STRUCTURE-ADAPTIVE PARALLEL SOLUTION OF SPARSE TRIANGULAR LINEAR SYSTEMS

EHSAN TOTONI

PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL UIUC

May 8, 2012
10th Charm++ Workshop

Sparse Triangular Solution

- Used in solution of linear systems, least squares
  - Many times iteratively
- Direct methods (after factorization)
  - many right-hand sides
  - Refinement of solution
- Iterative methods
  - kernel in Gauss-Seidel method
  - Preconditioners
    - E.g. Incomplete-Cholesky before Conjugate Gradient

Poor Parallelism

$$x_i = (b_i - \sum_{j=1}^{i-1} l_{ij}\, x_j)/l_{ii}, \quad i = 1, \ldots, n$$

- Minimal concurrency
  - Lots of structural dependencies
- Small work per data
  - Just one multiply-add for most entries!
- Sparse: Some parallelism

Poor Parallelism

- Slower than sequential!
  - HYPRE and SuperLU_DIST
  - Some progress in shared-memory
- Has to be done in parallel
  - Matrix is already distributed
    - E.g. by factorization
  - Memory is constrained
- Bottle-neck of many methods

Basics Approach

$$
\begin{bmatrix}
l_{11} & & & & & & & & \\
l_{21} & l_{22} & & & & & & & \\
 & & l_{33} & & & & & & \\
 & & l_{43} & l_{44} & & & & & \\
 & & & l_{54} & l_{55} & & & & \\
 & & & & & l_{66} & & & \\
 & & & & & l_{76} & l_{77} & & \\
l_{81} & & & & & & & l_{88} & \\
 & & & & & & & & l_{99}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix}
$$

P1 P2 P3

Basic Approach-Reordering

Independent rows

$$
\left[\begin{array}{ccc|ccc|ccc}
 & & l_{33} & & & & & & \\
l_{11} & & & & & & & & \\
l_{21} & l_{22} & & & & & & & \\
 & & & & & l_{66} & & & \\
 & & l_{43} & l_{44} & & & & & \\
 & & & l_{54} & l_{55} & & & & \\
 & & & & & & & & l_{99} \\
 & & & & & l_{76} & l_{77} & & \\
l_{81} & & & & & & & l_{88} &
\end{array}\right]
$$

