Turbostream: A CFD solver for many-core processors

Tobias Brandvik

Whittle Laboratory, University of Cambridge

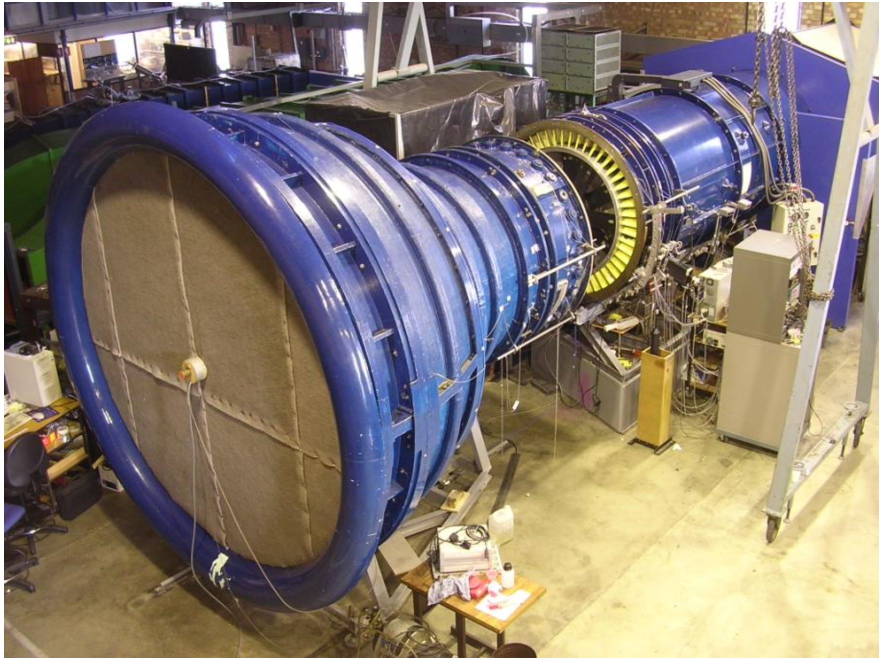
Aim

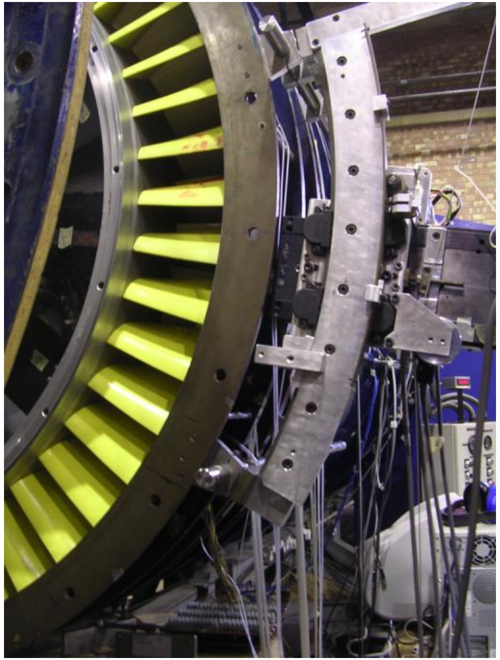
To produce an order of magnitude reduction in the run-time of CFD solvers for the same hardware cost

The Whittle Laboratory

- Turbomachinery research laboratory
- Does both experimental and computational work

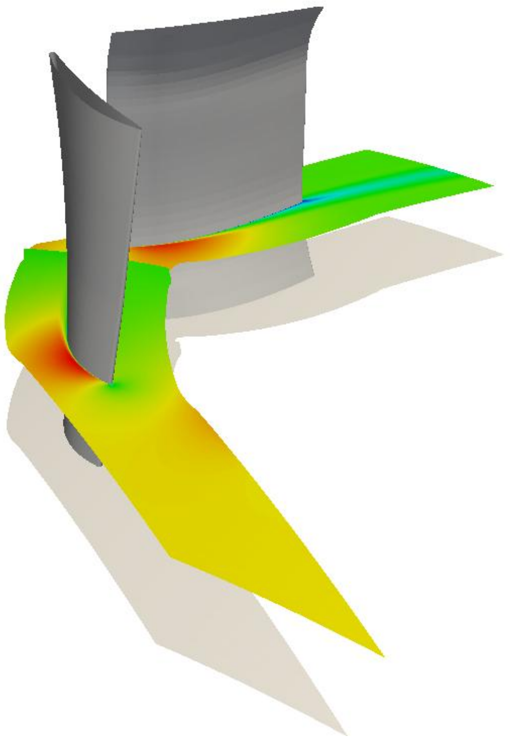
Deverson low-speed compressor rig

