HECToR

HIGH END COMPUTING TERASCALE RESOURCE

A Research Councils UK High End Computing Service

Parallelisation of CABARET

Phil Ridley

Numerical Algorithms Group Ltd, HECToR CSE

Outline

- Project background
- What is CABARET?
- Applications of CABARET
- Why develop the CABARET code?
- Code description and performance
- Conclusion

CABARET dCSE

- Started March 2009 working 50%
- Collaboration with the Whittle Laboratory, University of Cambridge Department of Engineering
- First dCSE project requiring full parallelisation of a code
- Ends Feb 2011

HECToR

RESEARCH COUNCILS UK

What is CABARET?

- Compact Accurately Boundary Adjusting high-Resolution Technique (CABARET)

- Method is based on an extension of the original second-order Upwind Leapfrog three-time-level advection scheme (Roe, 1993)

- Two-time-level non-oscillatory scheme for quasilinear hyperbolic conservation laws

HECToR

RESEARCH COUNCILS UK

Applications of CABARET

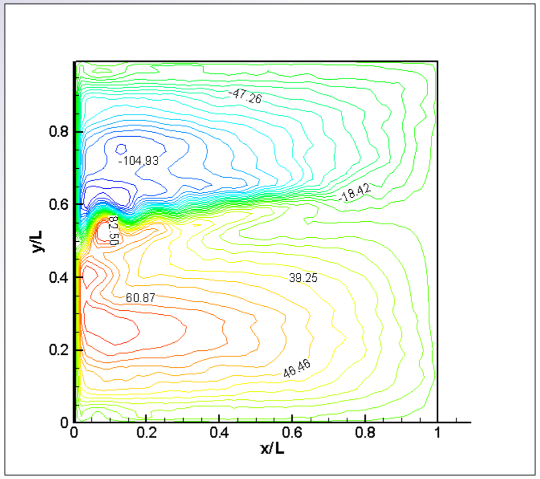
- Applicable to compressible unsteady Navier-Stokes Flow problems ranging from acoustic wave propagation and vortical flows to shock wave interaction

- In Re~10000 calculations the method gives a very good convergence without additional preconditioning, down to Mach numbers as low as M~0.05-0.1

CABARET LES ocean modelling

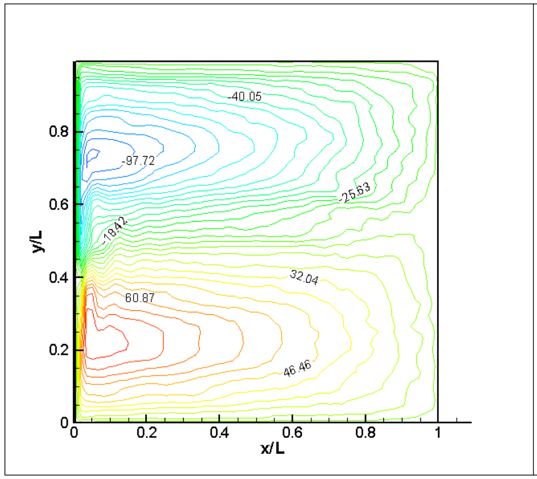
• Because of hydrodynamic instability, the boundary layer, which occurs at the left (western) domain boundary, is separated and a free jet in the eastward direction is developed
• Time-averaged stream function in the top layer of the 3-layer model, CABARET is ~ 30 times more efficient!