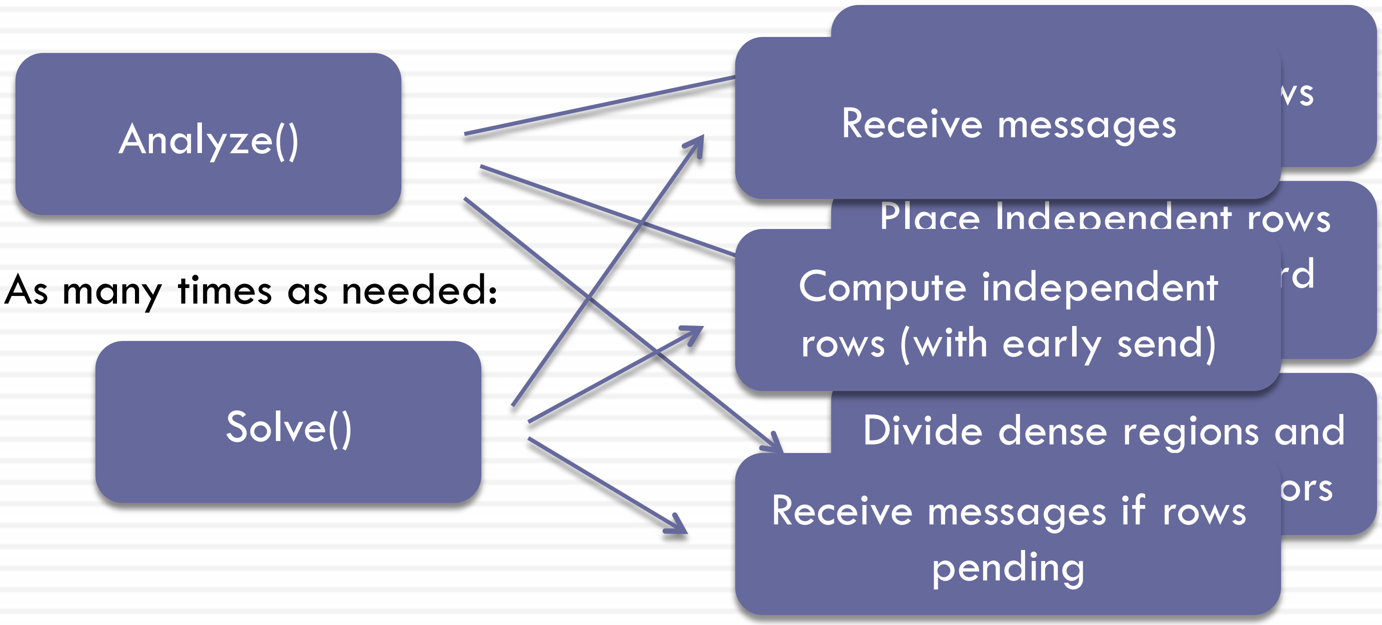
Early send

Dense Regions

Broadcast cost

$$
\begin{array}{c}
\\ \\ \\ \\ \\ \\ \\ \text{P4} \\ \text{P5}
\end{array}
\begin{bmatrix}
l_{11} & & & & & & & & \\
l_{21} & l_{22} & & & & & & & \\
 & & l_{33} & & & & & & \\
 & & l_{43} & l_{44} & & & & & \\
 & & & l_{54} & l_{55} & & & & \\
 & & & & & l_{66} & & & \\
 & & & & & l_{76} & l_{77} & & \\
l_{81} & l_{82} & l_{83} & & & & & l_{88} & \\
l_{91} & l_{92} & l_{93} & & & & & & l_{99}
\end{bmatrix}
$$

Algorithm- high level view



Implementation in Charm++

- Chare array for blocks of columns
  - Virtualization
  - Built-in round-roubin
  - Priority of data messages over compute
- Message aggregation
- Virtualization ratio trade-off

MPI implementation

- More effort but possible
- Multiple column blocks per processor
  - Virtualization illusion
  - Mapping
- MPI_Iprobe for priorities
- Give up virtualization
  - For easy programming
  - Some performance loss

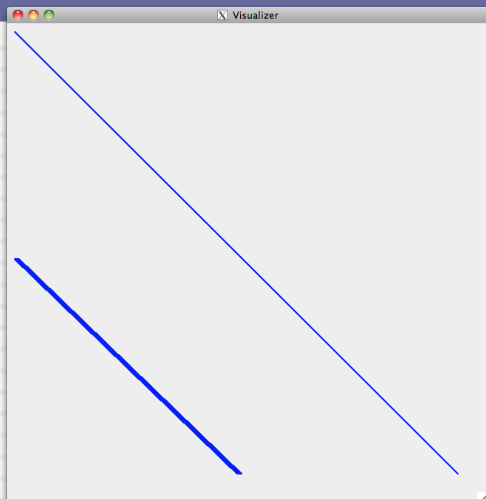
Evaluation

- Performance highly depends on matrix structure!
  - Real application matrices
  - Many different ones
  - Strong scaling (pretty small matrices!)
- Up to 512 nodes of BG/P
  - 1 core per node
  - Simple sequential kernel
- Comparison with standard packages
  - HYPRE, SuperLU_DIST

