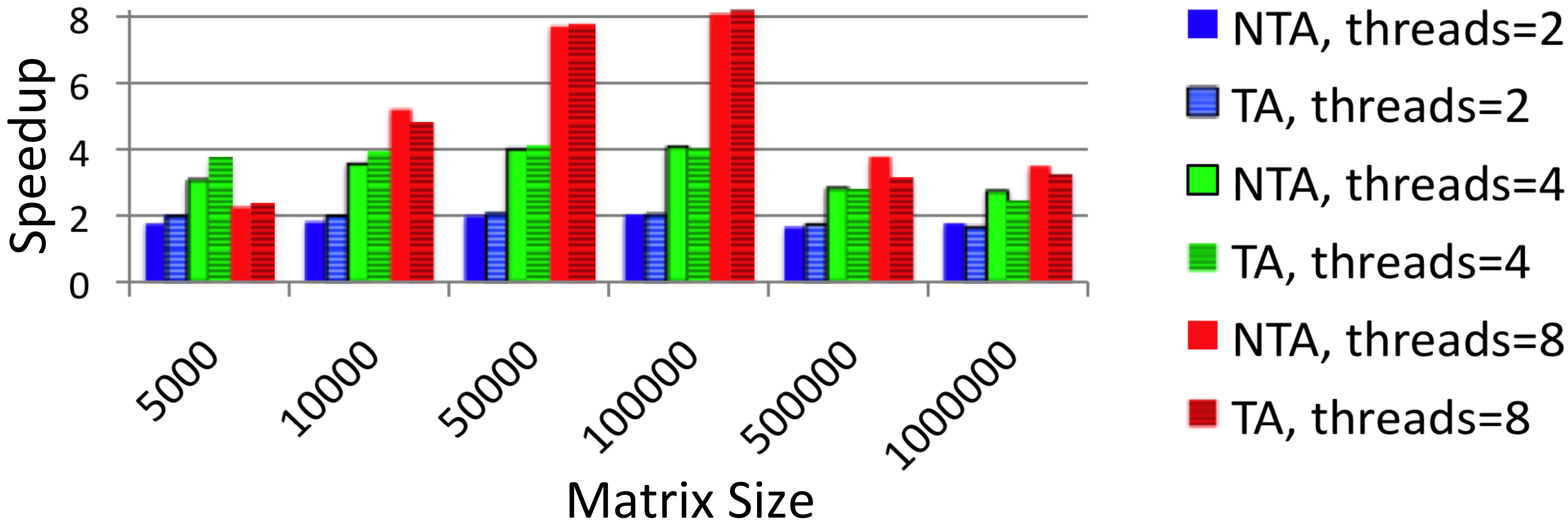


- **Results for good data locality matrices**
- **Active/aggressive barriers essential for scalability**

Factor 2: Thread Affinity

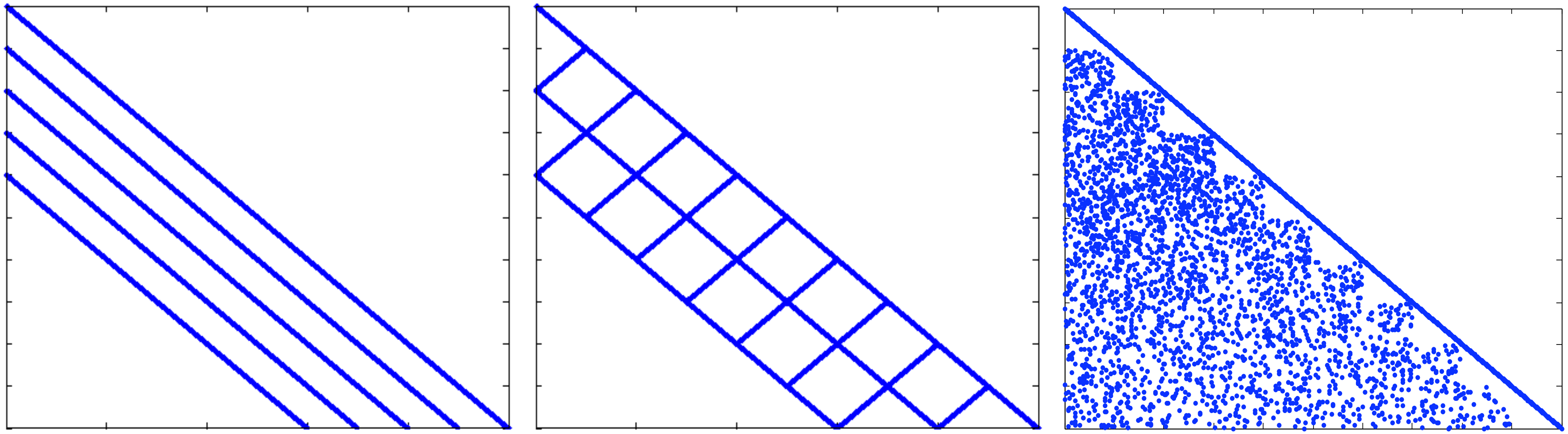
- Studied the importance of thread affinity
- Thread affinity allows threads to be pinned to cores
  - Less likely for threads to be switched (beneficial for cache utilization)
  - Ensures that threads are running on same socket

