

Acceleration of a 3D flow solver

Tobias Brandvik

PhD Student

Whittle Laboratory, Cambridge University

Background

Background

- Turbomachinery experiments are expensive
- CFD is used to explore design space
- Typical runtimes:
 - A few hours on a workstation (coarse, single blade)
 - Many days on a cluster (detailed,

Background

- Investigating many-core architectures
 - ClearSpeed, GPUs, Cell
- Promising initial results on GPUs

Status

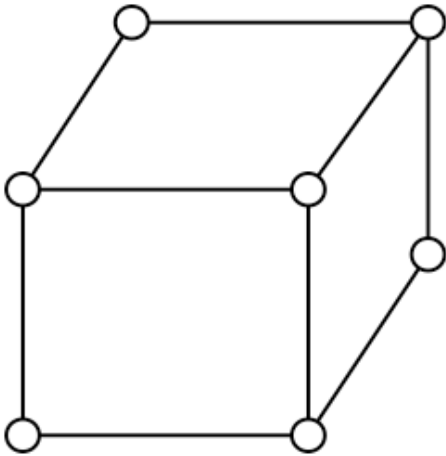
- Simplified version of widely used flow solver has been ported to one chip
- 2X speed up compared to single CPU core

CFD overview

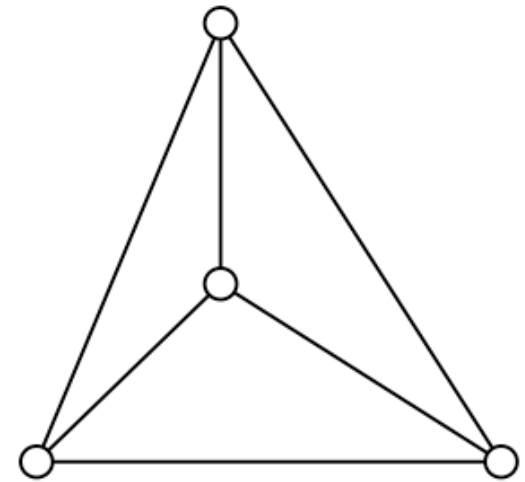
- Discretise surface geometry
- Create volume mesh
- Calculate flow
- Visualisation

Grids

- Two main grid types:



- Structured grids
- 3D arrays in memory



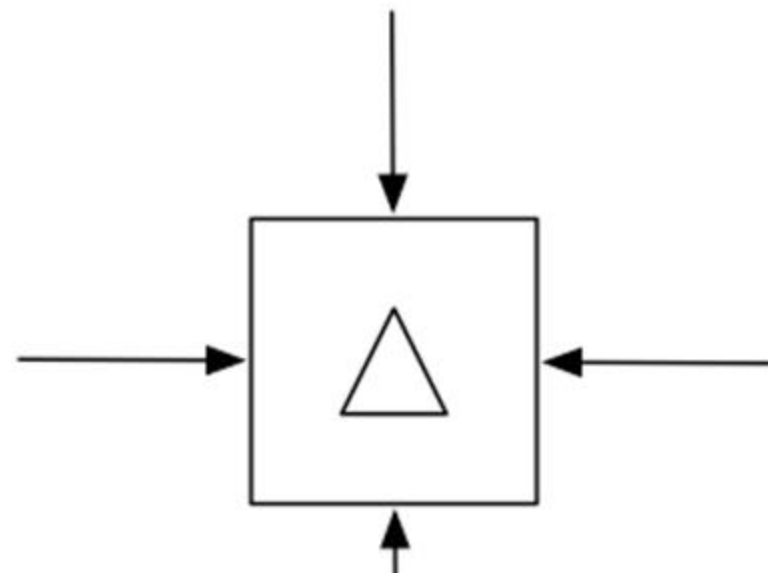
- Unstructured grids
- Need pointer lists for cor

- Will be focusing on structured grids

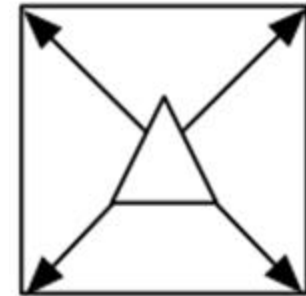
Solver algorithm

- Flow properties are stored at nodes
- Time-marching from initial guess to converged solution

