DNS of Turbulent Flows

Ning Li, NAG

Sylvain Laizet, Imperial College

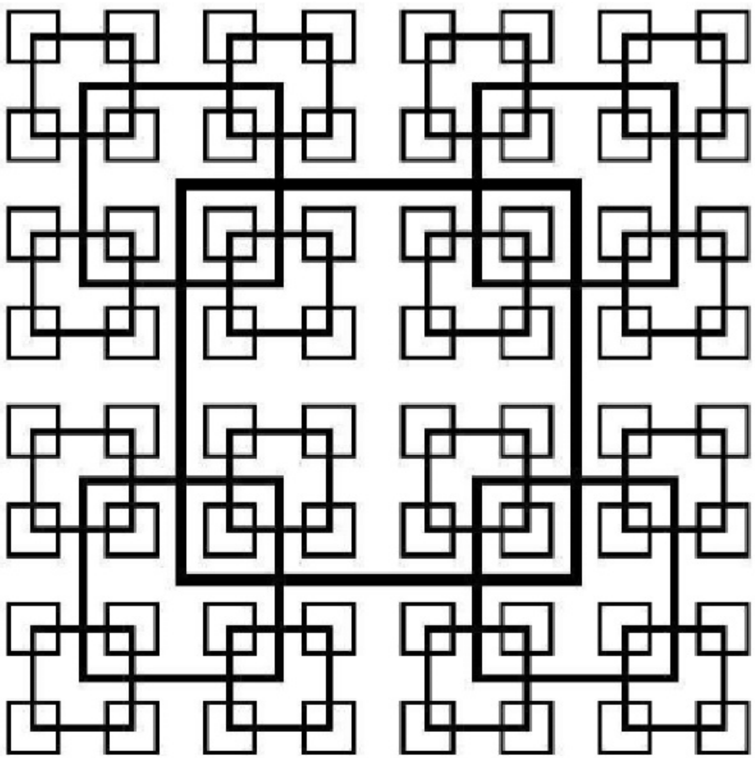
HECToR dCSE technical meeting

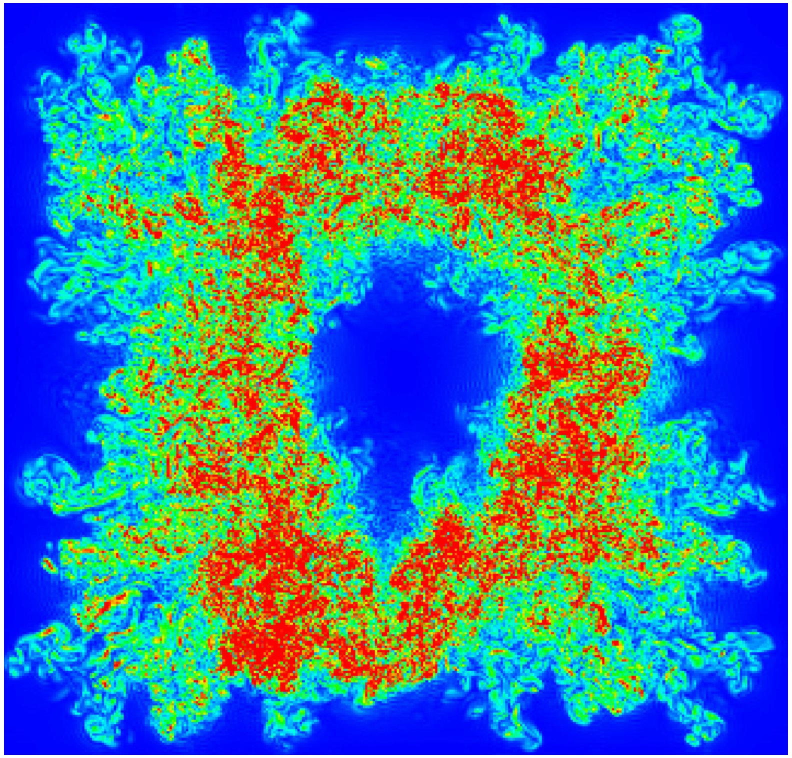
Manchester, 4-5 October 2011

Project Information

- Turbulence, Mixing and Flow Control Group at Imperial College
- PI – Prof. Vassilicos
- A previous dCSE on code Incompact3D successfully completed in early 2011 (16-month effort)
- Current project working on different but related CFD applications – Compact3D (6-month effort)
- Software framework development

