UNIVERSITY OF CAMBRIDGE

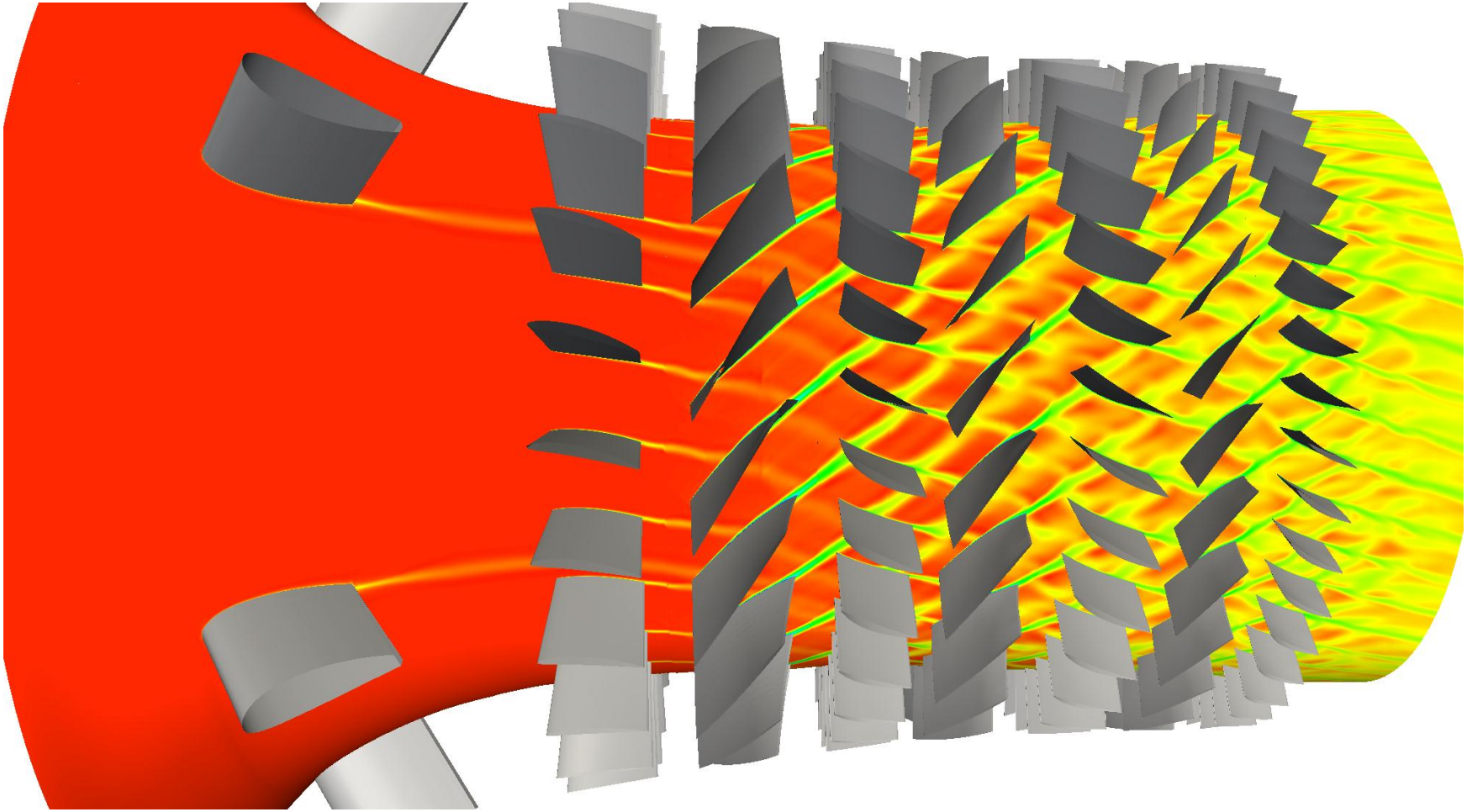
An Accelerated Navier-Stokes Solver for Flows in Turbomachines

Tobias Brandvik

Whittle Laboratory

University of Cambridge

A large-scale simulation



Overview

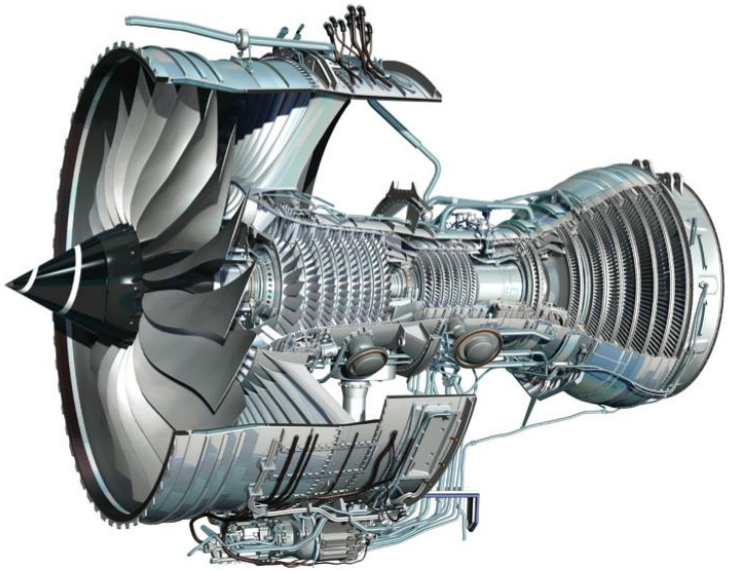
- PART I: Turbomachinery
- PART II: Stencil-based PDE solvers on multi-core
- PART III: A new solver for turbomachinery flows