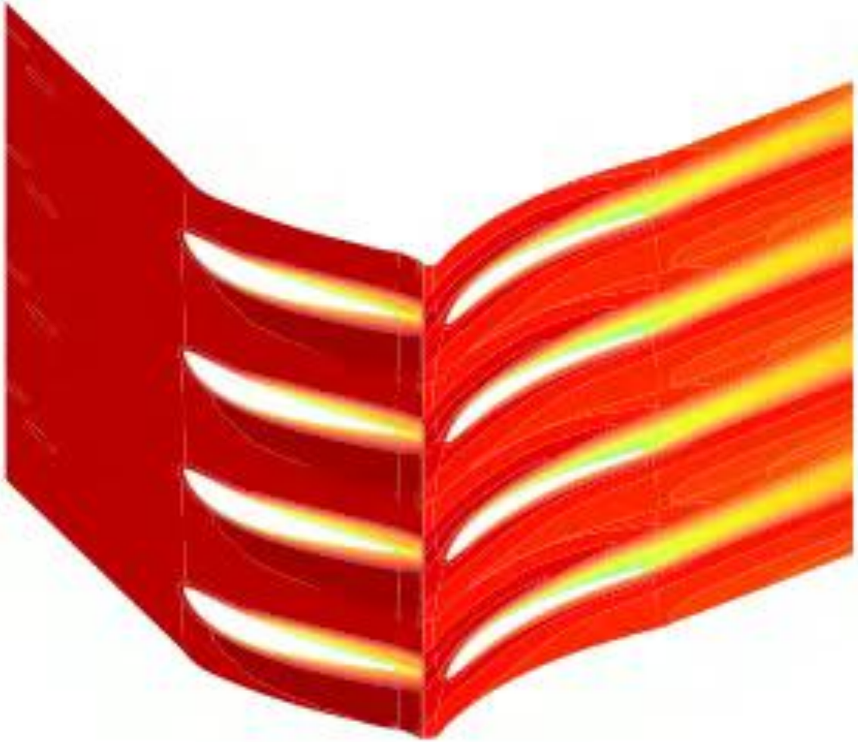




Deverson simulation

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





Structured grids

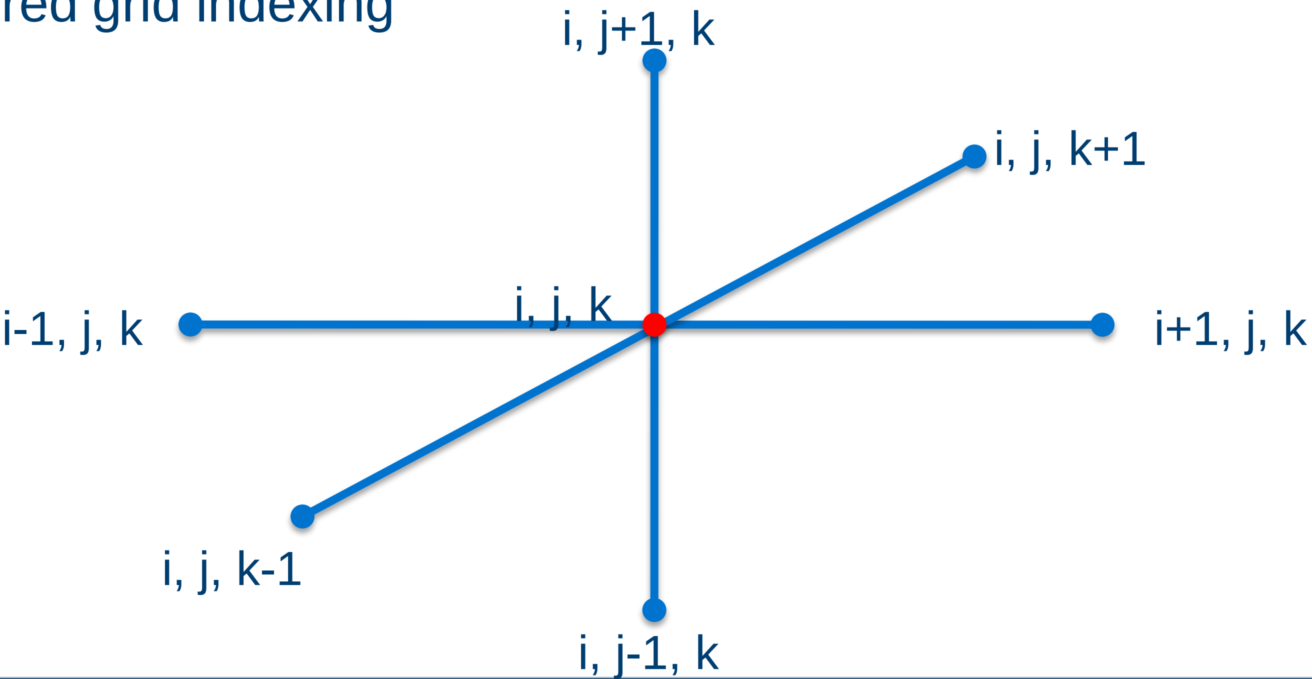
- The processor landscape is rapidly changing, but CFD codes can have a life-span of 30 years
- Our work focuses on structured grids
- Structured grid solvers: a series of stencil operations
- Stencil operations: discrete approximations of the equations

