2DECOMP&FFT
The Library Behind Incompact3D

Ning Li
NAG

Incompact3D
User Group Meeting

Imperial College London
24/04/2014

nag®
Experts in numerical algorithms and HPC services

About the Speaker

- PhD in Mechanical Engineering (University of Maryland)
  - Experimental and numerical studies of Turbulence
- Academic experience
  - Department of Aeronautics, Imperial College
- Employment
  - STFC
  - Numerical Algorithms Group (NAG)

Background Information

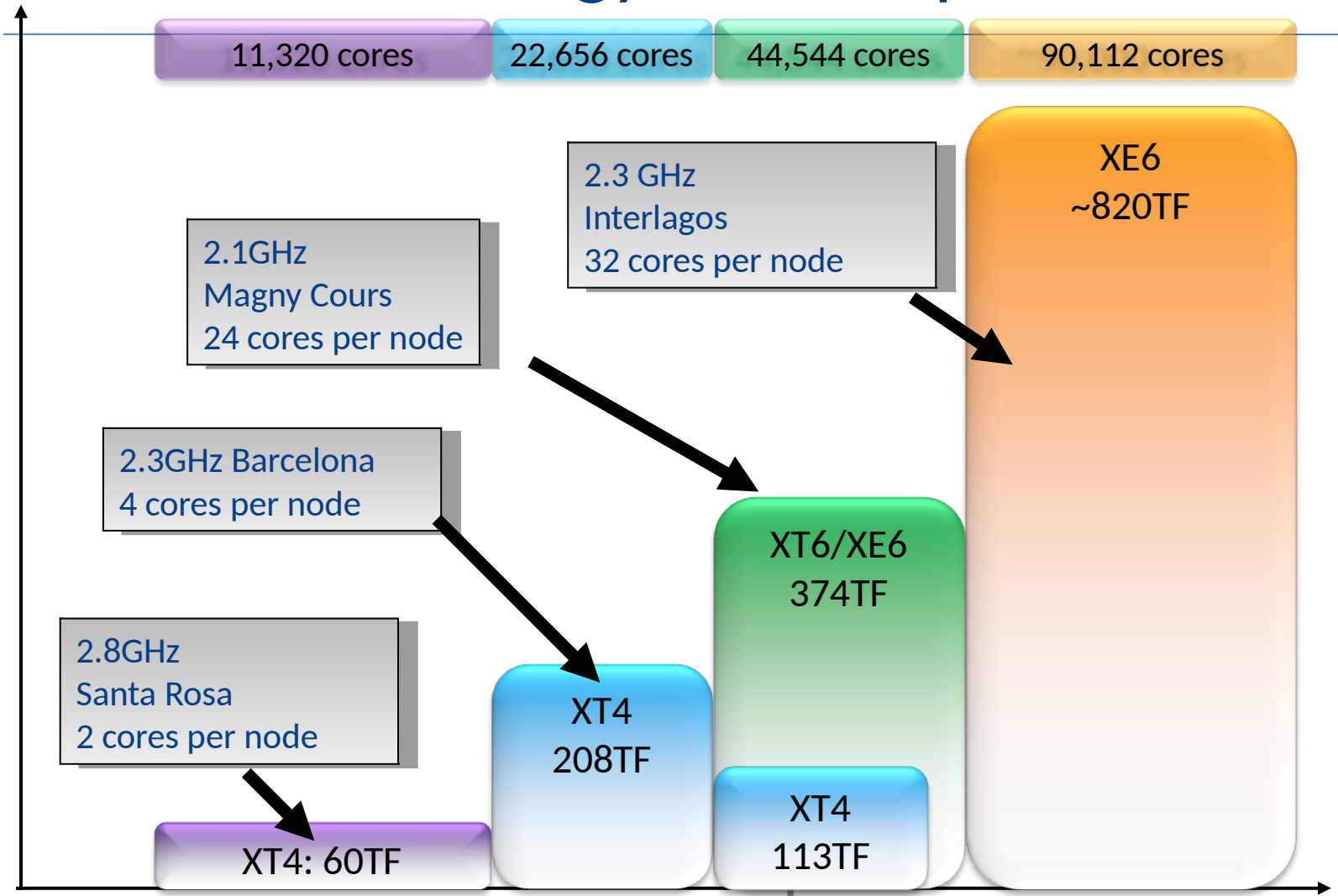
- NAG was awarded the Computational Science and Engineering (CSE) contract of HECToR back in 2007
- In addition to the day-to-day CSE tasks (porting, optimising, debugging, etc.) NAG provided a distributed CSE (dCSE) service
  - Dedicated software engineering support to help improve the capability of scientific applications
  - ~60 dCSE projects benefiting ~45 applications
- Incompact3D was one of the first applications benefited and received three rounds of support
  - A first project to refactor the parallel decomposition of Incompact3D – to improve scalability and productivity (16-month effort)
  - A second project to parallelise Incompact3D's sister codes – widen scientific applications (6-month effort)
  - A third project to introduce communication/computation overlap – further improve scalability (5-month effort)

Beyond a Single Application

- Applications benefiting from dCSE programme: Turbulence modelling; Ocean modelling; Weather/climate simulations; Magnetised plasmas; Atomistic molecular modelling; Ab initio quantum mechanics; Density functional theory; Quantum Monte Carlo; Biostatistics; Structure modelling ……
- The technical challenges were not isolated and common patterns could be found: High-performance parallel I/O; Dynamic partitioning and load balance; Shared-memory programming; common algorithms such as Fast Fourier Transform ……
- Reusable software components -> library solutions

- **2DECOMP&FFT** was derived from Incompact3D projects to support a wide range of CFD applications

HECToR Technology Roadmap



Incompact3D's Key Numerical Algorithms

- 3D Cartesian mesh
- 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 pressure Poisson solver
  - Using Fast Fourier Transform
- Both are spatially implicit schemes
  - Requires boundary-to-boundary data access
- How to parallelise these algorithms?
  - Parallelise each basic algorithm to work with distributed data
  - Or, transpose data required into local memory and apply serial algorithm

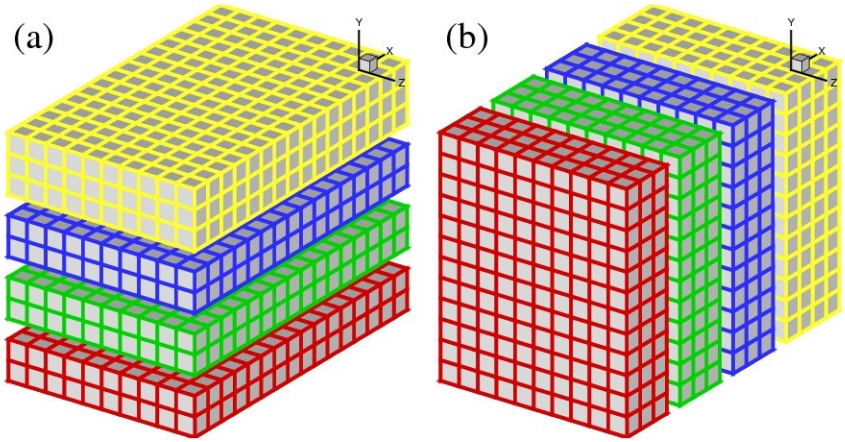
Transpose-based Method

- Existing serial algorithms remain unchanged.
- Only need to develop new communication code.
- Communication code and algorithms are often independent => library solution
- Communication library suitable for many spatially implicit numerical schemes
  - High-order compact finite difference scheme
  - Spectral method (Fourier, Chebyshev, ……)
  - Elliptic PDEs such as Poisson / Helmholtz