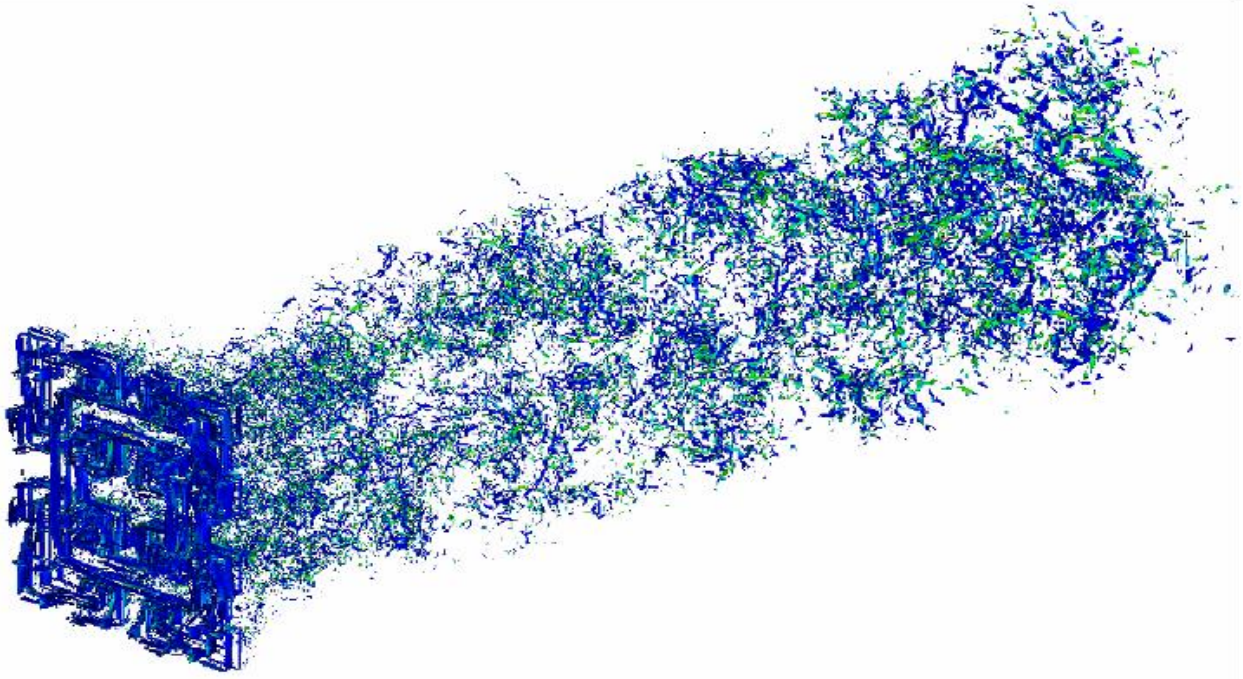
Scientific Background





- Turbulence generated by fractal objects

Typical Simulations



- 2881*360*360 ~ 270 million mesh points
- On 8100 HECToR cores
- >200,000 time steps (~1.5 second in real life)
- 4 TB data (only 200 full 3D data sets + some probes)

Incompact3D vs. Compact3D

- Incompressible vs. compressible solver
- Numerical methods fundamentally different
  - Transport equation of density for compressible case
  - Pressure Poisson solver for incompressible case
- Different parallelisation strategy
- Sharing some algorithms, such as derivative calculations
  - Based on compact finite difference method

Key Numerical Algorithms

- 6-order compact finite difference

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

- Spectral method for Poisson solver
  - Using Fast Fourier Transforms
- Both are spatially implicit schemes
- Transpose-based parallelisation required

2DECOMP&FFT

- Derived from the Incompact3D dCSE
- Framework to support the parallelisation of large-scale applications
  - that are based on Cartesian-topology mesh
  - that use spatially implicit numerical schemes
    - compact finite difference scheme; spectral method
- Now mature enough to be release as an open-source package (http://www.2decomp.org)
- Part of the Open Petascale Libraries