UNIVERSITY OF CAMBRIDGE

PART I: Turbomachinery

Turbomachinery

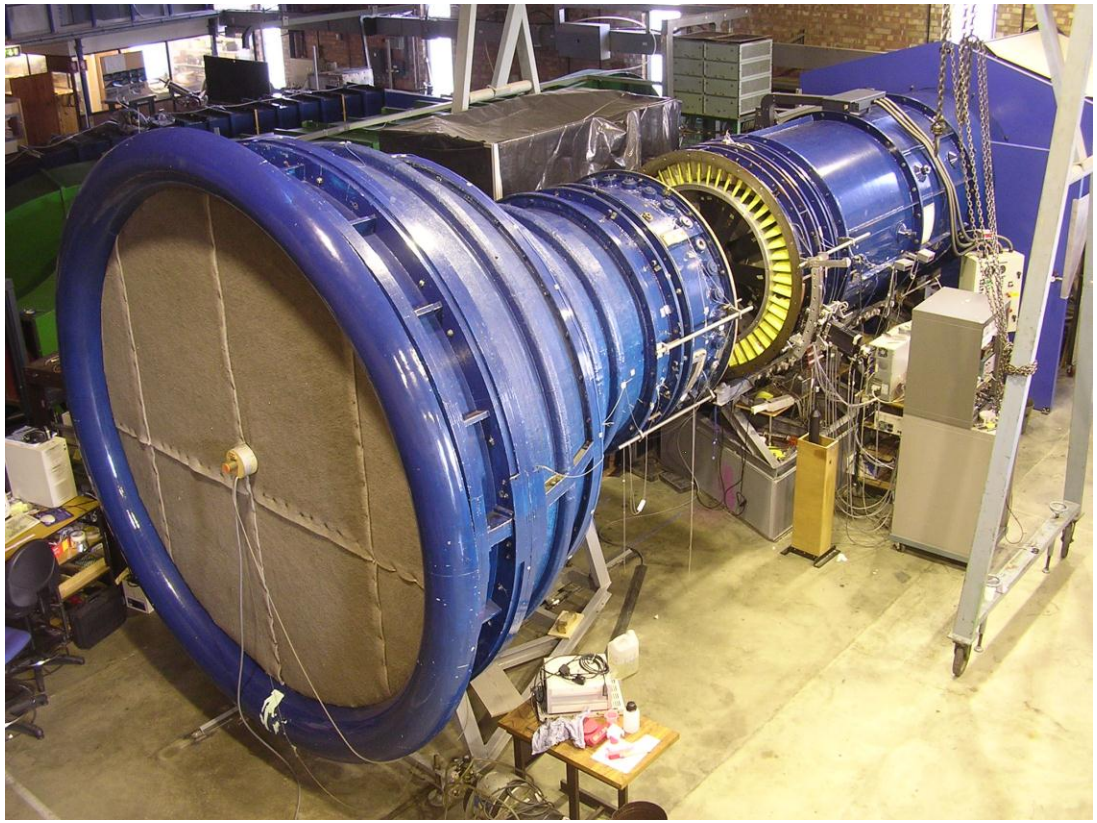
- Thousands of blades, arranged in rows
- Each row has a bespoke blade profile designed with CFD

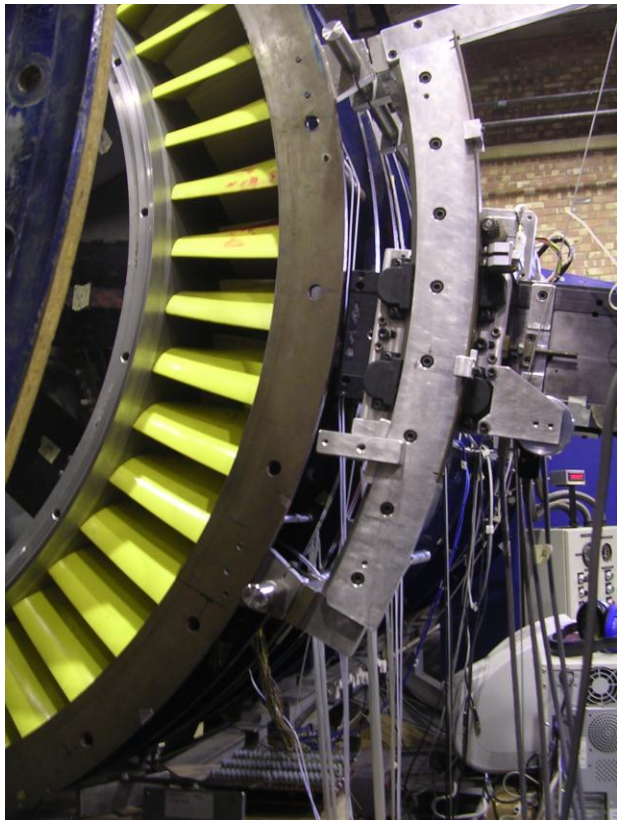


Research problem - surge prediction

Research problem - surge prediction

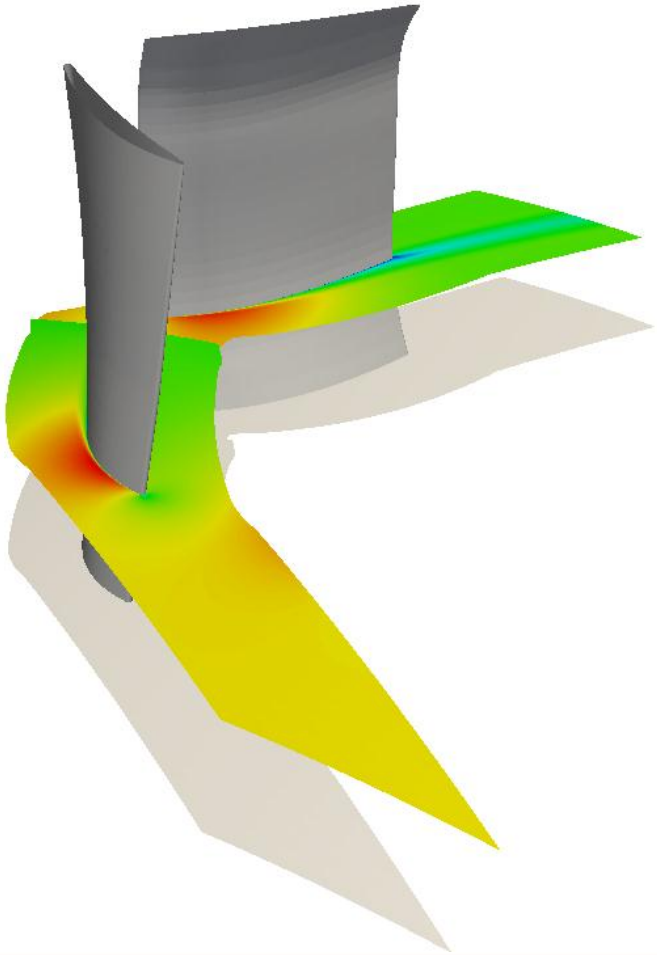
Deverson low-speed compressor rig





Deverson simulation

- Typical routine simulation
- Structured grid, steady state
- 3 million grid nodes
- 8 hours on four CPU cores
- 30 minutes on a GPU



