Oral presentation | Multi-phase flow

High performance computing-I

Tue. Jul 16, 2024 4:30 PM - 6:00 PM Room D

[6-D-01] Xcompact3D Revisited: A Codebase for Heterogeneous Architectures Based on a Distributed-memory Tridiagonal Matrix Algorithm

*Semih Akkurt[1], Sebastien Lemaire[2], Paul Bartholomew[2], Stefano Rolfo[3], Cedric Flageul[4], Filippo Spiga[5], Charles Moulinec[3], Sylvain Laizet[1] （1. Imperial College London, 2. EPCC, University of Edinburgh, 3. STFC Daresbury Laboratory, UKRI, 4. Universite de Poitiers, 5. NVIDIA Corporation）

Keywords: High performance computing, Compact finite difference schemes, GPU

Xcompact3D Revisited:
A Codebase for Heterogeneous Architectures Based on a Distributed-memory Tridiagonal Matrix Algorithm

Semih Akkurt [1] Sebastien Lemaire [2] Paul Bartholomew [2]
Stefano Rolfo [3] Cedric Flageul [4] Filippo Spiga [5] Sylvain Laizet [1]

[1]Imperial College London, London, UK
[2]The University of Edinburgh, EPCC, Edinburgh, UK
[3]STFC Daresbury Laboratory, UKRI, Warrington, UK
[4]Universite de Poitiers, Poitiers, France
[5]NVIDIA Corporation, Cambridge, UK

ICCFD12
14-19 July 2024, Kobe, Japan

Overview

- Xcompact3D is a high-order Navier-Stokes solver on structured grids.
- Time discretisation is based on fractional time stepping
- Spatial discretisation uses high-order compact finite difference schemes.
  - Compact schemes introduce a space-implicit coupling
  - Necessitates solving a batch of tridiagonal systems
- Poisson equation is solved in spectral space
  - Using 3D FFT transforms

High-order Compact Finite Difference Schemes

- Example from Xcompact3D based on 6th order compact FD schemes.
- We solve ~ 50 batches of such systems per time step.

$$\alpha f'_{i-1} + f'_i + \alpha f'_{i+1} = a\frac{f_{i+1} - f_{i-1}}{2\Delta x} + b\frac{f_{i+2} - f_{i-2}}{4\Delta x}$$

$$\begin{bmatrix} b_0 & c_0 & & & & \\ a_1 & b_1 & c_1 & & & \\ & a_2 & b_2 & c_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

S. K. Lele, Compact finite difference schemes with spectral-like resolution, Journal of computational physics 103 (1) (1992) 16–42.

Overview of the Xcompact3D Algorithm

Transport equation

$$\text{RHS}_x^u \leftarrow -\frac{1}{2}\left(u\frac{\partial u}{\partial x} + \frac{\partial uu}{\partial x}\right) + \nu\frac{\partial^2 u}{\partial x^2}$$

$$\text{RHS}^{(3)} \leftarrow \text{RHS}_x^{(3)} + \text{RHS}_y^{(3)} + \text{RHS}_z^{(3)}$$

Time integration

$$U^{*(3)} \leftarrow \Delta t \cdot \text{RHS}^{(3)} + U^{(3)}$$

Divergence of U^* at Pressure grid

Poisson Solver to obtain Pressure P

Gradient of Pressure

Pressure correction

Parallel Strategies for Solving Tridiagonal Systems

- Xcompact3D spends $\sim 70\%$ of runtime on solving tridiagonal systems.
- For each time-step there are around 50 batches of tridiagonal systems where each batch may contain around $\sim 10^{6/7}$ individual systems of size $\sim 10^{3/4}$ with a total DOF up to around $\sim 10^{12}$.
- Such a problem require ~ 1000 nodes
- 2 options for parallelisation
  - Serial TDS solvers with slab/pencil decomposition
  - Distributed TDS solvers

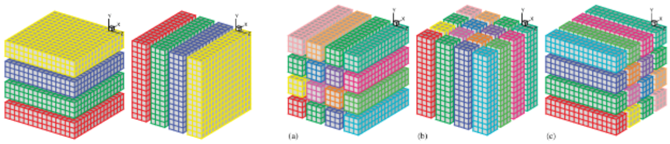


Figure: 1D/2D (Slab/Pencil) Domain Decomposition

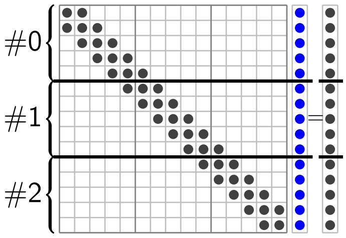
- Current version of Xcompact3D uses Thomas algorithm.
- Thus, Xcompact3D require 12 pairs of data transfers per iteration.
- These large scale data transfers can be costly.
- The best case behaviour can be modelled theoretically.

