Parallel banded preconditioners for non-symmetric linear system solvers

Murat Manguoglu, Ahmed Sameh, and Ananth Grama
Department of Computer Science, Purdue University

{mmanguog,sameh,ayg}@cs.purdue.edu

June 12, 2008

Acknowledgment: Funding for this work was provided by the US National Science Foundation

Theses of Talk

- *Building on our prior work demonstrating the effectiveness of Spike as a solver and preconditioner, we derive highly scalable parallel formulations and demonstrate performance on large parallel platforms.*

- *Using a pseudo-analytical performance model, we derive estimates of parallel time, show these estimates to be highly accurate, and use these estimates to argue scaling of Spike to much larger machine configurations.*

Layout of Presentation

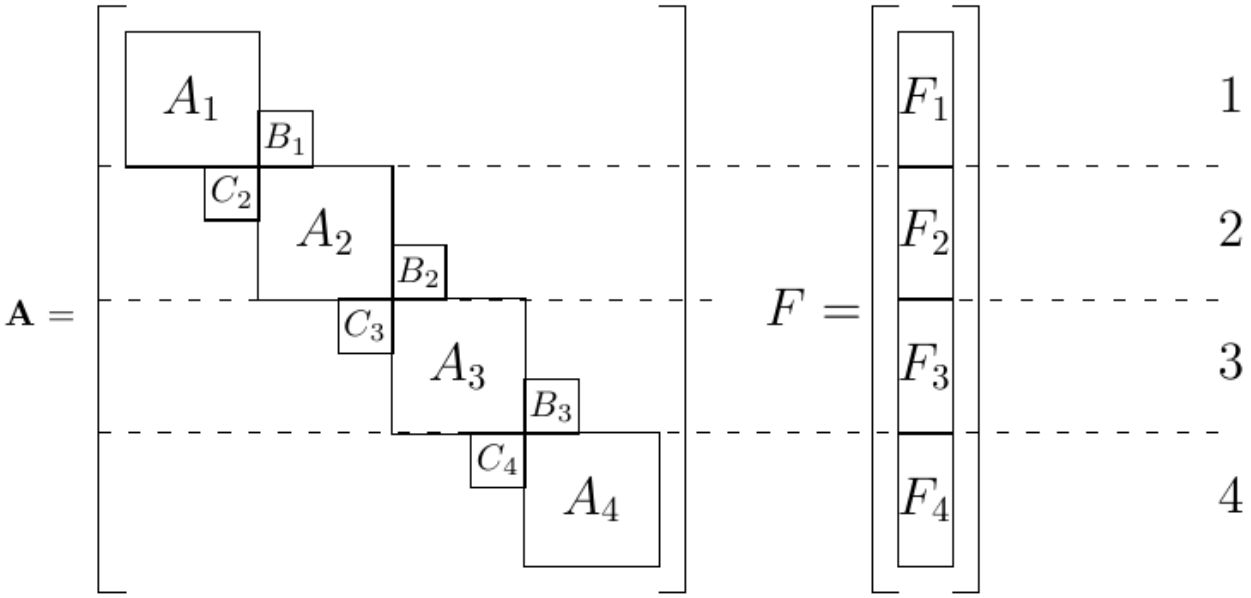
- Summary of Spike Performance (Serial)
- Spike Parallelization Strategy
- Parallel Performance Results
- Pseudoanalytical Performance Model for Spike
- Validation of Performance Model
- Limitations of Approach

Summary of Spike Solver

- Targeted to banded, or low-rank perturbations of banded systems (dense or sparse within the band).
- Banded approximations used as preconditioners.
- Spike is designed to optimize memory system as well as parallel performance.