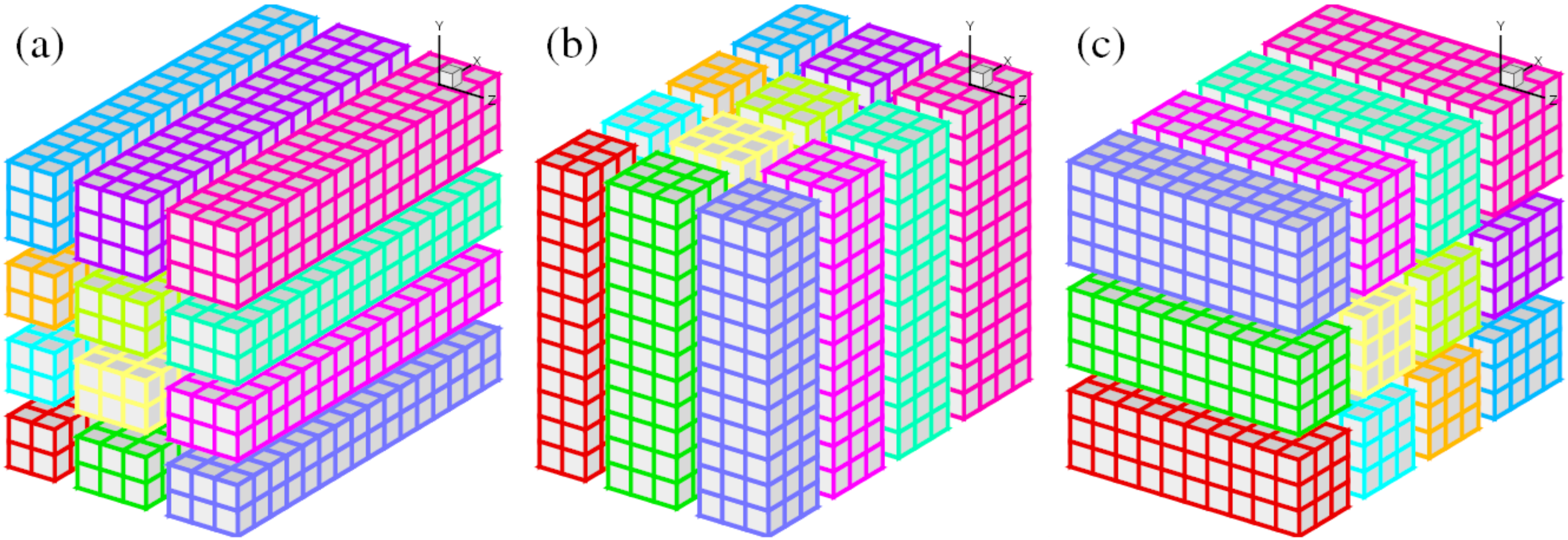
HECToR dCSE & UKTC

- Completed
  - Incompact3D (Vassilicos, Imperial)
  - EBL (Coleman, Southampton)
  - DSTAR (Luo, Southampton)
- Ongoing
  - Compact3D (Vassilicos, Imperial)
- Approved and to start
  - SS3F/SWT (Coleman, Southampton)
- Other UKTC code benefiting from dCSE work
  - SoFTaR (Jiang, Lancaster)
  - DNS/LES code (Chung, Warwick)
- All based on 2DECOMP&FFT (Except EBL - EPCC)

2DECOMP&FFT Features

- General-purpose 2D pencil decomposition (algorithm independent)
- Distributed Fast Fourier Transform
- Halo-cell communication
- Parallel I/O
- Various optimisations

