Improving the Scalability of our in-house Large Eddy Simulation (LES) code

Sponsored by National Science Foundation

By Chandra S. Martha, Yingchong Situ, Matt Louis, Gregory A. Blaisdell , Anastasios S. Lyrintzis and, Zhiyuan Li

Objectives

- **Identify the bottlenecks and hot-spots involved in our in-house LES code**
- **Evaluate its scalability and peak performance for realistic jet simulations**
- **Improve the single-core performance by efficient utilization of the memory hierarchy**
- **Explore and assess alternative parallelization strategies to better make use of current super computers**

Introduction

- Jet engine noise has been one of the most active areas of research.
- Computationally intensive LES tool allows accurate prediction of sound levels.
- LES of realistic simulations is within the reach of the state-of-the-art super computer architectures.
- However, scalability of the code to massive number of cores is critical to simulate realistic flows involving billions of grid points.

Testing Platform

- *Kraken* super computer at NICS, University of Tennessee.

Platform	Kraken
Cluster model	Cray XT5
Processor model	AMD "Istanbul"
Architecture	X86-64
Nodes (cores)	8,256 (99,072)
Socket per node	2
Cores per socket	6
Clock frequency	2.6 GHz
Memory	16 GB

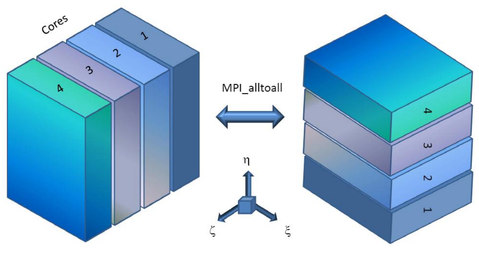
- Cray-PAT is used for profiling.

Governing Equations – Navier-Stokes

$$\frac{1}{J}\frac{\partial Q}{\partial t}+\frac{\partial}{\partial \xi}\left(\frac{F-F_v}{J}\right)+\frac{\partial}{\partial \eta}\left(\frac{G-G_v}{J}\right)+\frac{\partial}{\partial \zeta}\left(\frac{H-H_v}{J}\right)=0$$

- Q – vector of conservative flow variables
- F, G and H – Inviscid flux; F_v , G_v and H_v – Viscous flux vectors
- 4^{th} order, 4-stage Runge-Kutta method for time-advancement
- The flow is filtered in the three-directions to damp unresolved frequencies
- Compact differencing schemes to compute the derivative and filter resulting in tri-diagonal system of equations

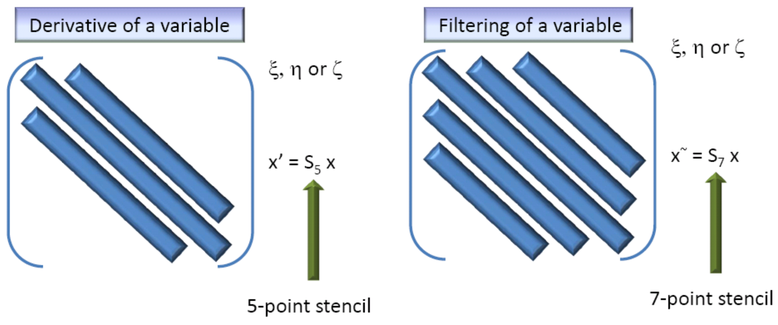
Transposition Scheme:
Partitioning of the Computational Domain.



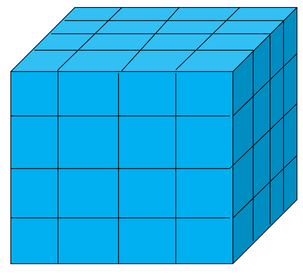
LAPACK is for the derivative and filtering operations.
Pros:
- Easy to implement

Cons:
- Parallelism is limited to one plane per processor
- *ALL-to-ALL* communication
- MPI message bytes are too high
- Poor scalability at large core counts

