

# Reducing Communication Overhead in Large Eddy Simulation of Jet Engine Noise

Y. Situ<sup>1</sup> L. Liu<sup>1</sup> C. S. Martha<sup>2</sup> M. E. Louis<sup>2</sup> Z. Li<sup>1</sup>  
A. H. Sameh<sup>1</sup> G. A. Blaisdell<sup>2</sup> A. S. Lyrintzis<sup>2</sup>

<sup>1</sup>Department of Computer Science  
Purdue University

<sup>2</sup>School of Aeronautics and Astronautics  
Purdue University

IEEE Cluster 2010

# Outline

- 1 Introduction and existing implementation
  - Project background
  - Fundamental methodology
  - Existing implementation
- 2 New tridiagonal linear system solver
  - Design goals
  - SPIKE-based tridiagonal linear system solver
- 3 Experimental results
- 4 Summary

# Outline

- 1 Introduction and existing implementation
  - Project background
  - Fundamental methodology
  - Existing implementation
- 2 New tridiagonal linear system solver
  - Design goals
  - SPIKE-based tridiagonal linear system solver
- 3 Experimental results
- 4 Summary

# Project Background

- Realistic simulation of jet engine noise using petaflop computing
  - Primarily funded by U.S. National Science Foundation's PetaApps program
  - Aims to advance jet noise prediction of modern turbofan aircraft engines
- Collaboration between Computer Science (CS) and Aeronautics & Astronautics Engineering (AAE) at Purdue
  - CS: parallel algorithms, programming models
  - AAE: physical models, numerical methods

# Project Background

- Realistic simulation of jet engine noise using petaflop computing
  - Primarily funded by U.S. National Science Foundation's PetaApps program
  - Aims to advance jet noise prediction of modern turbofan aircraft engines
- Collaboration between Computer Science (CS) and Aeronautics & Astronautics Engineering (AAE) at Purdue
  - CS: **parallel algorithms**, programming models
  - AAE: physical models, numerical methods

# Fundamental Methodology

- Large eddy simulation (LES)
  - Direct resolution of large-scale eddies
    - More accurate than Reynolds-averaged Navier–Stokes equations (RANS)
  - Turbulence model for small-scale eddies
    - Less computationally intensive than direct numerical simulation (DNS)

# Fundamental Methodology

- Governing equation: Navier–Stokes equations

$$\frac{\partial \mathbf{Q}}{\partial t} = -J \left( \frac{\partial}{\partial \xi} \left( \frac{\mathbf{F} - \mathbf{F}_v}{J} \right) + \frac{\partial}{\partial \eta} \left( \frac{\mathbf{G} - \mathbf{G}_v}{J} \right) + \frac{\partial}{\partial \zeta} \left( \frac{\mathbf{H} - \mathbf{H}_v}{J} \right) \right)$$

- 3D computational space

$$\mathbf{Q} = \mathbf{Q}(\xi, \eta, \zeta)$$

- Multiple flow variables

$$\mathbf{Q} = (\rho, \rho u_x, \rho u_y, \rho u_z, e)^T$$

- Time-advanced using classical Runge-Kutta method

# Basic Operations

- Pointwise computation
  - Example: local mean flow properties

$$\bar{\mathbf{Q}}^{(t)}(\xi, \eta, \zeta) = \frac{t}{t+1} \bar{\mathbf{Q}}^{(t-1)}(\xi, \eta, \zeta) + \frac{1}{t+1} \mathbf{Q}^{(t)}(\xi, \eta, \zeta) \quad \forall \xi, \eta, \zeta$$

- Embarrassingly parallel
- Requires no communication

# Basic Operations

- Diagonally-dominant tridiagonal linear systems
  - Spatial differentiation:  $\frac{\partial}{\partial \xi}$ ,  $\frac{\partial}{\partial \eta}$ ,  $\frac{\partial}{\partial \zeta}$  operators

$$\frac{1}{3}f'_{i-1} + f'_i + \frac{1}{3}f'_{i+1} = \frac{7}{9\Delta\xi}(f_{i+1} - f_{i-1}) + \frac{1}{36\Delta\xi}(f_{i+2} - f_{i-2})$$