2DECOMP&FFT Framework

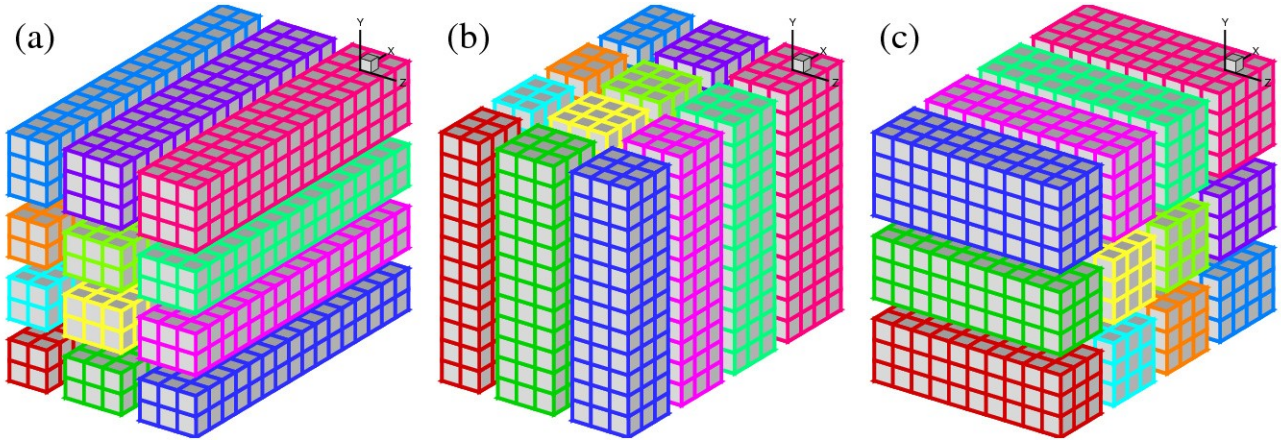
- Derived from Incompact3D work
- Main features
  - General-purpose 2D pencil decomposition & communication
  - Distributed Fast Fourier Transform
  - Halo-cell communication
  - Parallel I/O
  - Low-level optimisations
  - Support for overlap of communication and computation
- Designed to
  - Be scalable, flexible, user-friendly and portable
  - Simplify the development of large parallel applications

1D Slab Decomposition



- Procedures
  - ☐Apply algorithms in X-Z planes
  - ☐Transpose to state (b)
  - ☐Apply algorithms in Y direction
  - ☐Transpose back to state (a)
- Limitation
  - ☐For a N^3 mesh, can use no more than N cores
    - N ~ O(10^3) for many applications
    - Top supercomputers have ~ O(10^5) cores
  - ☐Memory constraint for large problems

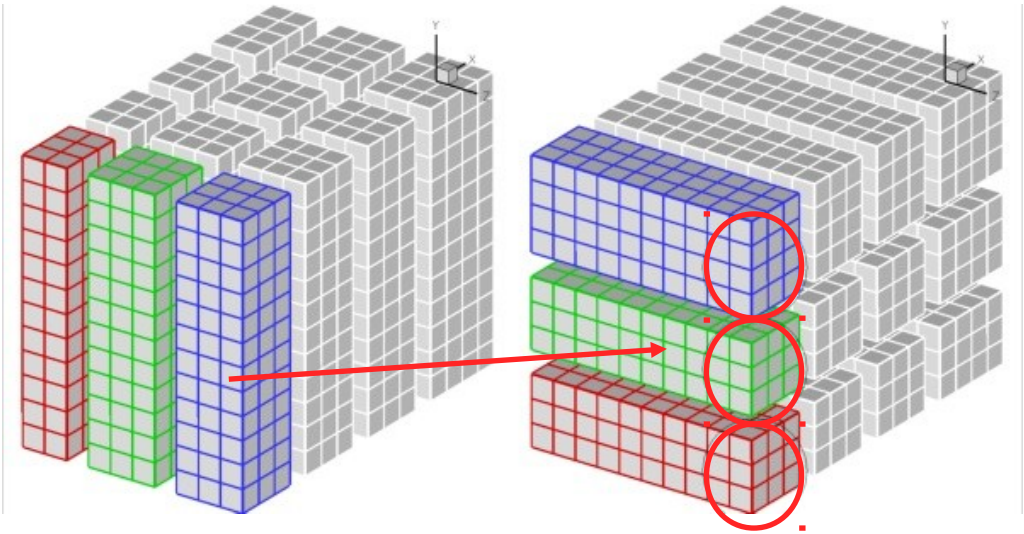
2D Pencil Decomposition

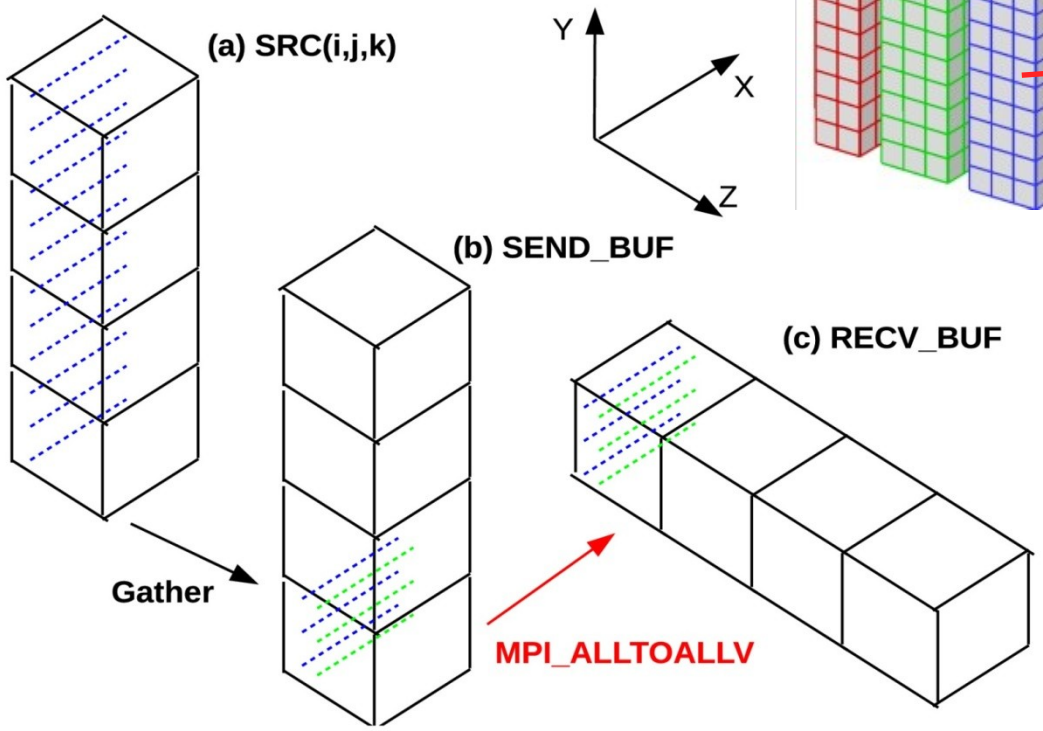


- 2D Pencil Decomposition
  - Also known as **block** or **drawer** decomposition
  - Apply basic algorithms (1D FFT, tri-diagonal solver, etc.) one direction at a time
  - 4 transpositions to traverse all states
    - (a) ⇔ (b) ⇔ (c) ⇔ (b) ⇔ (a)
  - Limitation: N^2 processes for N^3 mesh (much better scalability!)