Thread Affinity



- **Results for good data locality matrices, active barrier**
- **Thread affinity not as important as active barrier**
  - **But can be beneficial for some problem sizes**

Factor 3: Data Locality

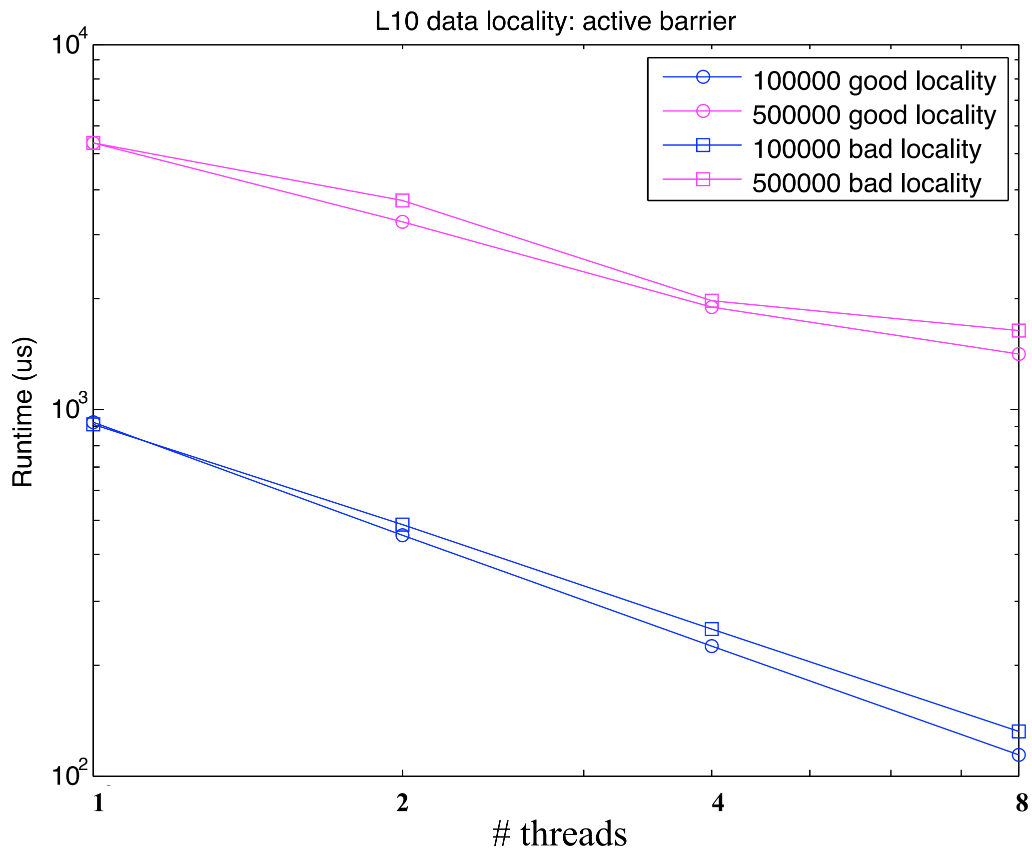


"Good" data locality — "Bad" data locality — Random

- **Examined three different types of matrices**
  - **Same number of rows per level**
  - **Same number of nonzeros per row**
- **Allowed us to explore how data locality affects performance**