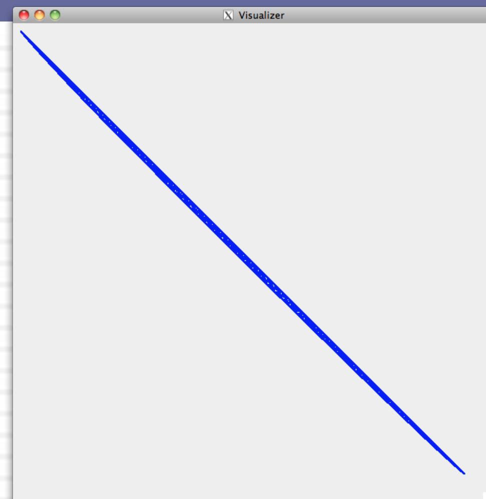
Benchmark matrices

Name	Dimension	Independent rows	Nonzeros	Nondiagonal Nonzero	Domain
circuit5M dc	3,523,317	674,311	10,631,719	4,110,848	circuit simulation
slu c-big	345,241	345,141	499,807	17,038	optimization
slu bbmat	38,744	6,735	17,819,183	15,762,657	fluid dynamics
nlpkkt120	3,542,400	1,814,400	50,194,096	46,651,696	optimization
…					

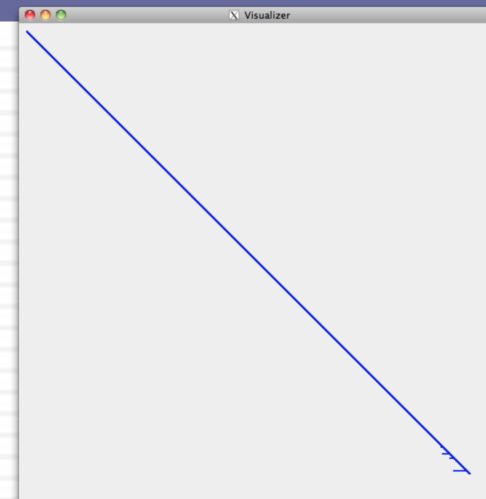
Matrix structures



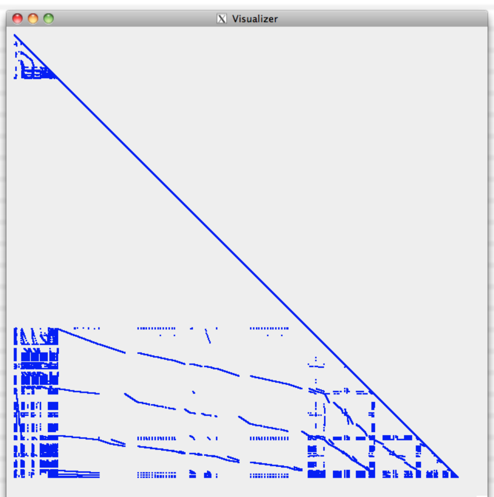