Basic operations involved



3-D SPIKE algorithm

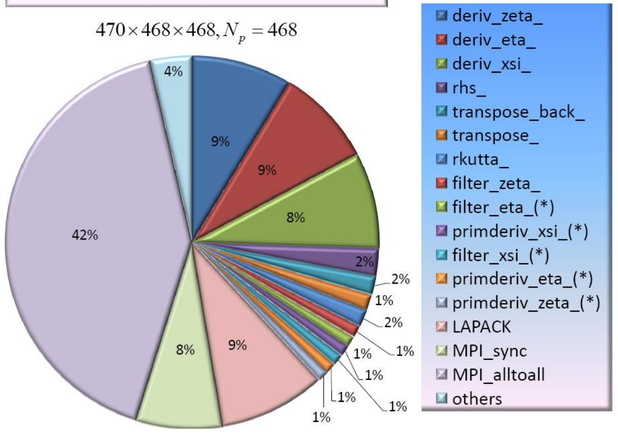


User developed modules to solve the system of equations
Pros:
- 3-D decomposition: Better parallelism
- Each processor only needs to communicate with its neighbors
- Gives rise to better efficiency for large processor counts

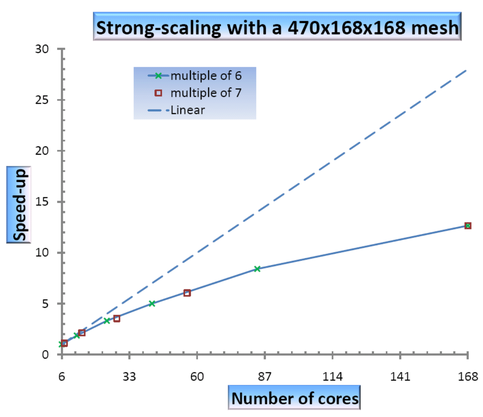
Cons:
- Requires iterative refinement (2-7 iterations)

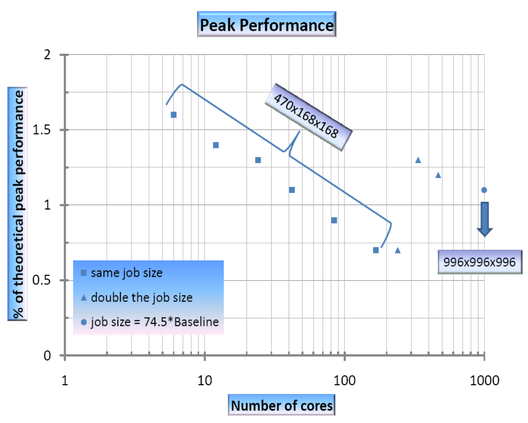
Function profiling - % of wall clock time