CABARET

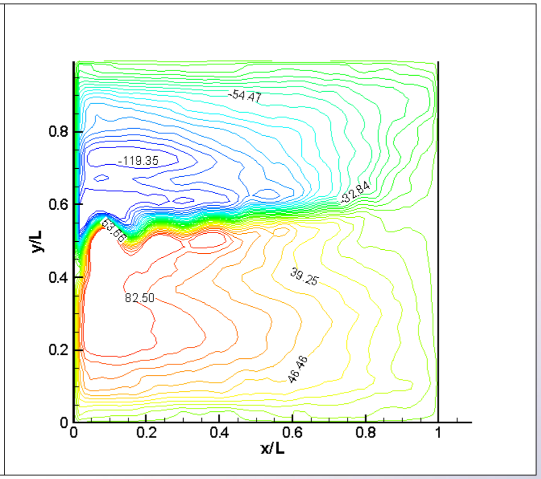


Grid 257x257

Conventional 2nd order central scheme



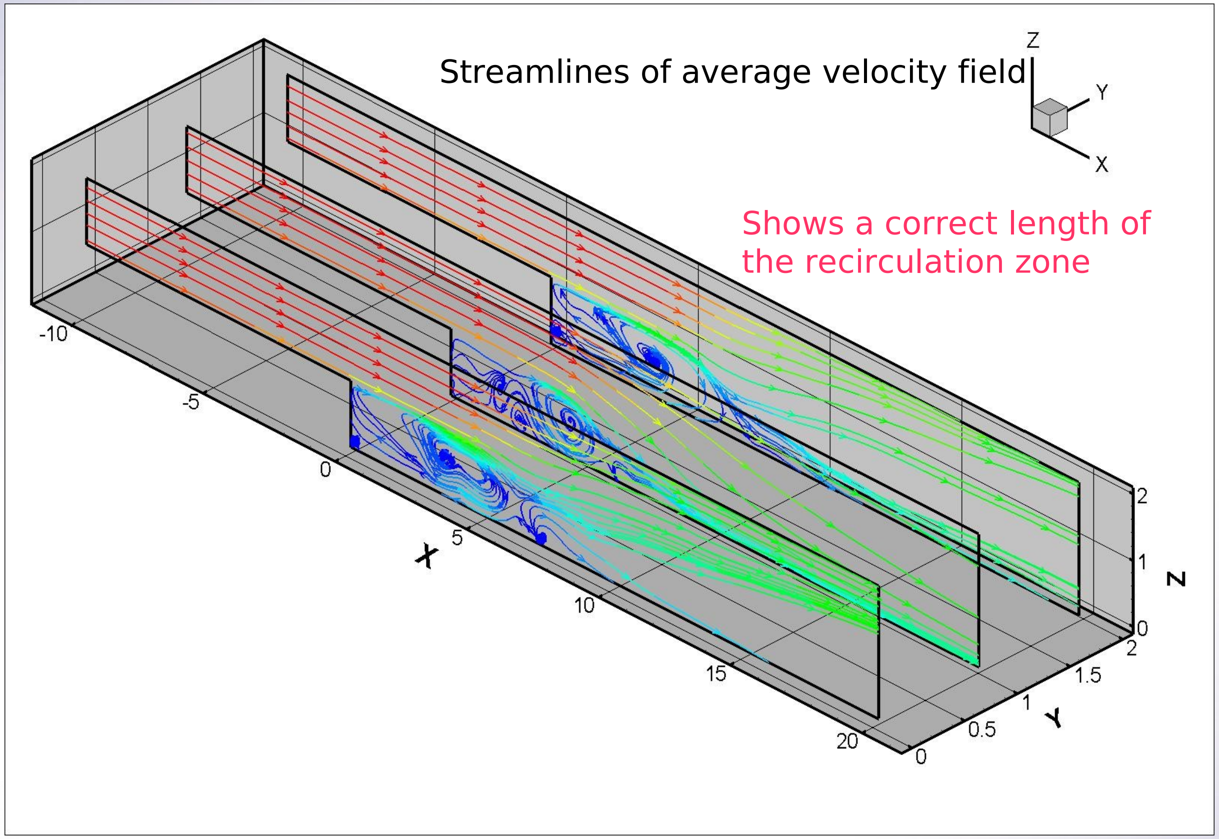
Grid 257x257



Grid 1025x1025

3-D backward facing step

Test case Low Mach number turbulent flow around a 3-D backward facing step



Re=5000
M =0.1
Grid: 10 points per step height size=1
Inflow BC: laminar

'Mature' solution courtesy of Prof. Vasily Kondakov, NSI, Moscow

Why develop parallel CABARET?