Structured grids on many-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

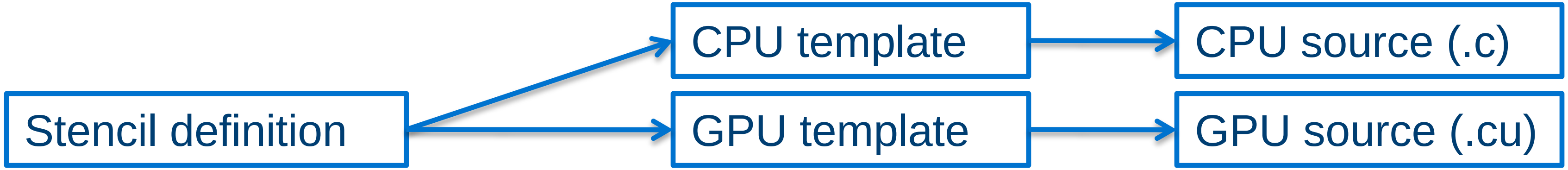
- Stencil definition:

```
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

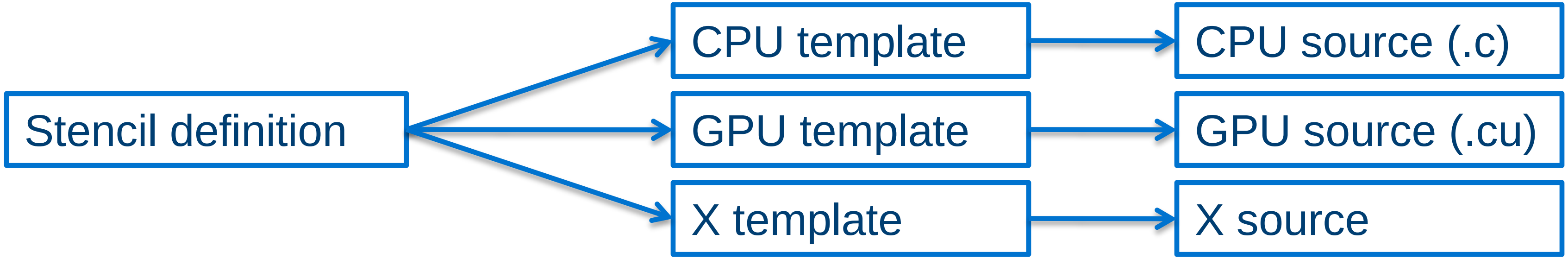
- The stencil definition is transformed at compile-time into code that can run on the chosen processor
- The transformation is performed by filling in a pre-defined template using the stencil definition



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