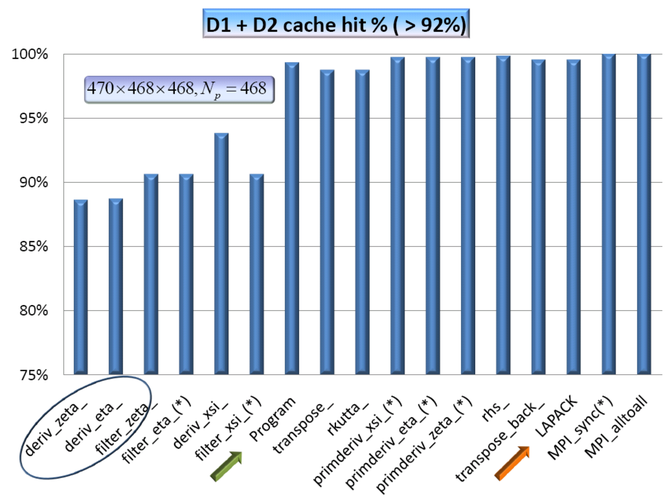
USER – 40%, LAPACK – 9%, MPI – 51%



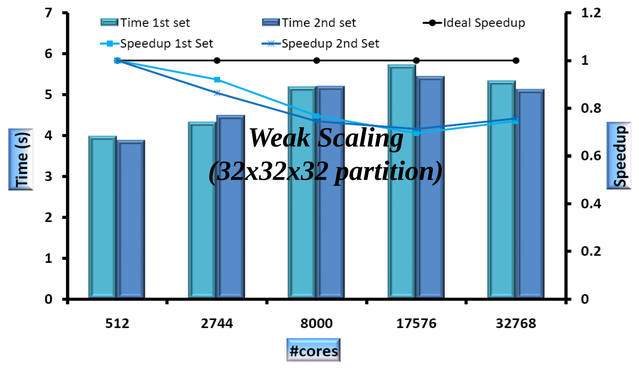
Performance with 1-D Transposition

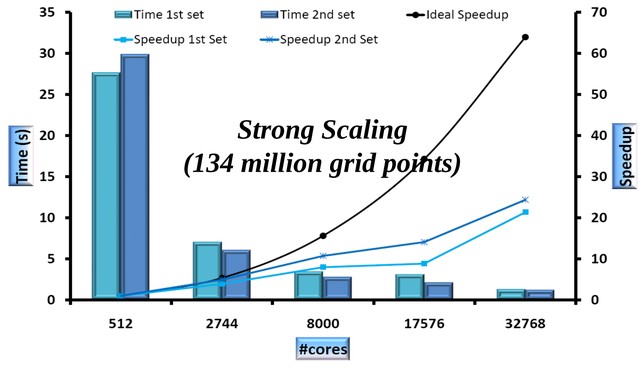


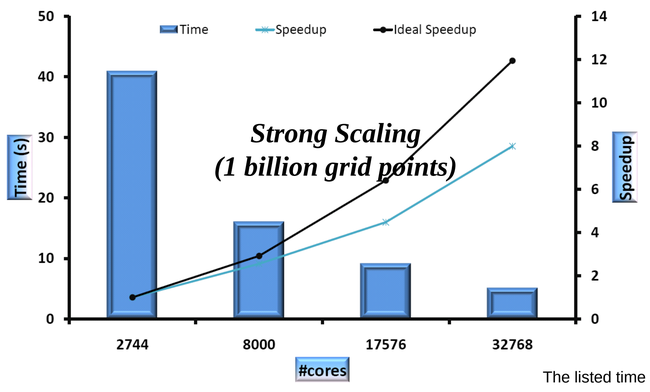


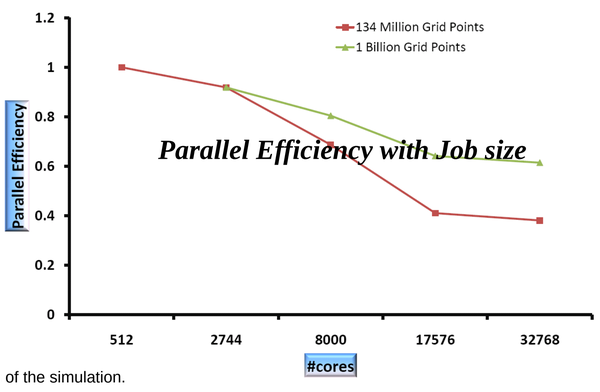


Performance with 3-D SPIKE









- The indices of the flow-field data array are *switched* to improve the cache access.
- SPIKE solver is used to eliminate to reduce the amount communicated data.

768x768x768	Cores	96	Time-steps		5
	Original	Index switch	Index switch+ spike		
			1-plane	1-variable	all variables
USER	287	224	163	168	168
MPI	53	58	52	28	29
LAPACK	100	100	66	67	67
TOTAL	**440**	**382**	**281**	**263**	**264**
Peak Perf	**1.80%**	**2.10%**	**3.30%**	**3.40%**	**3.40%**
MPI Msgs	105	105	1.7 M	2287	461
MPI Bytes	20 GB	20 GB	13 GB	13 GB	13 GB

Truncated SPIKE with 2 iterations for the derivative and 7 for the filtering.

The listed time corresponds to **10 time-steps** of the simulation.

Resource Provider TeraGrid™

School of Aeronautics and Astronautics