- Sixth-order accurate, required for long time advancement
- Spatial filtering: all three axial directions

$$\alpha_f \bar{f}_{i-1} + \bar{f}_i + \alpha_f \bar{f}_{i+1} = \sum_{n=0}^3 \frac{a_n}{2} (f_{i+n} + f_{i-n}) \quad (|\alpha_f| < 0.5)$$

- Spatial low-pass filter for suppressing numerical instability

# Basic Operations

- Diagonally-dominant tridiagonal linear systems
  - Spatial differentiation:  $\frac{\partial}{\partial \xi}$ ,  $\frac{\partial}{\partial \eta}$ ,  $\frac{\partial}{\partial \zeta}$  operators

$$\frac{1}{3}f'_{i-1} + f'_i + \frac{1}{3}f'_{i+1} = \frac{7}{9\Delta\xi}(f_{i+1} - f_{i-1}) + \frac{1}{36\Delta\xi}(f_{i+2} - f_{i-2})$$

- Sixth-order accurate, required for long time advancement
- Spatial filtering: all three axial directions

$$\alpha_f \bar{f}_{i-1} + \bar{f}_i + \alpha_f \bar{f}_{i+1} = \sum_{n=0}^3 \frac{a_n}{2} (f_{i+n} + f_{i-n}) \quad (|\alpha_f| < 0.5)$$

- Spatial low-pass filter for suppressing numerical instability

# Existing Implementation

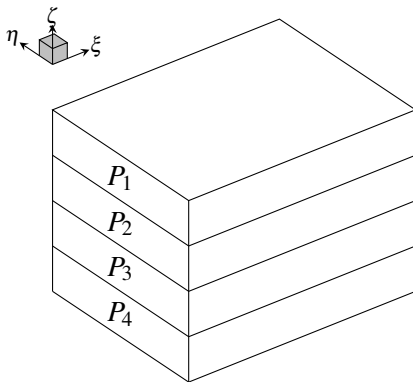


Figure: Computational space partitioning

# Existing Implementation

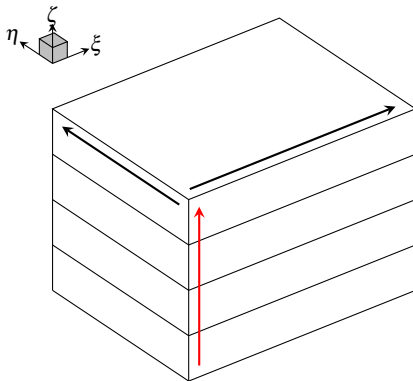


Figure: Tridiagonal linear systems across three directions

# Existing Implementation

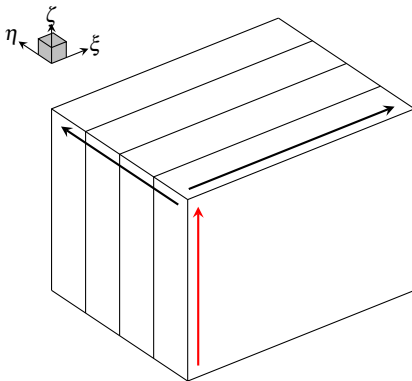


Figure: Transposition of computational space

# Existing Implementation

- Pros
  - Simplifies implementation
- Cons
  - Large amount of communication
    - $O(N^3/p)$  per node per time step
    - Long transmission delay
  - Complex communication pattern
    - High network congestion

# Outline

- 1 Introduction and existing implementation
  - Project background
  - Fundamental methodology
  - Existing implementation
- 2 **New tridiagonal linear system solver**
  - Design goals
  - **SPIKE-based tridiagonal linear system solver**
- 3 Experimental results
- 4 Summary

# Design Goals

- Significantly less communication
  - No more than  $O(N^2)$  per node per time step
- Simple communication pattern
  - Natural for linear array of nodes
  - Communication occurs only between neighboring nodes
- Easy to implement

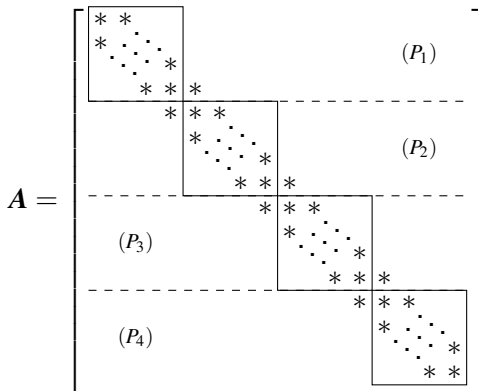
# SPIKE-based Tridiagonal Linear System Solver