•(1) Calculate cell change



(2) Distribute cell change



SIMD Parallelisation

- Structured grids are inherently data parallel
- Give each PE a few sub-blocks each
- Sub-block size dictated by the amount of PE memory

implementation

- Ideal

- Actual

Kernel types

- Two kernel types:
 - Vector operations

$$y[i] = f(x[i])$$

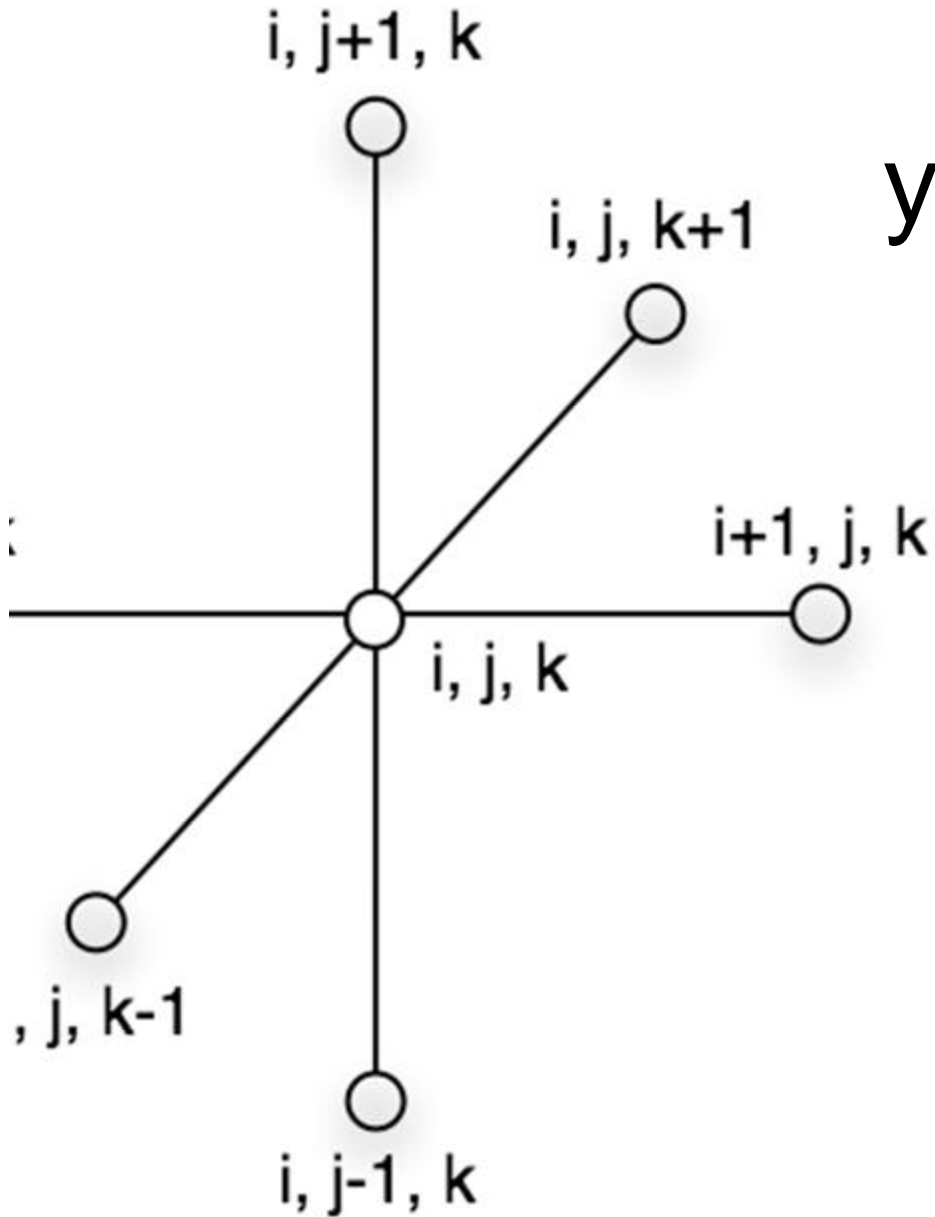
Example: Multiply flux sums by time step to get cell change