MPI Implementation

MPI_ALLTOALLV(sendbuf, sendcounts, sdispls, sendtype, recvbuf, recvcounts, rdispls, recvtype, comm)





- Best local transpose algorithm?
- Application data layout?
- Optimisation opportunities?

call transpose_y_to_z(in,out)

Decomposition and Communication API

- Initialisation/finalisation
- Global variables
  - To describe data distribution (pencils' size, starting/ending indices), for defining data structures
  - To describe the communication buffers
- Communication routines
  - transpose_x_to_y(in, out)
- Advanced decomposition API
  - To support multiple arbitrary global sizes in the same application
  - transpose_x_to_y(in, out, decomp)

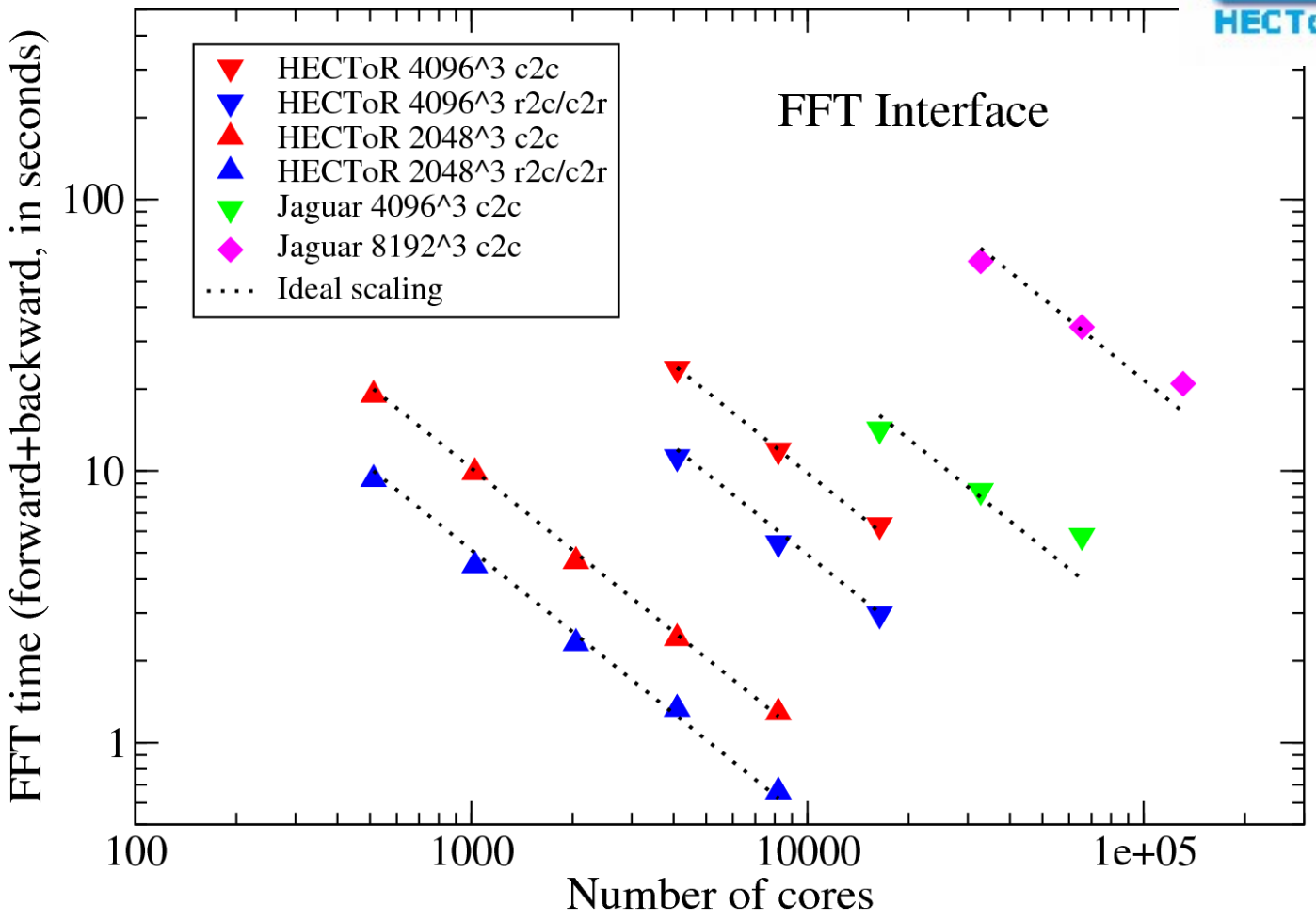
Review of Distributed FFT Libraries

Library	My comments
FFTW MPI	Slab decomposition
CRAFFT	Slab decomposition, evenly distributed data, Cray hardware only
BGL3FFT	C2C only, prime factor of 2 only, Blue Gene hardware
Plimpton's	C2C only, user callable communication routines
FFTE	Prime factor 2, 3 and 5 only
P3DFFT	R2C/C2R only
PFFT	FFTW-like (very complex) API, otherwise very good

Distributed FFT API

- Built on top of the 2D decomposition API
  - 1D FFT one direction at a time
  - Transpose as needed
- Support both complex and real transforms
- Portable – interface with all popular FFT libraries
  - 2DECOMP&FFT handles data communication
  - Let external library (highly optimised) compute
  - FFTW, ACML, MKL, ESSL, CUFFT…
- User-friendly API
  - Call decomp_2d_fft_3d(in, out, direction)
  - Utility function to help set up data structures
- Scale to tens of thousands of cores.