PART II: Stencil-based PDE solvers

Aim

To produce an order of magnitude reduction in the run-time of CFD solvers for the same hardware cost

Aim

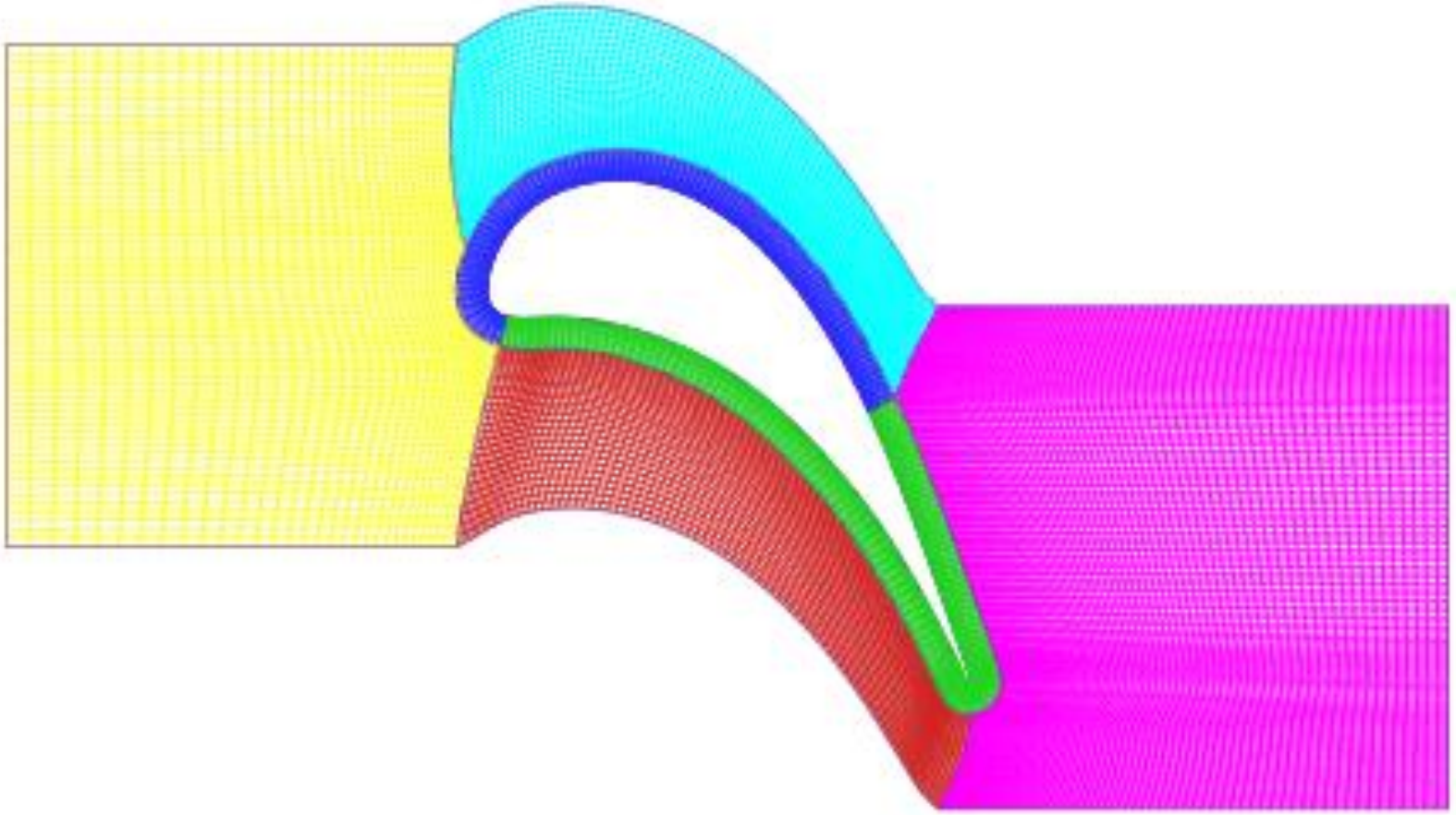
To produce an order of magnitude reduction in the run-time of CFD solvers for the same hardware cost

And maintain performance portability across current and **future** multi-core processors

Structured grids

- Our work focuses on multi-block structured grids
- Structured grid solvers: a series of stencil operations
- Stencil operations: discrete approximations of the equations

Solving PDEs on multi-core processors



Solving PDEs on multi-core processors

Navier-Stokes

Solving PDEs on multi-core processors

Navier-Stokes

Finite volume, explicit time stepping

Solving PDEs on multi-core processors

Navier-Stokes

Finite volume, explicit time stepping

Second order central differences, Scree scheme

Solving PDEs on multi-core processors

Navier-Stokes

Finite volume, explicit time stepping

Second order central differences, Scree scheme

Set flux, Sum flux, Shear Stress ...

Solving PDEs on multi-core processors

Navier-Stokes

Finite volume, explicit time stepping

Second order central differences, Scree scheme

Set flux, Sum flux, Shear Stress ...

Multigrid, Mixing plane, Sliding plane ...

Solving PDEs on multi-core processors

Navier-Stokes

Finite volume, explicit time stepping

Second order central differences, Scree scheme

Stencil operations

Set flux, Sum flux, Shear Stress ...

Multigrid, Mixing plane, Sliding plane ...

Solving PDEs on multi-core processors

Navier-Stokes

Finite volume, explicit time stepping

Second order central differences, Scree scheme

Stencil operations

Set flux, Sum flux, Shear Stress ...

Non-stencil operations

Multigrid, Mixing plane, Sliding plane ...