Summary of Spike Solver

Solving $Ax = F$ with four partitions:



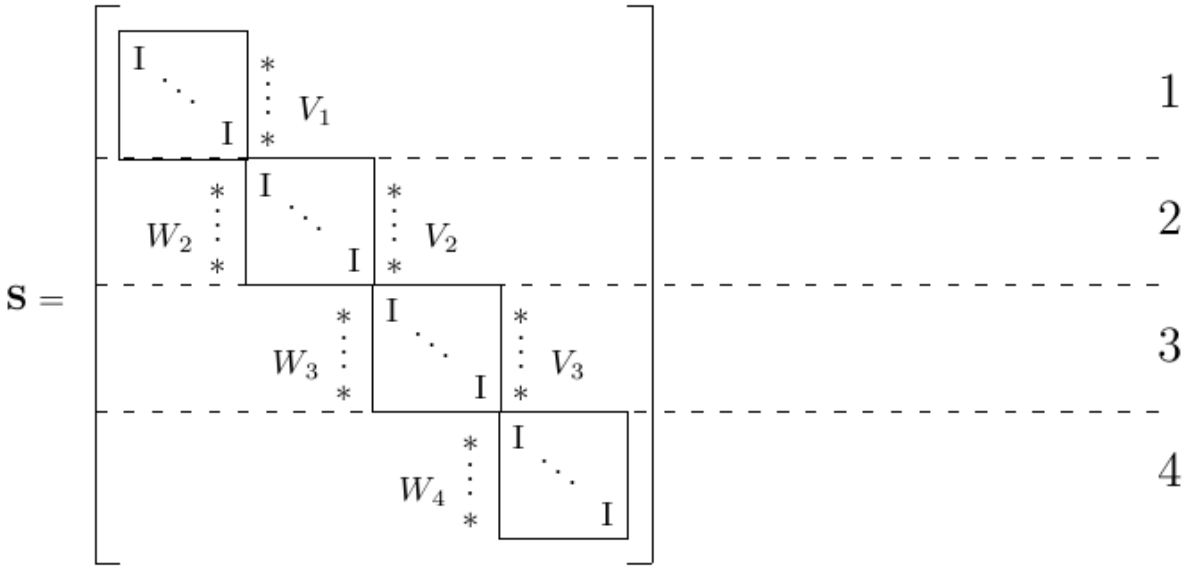
Partitioning of the matrix A and the RHS F, with $p = 4$.

Summary of Spike Solver

- Spike factorization is: $A = D\ S$, where D is a block-diagonal matrix consisting only of the diagonal blocks A_j,

$$D = \mathrm{diag}(A_1, ..., A_p),$$

- Matrix S has the following structure:



Summary of Spike Solver

- The spikes V_j and W_j are given by

$$V_j = (A_j)^{-1} \begin{bmatrix} 0 \\ I_m \end{bmatrix} B_j, \quad \text{and} \quad W_j = (A_j)^{-1} \begin{bmatrix} I_m \\ 0 \end{bmatrix} C_j. \qquad (1)$$

- Spikes V_j and W_j, $j = 2, ..., p-1$, are generated by solving,

$$A_j \left[V_j, W_j \right] = \begin{bmatrix} 0 & C_j \\ \vdots & 0 \\ 0 & \vdots \\ B_j & 0 \end{bmatrix}. \qquad (2)$$

- Solving the system $AX = F$ now reduces to two steps: $DG = F$ and $SX = G$.

Spike Performance (Serial)

- Spike used as a preconditioner with BiCGStab as the iterative solver.

- Comparison with:
  - ILUPACK: Multilevel ILU (Bollhofer)
    `http://www.math.tu-berlin.de/ilupack/`
  - ILUT: Incomplete LU Factorization from Sparsekit (Saad)
    `http://www-users.cs.umn.edu/~saad/software/SPARSKIT/spa`
  - ILUT-I: Improved ILUT (Benzi) (reorder using HSL-MC64 to maximize the product of the diagonals and scale the matrix, apply symmetric RCM reordering, incomplete factorization via ILUT)

Spike-Based Preconditioning: Preprocessing

Extracting a banded preconditioner:

- reorder using HSL-MC64 to make the diagonal zero free
- reorder $\|A\| + \|A^T\|$ using HSL-MC73 to place larger elements closest to the main diagonal
- extract a banded preconditioner, such that 99.9% of the weight is inside the band
- factorize the banded preconditioner

Spike Performance

Test Problems:

Matrix Name	*Application*	*n*	*nnz*
1. ASIC_680K	*Circuit Simulation*	*680,000*	*2, 638, 997*
2.DC1	*Circuit Simulation*	*116, 835*	*766, 396*
3. FINAN512	*Econometrics*	*74, 752*	*596, 992*
4. H2O	*Quantum Chemistry*	*67, 024*	*2, 216, 736*
5. 2D_54019_HIGHK	*Device Simulation*	*54, 019*	*996, 414*
6. APPU	*NASA Benchmark*	*14, 000*	*1, 853, 104*