Sandia National Laboratories

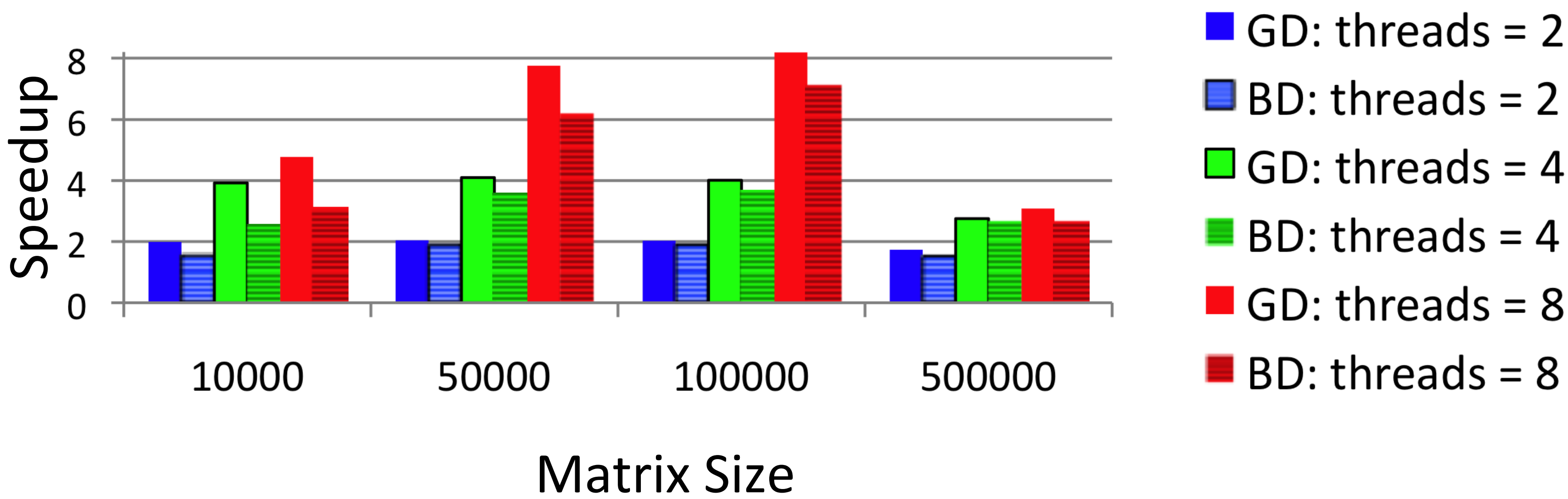
Data Locality: Good vs. Bad



- Results for good (GD) vs. bad data (BD) locality matrices
- Active barrier

Sandia National Laboratories

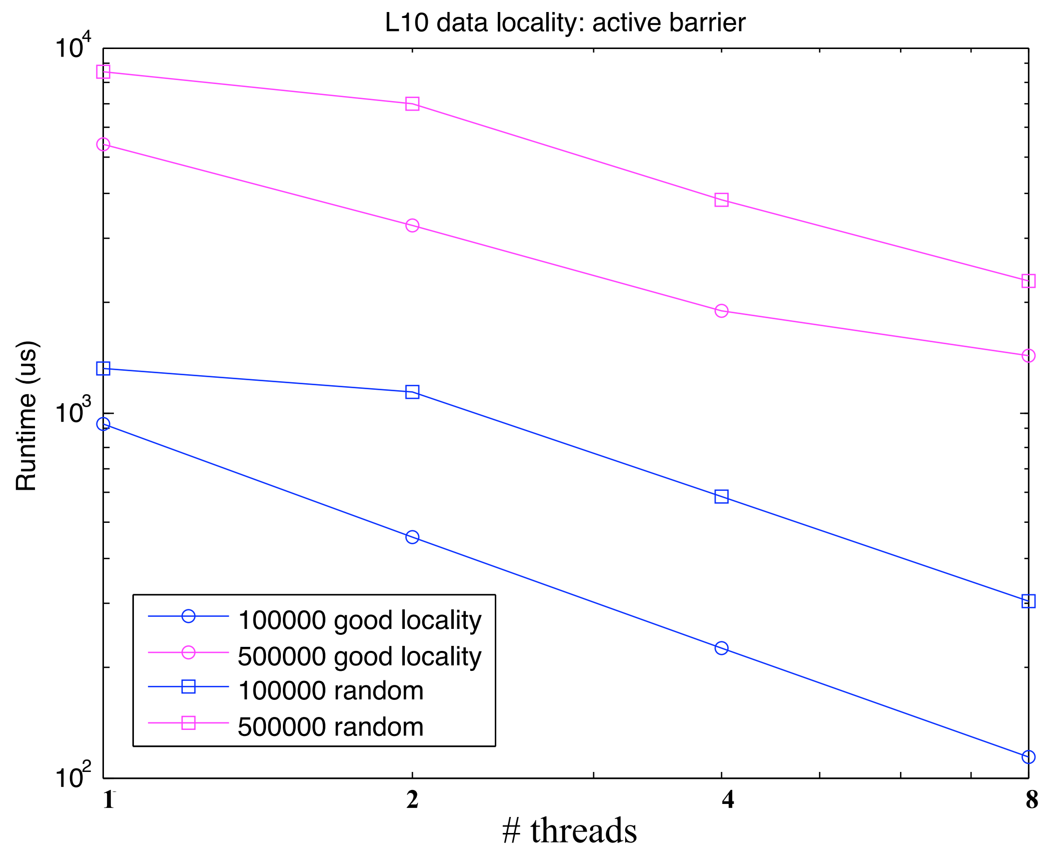
Data Locality: Good vs. Bad



- **Results for good (GD) vs. bad data (BD) locality matrices**