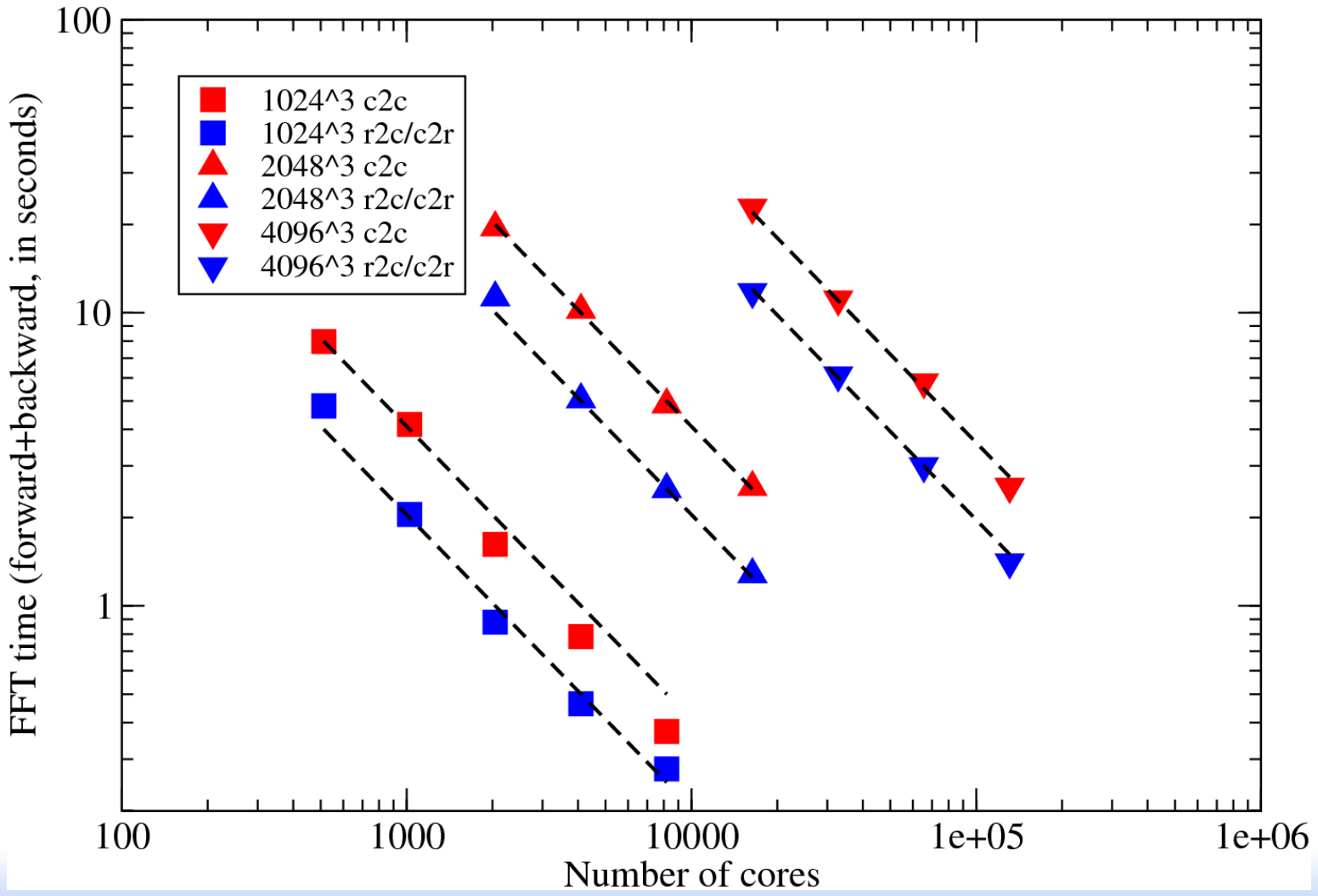
FFT on Cray XT



This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

FFT on IBM BlueGene/P

JÜLICH FORSCHUNGSZENTRUM



nag®

Distributed Poisson Solver for Incompact3D

- Based on spectral method
  - Basic steps
    - 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 the standard 3D FFT library even for non-periodic conditions
  - Depending on B.C., pre- and post-processing steps may require global transpositions (using the decomposition API)

Distributed Poisson Solver for Incompact3D

- Based on spectral method
  - Basic steps
    - 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 the standard 3D FFT library even for non-periodic conditions
  - Depending on B.C., pre- and post-processing steps may require global transpositions (using the decomposition API)

Distributed Poisson Solver(2)

- Based on matrix decomposition method
  - Idea:
    - 3D finite difference discretisation of Poisson's equation -> matrix with 7 diagonal lines
    - Apply Fourier analysis in one direction -> a number of penta-diagonal problems
    - FFT in another direction -> many tri-diagonal problems (fast solver available)
  - Implementation ： FFT in X → FFT in Y → tri-diagonal solver in Z → inverse FFT in Y → inverse FFT in X
  - For non-periodic data sets
    - Use discrete sine/cosine/quarter-wave transforms
    - In practice, use standard FFT library, with pre- and post-processing
  - Serial implementation: FISHPACK, FFTPACK
  - Parallel implementation: transpose between sub-steps.

Halo-cell Communication

- A second set of communication API
- Allows neighbouring pencils to talk to each other using MPI point-to-point communication
- Allows explicit numeric schemes to be used in the context transpose-based codes
  - Stencil-based FD/FV schemes
  - Particle tracking (interpolation)
  - Any calculations where accuracy isn't an issue: e.g. LES sub-grid scale models
- Periodic B.C. support

Parallel I/O

- Support typical I/O requirements
  - Read/write full 3D data sets
  - Cut 2D planes
  - Save 3D data set at reduced resolution
  - Save a 3D sub-domain
- Implemented using MPI-IO
  - Functions assisting checkpoint / restart
- Data stored in natural arrays orders
  - Easy pre- and post-processing
  - Independent on number of processors
- More features that may be implemented at library level:
  - Statistical computations
  - Simple 2D visualisation
  - Other parallel I/O models (multiple writers etc.)

Optimisation Techniques

- ☐Point-to-point communications (preferably non-blocking)
- ☐System V shared-memory communication
- ☐MPI scatter/gather to emulate shared-memory operations
- ☐Padded ALLTOALL optimisation
- ☐One-sided communication
- ☐Overlap of communications and computations (OCC)
  - Using OpenMP threads
  - Using non-blocking collectives (MPI 3.0 feature)
- ☐Flexible data layout (any i,j,k order; stride-1 layout)
- ☐Hybrid MPI/threaded
- ☐Combinations of some of the above

- ☐**Perform these in low-level libraries, not in user codes!**

Padded ALLTOALL Optimisation

- Cray's MPI has optimisations for ALLTOALL on small messages
  - ALLTOALL can be 3-4 times faster than ALLTOALLV
- For unevenly distributed data:
  - Either use ALLTOALLV
  - Or, pad the messages to equal-sized packages and use ALLTOALL
  - Ideally, auto-tuning to select faster algorithm
- Implemented in 2DECOMP&FFT (compile-time flag)