• Original Fortran 90 code was developed around 1998

• Can handle up to a million grid points

• To resolve higher resolution grids required in high fidelity LES simulations where Re>100000 we require 10 million grid points

• Locality of algorithm will lend itself well to distributed processing – develop parallel code

Data decomposition

- CABARET is similar to a finite difference / finite volume calculation

- Parallel version of the code already exists for a structured orthogonal grid

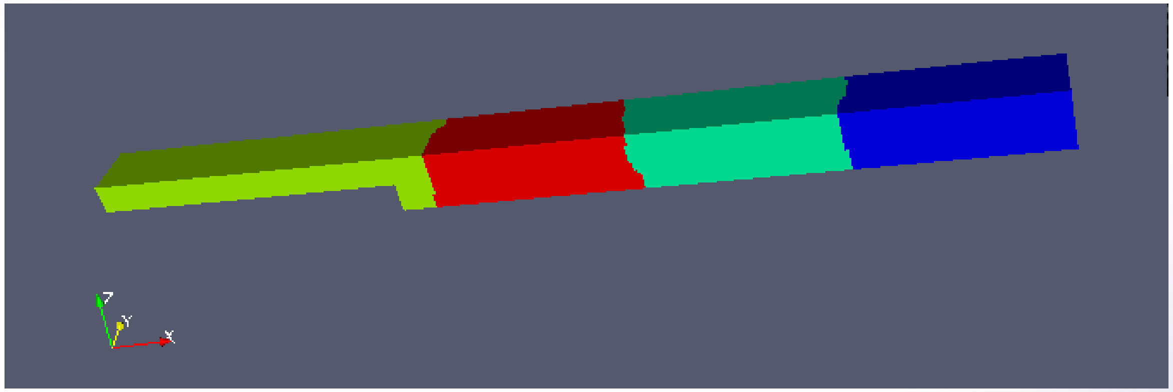
- This project is concerned with developing a parallel version based on an irregular hexahedral grid

- Next step is implementing a tetrahedral cell structure

Parallel Data Decomposition

• For the data decomposition we use calls to the graph partitioner Metis

• Metis produces minimal edge cuts for each partition – thus minimising MPI communications



E.g. Four part decomposition for the backstep case

HECToR

RESEARCH COUNCILS UK

Code description

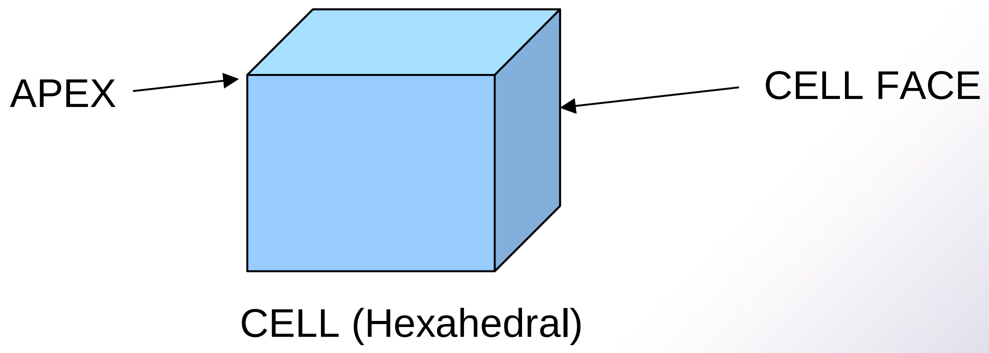
- PHASE1 - conservative predictor step
- VISCOSITY - computation of the cell centred viscous terms
- PHASE2 / MODULE- extrapolation step where the local cell-based characteristic splitting is performed
- BOUND - applying physical boundary conditions for the boundary cells
- PHASE3 - conducting the conservative corrector step