- To solve a tridiagonal linear system  $A\mathbf{x} = \mathbf{f}$  where

$$A = \begin{bmatrix} \begin{array}{cccc} * & * & & \\ * & \cdot & \cdot & \\ & \cdot & \cdot & * \\ & & * & * \end{array} & & & \\ & * & * & * & & & & \\ & * & \cdot & \cdot & \cdot & & & \\ & & * & * & * & & & \\ & & & * & * & * & & \\ & & & & * & * & * & \\ & & & & & * & * & * \\ & & & & & & * & * & * \end{array} \begin{matrix} (P_1) \\ (P_2) \\ (P_3) \\ (P_4) \end{matrix}$$

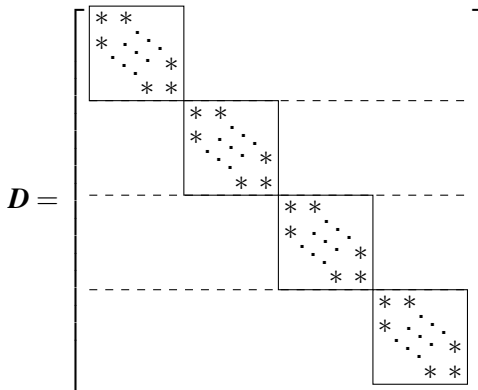
# SPIKE-based Tridiagonal Linear System Solver

- To solve a tridiagonal linear system  $A\mathbf{x} = \mathbf{f}$  where



# SPIKE-based Tridiagonal Linear System Solver

- Factorize  $A$  into  $A = DS$  where











# SPIKE-based Tridiagonal Linear System Solver

- Write  $Sx = g$  in full. Observe that **highlighted** elements capture coupling between neighboring partitions:

$$Sx = \begin{bmatrix} 1 & & & & \vdots \\ & \ddots & & & \\ & & 1 & v_1^{(b)} & \\ \text{-----} & & w_2^{(t)} & 1 & v_2^{(t)} \\ & & \vdots & \ddots & \vdots \\ & & w_2^{(b)} & & 1 & v_2^{(b)} \\ \text{-----} & & & & w_3^{(t)} & 1 & v_3^{(t)} \\ & & & & \vdots & \ddots & \vdots \\ & & & & w_3^{(b)} & & 1 & v_3^{(b)} \\ \text{-----} & & & & & & w_4^{(t)} & 1 & \ddots \\ & & & & & & \vdots & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} x_1^{(t)} \\ \vdots \\ x_1^{(b)} \\ x_2^{(t)} \\ \vdots \\ x_2^{(b)} \\ x_3^{(t)} \\ \vdots \\ x_3^{(b)} \\ x_4^{(t)} \\ \vdots \\ x_4^{(b)} \end{bmatrix} = \begin{bmatrix} g_1^{(t)} \\ \vdots \\ g_1^{(b)} \\ g_2^{(t)} \\ \vdots \\ g_2^{(b)} \\ g_3^{(t)} \\ \vdots \\ g_3^{(b)} \\ g_4^{(t)} \\ \vdots \\ g_4^{(b)} \end{bmatrix} = g$$





# SPIKE-based Tridiagonal Linear System Solver

- Solve  $\hat{S}\hat{x} = \hat{g}$  using block Jacobi iteration:

$$\hat{x} = \begin{bmatrix} 1 & v_1^{(b)} & & & & \\ w_2^{(t)} & 1 & & & & \\ & & 1 & v_2^{(b)} & & \\ & & w_3^{(t)} & 1 & & \\ & & & & 1 & v_3^{(b)} \\ & & & & w_4^{(t)} & 1 \end{bmatrix}^{-1} \left( \hat{g} - \begin{bmatrix} 0 \\ v_2^{(t)} x_3^{(t)} \\ w_2^{(b)} x_1^{(b)} \\ v_3^{(t)} x_4^{(t)} \\ w_3^{(b)} x_2^{(b)} \\ 0 \end{bmatrix} \right)$$

- Fast convergence due to exponentially small  $v_k^{(t)}$  and  $w_k^{(b)}$ 
  - $A$  is diagonally-dominant
- Constant amount of communication per right-hand side between neighboring nodes