Spike Performance

Comparison to ILUPACK AMF/PQ preconditoners on an uniprocessor

Method\Matrix Number	1	2	3	4	5	6
ILUPACK-AMF	>600 s	Conv.	Best	Conv.	Conv.	Conv.
ILUPACK-PQ	>600 s	Conv.	Conv.	Best	Conv.	Conv.
WSO	Best	Best	Conv.	Conv.	Best	Best

Outer Iterative Solver: unrestarted GMRES, ILUPACK Parameters: droptol : 1e-1 , bound for inv(L), inv(U): 10 , elbow space: 100

Spike Performance

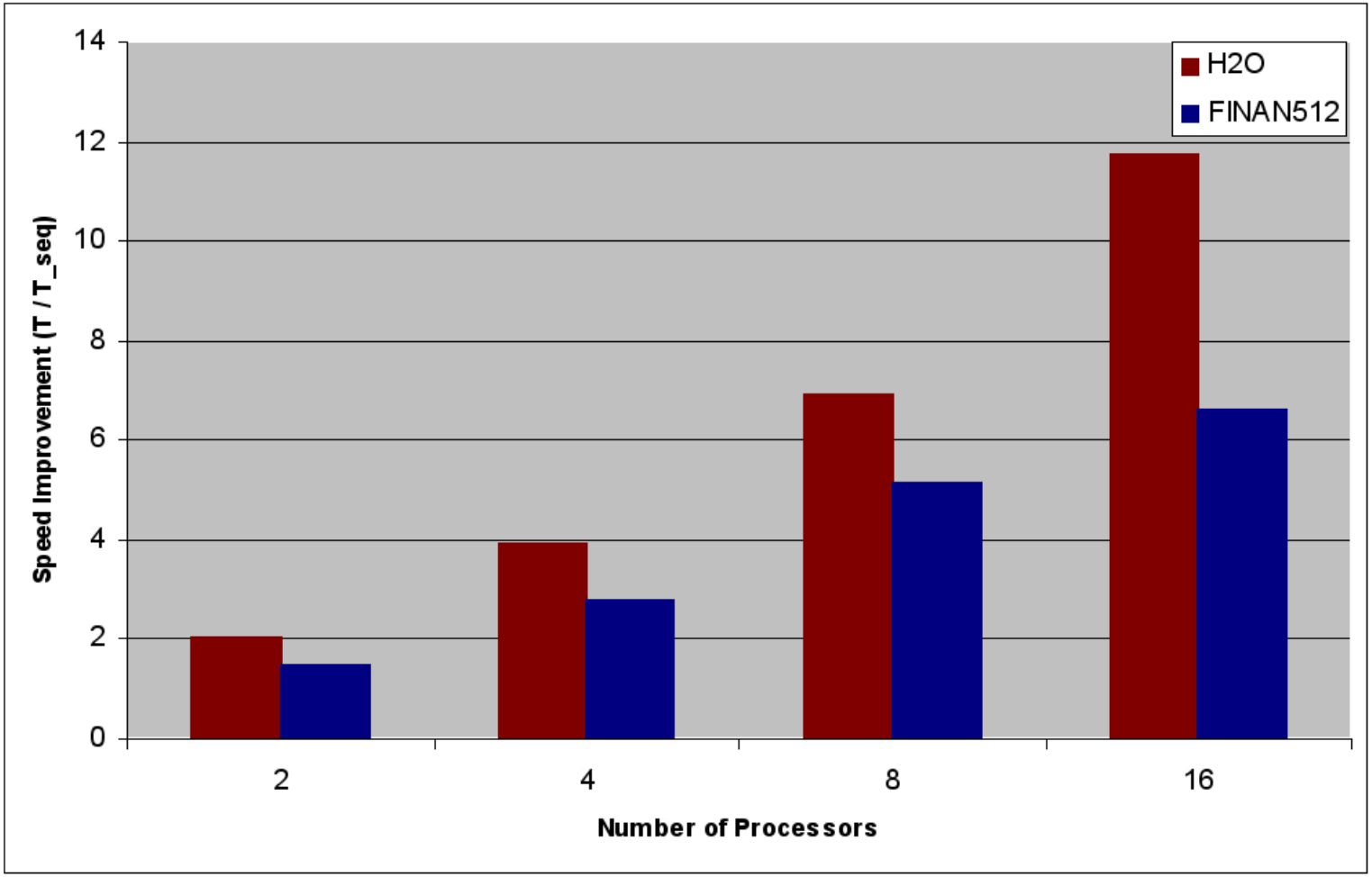
Comparison to ILUT and Improved-ILUT Preconditioners on an uniprocessor