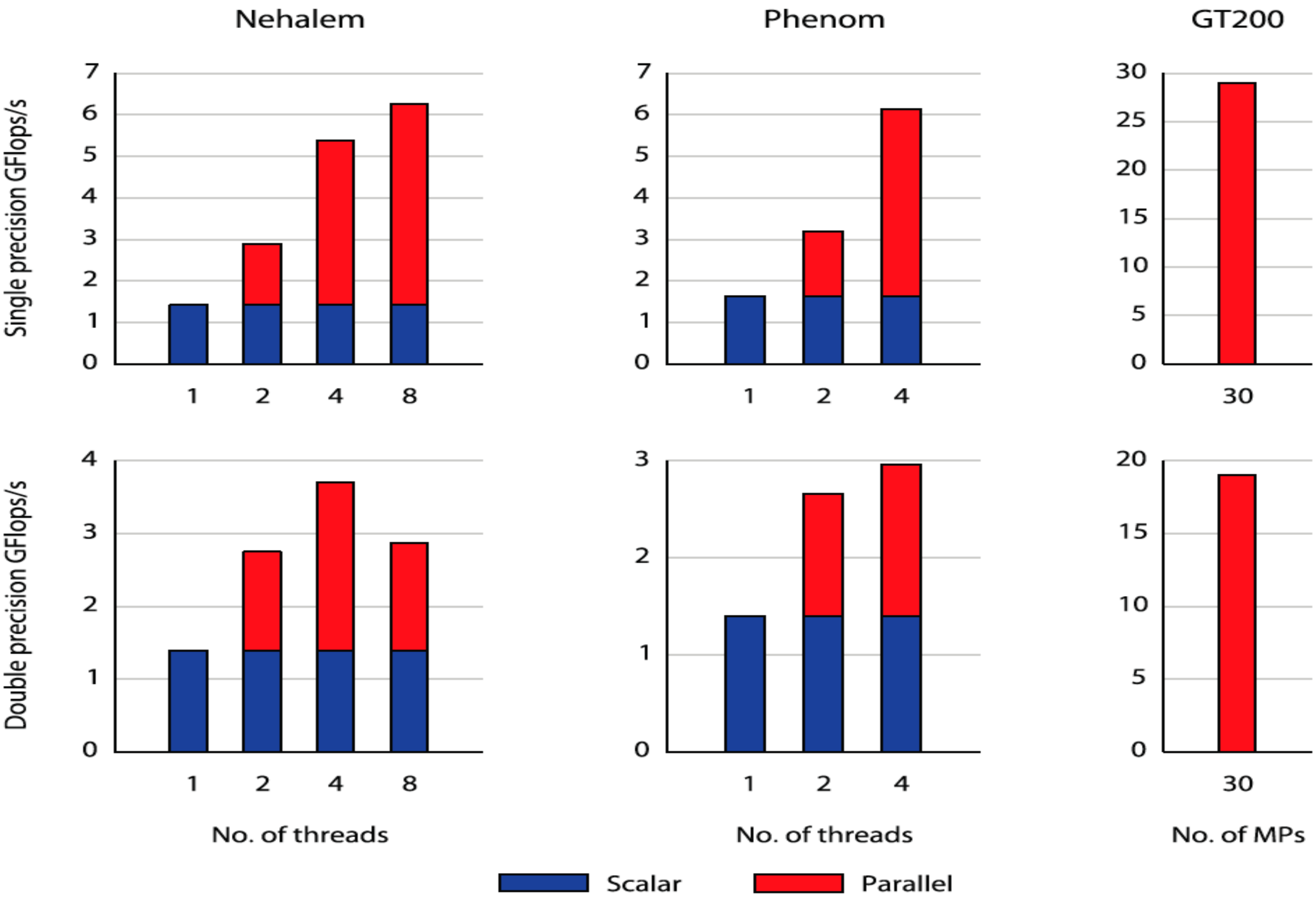


Implementation details

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

Performance results for simple kernel

- Intel Nehalem 2.66 GHz
- AMD Phenom II 3.0 GHz
- NVIDIA GT200 (Quadro Fx 5800)
- Heat conduction benchmark kernel


Nehalem
Phenom
GT200
Single precision GFlops/s
Double precision GFlops/s
7
6
5
4
3
2
1
0
30
25
20
15
10
5
0
1
2
4
8
30
No. of threads
No. of MPs
Scalar
Parallel

The new solver

- 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