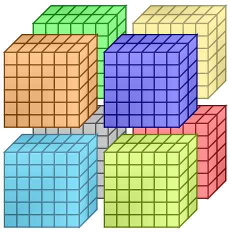
512^3 Domain - Theoretical Scalability Predictions

# Nodes	# GPUs	Comp. BW	Comm. BW	Comp. Time	Comm. Time	Total Time
1	1	332 GiB	0 GiB	0.300 s	0 s	0.300 s
	2	166 GiB	3 GiB	0.150 s	0.015 s	0.165 s
	4	83 GiB	2.25 GiB	0.075 s	0.004 s	0.079 s
2	8	42 GiB	1.31 GiB	0.038 s	0.060 s	0.098 s

Parallelisation Strategy - Distributed Tridiagonal Solvers

- Examples algorithms are PDD (Sun 1995) and SPIKE (Polizzi and Sameh 2007)
- Our strategy combines Hybrid Thomas-PCR (Laszlo 2016) and PDD (Sun 1995) algorithms
- 1/2/3 D domain decomposition, no all-to-all communication between sub-domains.
- A distributed solver is implemented along the direction that is split between ranks





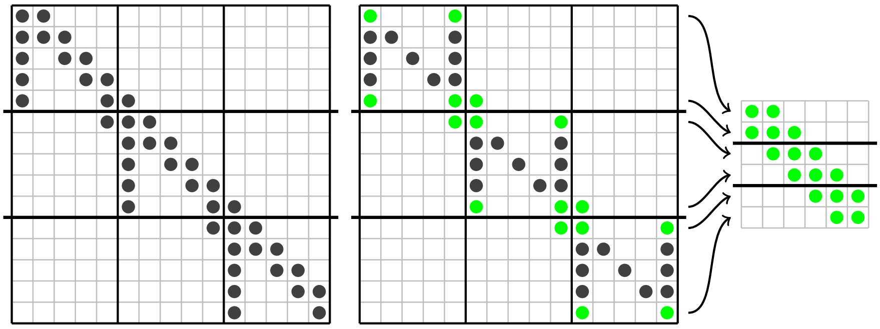


Figure: Modified Thomas algorithm [1]. Each subdomain carries out a local forward and backward phases resulting in a reduced system.

[1] László, Giles, and Appleyard. 2016. Manycore Algorithms for Batch Scalar and Block Tridiagonal Solvers. ACM Trans. Math. Softw.

Parallel Diagonally Dominant (PDD) Algorithm

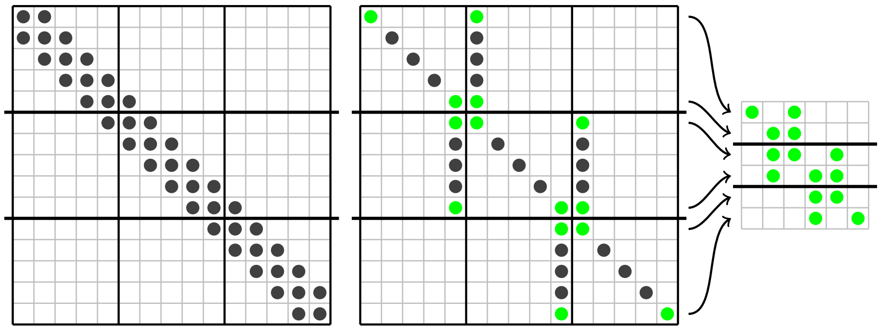


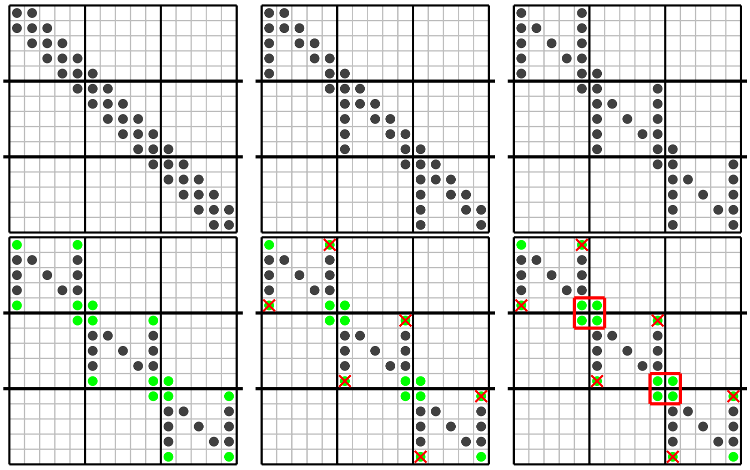
Figure: Parallel diagonally dominant algorithm [2]. Each region is multiplied by its local inverse which results in a reduced penta-diagonal system with first and last data points from each rank.