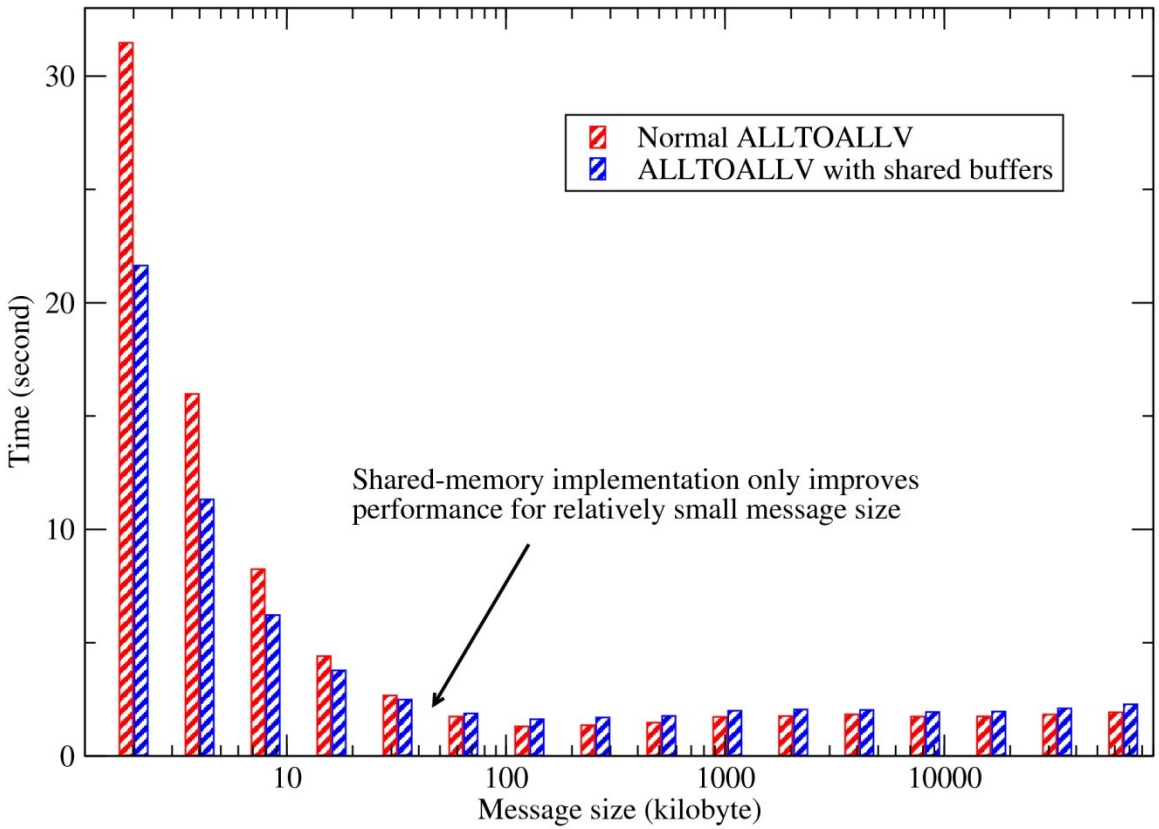
Shared-memory Programming

- Modern systems have multi-core CPUs sharing local memory
- Supercomputer's interconnects often prefer to handle small amount of large messages

- When applying to MPI_ALLTOALLV communication
  - Cores on the same node use shared sent/receive buffers
  - Only one core per node participates communication

- Implemented using System V IPC API
- Transparent to users (flag at compile time)
- Implementations: D. Tanqueray (Cray); FreeIPC by I. Bush (NAG)
- Worked well on Cray XT6, not XE6; also reported to work well on BG/Q

Shared-memory Performance



- Helpful when message size is small
- Most effective for networks with higher latency

Overlap Communication Computation

- OCC can be achieved using several techniques: OpenMP threads; non-blocking one-sided communication; non-blocking MPI collective, etc.

- Using non-blocking MPI collectives
  - Existing MPI_ALLTOALL etc. are blocking operations
  - MPI_IALLTOALL etc. introduced in MPI 3.0
  - LibNBC – a prototype implementation was initially used to add this feature to 2DECOMP&FFT
  - Some network hardware supports 'offloading'
  - Or use software means (MPI_TEST) to progress communication

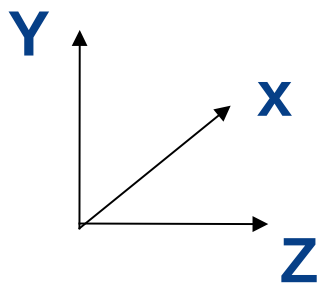
OCC in 3D FFT (Coarse-grained)

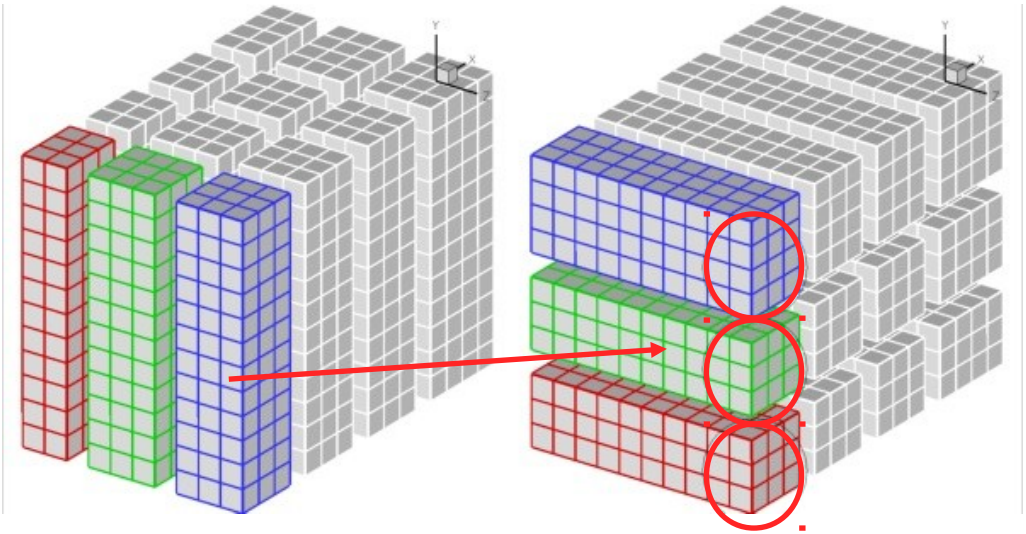
- For multiple independent 3D FFTs, overlap communications of one FFT with computations of another.
  - Assume 2D pencil decomposition
  - 1D FFT in X for v_1
  - Transpose X to Y for v_1 (blocking)
  - 1D FFT in Y for v_1
  - Start transpose Y to Z for v_1 (non-blocking)
  - Do loop k = v_2 to v_n
    - 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 finish
    - 1D FFT in Z for v_(k-1)
  - End do
  - Finish work for v_n

up to 15% performance gain on XT6/XE6

nag®

OCC in 3D FFT (Fine-grained)





- For a single distributed 3D FFTs
  - Assume 2D pencil decomposition
  - Further partition data in pencil into even smaller chunks
  - Transpose one plane of data while computing on another
  - For this to work well, very large data set required so that latency won't take over
  - Data layout in memory key for performance

OCC API

- Without 2DECOMP&FFT
  - Pack all-to-all send buffer from input 'in'
  - call MPI_ALLTOALL(V)
  - Unpack all-to-all receive buffer to output 'out'
- With 2DECOMP&FFT
  - Blocking version
    - call transpose_x_to_y(in, out)
  - OCC version, emulating the non-blocking MPI APIs
    - call transpose_x_to_y_start(handle, in, out, …)
    - Perform unrelated computation (cannot change ‘in’, cannot safely use ‘out’)
    - call transpose_x_to_y_wait(handle,…)

Flexible Data Layout

- 2DECOMP&FFT uses standard 3D arrays by default
- Flexible data layout (special branch of code)
  - Swap dimensions of 3D arrays
    - Work on leading dimension for cache efficiency
    - Legacy applications may have swapped dimensions already for historical reason (vector length etc.)
    - External library might impose constraint
  - Linear storage (1D buffers)
  - Additional cost for swapping, but most cost can be absorbed by the local transpose (packing/unpacking MPI buffers)