Solving PDEs on multi-core processors

Navier-Stokes

Finite volume, explicit time stepping

Second order central differences, Scree scheme

Stencil operations — Set flux, Sum flux, Shear Stress ... — **90% of run-time**

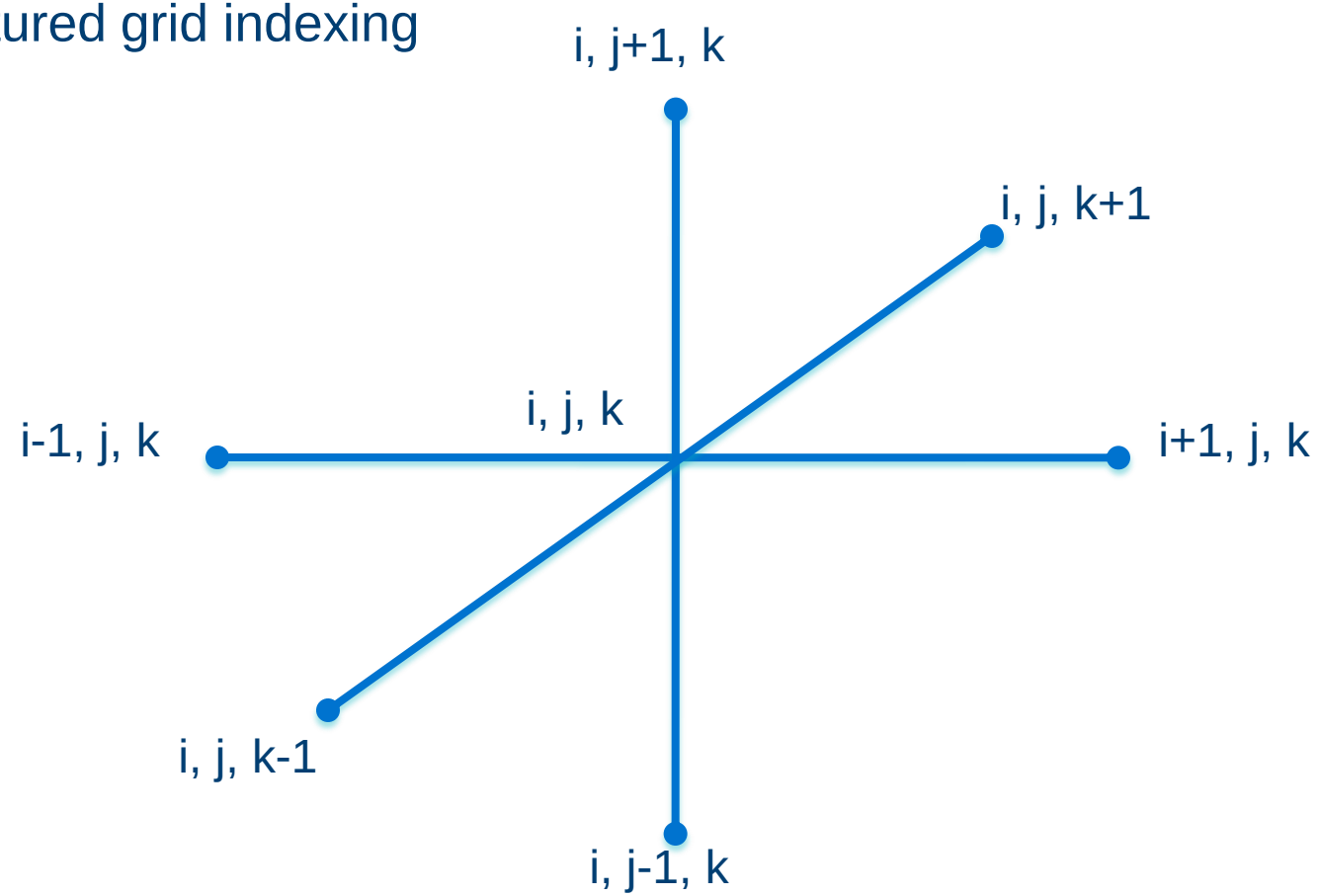
Non-stencil operations — Multigrid, Mixing plane, Sliding plane ... — **10% of run-time**

Stencil operations on multi-core processors

- Single implementation?
- Multiple implementations?
- Alternative:
  - High level language for stencil operations
  - Source-to-source compilation

Stencil example

- Structured grid indexing



Stencil example

- $\frac{\partial^2 u}{\partial x^2}$ in Fortran

```
DO K=2,NK-1
  DO J=2,NJ-1
    DO I=2,NI-1
      D2UDX2(I,J,K) = (U(I+1,J,K) - 2.0*U(I,J,K) +
     &                 U(I-1,J,K))/(DX*DX)
    END DO
  END DO
END DO
```

Stencil example

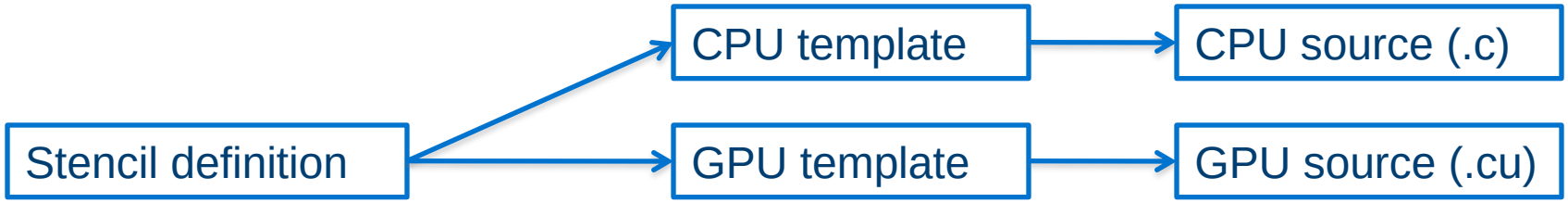
- Stencil definition:

```
input_scalars = ["dx"]
input_arrays = ["u"]
output_arrays = ["d2udx2"]

inner_calc = [
{"lvalue": "d2udx2"
 "rvalue": """u[1][0][0] - 2.0f*u[0][0][0] +
             u[-1][0][0])/(dx*dx)"""
]
```