2D Pencil Decomposition

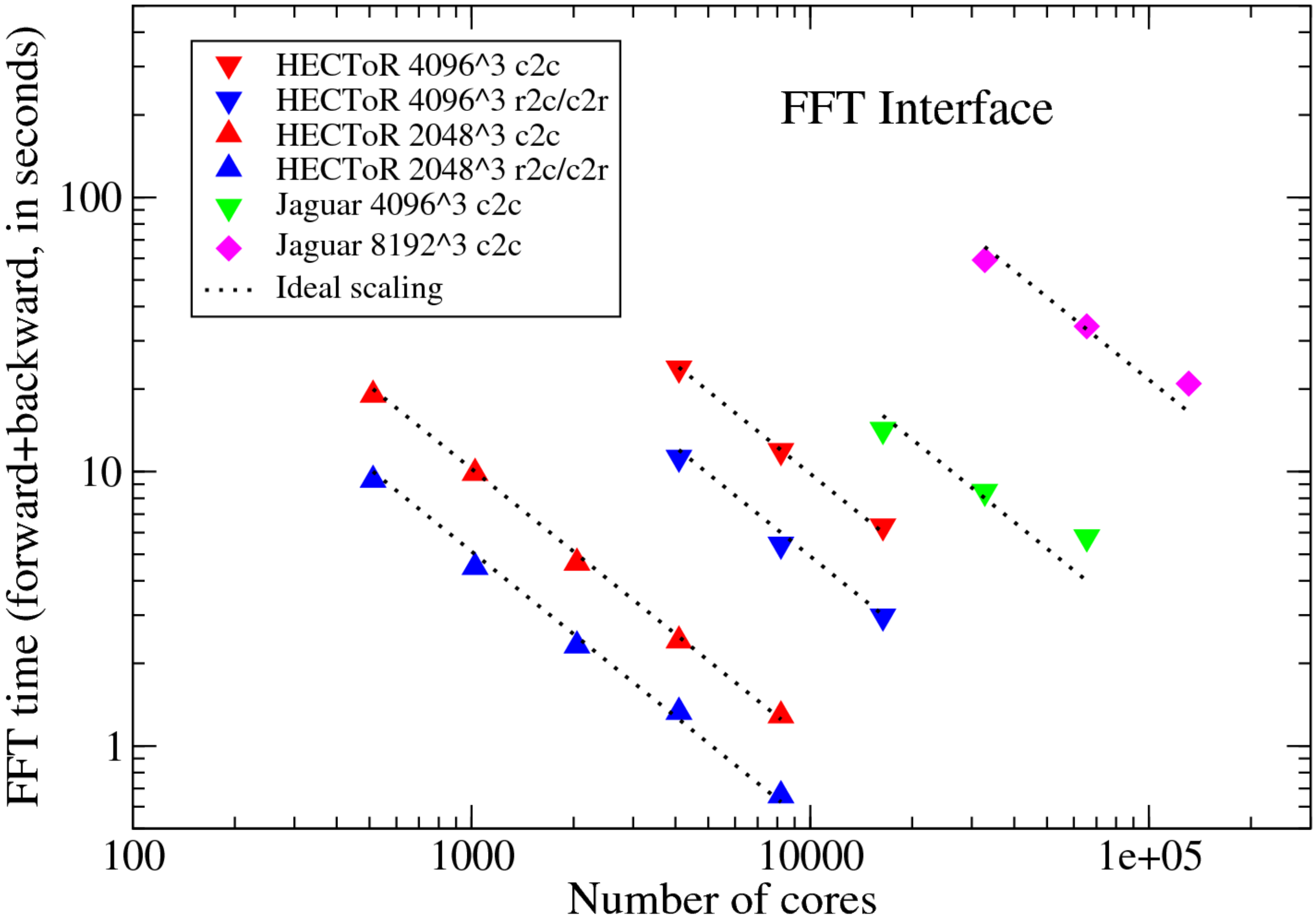


- Much better scalability than 1D decomposition
- 4 transposes to traverse 3 states
- Black-box implementation to hide most communication details