Incompact3D - Implementation Details

- Finite Difference (high-order compact scheme)
  - Solving tri-diagonal system while evaluating spatial derivatives or doing spatial interpolations
  - Using the transposition routines directly
- Spectral-based pressure Poisson solver using the 3D FFT library

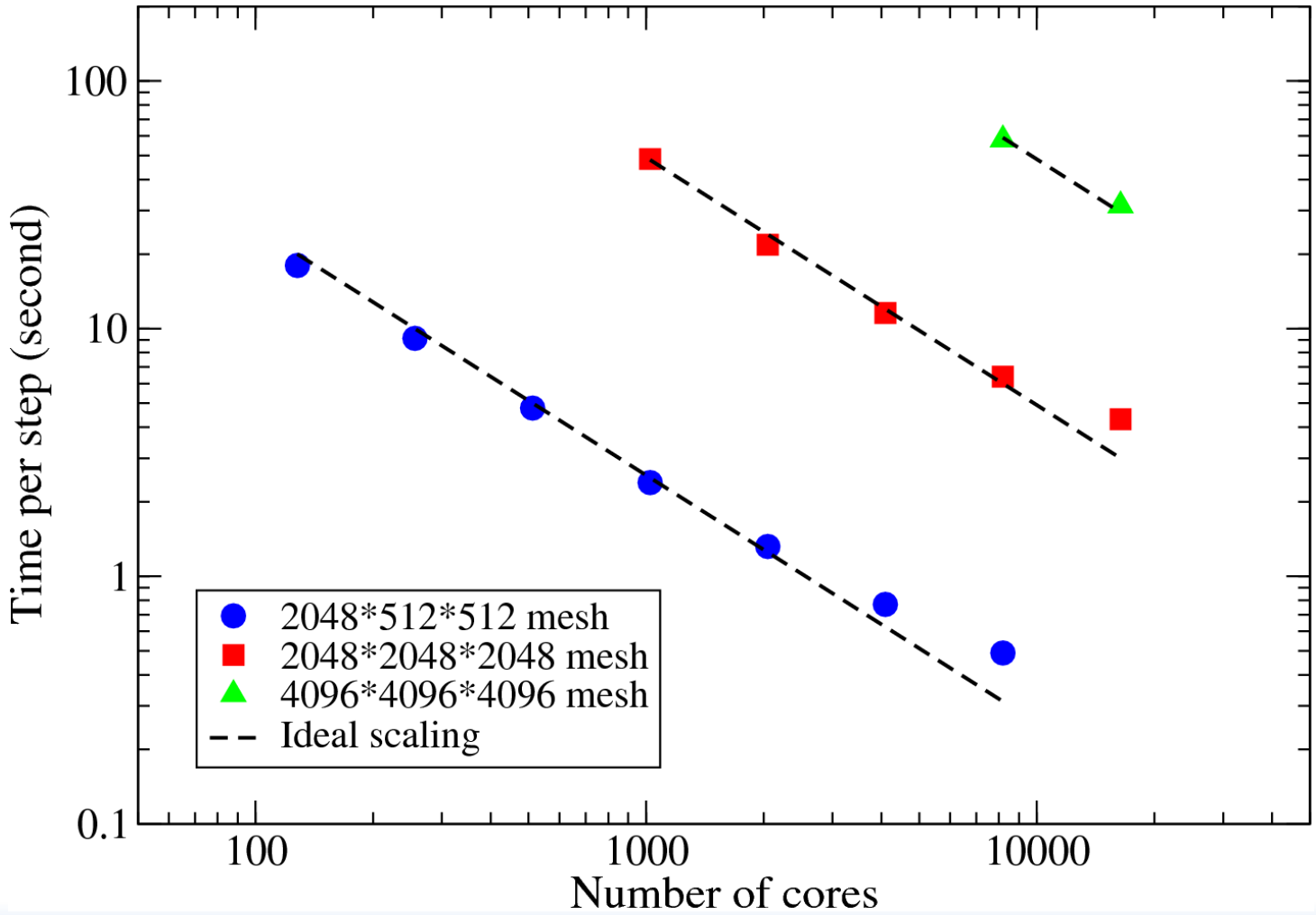
- The base DNS code uses up to 66 global transpositions per time step, depending on B. C.