Main loops

- All calculations are local to a cell and it's six nearest neighbour cells

- All main loops involve NSIDE calculations apart from phase1

- Finite volume calculation in phase1 loops over the APEXes



MPI communications

- Irregular decomposition – local numbering on each cpu is non-contiguous

- Global to local mapping for SIDEs,APEXes, CELLs and boundary SIDEs

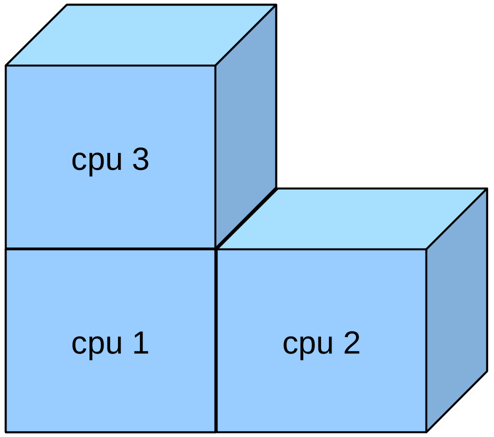
- Decomposition is optimised for the SIDEs

Partition connectivity

- Each cpu stores connectivity between neighbouring cpus and their SIDEs - NEIGH(I)

- Connectivity local to each cpu –

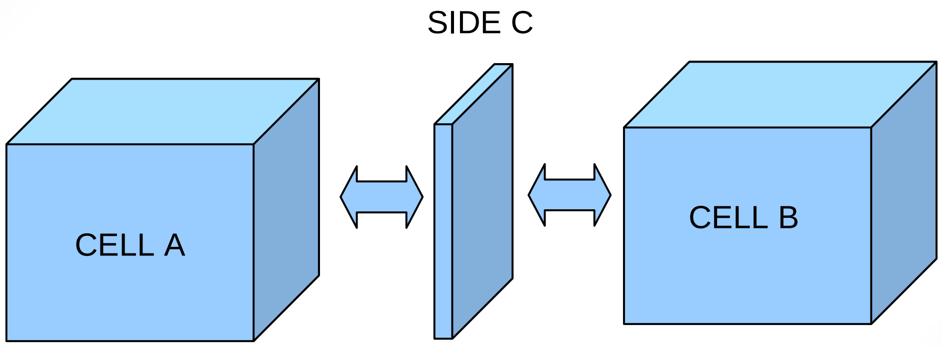
cpu 1 : connect(2:3)=1, cpu 2 : connect(1)=1, cpu 3 : connect(1)=1



SIDE connectivity

- K=SIDELINK(L,I) where K and L are local SIDE numbers and I is the neighbouring partition

- K and L both map to the same global SIDE number



Vectorisation

- All the main loops are SIDE based
  - DO L=1,NSIDE

```
......
IF (GEMSIDECELL(L,1)/=0)...
......
IF (GEMSIDECELL(L,3)==1234)...
......
END DO
```

will not vectorise !

Pointers

- Main arrays for the grid data, flux-type and conservative terms are allocatable
  - REAL(KIND=8), TARGET, ALLOCATABLE :: CELL(:)
  - INTEGER, TARGET,ALLOCATABLE :: GEMCELLSIDE(:)

- Use Fortran90 data types
  - TYPE TRANSFERS
  - INTEGER, POINTER :: INTBLOCK(:)
  - REAL(KIND=DP), POINTER :: REALBLOCK(:)
  - END TYPE TRANSFERS
  - TYPE (TRANSFERS) :: TRANSFER1
  - ALLOCATE(TRANSFER1%INTBLOCK(4*NCELL))
  - ALLOCATE(TRANSFER1%REALBLOCK(4*NCELL))

Transfers with pointers

- Set pointers
  - TRANSFER1%INTBLOCK=>GEMCELLSIDE
  - TRANSFER1%REALBLOCK=>CELL

- Can pass with separate calls
  - CALL MPI_ISSEND(TRANSFER1%INTBLOCK,..
  - CALL MPI_ISSEND(TRANSFER1%REALBLOCK,..