Distributed FFT

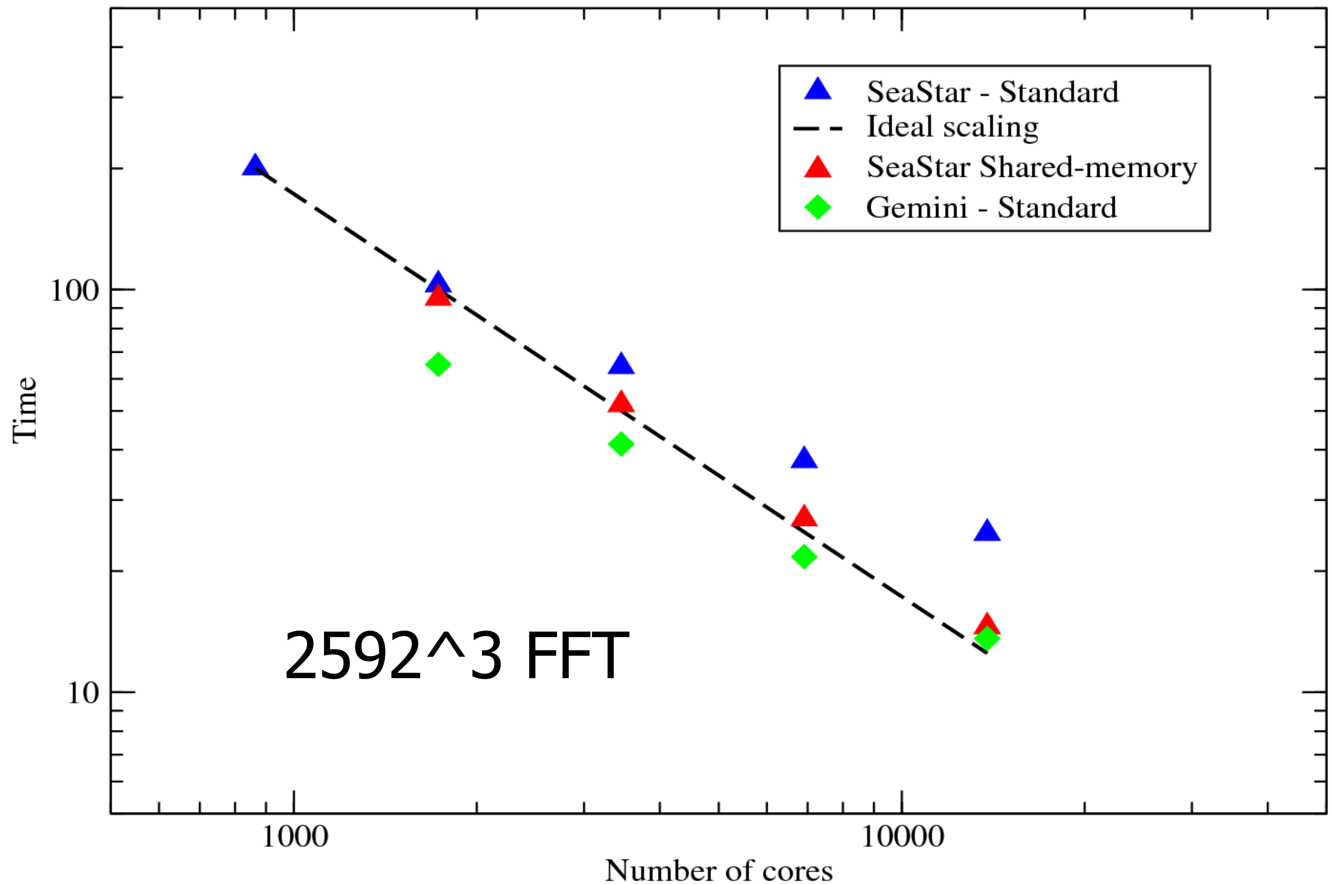
- Built on top of the 2D decomposition API.
- Support complex and real transforms
- Portable - interface with popular FFT libraries
  - FFTW, ACML, MKL, ESSL, etc.
- User-friendly API
  - call decomp_2d_fft_3d(in, out, direction)
  - Utility functions to help set up data structures
- Scale to tens of thousand of cores.