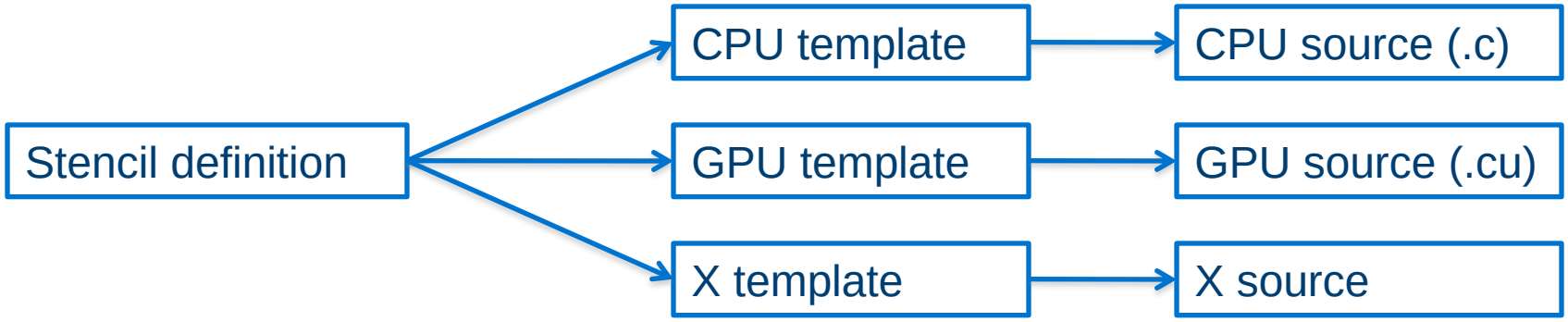
Source-to-source compilation

- The stencil definition is transformed at compile-time into code that can run on the chosen processor
- The transformation is performed by filling in a pre-defined template using the stencil definition



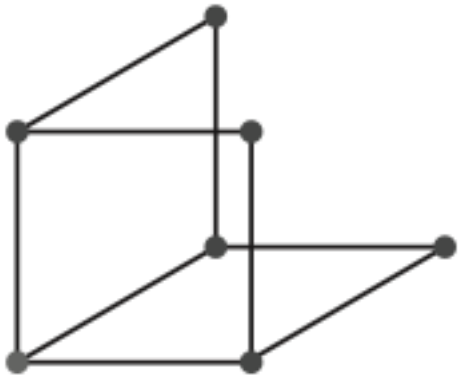
Source-to-source compilation

- The stencil definition is transformed at compile-time into code that can run on the chosen processor
- The transformation is performed by filling in a pre-defined template using the stencil definition

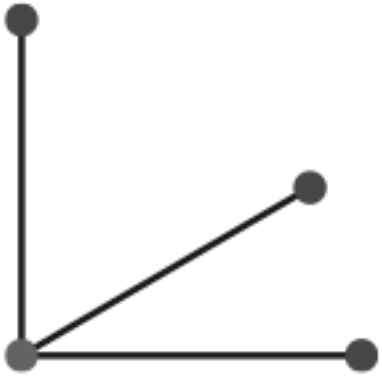


Stencil example

- Other stencils are in principle the same



Set fluxes



Sum fluxes

Smoothing

Implementation details

- There are many optimisation strategies for stencil operations (see paper from Supercomputing 2008 by Datta et al.)
- CPUs:
  - Parallelise with pthreads
  - SSE vectorisation
- GPUs:
  - Cyclic queues