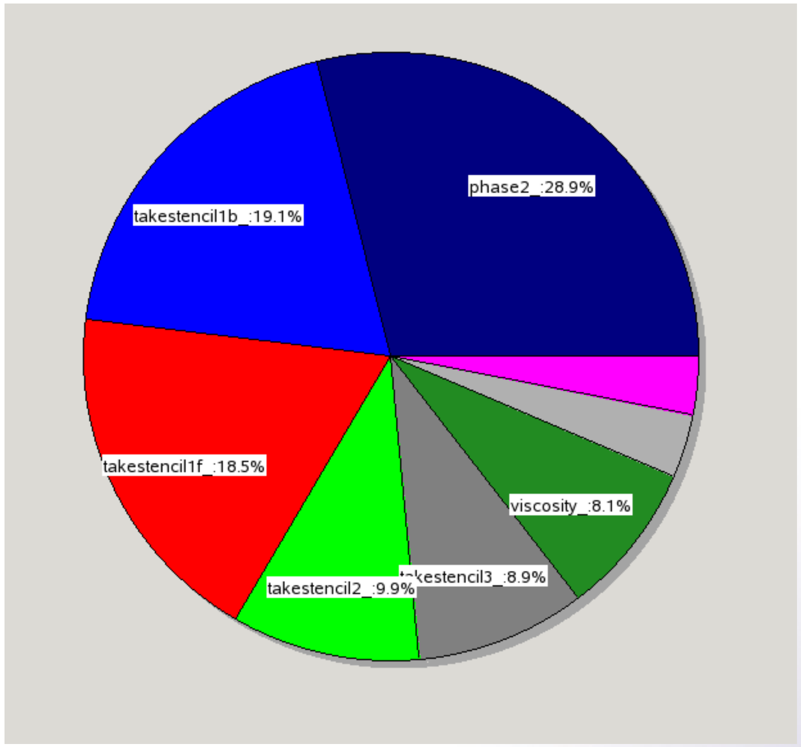
- Why not send the TRANSFER data type as an MPI_TYPE
  - CALL MPI_TYPE_CREATE_STRUCT((2, BLOCKCOUNTS, OFFSETS, OLDTYPES,TRANSFERTYPE, IERR)
  - CALL MPI_TYPE_COMMIT(TRANSFERTYPE, IERR)

Initial performance

- Test initial parallelisation with the following HECToR compilers and optimisation flags
  - Cray FFLAGS = -O3 -Oaggress -Omsgs
  - Pgf90 FFLAGS = -Minfo -Mneginfo -Mextend -fast -Munroll=n:4 -Mipa=fast,inline -O3 -tp barcelona-64
  - Pathscale FFLAGS = -Ofast -LNO:full_unroll=4 -march=barcelona -OPT:malloc_algorithm=1 -LNO:simd_verbose=ON
  - Gfortran FFLAGS = -march=barcelona -ffast-math -funroll-loops -O3 -ffixed-line-length-72 -ftree-vectorizer-verbose=2
- See http://www.hector.ac.uk/cse/reports/compilers.php for compiler performance results on a variety of other codes