FFT on HECToR Phase 2a

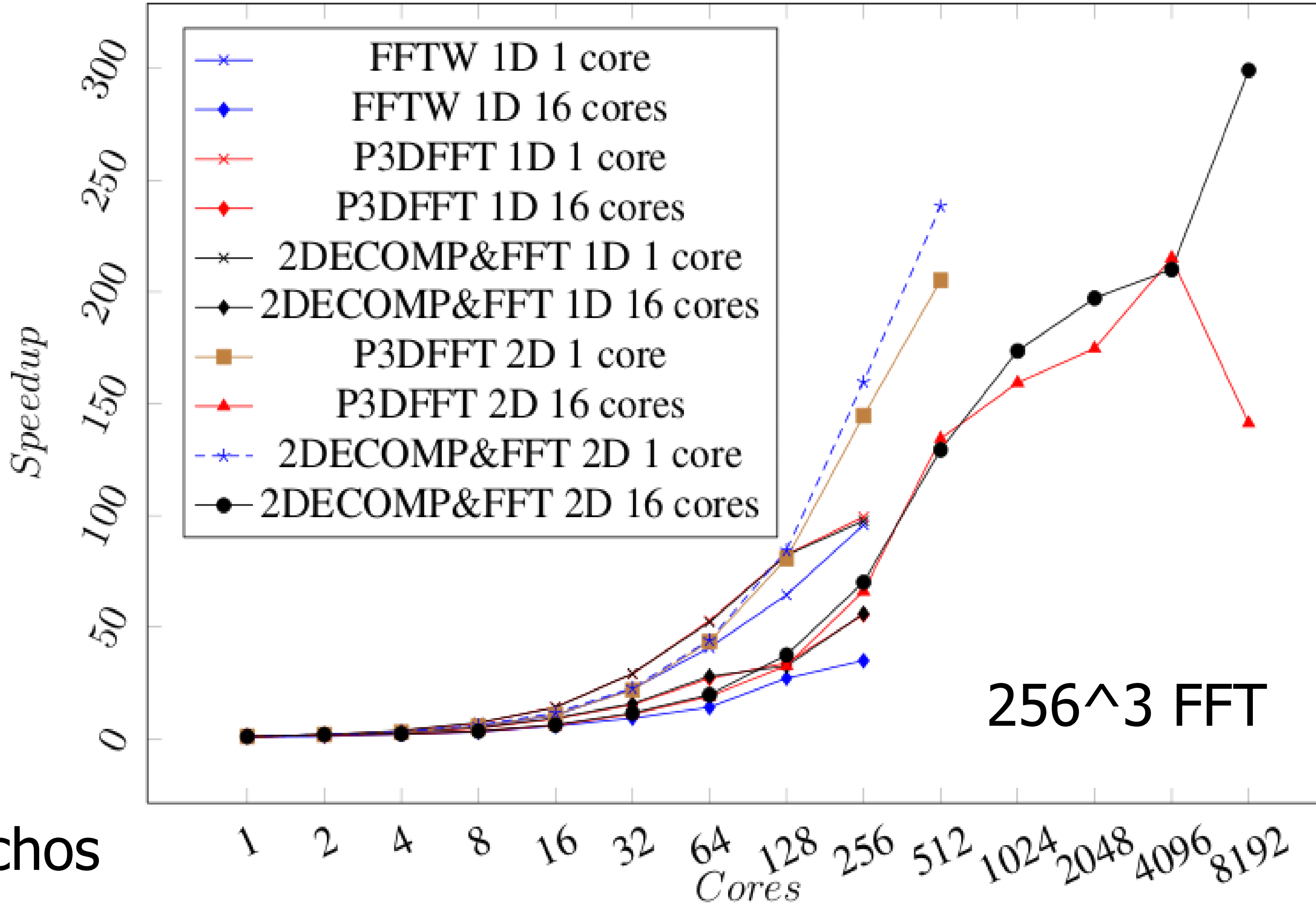


FFT on HECToR Phase 2b

XE6 Performance - SeaStar vs. Gemini



2DECOMP&FFT vs. P3DFFT



E. Brachos

Poisson Solver

- Based on spectral method
  - Pre-processing in physical space
  - 3D forward FFT
  - Pre-processing in spectral space
  - Solve the Poisson's equation in spectral space (algebraic operations)
  - Post-processing in spectral space
  - 3D inverse FFT
  - Post-processing in physical space
- Use standard FFT even for non-periodic B.C.
- Built on top of the FFT library
- Also requires global transpositions

Halo-cell Support

- One of the WPs in the current dCSE
- A second set of communication code
- Neighbouring pencils to talk to each other via MPI_SEND/RECV
- Allows explicit numerical schemes to be used in global transposition code
  - Stencil-based FD/FV schemes; particle tracking (interpolation)
- Periodic B.C. support

Parallel I/O

- Typical I/O requirements
  - Read/write full 3D data sets
  - Cut 2D planes
  - Save 3D data sets at reduced resolution
  - Helper functions for checkpointing/restart
- Implemented using MPI-IO
- Data storage in natural order and independent on number of processors
- For the future
  - Statistics computations
  - Simple 2D visualisation
  - Other parallel IO models (multiple writers etc.)

Optimisations

- Point-to-point (preferably non-blocking)
- System V shared-memory communication
- Scatter/gather to emulate SHM
- One-sided communication
- Padded ALLTOALL optimisation
- Overlap of communications and computations
  - Using OpenMP threads
  - Non-blocking ALLTOALL(V)
- Flexible data layout (any i,j,k order; stride-1 layout)
- Hybrid MPI/threaded
- Combinations of some of the above

OCC

- Lucian Anton showed yesterday how this can be done using OpenMP thread
- Pure MPI implementation also possible
- Use non-blocking MPI collective operations
  - MPI_IALLTOALL etc. to appear in MPI-3
  - libNBC – a prototype implementation
    - Implemented using existing MPI 1 functions (non-blocking send/recv)
    - Explicit MPI_TEST may be required
  - Support by some recent Infiniband and MVAPICH2

K. Kandalla et al., Computer Science - Research and Development, Vol. 26, 2011

OCC in 3D FFT

To overlap communications of one FFT with computations of another FFT, assuming 2D decomp.

- 1D FFT in X for v_1
- Transpose X to Y for v_1 (blocking)
- 1D FFT in Y for v_1
- Start transpose Y to Z for v_1 (non-blocking)
- Do loop k = v_2 to v_N
  - 1D FFT in X for v_k
  - Transpose X to Y for v_k (blocking)
  - 1D FFT in Y for v_k
  - Start transpose Y to Z for v_k (non-blocking)
  - Wait for communication v_(k-1) to complete
  - 1D FFT in Z for v_(k-1)
- End do
- Finish v_k

With 2DECOMP&FFT, up to 15% performance gain on HECToR

OCC in 3D FFT (continued)