- **Active Barrier**

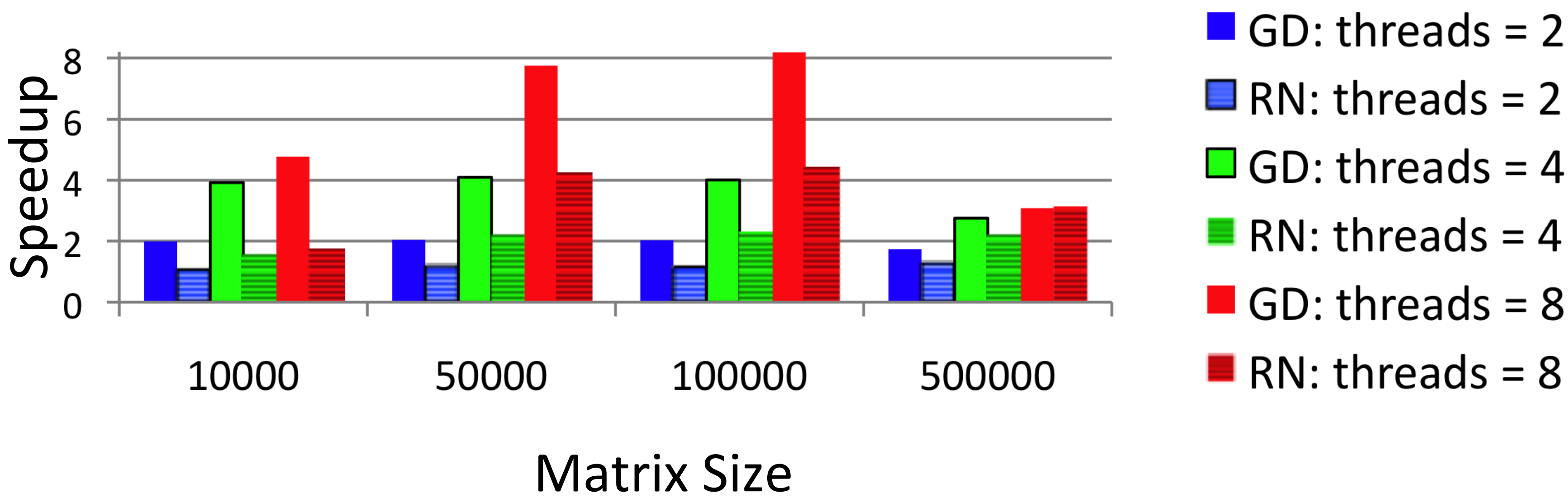
Sandia National Laboratories

Data Locality: Good vs. Random



- **Results for good data locality vs. random matrices**
- **Active barrier**

Data Locality: Good vs. Random



- **Results for good data locality (GD) vs. random (RN) matrices**
- **Active Barrier**