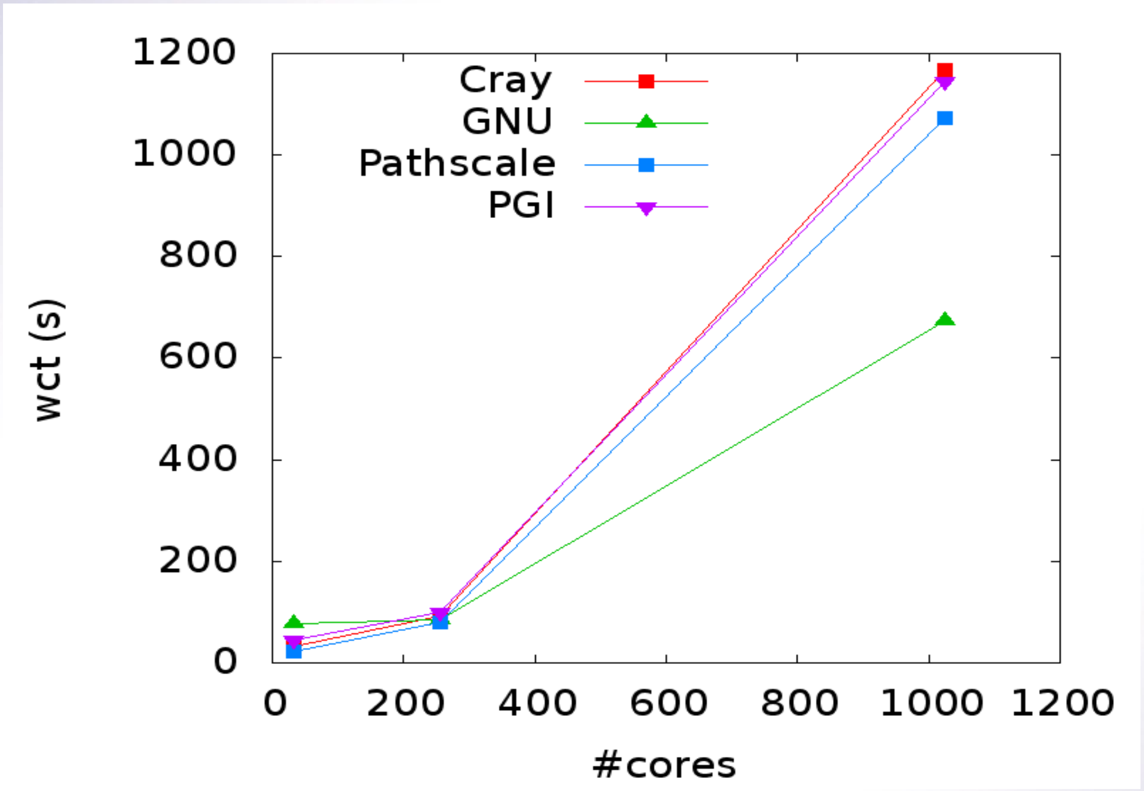
What takes most time?