# SPIKE-based Tridiagonal Linear System Solver

- Solve  $\hat{S}\hat{x} = \hat{g}$  using block Jacobi iteration:

$$\hat{x} = \begin{bmatrix} 1 & v_1^{(b)} & & & & \\ w_2^{(t)} & 1 & & & & \\ & & 1 & v_2^{(b)} & & \\ & & w_3^{(t)} & 1 & & \\ & & & & 1 & v_3^{(b)} \\ & & & & w_4^{(t)} & 1 \end{bmatrix}^{-1} \left( \hat{g} - \begin{bmatrix} 0 \\ v_2^{(t)} x_3^{(t)} \\ w_2^{(b)} x_1^{(b)} \\ v_3^{(t)} x_4^{(t)} \\ w_3^{(b)} x_2^{(b)} \\ 0 \end{bmatrix} \right)$$

- Fast convergence due to exponentially small  $v_k^{(t)}$  and  $w_k^{(b)}$ 
  - $A$  is diagonally-dominant
- Constant amount of communication per right-hand side between neighboring nodes

# SPIKE-based Tridiagonal Linear System Solver

- Solve  $\hat{S}\hat{x} = \hat{g}$  using block Jacobi iteration:

$$\hat{x} = \begin{bmatrix} 1 & v_1^{(b)} & & & & \\ w_2^{(t)} & 1 & & & & \\ & & 1 & v_2^{(b)} & & \\ & & w_3^{(t)} & 1 & & \\ & & & & 1 & v_3^{(b)} \\ & & & & w_4^{(t)} & 1 \end{bmatrix}^{-1} \left( \hat{g} - \begin{bmatrix} 0 \\ v_2^{(t)} x_3^{(t)} \\ w_2^{(b)} x_1^{(b)} \\ v_3^{(t)} x_4^{(t)} \\ w_3^{(b)} x_2^{(b)} \\ 0 \end{bmatrix} \right)$$

- Fast convergence due to exponentially small  $v_k^{(t)}$  and  $w_k^{(b)}$ 
  - $A$  is diagonally-dominant
- Constant amount of communication per right-hand side between neighboring nodes

# SPIKE-based Tridiagonal Linear System Solver

- Advantages over transposition
  - Low network traffic
    - $O(N^2)$  per node per time step
  - Restricts communication to between neighboring nodes
    - No global data reslicing and redistribution
- Implementation considerations
  - Precompute number of iterations needed for desired accuracy
    - Avoid tracking residual at runtime
  - Overlap communication and computation
    - Use asynchronous communication
    - Start computation even when only partial data are available
    - Issue wait commands as late as possible

# SPIKE-based Tridiagonal Linear System Solver

- Advantages over transposition
  - Low network traffic
    - $O(N^2)$  per node per time step
  - Restricts communication to between neighboring nodes
    - No global data reslicing and redistribution
- Implementation considerations
  - Precompute number of iterations needed for desired accuracy
    - Avoid tracking residual at runtime
  - Overlap communication and computation
    - Use asynchronous communication
    - Start computation even when only partial data are available
    - Issue wait commands as late as possible

# SPIKE-based Tridiagonal Linear System Solver

- Advantages over transposition
  - Low network traffic
    - $O(N^2)$  per node per time step
  - Restricts communication to between neighboring nodes
    - No global data reslicing and redistribution
- Implementation considerations
  - Precompute number of iterations needed for desired accuracy
    - Avoid tracking residual at runtime
  - Overlap communication and computation
    - Use asynchronous communication
    - Start computation even when only partial data are available
    - Issue wait commands as late as possible

# Outline

- 1 Introduction and existing implementation
  - Project background
  - Fundamental methodology
  - Existing implementation
- 2 New tridiagonal linear system solver
  - Design goals
  - SPIKE-based tridiagonal linear system solver
- 3 **Experimental results**
- 4 Summary

## Experiment Setup

Table: Experiment setup

Grid size	$768 \times 768 \times 768 \approx 4.5 \times 10^8$
Time steps	10
Computing platforms	Ranger, Kraken
Number of nodes	$24^1, 32, 48, 64, 96$