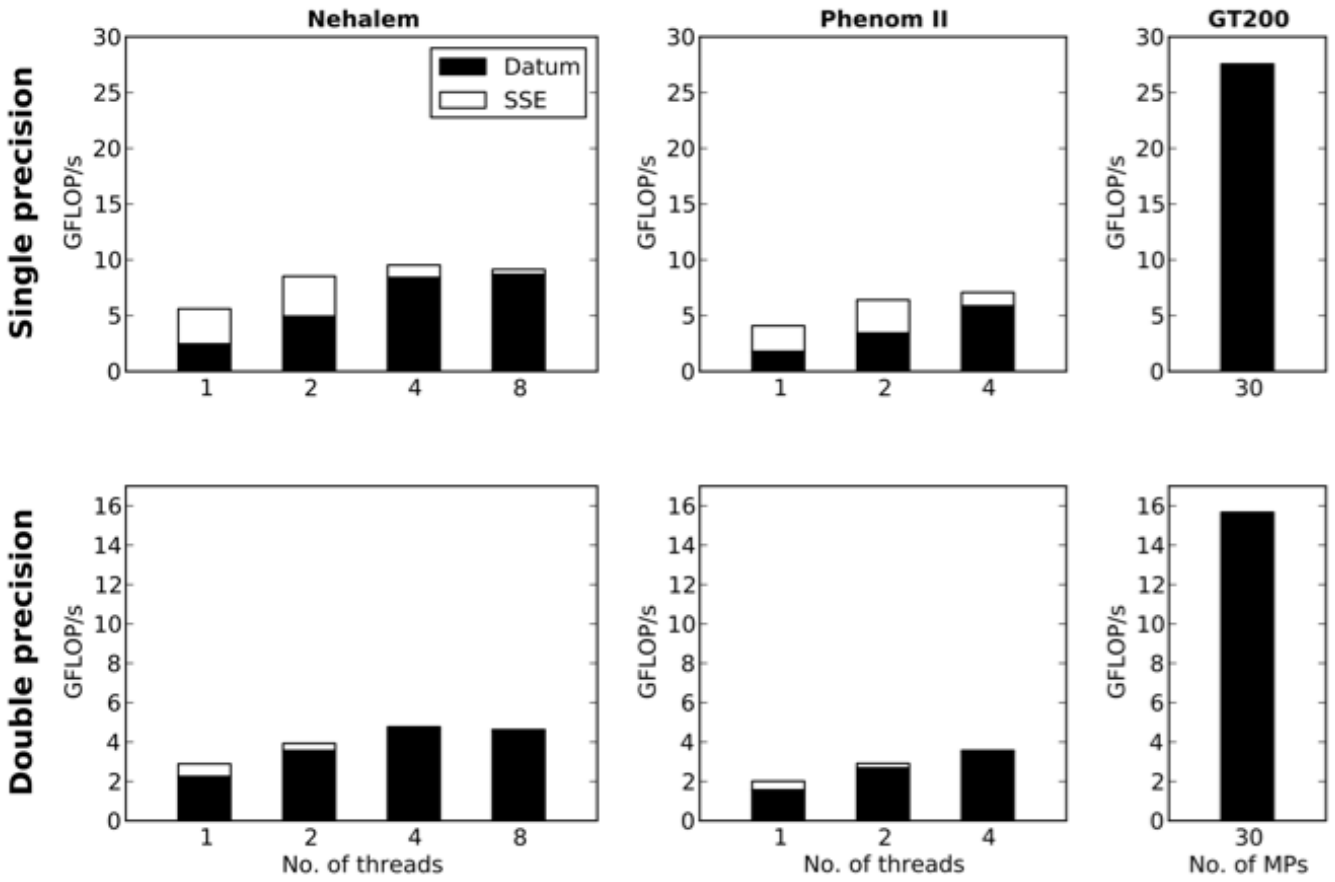
Software framework

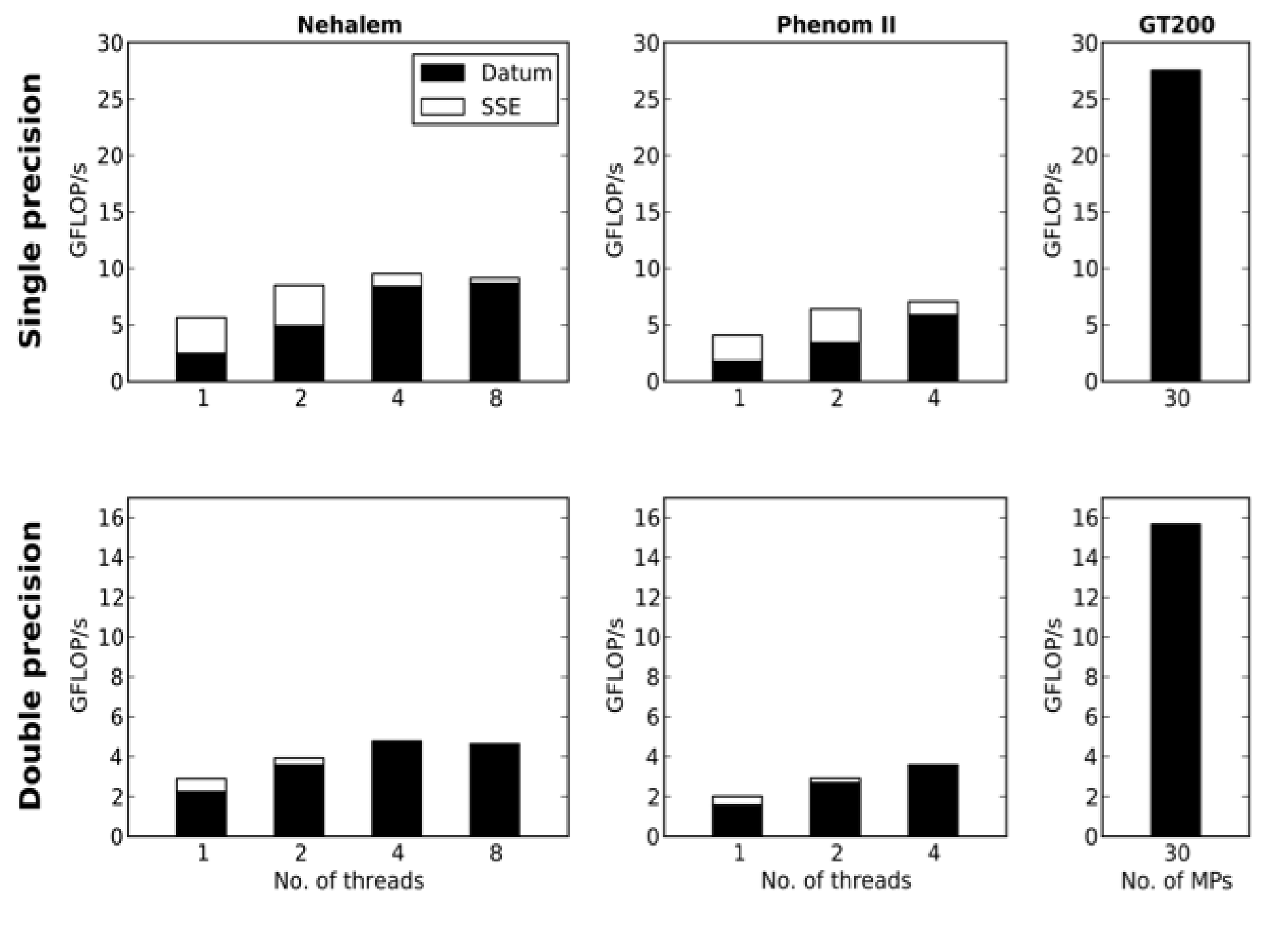
- Framework: Templates + Run-time library
- Library provides:
  - Uniform API for both CPUs and GPUs
  - MPI
  - Reductions
  - Parallel sparse matrix vector multiplication

Testbed

- CPU 1: Intel Core i7 920 (2.66 GHZ)
- CPU 2: AMD Phenom II X4 940 (3.0 GHz)
- GPU: NVIDIA GTX 280

Stencil benchmark




Nehalem
Phenom II
GT200
Single precision
Double precision
Datum
SSE
GFLOP/s
No. of threads
No. of MPs

UNIVERSITY OF CAMBRIDGE

PART III: A new solver for turbomachinery flows

Turbostream

- We have implemented a new solver that can run on both CPUs and GPUs
- The starting point was an existing solver called TBLOCK
- The new solver is called Turbostream

TBLOCK

- Developed by John Denton
- Blocks with arbitrary patch interfaces
- Simple and fast algorithm
- 15,000 lines of Fortran 77
- Main solver routines are only 5000 lines

Denton Codes

- TBLOCK is the latest of the “Denton Codes”
- Dates back to the late 1970s
- Used for some part of the design process at most turbomachinery manufacturers

Turbostream

- 3000 lines of stencil definitions (~15 different stencil kernels)
- Code generated from stencil definitions is 15,000 lines
- Additional 5000 lines of C for boundary conditions, file I/O etc.
- Source code is very similar to TBLOCK – every subroutine has an equivalent stencil definition

Turbostream performance

Processor	TBLOCK performance	Turbostream performance
Nehalem	1.21	1.48
Phenom II	1	0.89
GT200	-	10.2