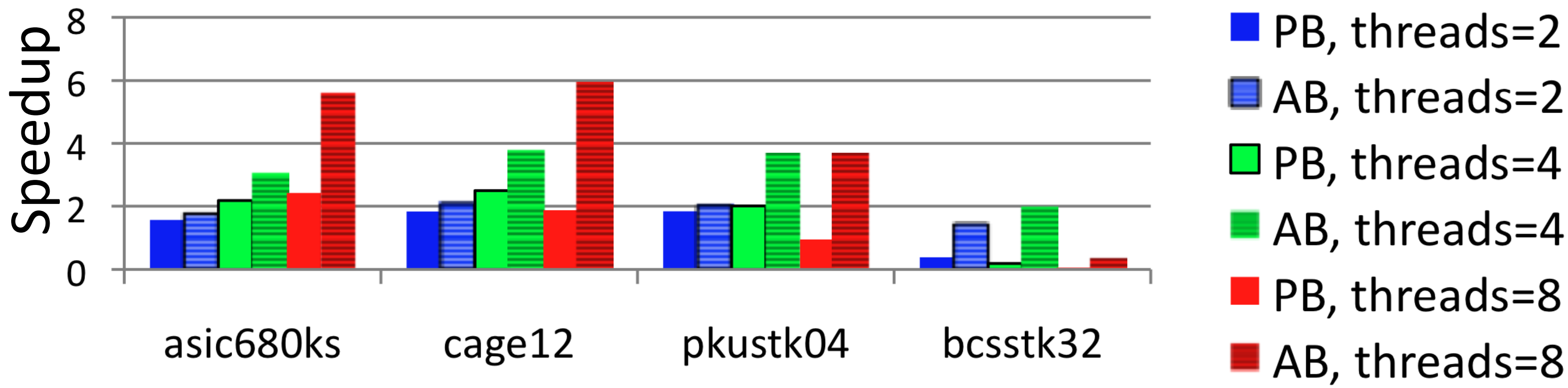
Sandia National Laboratories

More Realistic Problems

Name	N	nnz	N / nlevels	Application area
asic680ks	682,712	2,329,176	13932.9	circuit simulation
cage12	130,228	2,032,536	1973.2	DNA electrophoresis
pkustk04	55,590	4,218,660	149.4	structural engineering
bcsstk32	44,609	2,014,701	15.1	structural engineering

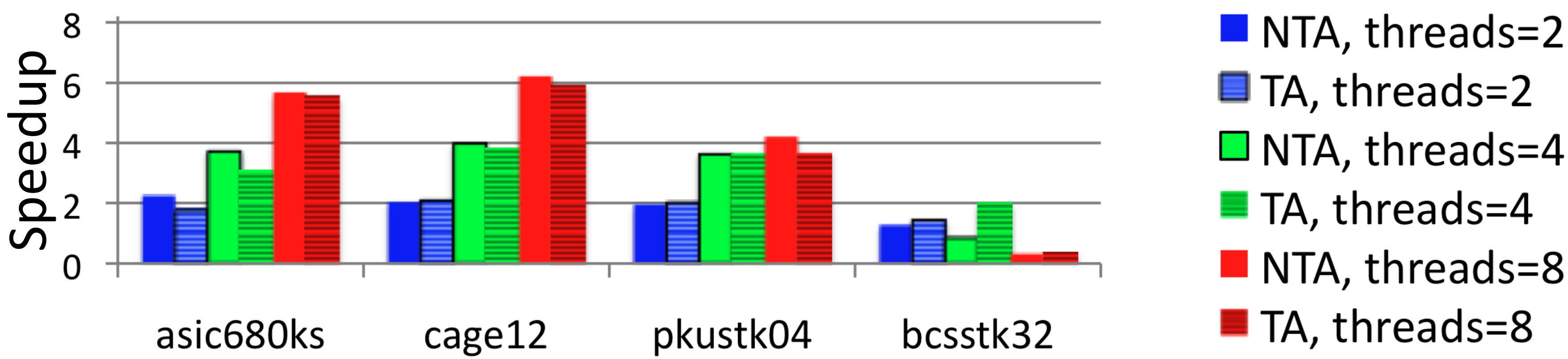
- **Symmetric matrices**
- **Incomplete Cholesky factorization (no fill)**
- **Average size of level important**