Method\Matrix Number	1	2	3	4	5	6
ILUT(1e-1, n)	Fail	Conv.	Best	Best	Fail	Conv.
ILUTI(1e-1,n)	Conv.	Conv.	Conv.	Conv.	Conv.	Conv.
ILUT(1e-3,n)	Fail	Conv.	Conv.	Conv.	Fail	>600s
ILUTI(1e-3,n)	Conv.	Conv.	Conv.	Conv.	Conv.	>600s
ILUT(0,k)	Fail	>600s	Conv.	>600s	Conv.	>600s
ILUTI(0,k)	Conv.	>600s	Conv.	>600s	Conv.	>600s
WSO	Best	Best	Conv.	Conv.	Best	Best

Outer Iterative Solver : BiCGStab

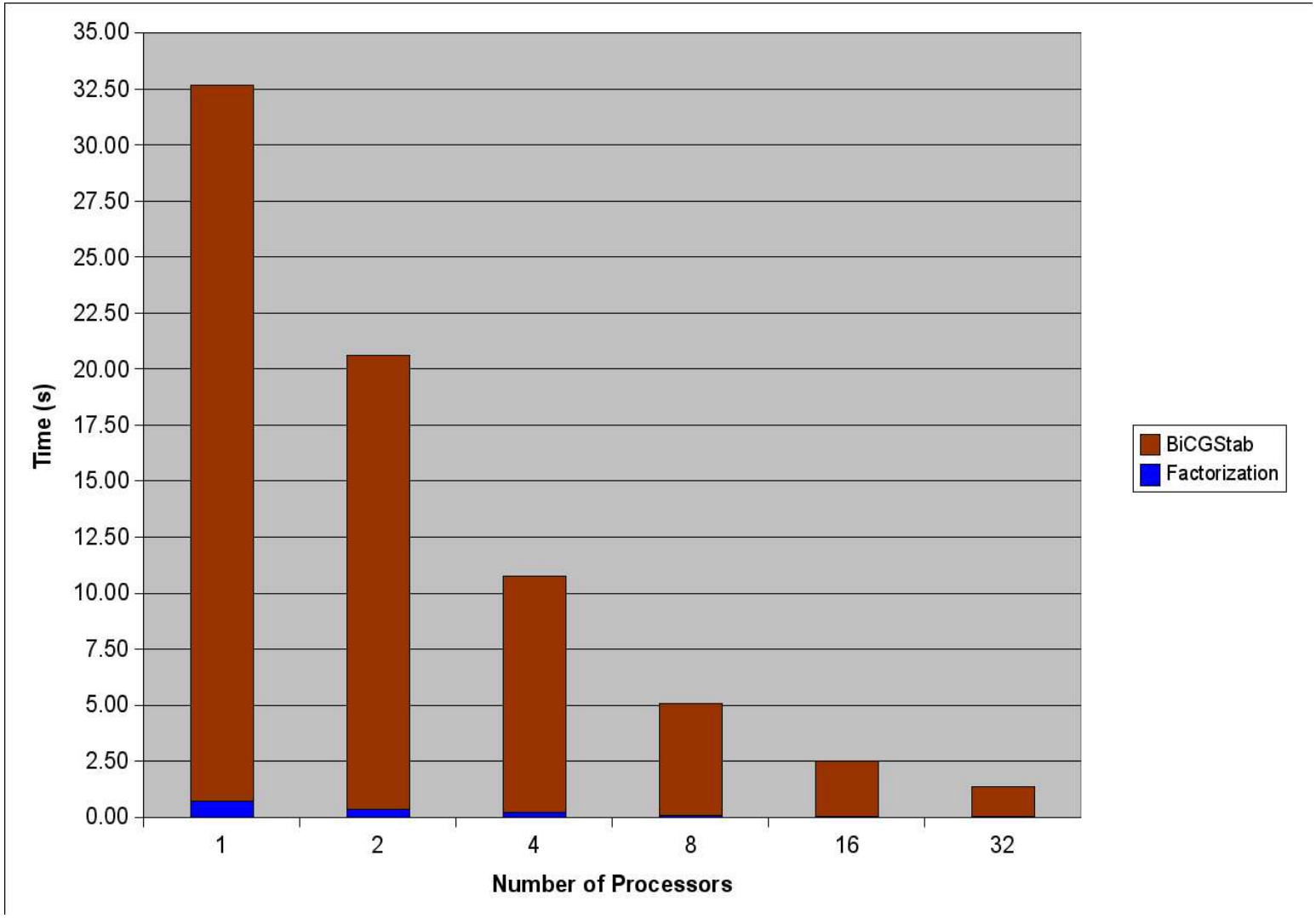
Spike Performance

WSO solver speedup, SGI Altix:



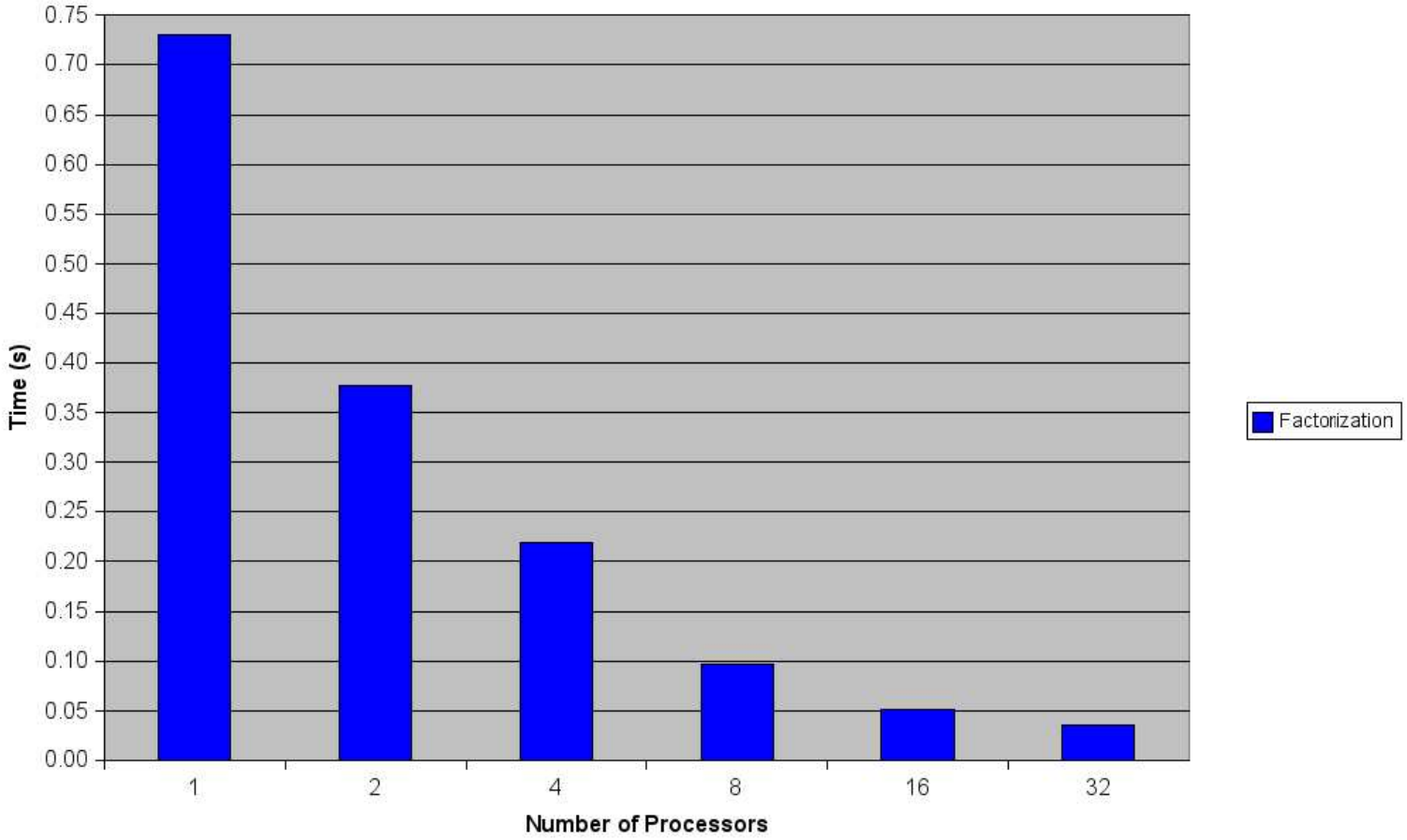
Spike Performance

Reservoir Simulation (SPE10 benchmarks): Problem 1: $N = 2,244,000$ Problem 2 : $N = 2,462,265$.