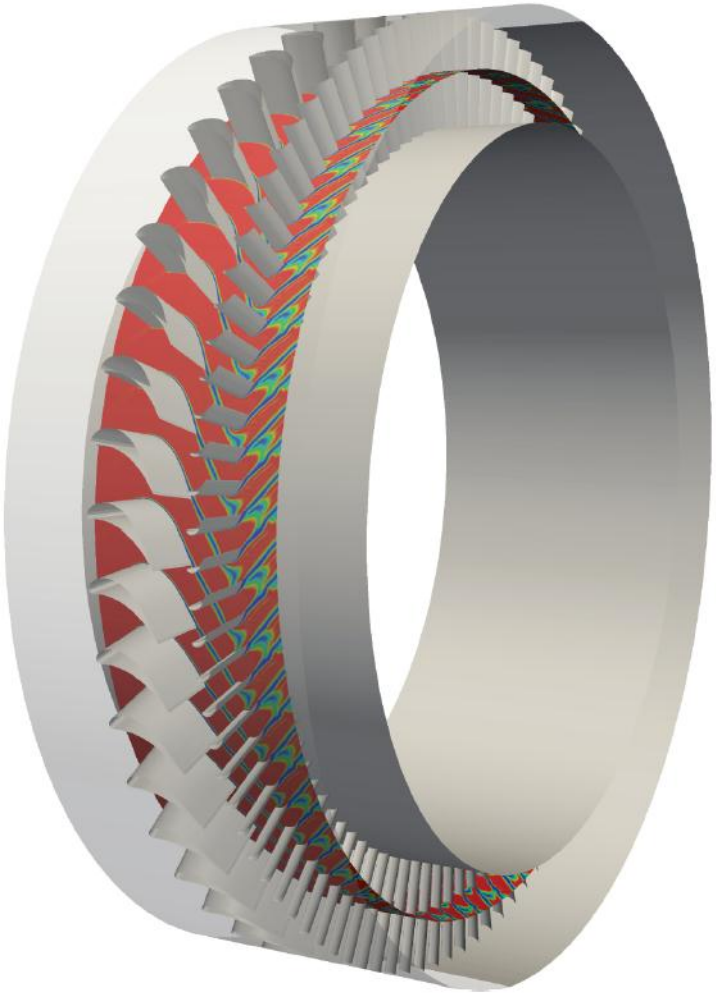
- TBLOCK runs in parallel on all four CPU cores using MPI
- Initial Fermi results are 2x faster – larger shared memory

Multi-processor performance

- University of Cambridge became a NVIDIA CCoE in December 2008
- NVIDIA donated 32 S1070s which were added to existing “Darwin” supercomputer
- Nehalem-based CPU nodes from Dell
- QDR Infiniband