(a) nlpkkt120



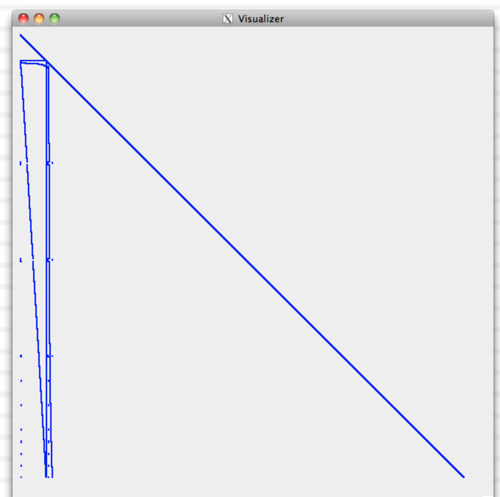
(b) Geo_1438



(c) slu_c-big

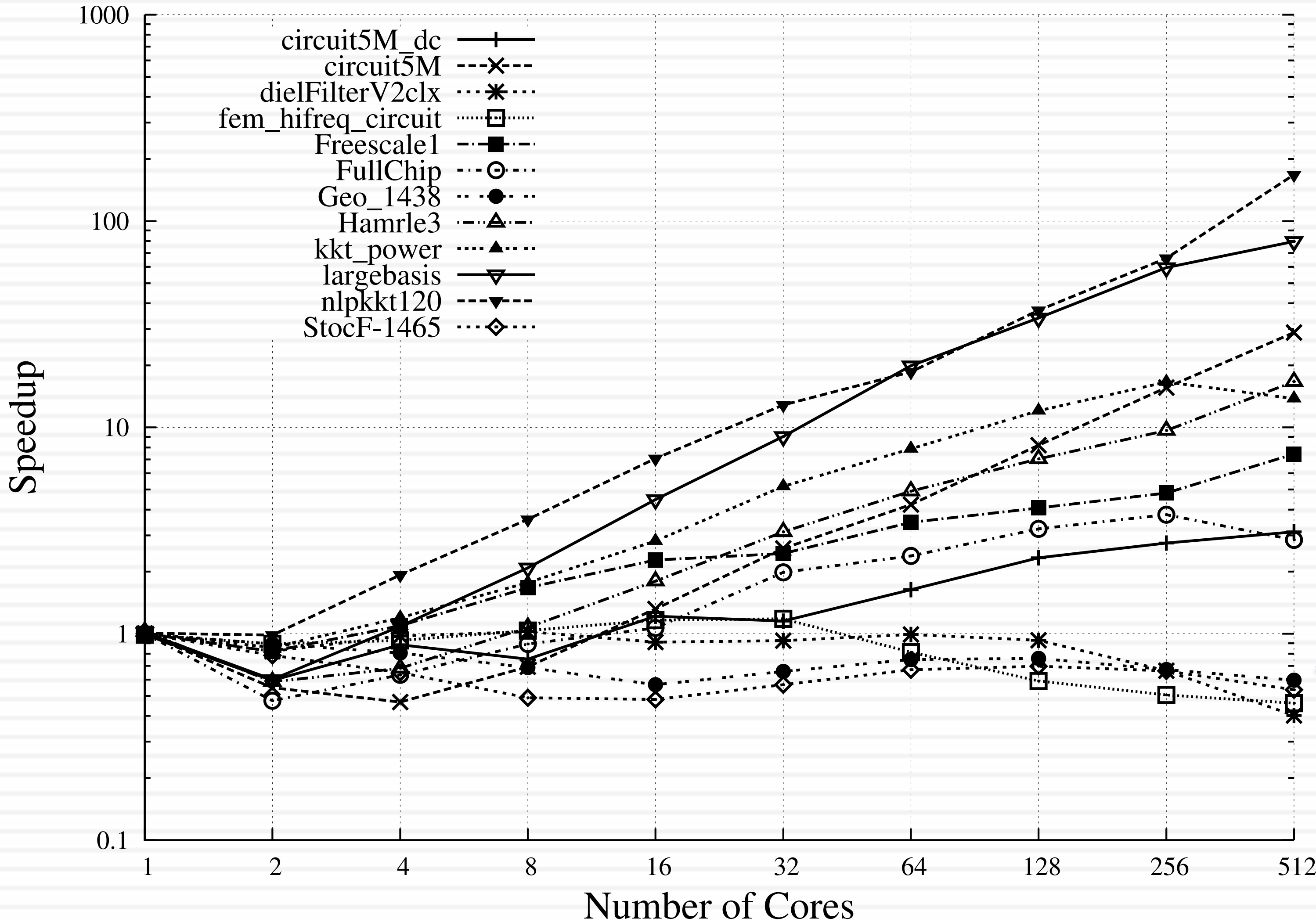


(d) Freescale1



(e) circuit5M

No-fill Incomplete-LU scaling

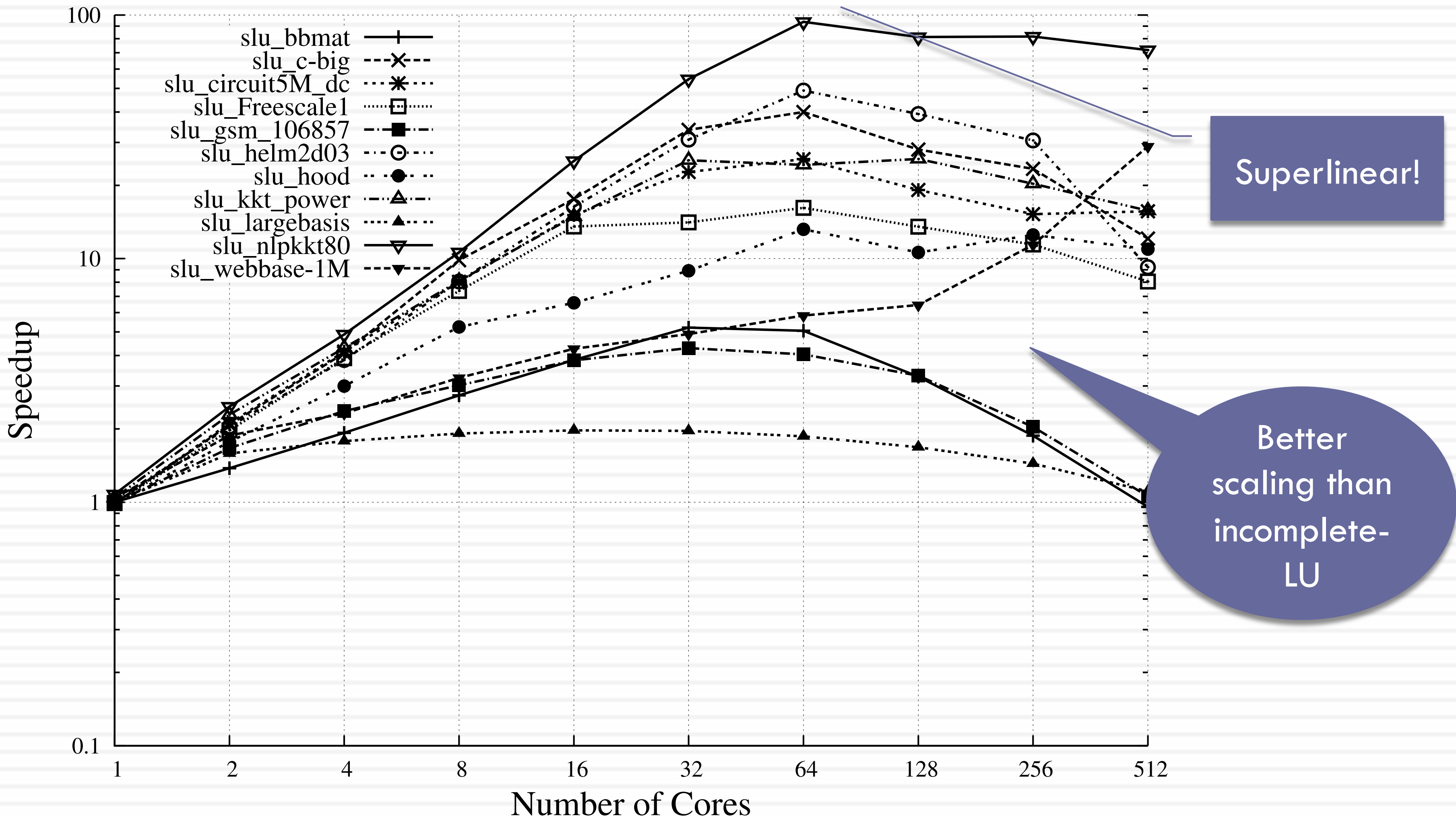