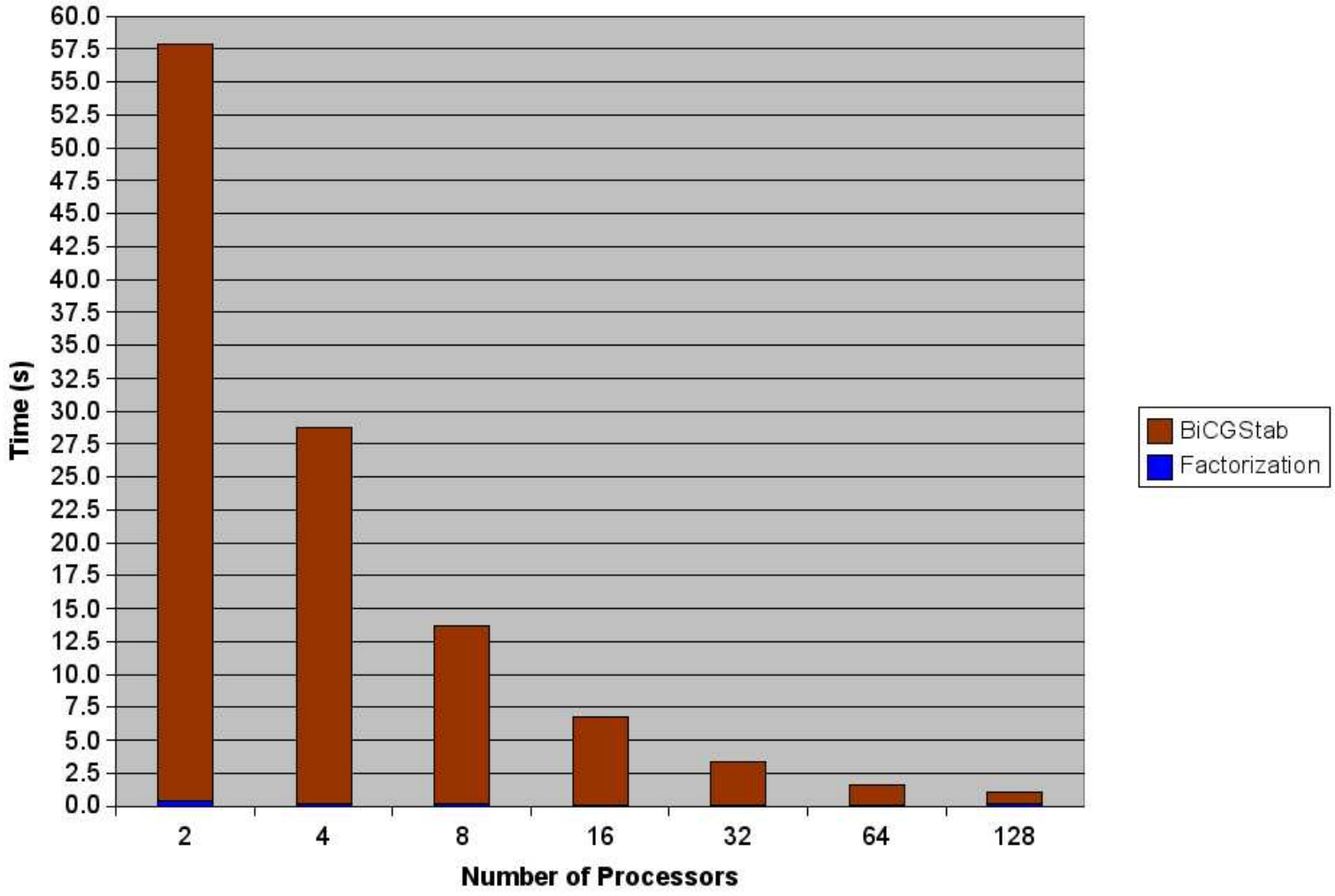
Spike Performance

Spike Factorization Performance:



Spike Performance

Reservoir Simulation Benchmark 2:



Spike: Parallel Steps

1. Compute the LU (via DDBTRF)
   - $L_j U_j \leftarrow A_j$ for $j = 1, 2, 3, 4$
2. Compute spikes (via DTBTRS)
   - Solve for V_j: $L_j U_j V_j = \begin{bmatrix} 0 & \cdots & 0 & B_j^T \end{bmatrix}^T$ for $j = 1, 2, 3$
   - Solve for W_j: $L_j U_j W_j = \begin{bmatrix} C_j^T & 0 & \cdots & 0 \end{bmatrix}^T$ for $j = 2, 3, 4$.
3. Communicate spike tips $W_j^{(t)}$ (processor j sends to processor $j-1$, for $j = 2, 3, 4$). $W_j^{(t)}$ is the top $k \times k$ part of W_j.
4. Factorize the reduced system (via DGETRF)

$$\begin{pmatrix} I & V_j^{(b)} \\ W_j^{(t)} & I \end{pmatrix} \tag{3}$$

5. Modify the right hand side by solving (via DTBTRS) : $L_j U_j g_j = f_j$ ($j = 1, 2, 3, 4$)

6. Communicate the modified right hand side tips $g_j^{(t)}$ (processor j sends to processor $j-1$ for $j = 2, 3, 4$).

7. Solve the reduced system