- Also heavily rely on 2DECOMP&FFT for parallel I/O

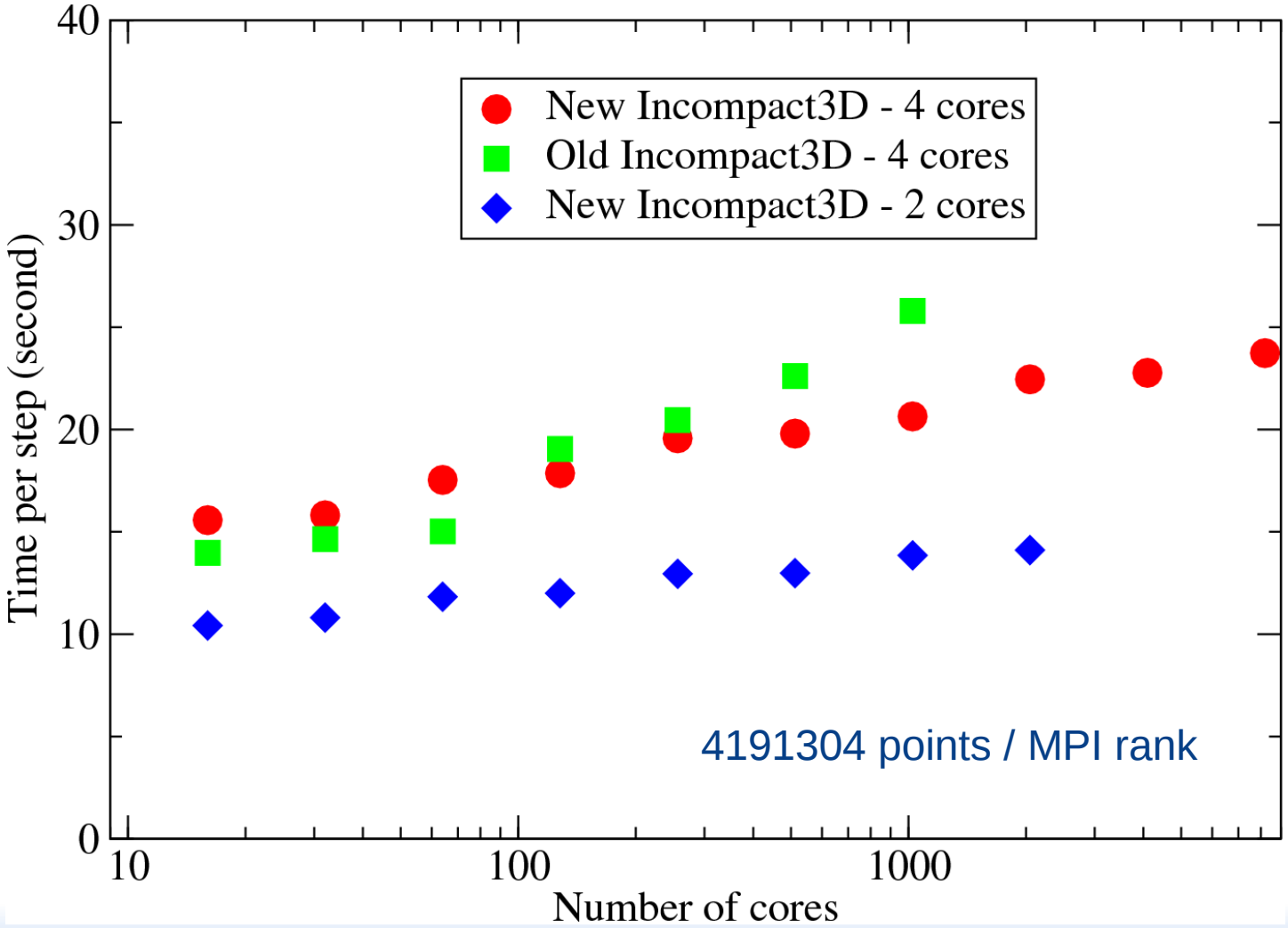
Incompact3D – Typical Simulations

- Was based on 1D slab decomposition before the supporting projects
- Typical mesh: 2881*360*360
  - No more than 360 cores
  - Weeks to complete large production runs
- Rewritten using 2DECOMP&FFT
  - Production runs on HECToR using 8000-16000 cores
  - Days to complete a large simulation, much improved productivity

Incompact3D's Strong Scaling on HECToR



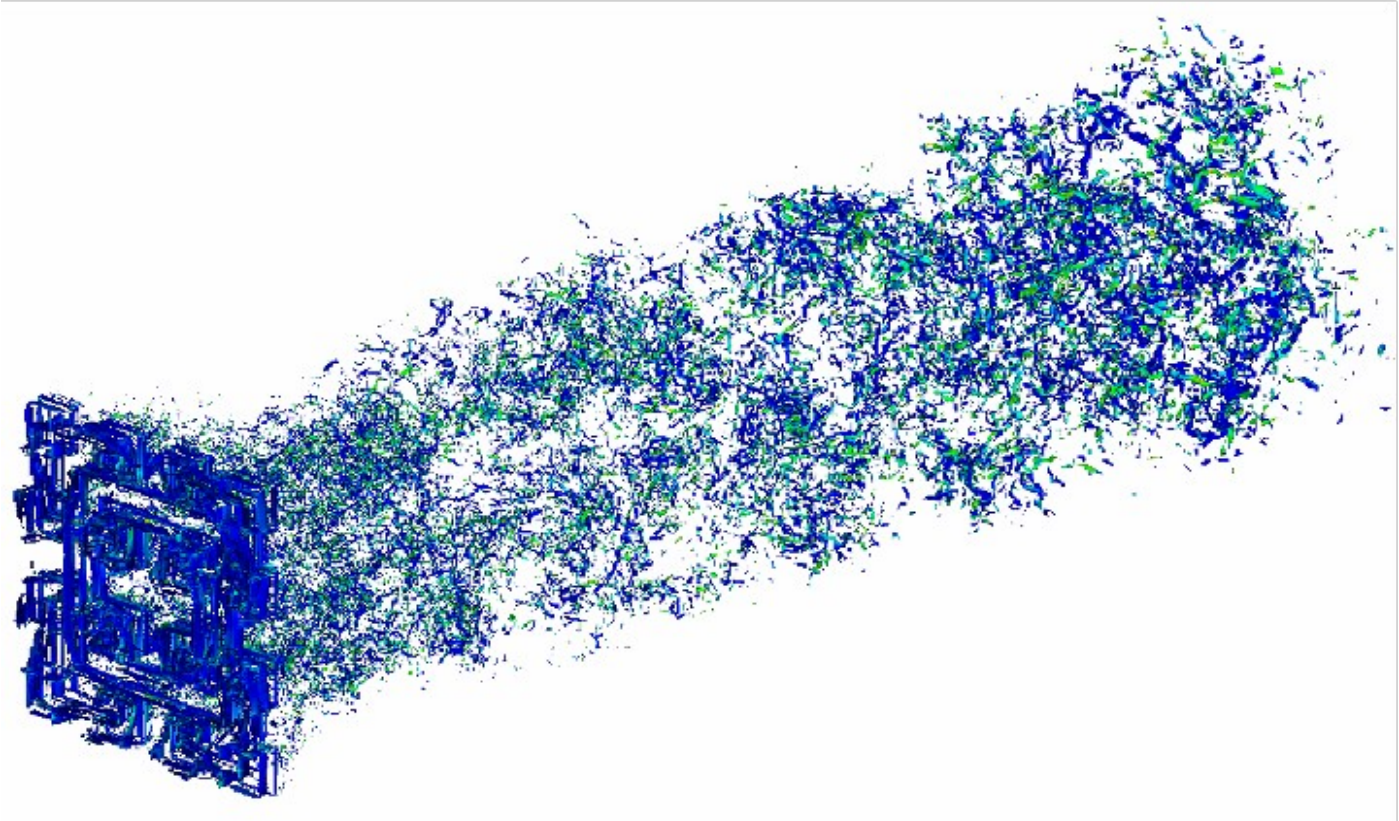
Incompact3D's Weak Scaling on HECToR



Impact – Within the United Kingdom

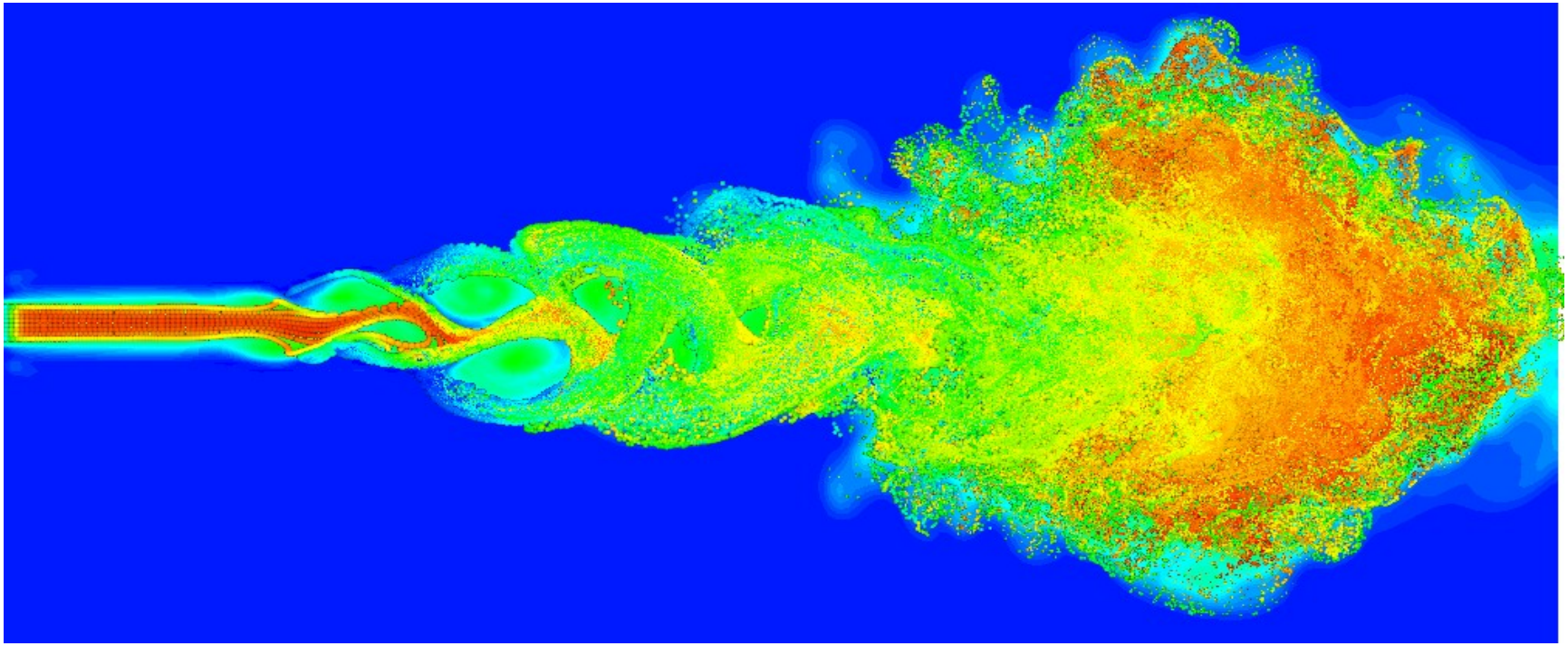
- Close collaborations with the UK Turbulence Consortium
- UKTC codes upgraded, based on my library or similar strategy
  - Incompact3D/Compact3D (Imperial)
  - DSTAR (Southampton)
  - SS3F/SWT (Southampton)
  - EBL (Southampton) * separate code base
- Other codes benefited indirectly
  - SoFTaR (Lancaster)
  - DNS/LES code (Warwick)