[2]. X.-H. Sun. 1995. Application and accuracy of the parallel diagonal dominant algorithm, Parallel computing 21 (8) 1241–1267

Proposed Strategy

- Developed a customised strategy
- Specifically for high-order compact finite difference schemes
- Diagonally dominant tridiagonal systems reduce the communication requirement.
- A specialist data structure improves the performance
  - Applicable to all directions

Proposed Strategy



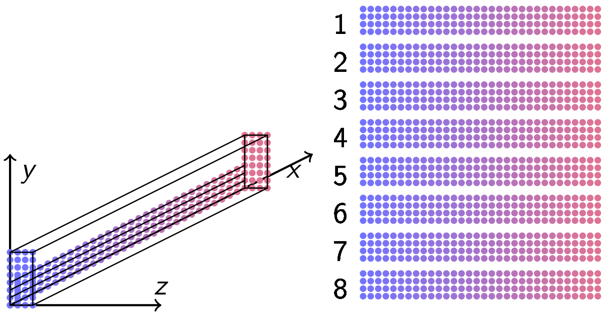


Figure: Pencil grouping data structure for vectorisation and thread parallelism support.

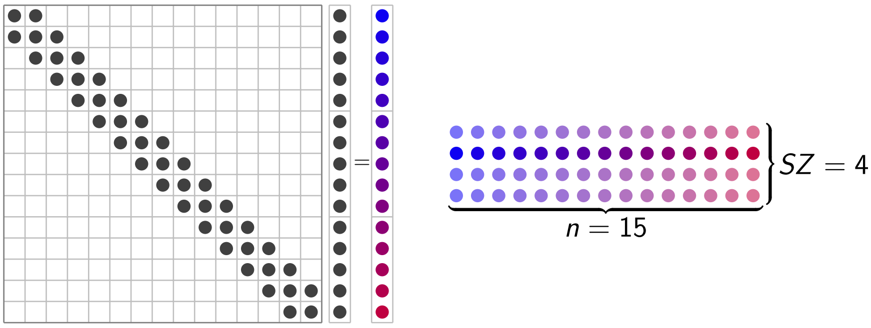


Figure: A single pencil group with $SZ = 4$ and $n = 15$. Colour coding demonstrates the exact distribution of a single RHS in memory. The data layout is in column major order, thus each subsequent data point in a single tridiagonal system is separated from each other by SZ many data points in memory.

Pencil Grouping Approach - Specialist Data Structure

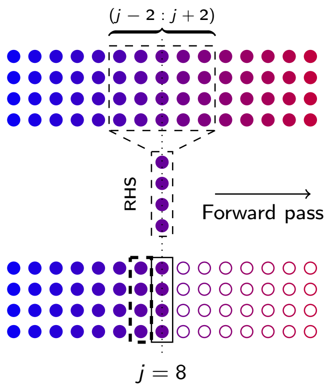


Figure: Visualisation of the fused forward pass and RHS construction operation. Input field is shown on top, and the output field is at the bottom. The RHS at each j is constructed using the input field values at $j - 2$ to $j + 2$. Then the forward pass is applied using the output field values at $j - 1$. SZ many lines are processed concurrently in a CPU core or in a GPU SM.

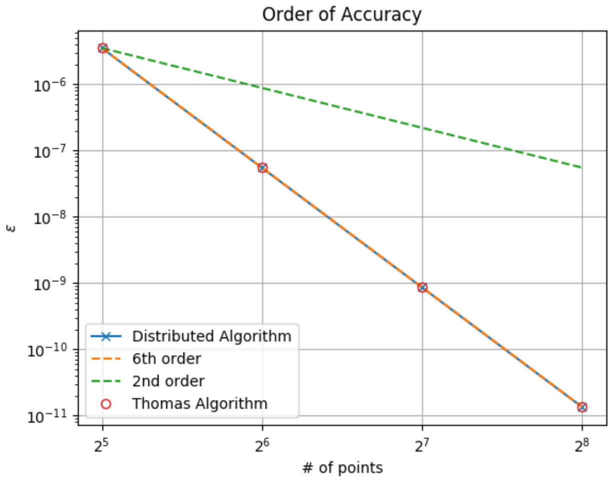


Figure: Order of Accuracy comparison.