$$\begin{pmatrix} I & V_j^{(b)} \\ W_{j+1}^{(t)} & I \end{pmatrix} \begin{pmatrix} x_j^{(b)} \\ x_{j+1}^{(t)} \end{pmatrix} = \begin{pmatrix} g_j^{(b)} \\ g_{j+1}^{(t)} \end{pmatrix} \tag{4}$$

8. Communicate the reduced system solution $g_{j+1}^{(t)}$ (processor j sends to processor $j+1$ for $j = 1, 2, 3$).

9. Retrieve x_j ($j = 1, 2, 3, 4$) (via DGEMV) $x_j = f_j - V_j x_{j+1}^{(t)} - W_j x_{j-1}^{(b)}$ ($V_4 = 0$ and $W_1 = 0$)

Analytical Characterization of Spike Steps

Notation:

- $k = kl = ku$ (upper and lower bandwidths identical)
- the dimension of the matrix is N
- the dimension of the partitioned blocks are $n = N/p$.

Stage	Description	Cost
1	Factorize the Diagonal Blocks	$\alpha_1 nk^2 + \beta_1 nk$
2	Compute Spikes	$\alpha_2 nk^2 + \beta_2 nk$
3	Communicate Tips of Spikes	$\alpha_3 k^2(p-1) + \beta_3 k^2 + \gamma_3$
4	Factorize The Reduced System	$\alpha_4 k^3 + \beta_4 k^2$
5	Modify the Right Hand Side	$\alpha_5 nk$
6	Communicate Tips of MRHS	$\alpha_6 k(p-1) + \beta_6 k + \gamma_6$
7	Solve the Reduced System	$\alpha_7 k^2$
8	Comm. Soln. of Reduced System	$\alpha_8 k(p-1) + \beta_8 k + \gamma_8$
9	Retrieve the Solution	$\alpha_9 nk + \beta_9 n$