Good

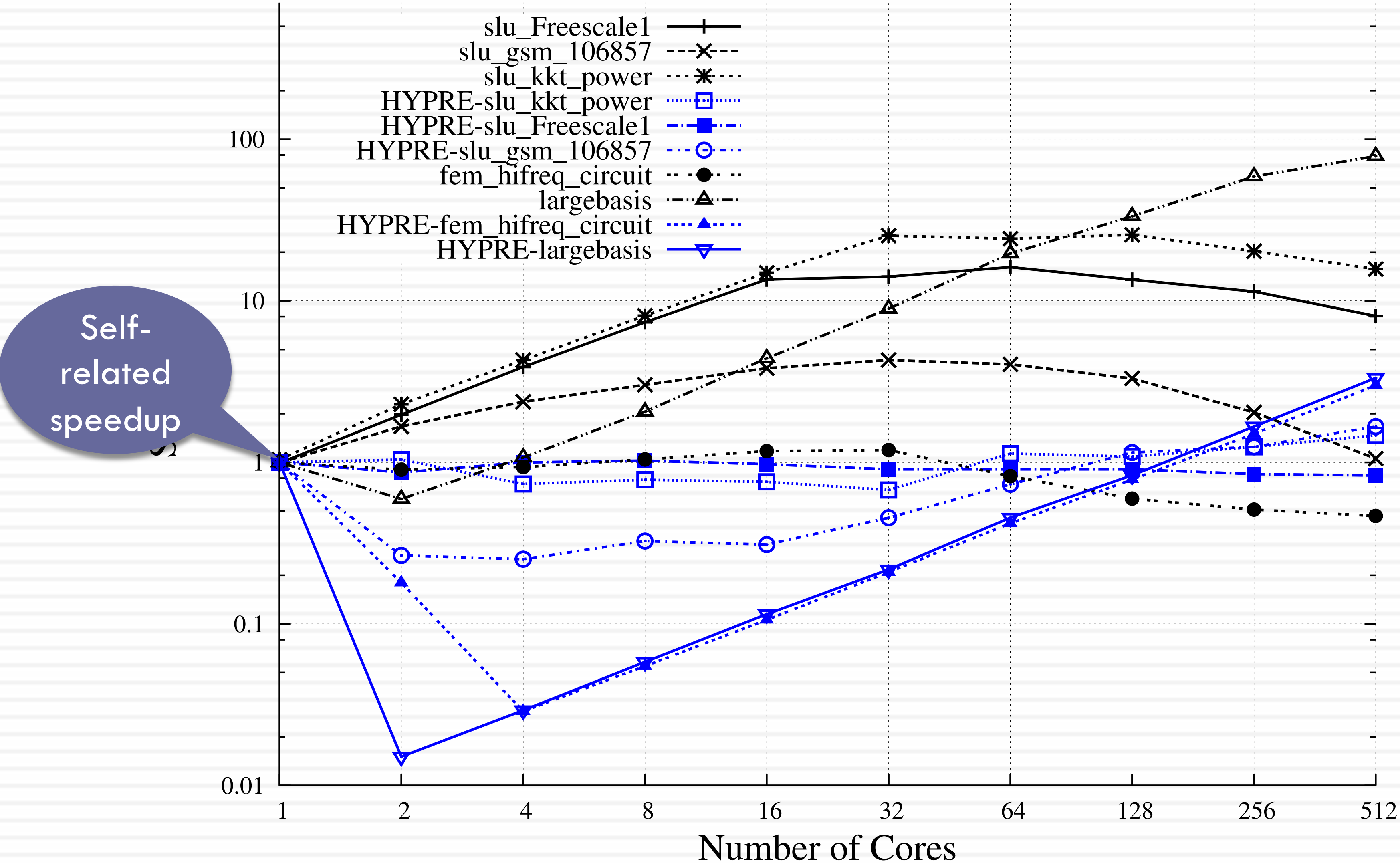
Moderate

Poor

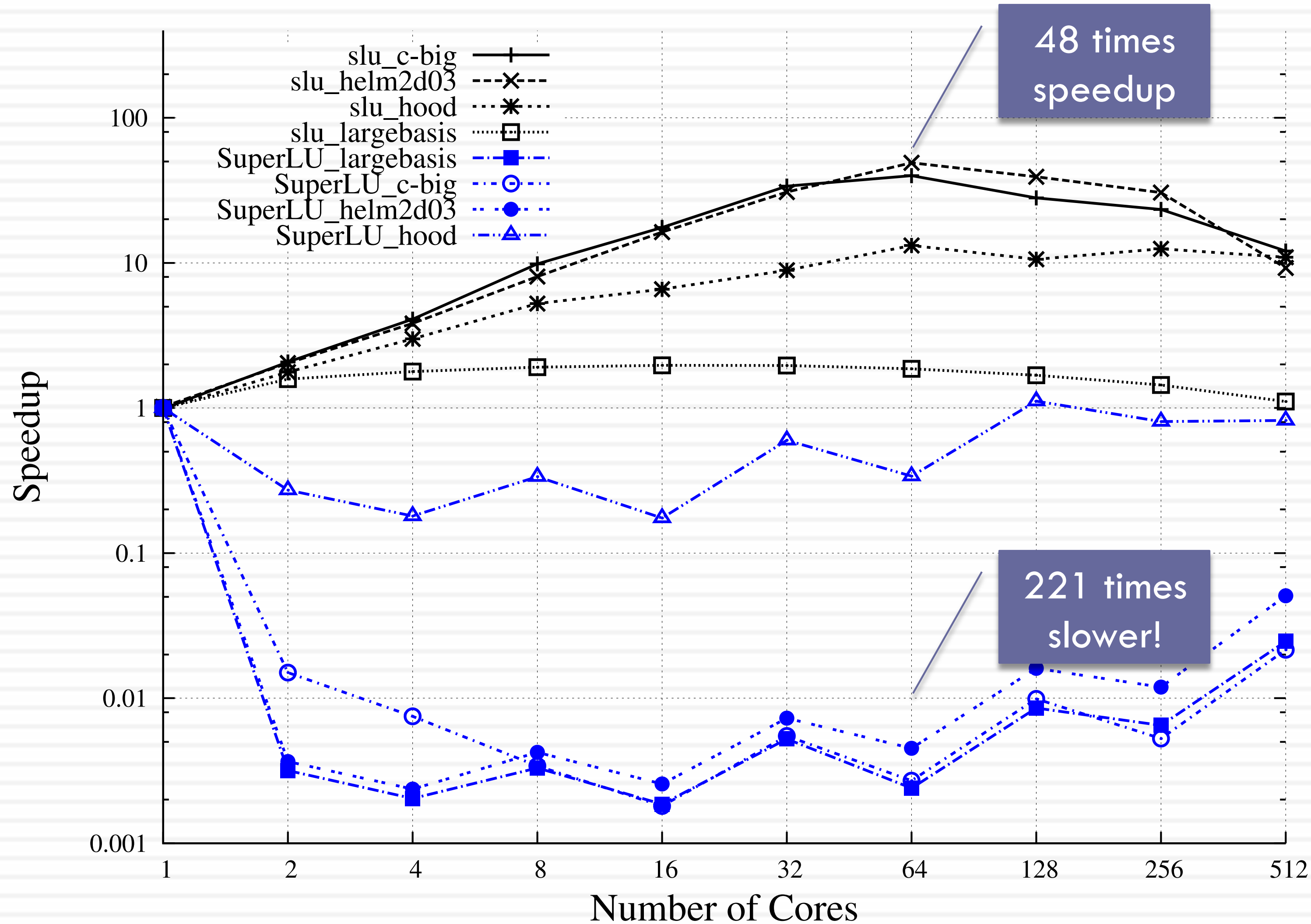
Complete-LU scaling



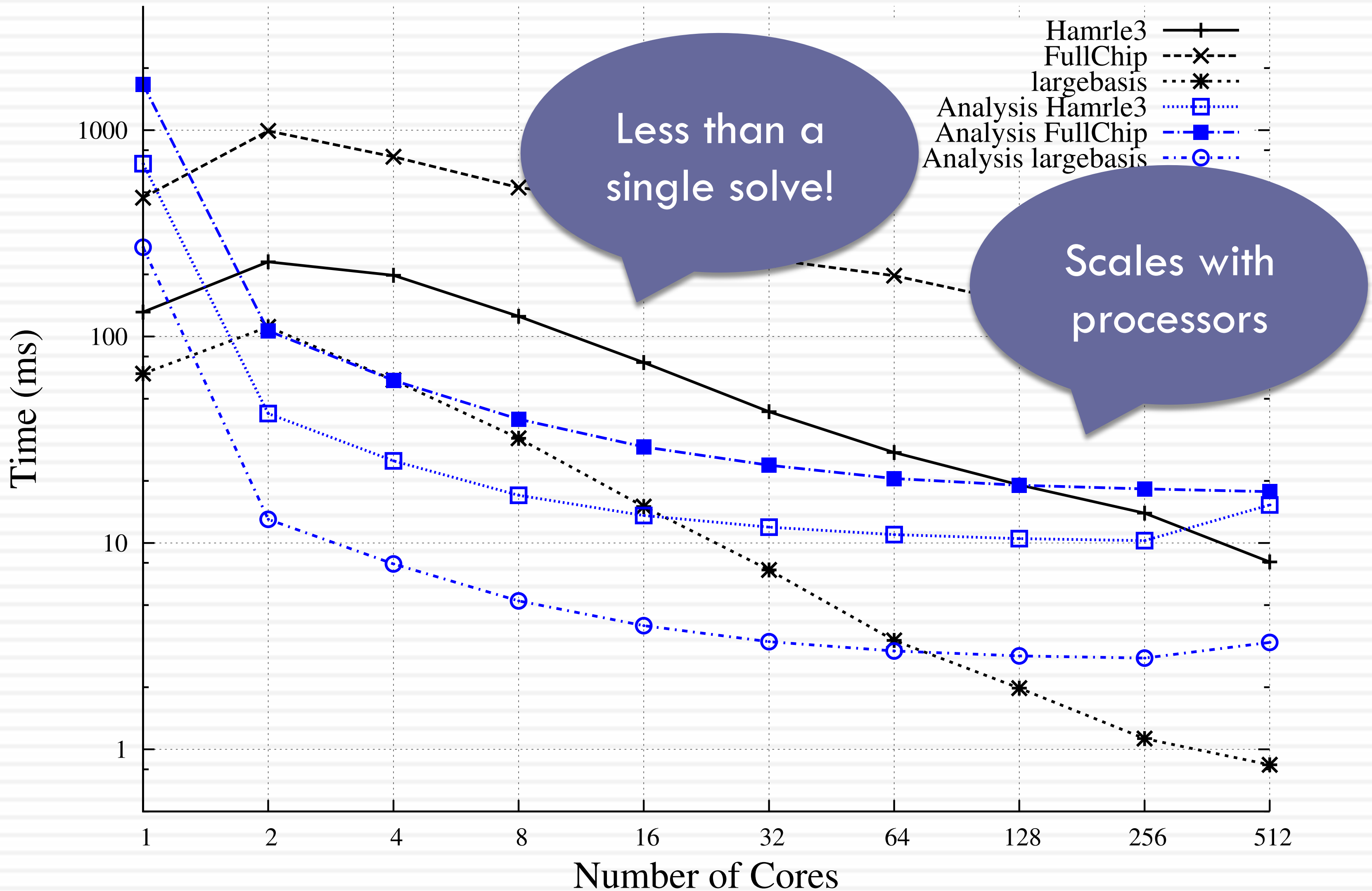
Comparison to HYPRE



Comparison to SuperLU_DIST



Analysis time



Conclusion

- Parallel solution of sparse triangular systems
  - Needed for many solvers
  - Notoriously hard to parallelize!
- A novel parallel algorithm
  - Many heuristics
    - Analysis and reordering
- Implementation in Charm++
  - Useful features such as virtualization

Future work

- Mapping column blocks to processors
  - Better balance
  - Less communication latency
- Smart priorities
  - Different blocks
  - data messages
- Virtualization ratio
- Message aggregation

Questions?