Capabilities

- Explicit scheme
- Variable time steps and multi-grid for convergence acceleration
- Time-accurate solutions using Jameson’s Dual Time Stepping procedure
- Multi-stage calculations using mixing planes or sliding planes

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

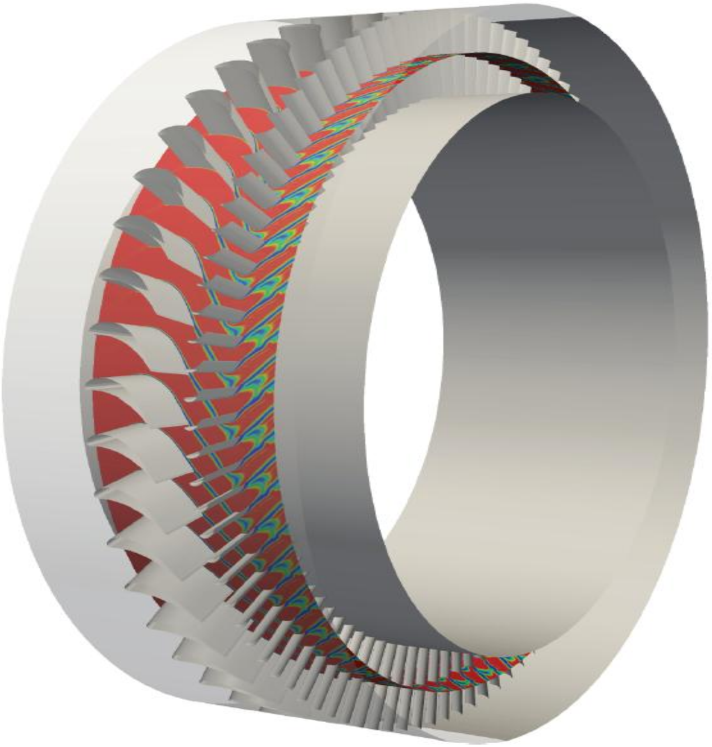
Single-processor performance

Solver	Processor	Time/node/step
TBLOCK	Intel Xeon 2.33 GHz	$5.1 \cdot 10^{-7}$ s