Realistic Problems: Barriers



- **Problems with larger average level size scale fairly well**
- **Active/aggressive barrier important**

Realistic Problems: Thread Affinity



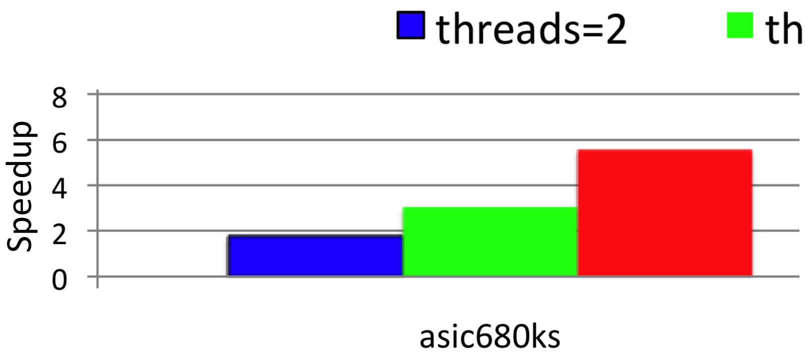
- Problems with larger average level size scale fairly well
- Thread affinity not particularly important

Level Set Triangular Solver Extension

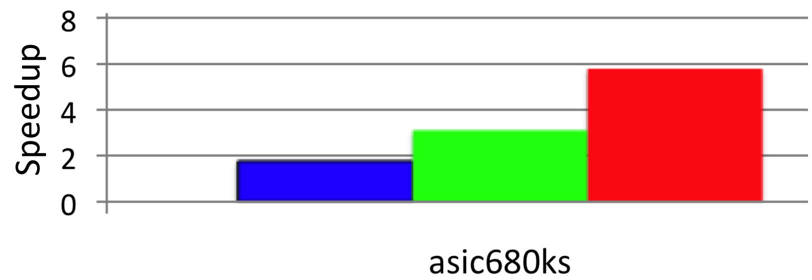
- Algorithm scales when average level size is high
- Couple factors hurt performance for small average level size
  - Many levels, many synchronization points
  - Not enough work in small levels (barrier cost significant)

- Implemented simple extension to address these problems
  - Serialize small levels below a certain threshold
  - Merge consecutive serialized levels
  - Reducing levels reduces synchronization points