- Stencil operations

$$y[i] = f(x[i \pm (1, 2, 3, \dots)])$$

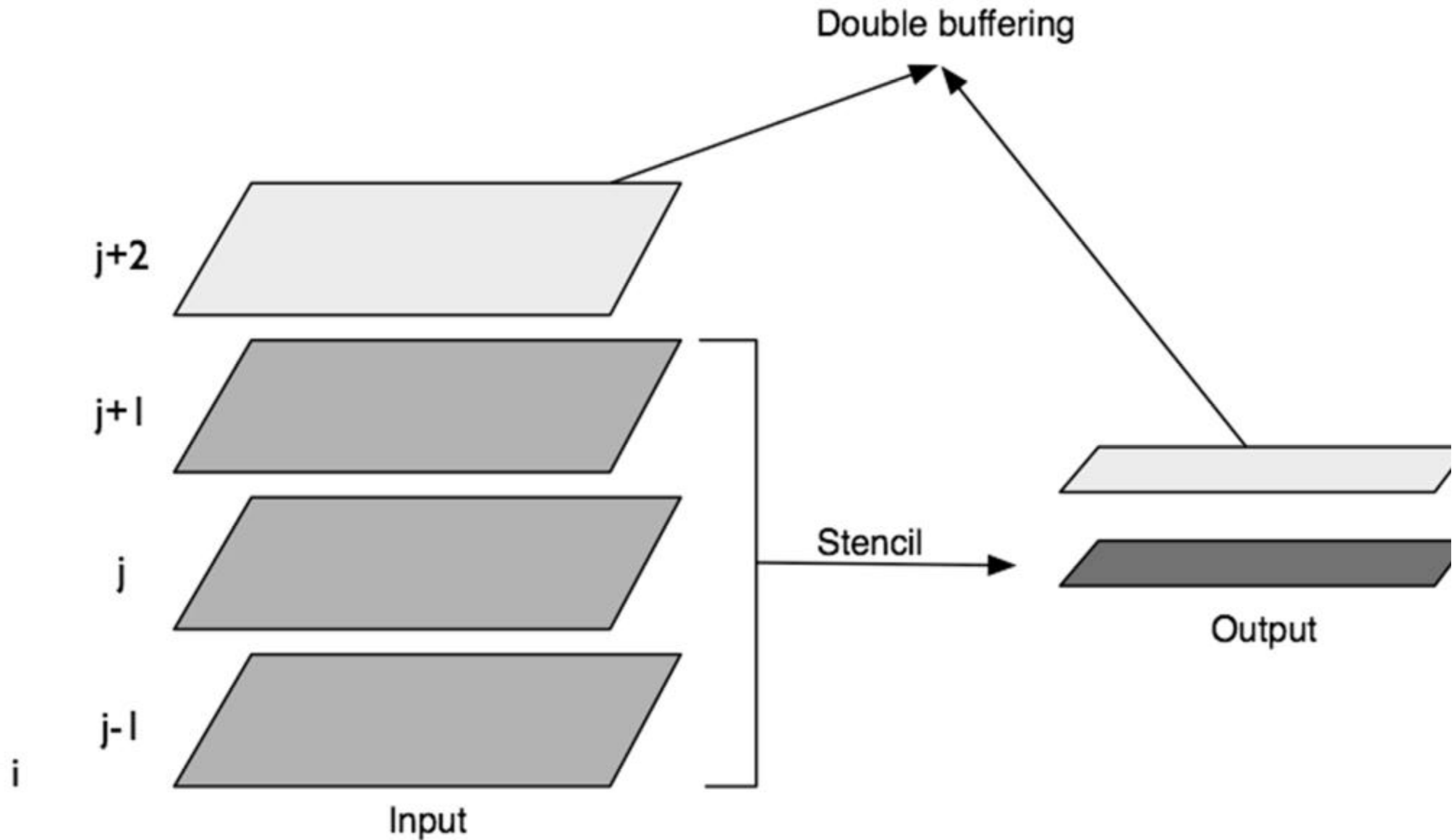
Example: Smooth node values to maintain stability