Turbostream	NVIDIA GT200	$2.7 \cdot 10^{-8}$ s

- TBLOCK uses all four cores on the CPU through MPI
- Turbostream is ~20 times faster

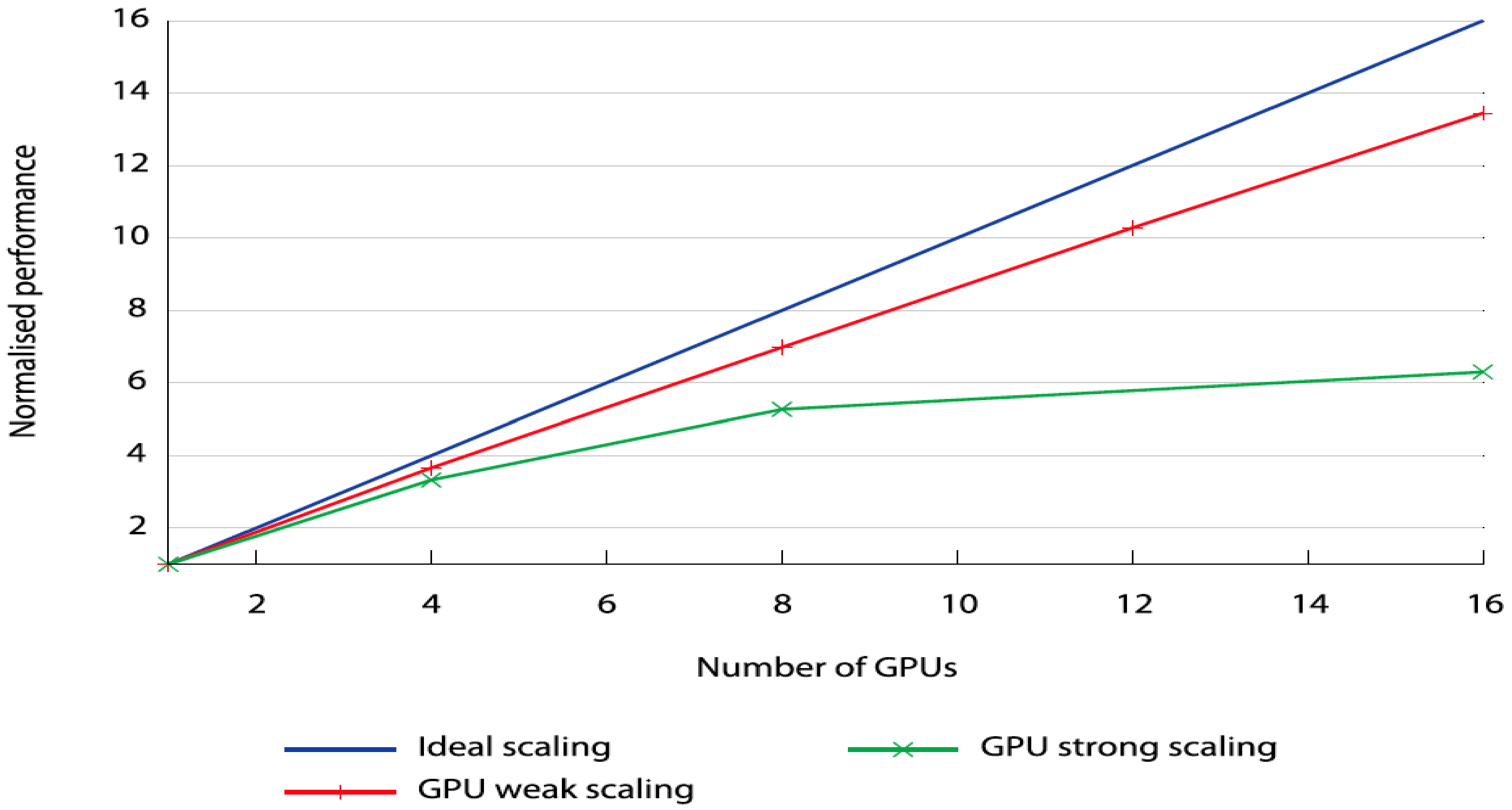
Multi-processor performance

- Benchmark case is an unsteady simulation of a turbine stage



Multi-processor performance

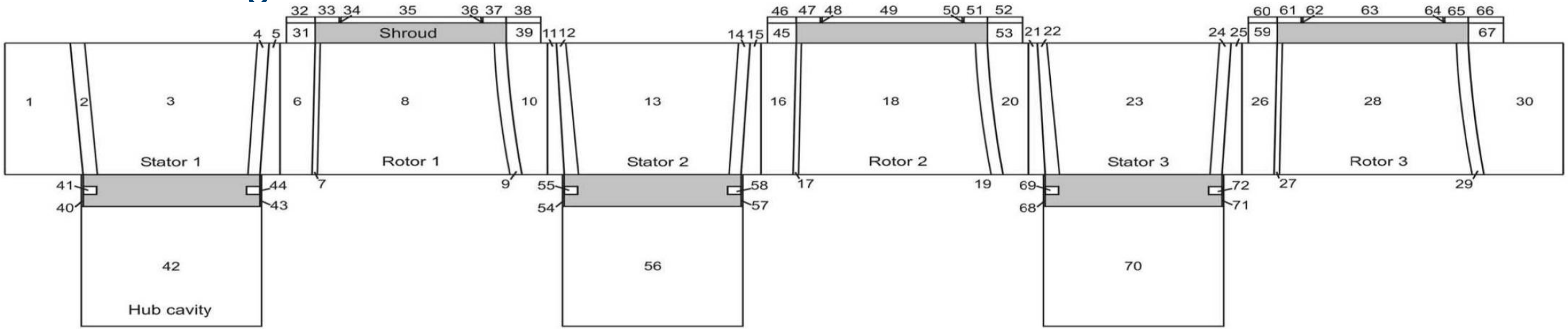
- 16 NVIDIA G200 GPUs, 1 Gb/s Ethernet
- Weak scaling: 6 million grid nodes **per GPU**
- Strong scaling: 6 million grid nodes **in total**