Level Set Triangular Solver Extension



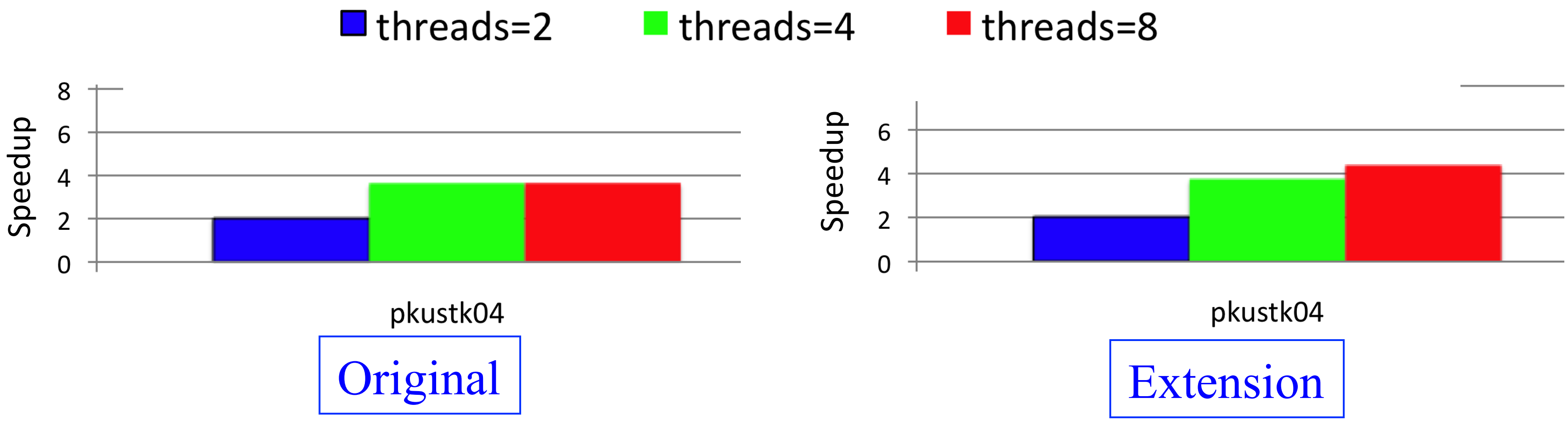
Original



Extension

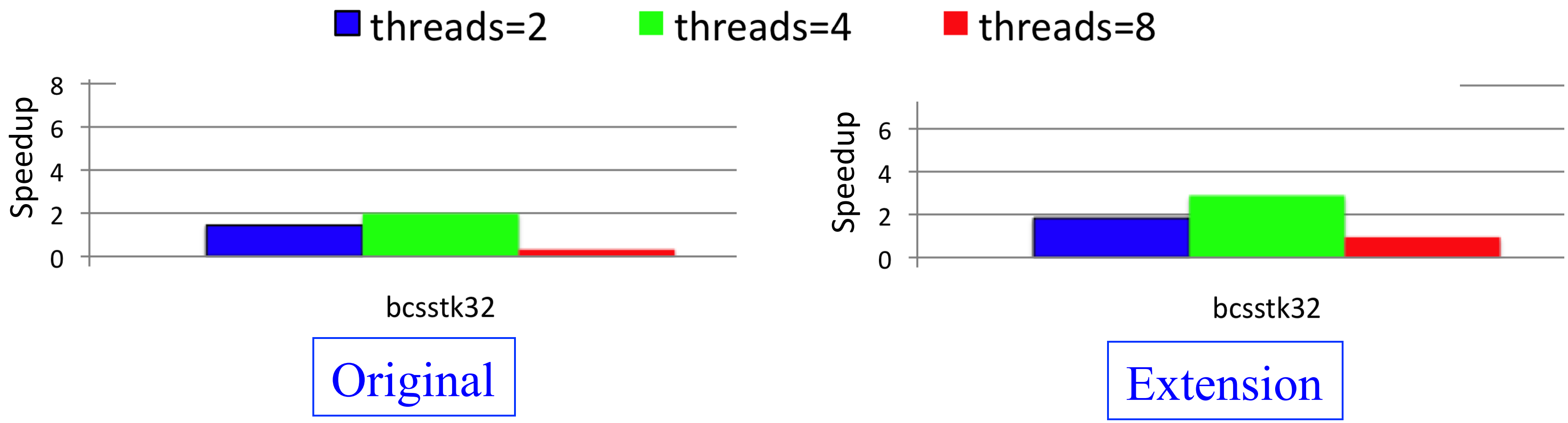
- **Very slight improvement for problem that scale well**
  - **Not many small levels**
  - **Can reduce speedup if too aggressive in serialization**

Level Set Triangular Solver Extension



- Slight improvement for problem that originally did not scale quite so well
  - More small levels

Level Set Triangular Solver Extension



- **Significant improvement for problem that originally did not scale well**
  - **Many small levels**
  - **Great reduction in synchronization points**
- **Still does not scale well for 8 threads**

Summary/Conclusions

- Presented threaded triangular solve algorithm
  - Level scheduling algorithm
- Studied impact of three factors on performance
  - Barrier type most important
- Good scalability for simple matrices and two realistic problems
- Scalability related to average level size
  - Simple extension to improve results when level sizes are small
  - Better algorithms needed for matrices with small average level size
- Algorithms being implemented in Trilinos
  - http://trilinos.sandia.gov