Smoothing kernel

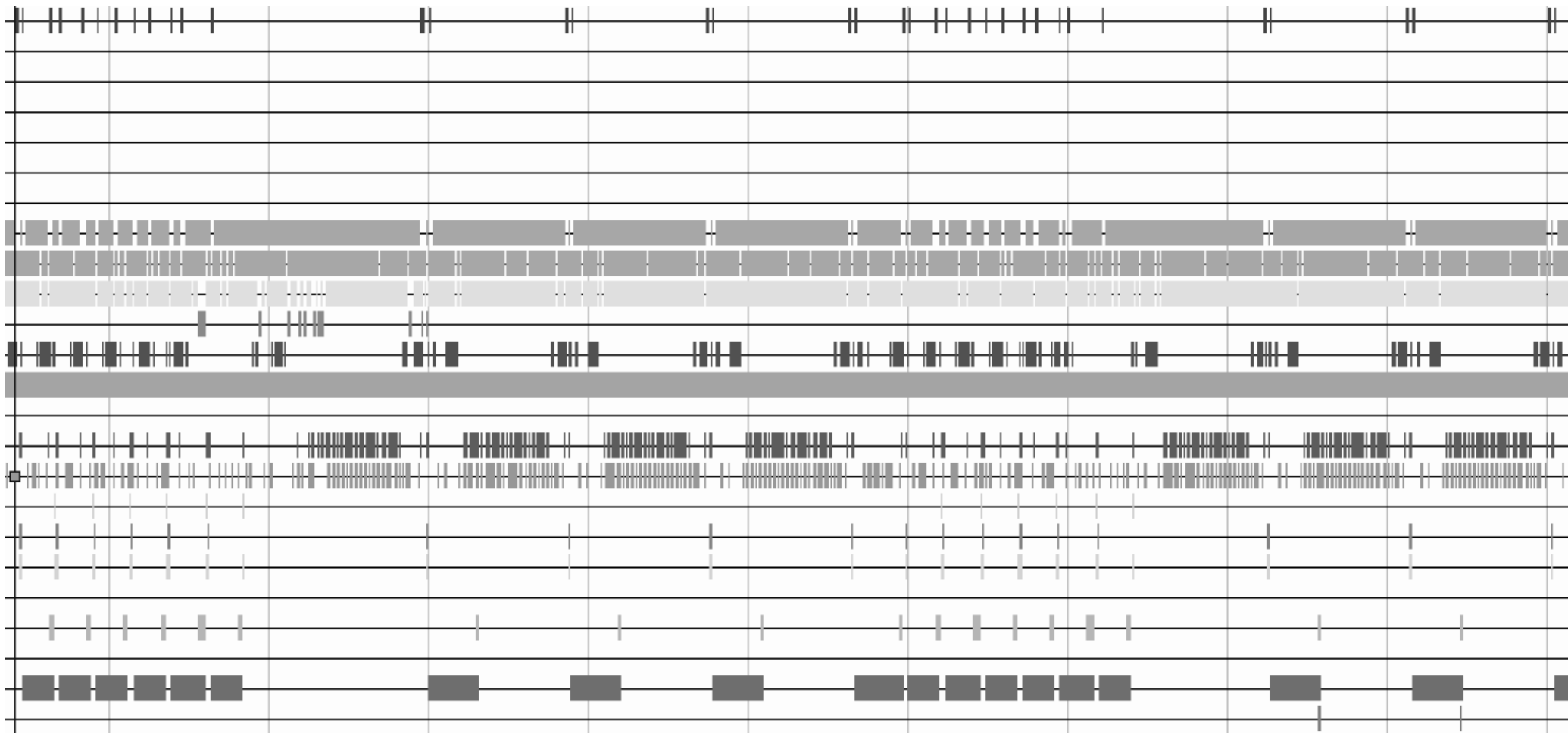


$$y[i, j, k] = (1-sf) \cdot x[i, j, k] + sf \cdot (\text{neighbours})$$

Smoothing kernel



Profiler



Optimisation tips

- No silver bullet, follow usual recommendations:
 - Double-buffer
 - Use vector math library
 - Keep DMA transfers to multiples of 64 bytes
 - Use as *char* and *short* instead of *int* for loop counters and array indices if possible
- Cumbersome edit-compile-debug cycle with host code

Development tips