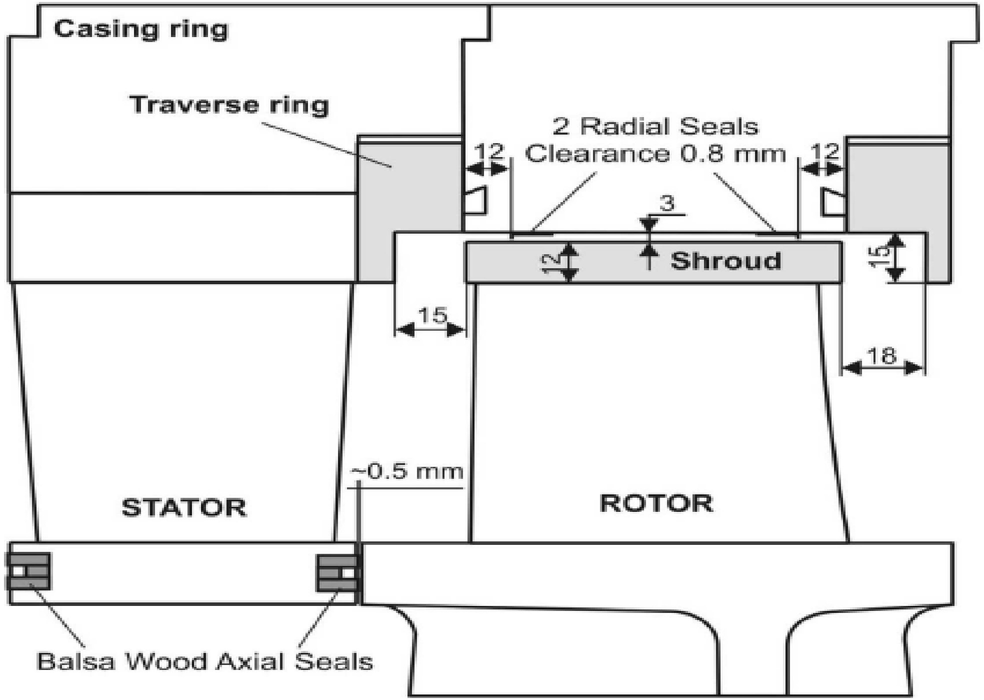
16
14
12
10
8
6
4
2
Normalised performance
2
4
6
8
10
12
14
16
Number of GPUs
Ideal scaling
GPU weak scaling
GPU strong scaling

Test case

- Three-stage low-speed turbine case from Rosic et al. (2006)
- Used to demonstrate the importance of hub and shroud leakages in multi-stage turbines

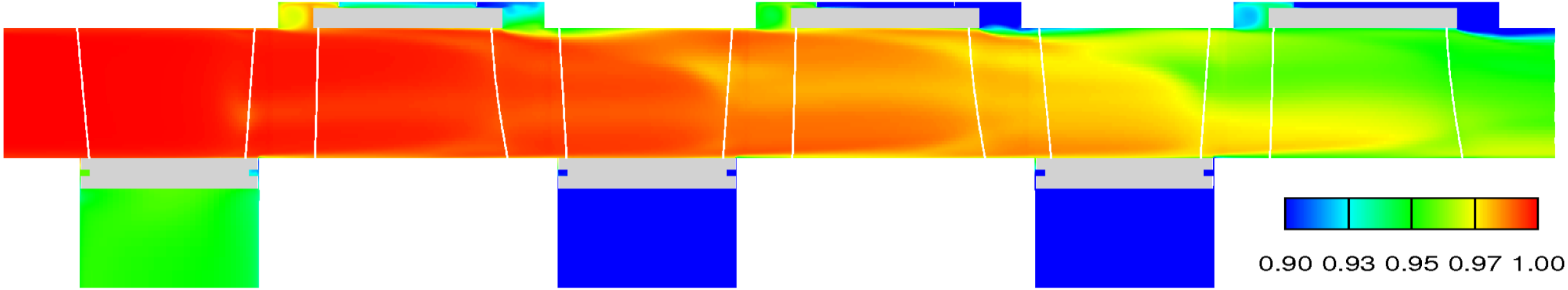


Single stage geometry



Results

Pitch-wise averaged entropy function:

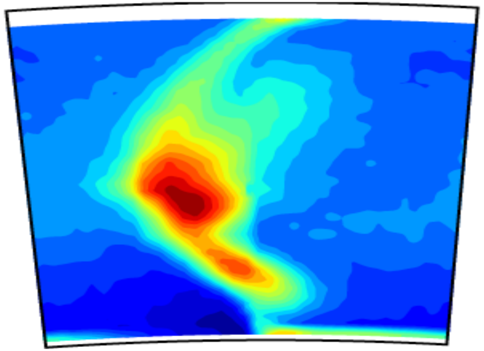