Strong Scaling - Distributed Tridiagonal Solvers

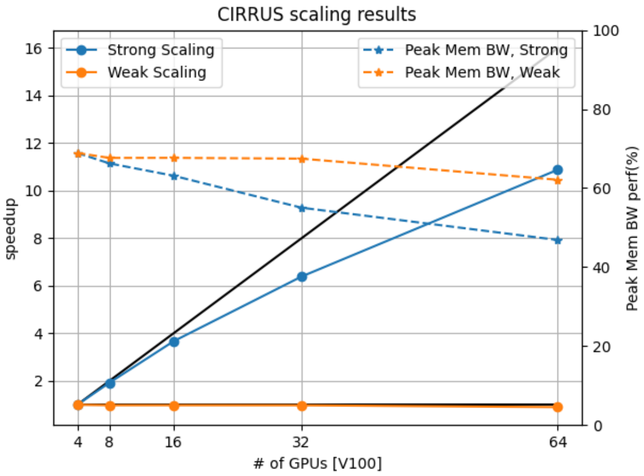


Figure: Strong Scaling on a GPU cluster.

Strong Scaling - Distributed Tridiagonal Solvers

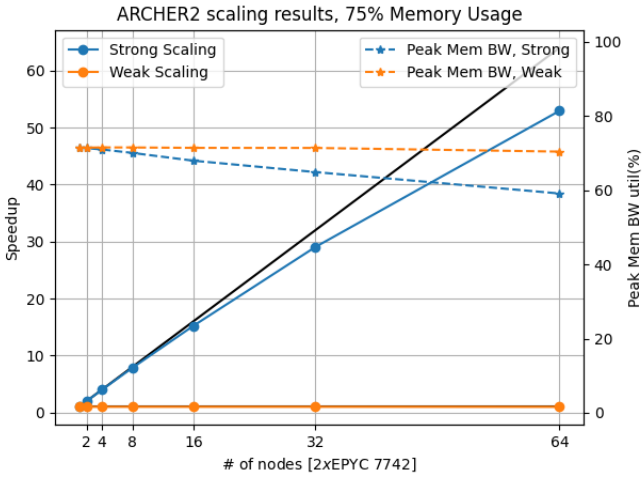


Figure: Strong Scaling on a CPU cluster.

- Backend based strategy to target CPU/GPU clusters.
  - CUDA Backend: MPI + CUDA for Nvidia GPUs
  - OpenMP Backend: MPI + OpenMP for CPUs
- Backend is specified at runtime
- Higher level algorithmic description:
  - Uses methods all backends implement
  - A single implementation for the overall algorithm

x3d2 - The new framework

- Pressure correction in Xcompact3D requires a Poisson solver
- Poisson solver based on distributed-memory 3D FFTs
  - CUDA Backend uses cuFFTMp
  - OpenMP Backend uses 2DECOMP&FFT
- Distributed-memory FFTs require MPI all-to-all type communication
- A 1D decomposition necessitates 2x MPI all-to-all communication
- An iterative Poisson solver is under active development

Taylor-Green vortex

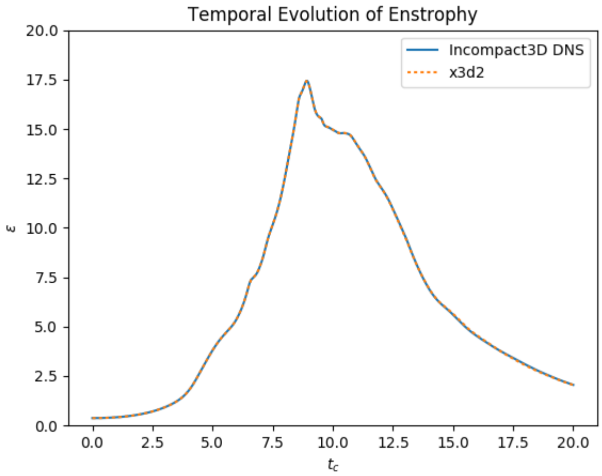


Figure: DNS of TGV on a 768^3 mesh, 1.8Bn DOF, $Re = 2500$.

- Global communications are eliminated for the tridiagonal solvers.
  - Few layers halo-data communication between previous and next ranks
- Excellent weak and strong scalability for tridiagonal solvers
- Future work
  - Iterative Poisson solver
  - Wind turbine simulations

Thank you!