- Implementation details
  - New API in 2DECOMP&FFT
  - Use pencil decomposition
  - In each sub-step, compute loops of 1D FFTs (instead of using FFTW's advanced interface), so that MPI_TEST can be inserted to progress the communication.
  - Wait for better hardware/MPI library support
- Other ways to achieve OCC
  - Partition data in pencils to even smaller chunks; compute and transpose one plane at a time
  - ……

Flexible Data Layout

- Use standard 3D arrays by default
- Flexible data layout
  - Swap dimension of 3D array
    - Work on leading dimension for cache efficiency
    - Legacy applications may have swapped dimension for historical reasons (vector length etc.)
    - External library might impose constraint
  - Linear storage (1D buffers)
  - Additional cost for swapping, but most can be absorbed by the algorithm packing/unpacking MPI buffers

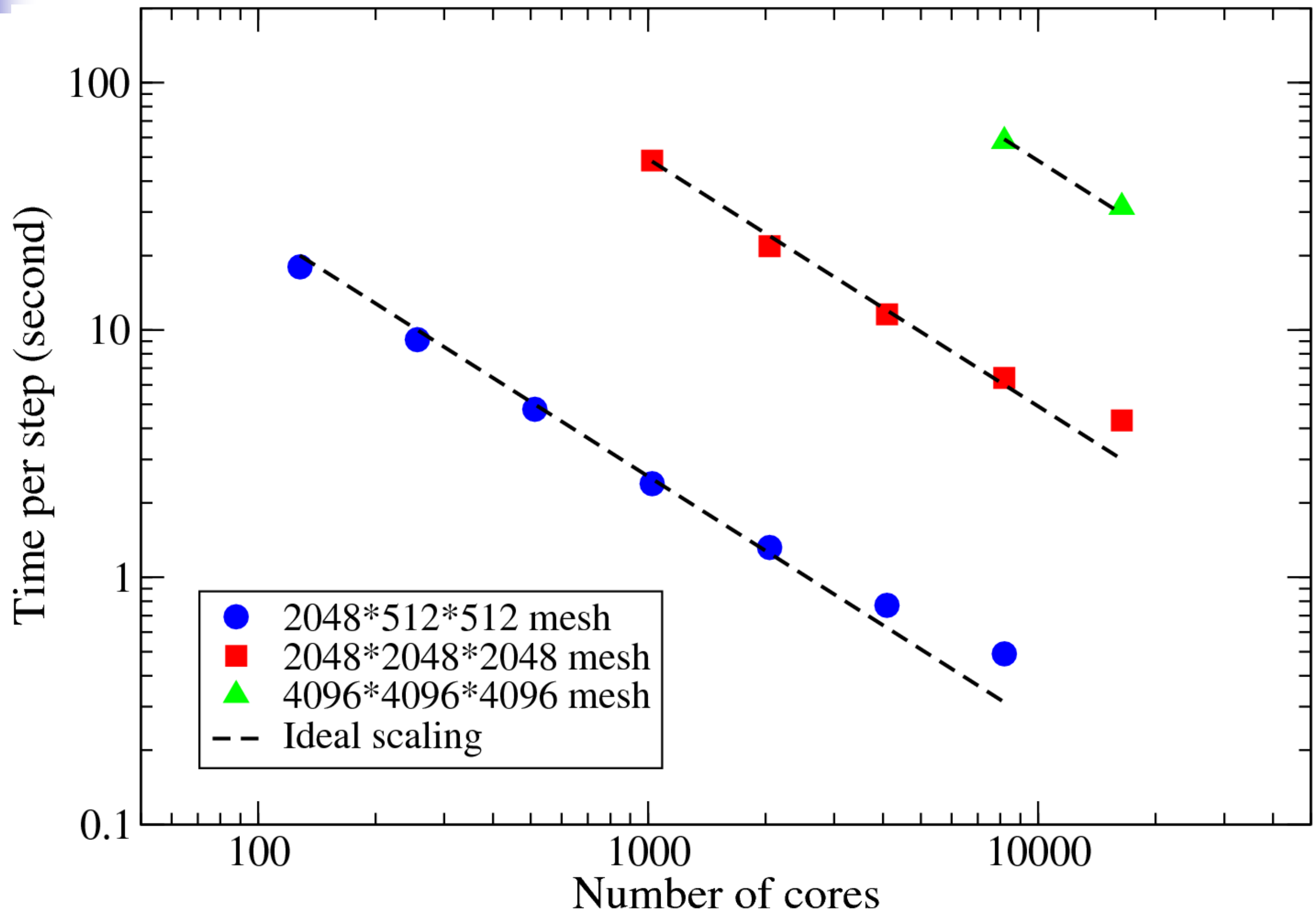
Incompact3D Details

- Was based on 1D slab decomposition
- Typical mesh: 2881*360*360
  - No more than 360 cores
  - Weeks to complete a simulation
- Rewritten using 2DECOMP&FFT
  - Regular production runs using 8000-16000 HECToR cores
  - Days to complete a simulation
- Expect similar benefits for Compact3D