- Steady calculation with mixing planes

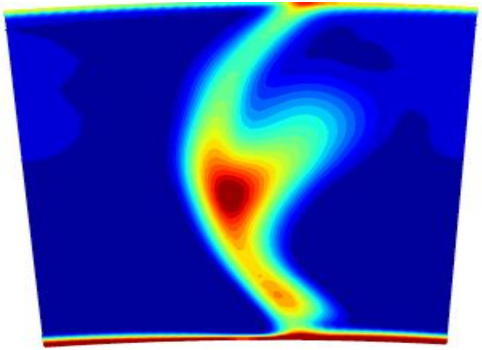
Stator 3 exit

C_{po} contours, stator 3

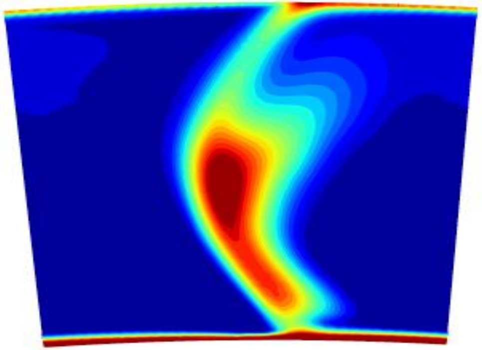
Experiment



TBLOCK



Turbostream



Conclusions

- The switch to many-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 ~20 times faster than TBLOCK when running on an NVIDIA GPU as compared to a quad-core Intel CPU
- 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 URANS completes overnight on a modest cluster