Analytical Characterization of Spike Steps: Observations

- Stage 1 and Stage 2 cost $O(nk^2)$ computation and $O(nk)$ memory references.
- Stage 4 has $O(k^3)$ computation an $O(k^2)$ memory references.
- Stage 5 has $O(nk)$ computation and $O(nk)$ memory references.
- Stage 7 has $O(k^2)$ computation and $O(k^2)$ memory references.
- Stage 9 has $O(nk)$ computation for matrix vector product and $O(n)$ computation for vector addition.
- In Stages $3, 6, 8$ we model the cost by $O(data)$ for the communication of data, $O(data \times (p-1))$ for network saturation

Training the Analytical Model

- Digonally dominant toeplitz system of dimension $5,000,000$.
- Training platform: Ranger Sun Constellation Linux Cluster at TACC ($3,936$ nodes, each node has 16 cores of AMD Barcelona Processor. Interconnect is Infiniband.
- Train using $16, 32, 64$ processors ($1, 2, 4$ nodes respectively) for bandwidths $k = 15, 25, 35$.
- Using a least squares approximation we find the following parameters $\alpha_i, \beta_i, \gamma_i$ for $i = 1, .., 9$.

Cost Model Parameters

Training yields the following parameters:

i	α_i	β_i	γ_i
1	8.62×10^{-10}	2.61×10^{-8}	-
2	3.96×10^{-8}	3.66×10^{-7}	-
3	9.07×10^{-14}	1.21×10^{-6}	9.48×10^{-4}
4	1.65×10^{-29}	7.43×10^{-6}	-
5	2.85×10^{-8}	-	-
6	4.31×10^{-9}	4.10×10^{-7}	3.59×10^{-5}
7	1.19×10^{-7}	-	-
8	2.57×10^{-7}	3.39×10^{-31}	6.42×10^{-4}
9	8.86×10^{-8}	9.51×10^{-9}	-

Note: Parameters also tell us what regime our algorithm operates in: is the computation memory or processor bound? is communication latency or bandwidth bound, etc.

Cost Model Verification

Validation on two systems – one using the same toeplitz system used in training, the other, a toeplitz system of dimension $10,000,000$. We verify the model for 128, 246, 512, and $1,024$ processors.

N	k	p	Observed	Model	Error
$5,000,000$	35	128	2.78	2.64	0.13
$5,000,000$	25	128	1.55	1.49	0.06
$5,000,000$	15	128	0.70	0.66	0.04
$5,000,000$	35	256	1.49	1.33	0.16
$5,000,000$	25	256	0.79	0.75	0.04
$5,000,000$	15	256	0.35	0.33	0.02
$5,000,000$	35	512	0.67	0.67	0.00
$5,000,000$	25	512	0.38	0.38	0.00
$5,000,000$	15	512	0.20	0.17	0.03
$5,000,000$	35	$1,024$	0.37	0.35	0.02
$5,000,000$	25	$1,024$	0.21	0.20	0.01
$5,000,000$	15	$1,024$	0.10	0.09	0.01

Cost Model Verification (contd)

N	k	p	Observed	Model	Error
$10,000,000$	35	128	5.36	5.27	0.09
$10,000,000$	25	128	3.03	2.98	0.06
$10,000,000$	15	128	1.36	1.31	0.05
$10,000,000$	35	256	2.78	2.64	0.13
$10,000,000$	25	256	1.55	1.49	0.06
$10,000,000$	15	256	0.68	0.66	0.02
$10,000,000$	35	512	1.51	1.33	0.18
$10,000,000$	25	512	0.79	0.75	0.04
$10,000,000$	15	512	0.35	0.33	0.02
$10,000,000$	35	$1,024$	0.69	0.68	0.01
$10,000,000$	25	$1,024$	0.45	0.38	0.07
$10,000,000$	15	$1,024$	0.19	0.17	0.02

Cost Model Verification

- Model yields outstanding accuracy in predicting performance well beyond training machine size.

- Model predicts linear scaling (linearly increasing speedup when keeping problem size per processor constant)!

Spike is an excellent candidate for emerging ultrascale platforms.

Limitations of Approach

- Serial performance is harder to characterize than parallel performance!
  - Operate in stable region of serial performance
- Analytical models may vary across platforms
  - Separate models for message passing and shared address space machines
- Granularity of aggregates
  - It may not always be possible to clearly identify and analytically characterize various steps in a parallel algorithm.

Thank You!