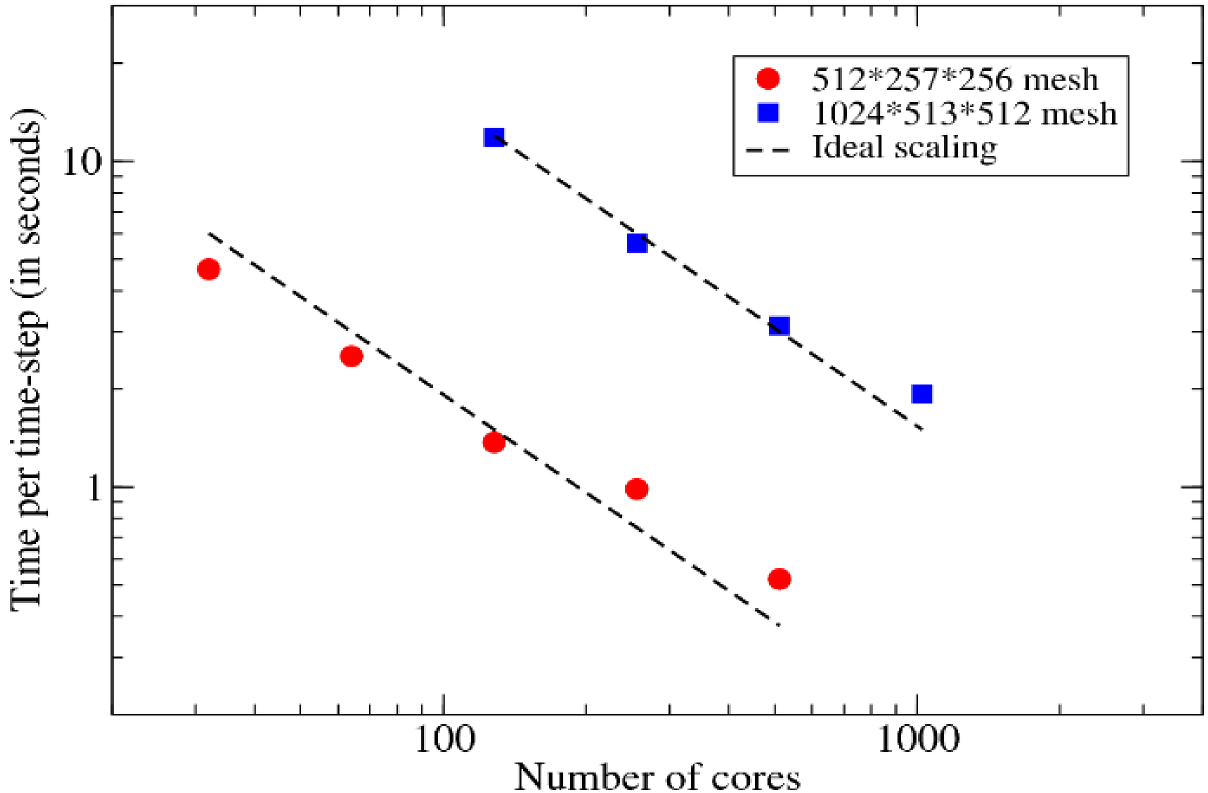
Incompact3D Details

- Compact Finite Difference scheme
  - Need to solve tri-diagonal system for spatial derivatives and interpolations
  - Use global transposition code extensively
  - Same applies to Compact3D
- Spectral Poisson Solver
  - Use 3D FFT
  - Also use global transposition to map internal data
- 66 calls to global transpositions per time step
- Halo-cell support for proposed particle tracking
- Tailor-made routines to take running average of low-resolution statistics (a second decomposition)
- Use 2DECOMP&FFT for I/O as well

Incompact3D on HECToR



Warwick DNS/LES code



- Parallelised by Edward Hurst using 2DECOMP&FFT
- Scaling well to 1024 cores on HECToR

Concluding Remarks

- Building scalable applications become increasingly challenging on modern supercomputers.
- Application developers need right tools/libraries.
- 2DECOMP&FFT is doing well, in CFD area in particular.