One iteration of CABARET algorithm



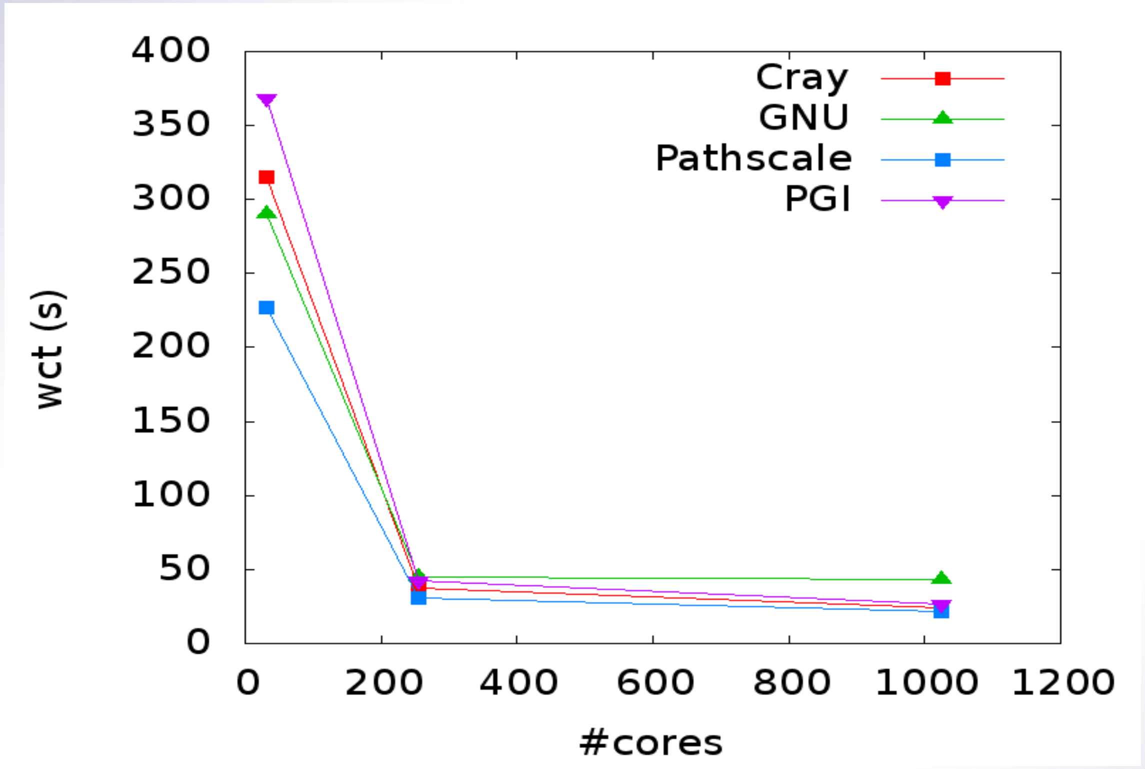
Performance

Initialisation



Backward facing step case (fixed problem size) with NAPEX=111741 NCELL =100000 NSIDE =311400 270 iterations

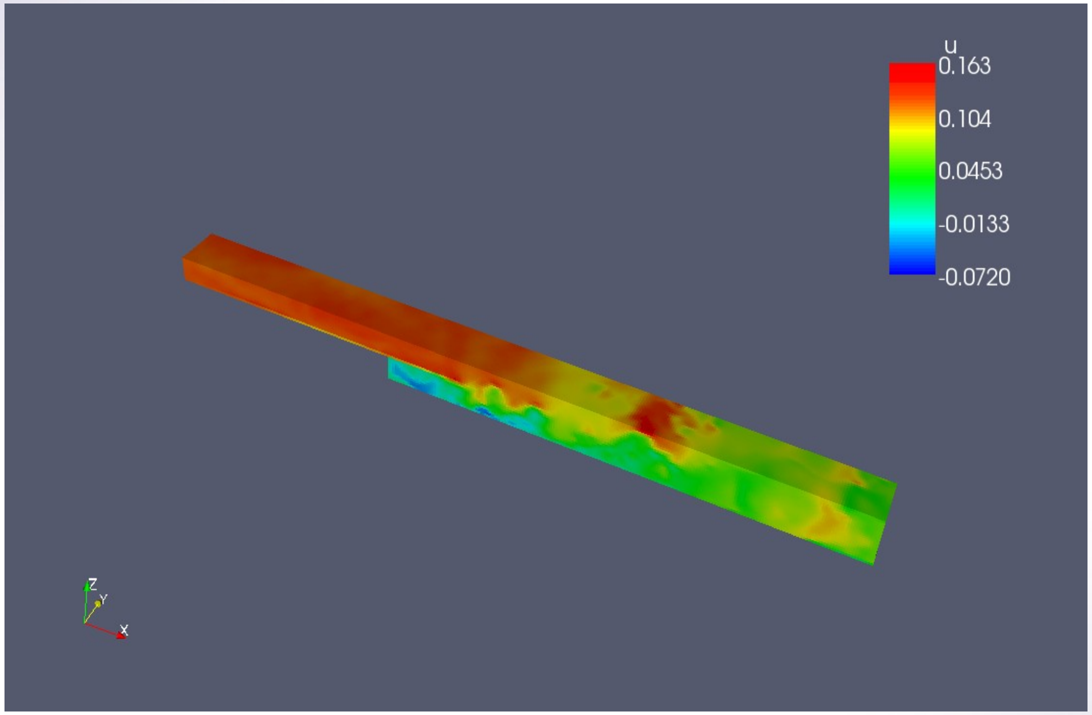
Performance



Backward facing step case (fixed problem size) with NAPEX=111741 NCELL =100000 NSIDE =311400 270 iterations

3-D backward facing step

Paraview plot of x component of velocity



Backward facing step case with NAPEX=111741 NCELL =100000 NSIDE =311400 mature solution

Conclusion

- CABARET method up to 30 times more efficient for some CFD applications - on coarser grids

- For Re~10000 and subsonic M - method gives good convergence without preconditioning

- Parallel code uses irregular domain decomposition

- Main loops will not vectorise – further work!