<sup>1</sup>Data not available on Kraken due to memory limit

# Experimental Results

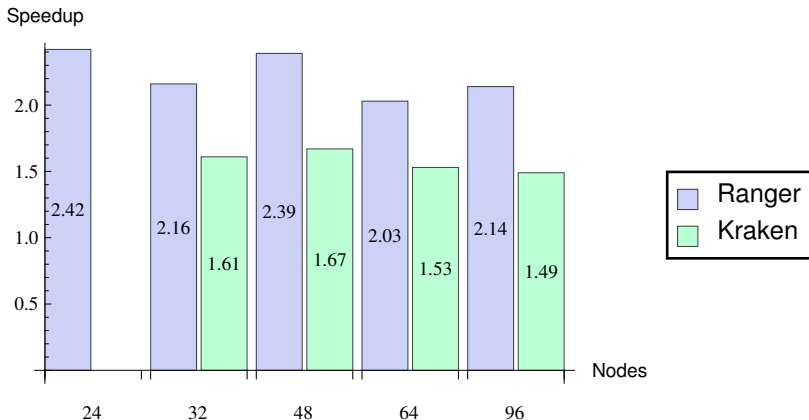


Figure: Speedup achieved on Ranger and Kraken

# Experimental Results

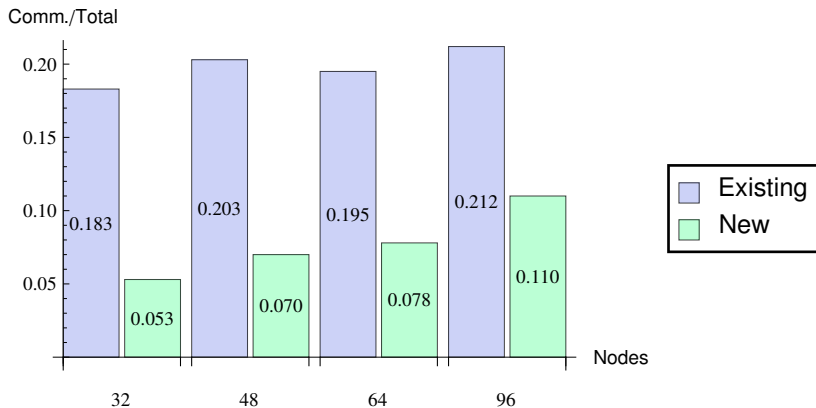


Figure: Communication-to-total ratios on Kraken

# Experimental Results

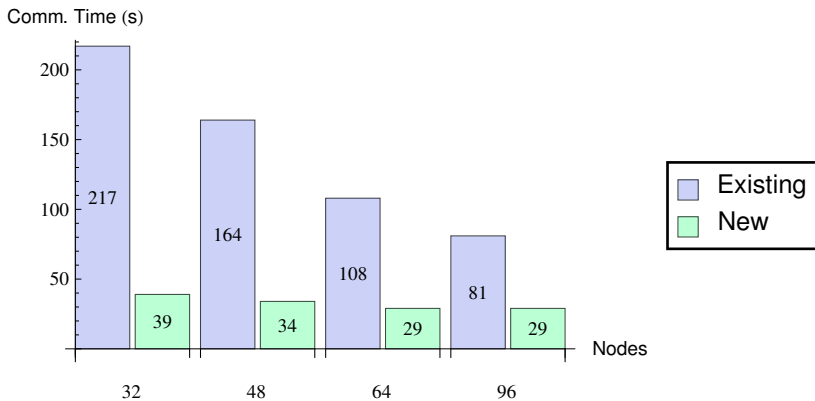


Figure: Communication times on Kraken

# Outline

- 1 Introduction and existing implementation
  - Project background
  - Fundamental methodology
  - Existing implementation
- 2 New tridiagonal linear system solver
  - Design goals
  - SPIKE-based tridiagonal linear system solver
- 3 Experimental results
- 4 Summary

# Summary

- SPIKE-based tridiagonal linear system solver
  - Requires only local communication
  - Efficient in empirical experiments
  - Easily extensible to 3D partitioning
  
- Future work
  - Exploit higher degree of parallelism via 3D partitioning
    - Prototype completed and in testing
  - Explore more complex topologies

# For Further Reading



A. Uzun.

3-D Large Eddy Simulation for Jet Aeroacoustics.

PhD dissertation, School of Aeronautics and Astronautics,  
Purdue University, 2003.



E. Polizzi, A. H. Sameh.

A Parallel Hybrid Banded System Solver: the SPIKE  
Algorithm

*Parallel Computing*, 32(2):177–192, 2006.