Incompact3D (DNS & LES)



- Vortical structures behind a fractal grid

DSTAR (Turbulence & Reaction)

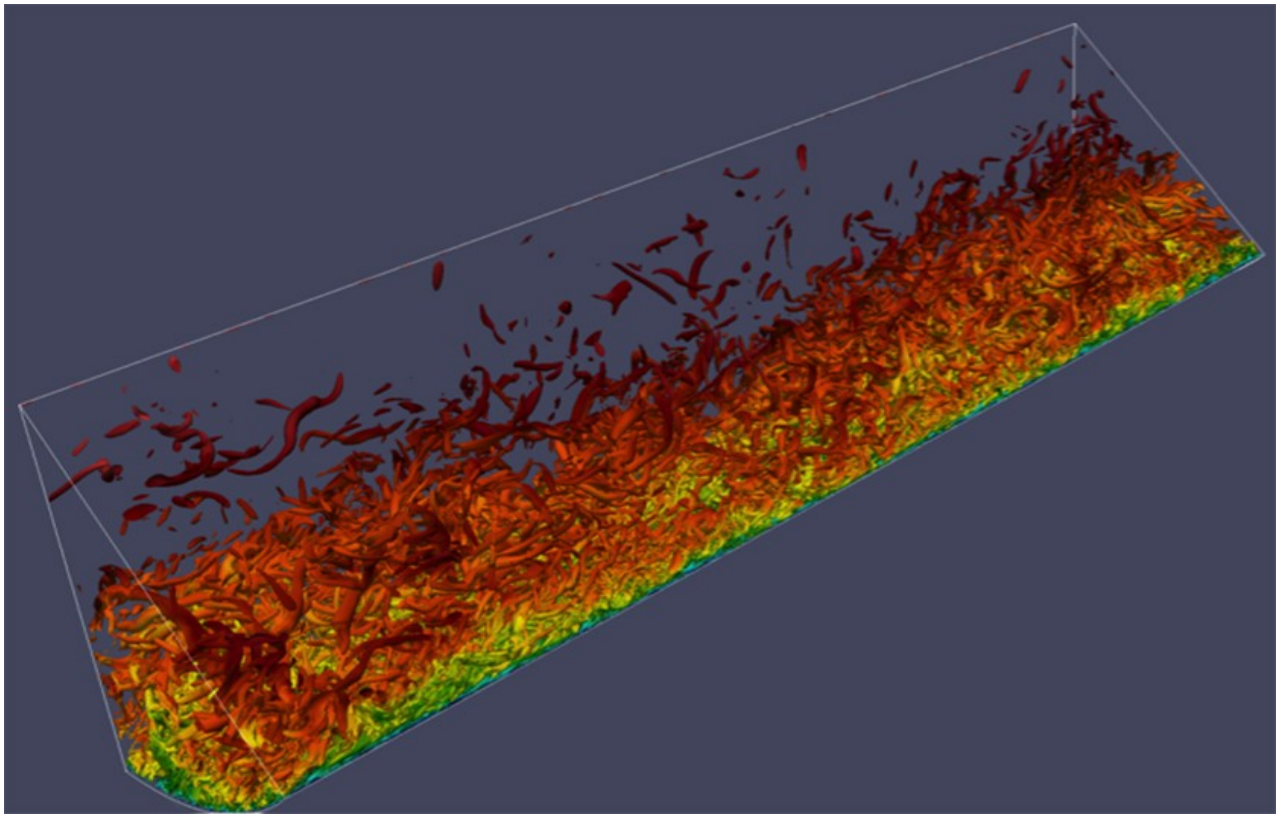


- Luo (Southampton): numerical simulation of flame interacting with water droplets.

Impact - International

- Cray User Group paper published in 2010
  - 20+ citations from scientific papers
- 2DECOMP&FFT open-sourced in 2011
  - 1000+ downloads http://www.2decomp.org
  - Used worldwide
- Applications
  - High resolution DNS of turbulence, up to 12288^3 grid (Nagoya/K computer)
  - 3dt – DNS of turbulence, 4096 grid, 524k threads (RWTH Aachen/JUQUEEN)
  - ……

HRPIPE (DNS of High-Re Pipe Flow)



- Delft/GCS: high resolution pipe flow simulations (Re up to 75,000)
- 7.6 billion grid points on 24,000 nodes

CINECA

- FFT training for PRACE
- Auto-tuning algorithm to select the best decomposition method
- Upgraded CFD code BlowupNS using 2DECOMP&FFT
- First attempt to apply 2DECOMP&FFT to a molecular dynamics code

Parallel Spectral Numerical Methods

- Solving ODEs/PDEs using pseudo spectral method
  - Course material developed at University of Michigan.
  - Introduce numerical methods and scientific computing
  - http://en.wikibooks.org/wiki/Parallel_Spectral_Numerical_Methods
- Numerical simulation of Klein-Gordon equation
  - High-resolution simulations to capture its blow-up behaviour

The Future

- Some of the technical areas covered by these projects remain hot research topics.
- Ongoing effort:
  - Co-processing for visualisation of large data set using Paraview Catalyst.
- Selected recent papers:
  - SC13 - “Petascale Direct Numerical Simulation of Turbulent Channel Flow on up to 786K Cores” Optimise cache reuse, memory access and communication for the global data transposes.
  - PPoPP '14 - “Designing and auto-tuning parallel 3-D FFT for computation-communication overlap” Parameterise and auto-tune non-blocking alltoall.

Conclusion

- Researchers can benefit from dedicated software engineering support
  - Improved productivity
  - Wider range of applications
- 2DECOMP&FFT framework
  - Powerful and user-friendly platform for building CFD applications
  - Widely adopted by the research community
- Collaboration opportunities?