Multi-processor performance

- Benchmark case is an unsteady simulation of a turbine stage

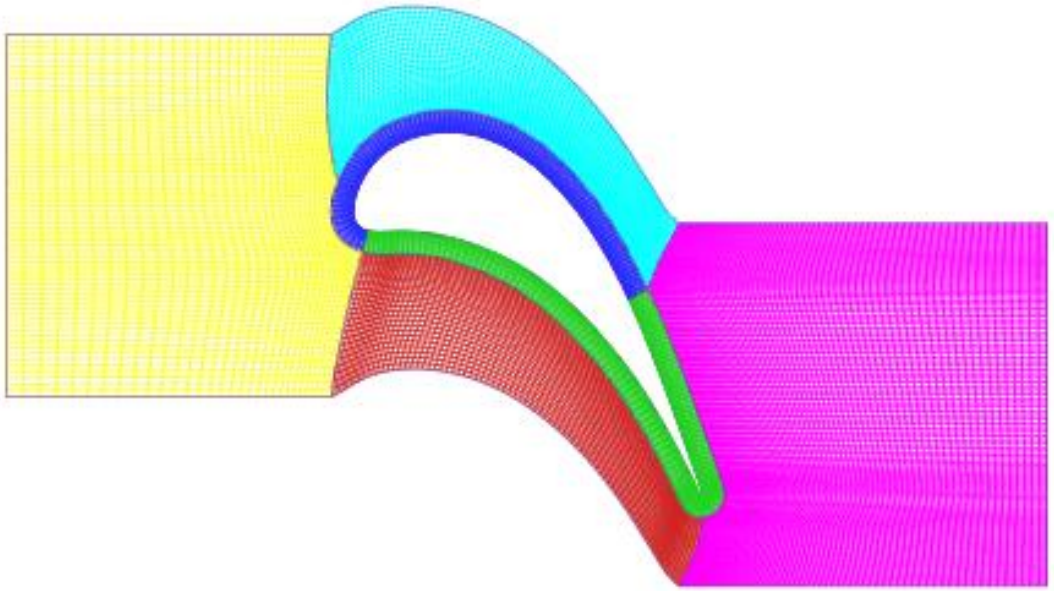


Multi-processor implementation

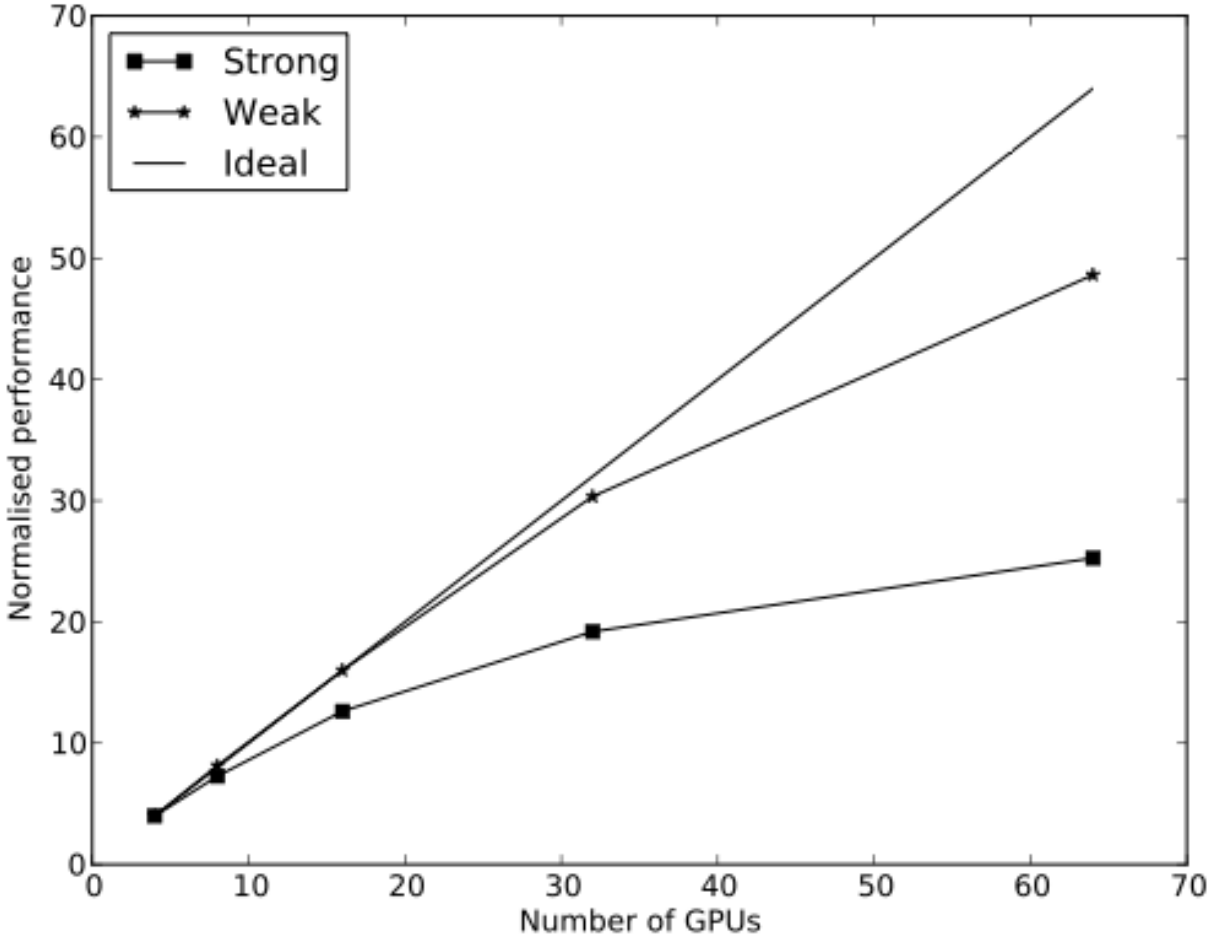
1 Stencil kernels

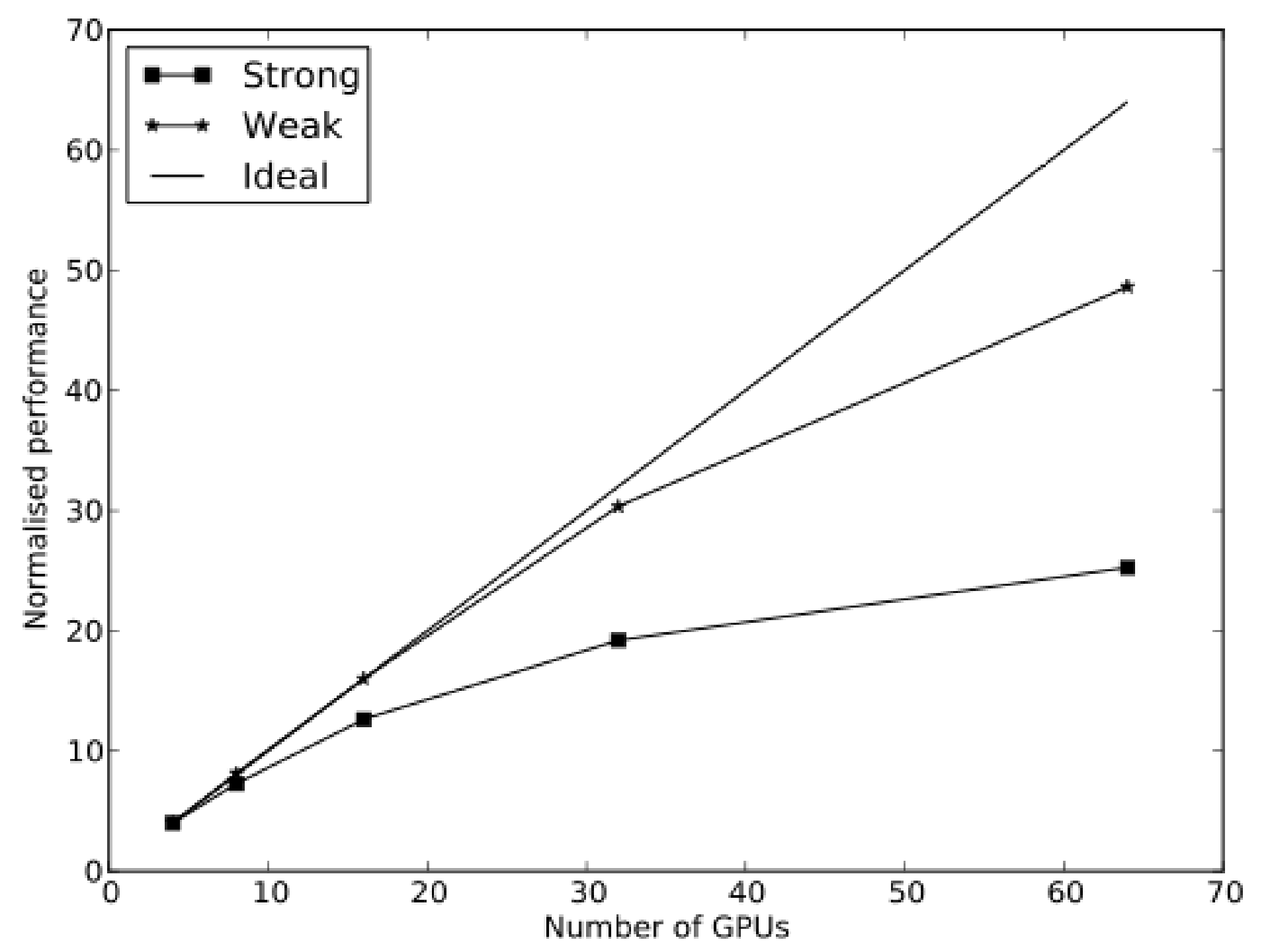
2 Boundary conditions

3 Exchange halo nodes (MPI)



Multi-processor performance




Strong
Weak
Ideal
Normalised performance
Number of GPUs
0
10
20
30
40
50
60
70

Conclusions

- The switch to multi-core processors enables a step change in performance, but existing codes have to be rewritten
- The differences between processors make it difficult to hand-code a solver that will run on all of them
- We suggest a high level abstraction coupled with source-to-source compilation

Conclusions

- A new solver called Turbostream, which is based on Denton's TBLOCK, has been implemented
- Turbostream is ~10 times faster than TBLOCK when running on an NVIDIA GPU as compared to a quad-core Intel or AMD CPUs
- Single blade-row calculations almost interactive on a desktop (10 – 30 seconds)
- Multi-stage calculations in a few minutes on a small cluster ($10,000)
- Full annulus unsteady calculations complete overnight on a modest cluster ($100,000)

More information

- www.turbostream-cfd.com
- SBLOCK: A Framework for Efficient Stencil-Based PDE Solvers on Multi-core Platforms
- An Accelerated 3D Navier-Stokes Solver for Flows in Turbomachines