Improving Locality of Unstructured Mesh Algorithms on GPUs

András Attila Sulyok[1] Gábor Dániel Balogh[1]
István Zoltán Reguly[1] Gihan R Mudalige[2]

[1]Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

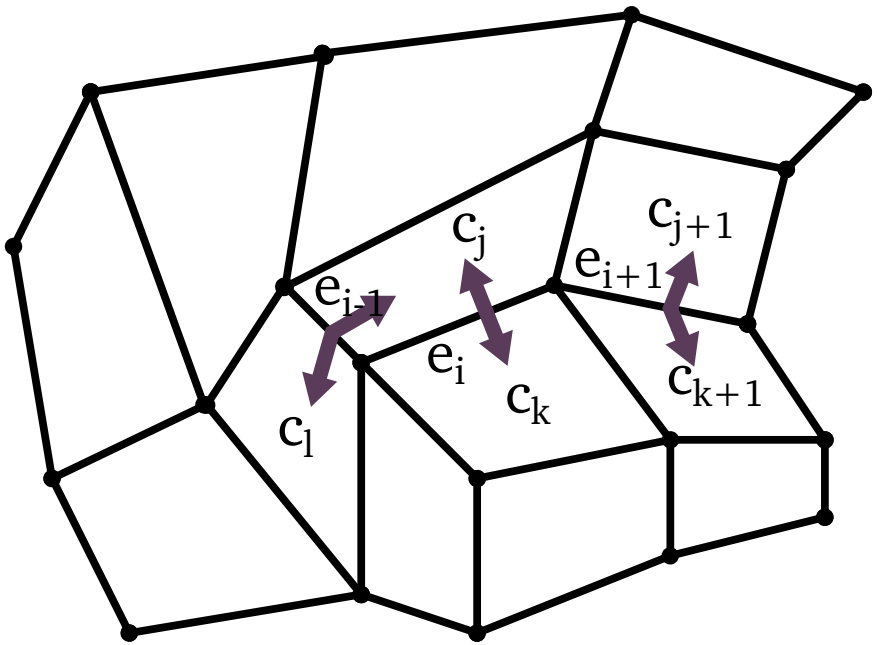
[2]The University of Warwick
Department of Computer Science

21/06/2018

Introduction

- Problem setting:
  - algorithms on unstructured meshes
  - running on GPUs
  - memory access is irregular
- Our approach: increasing locality
  - cache into shared memory
  - use two-level colouring
  - partition the mesh beforehand
- Using an automatic parallelisation framework
- Nvidia Pascal, Volta GPUs

Unstructured mesh



An unstructured mesh example

Cells: ... | j | l | j | k | j+1 | k+1 | ...

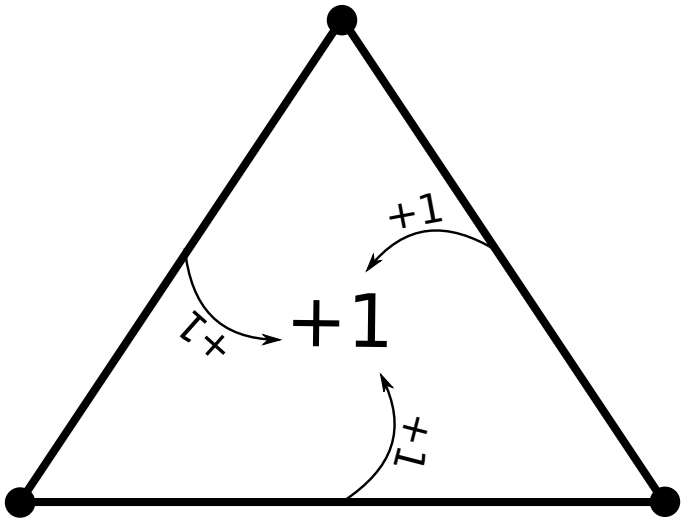
Edges: i-1 i i+1

A corresponding mapping

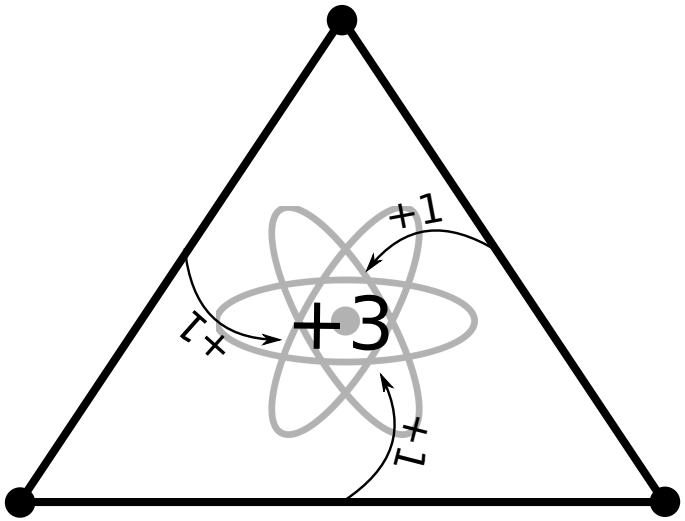
Test applications

- **Airfoil**
  - 2D inviscid airfoil flow simulation
  - University of Oxford
- **Volna**
  - 2D shallow-water tsunami simulation
  - University College London
- **Bookleaf**
  - 2D Lagrangian hydrodynamics application
  - Atomic Weapons Establishment
- **LULESH**
  - 3D Lagrangian hydrodynamics application
  - Lawrence Livermore National Laboratory
- **miniAero**
  - 3D Finite Volume code solving the compressibel Navier-Stokes equations
  - Sandia National Laboratories

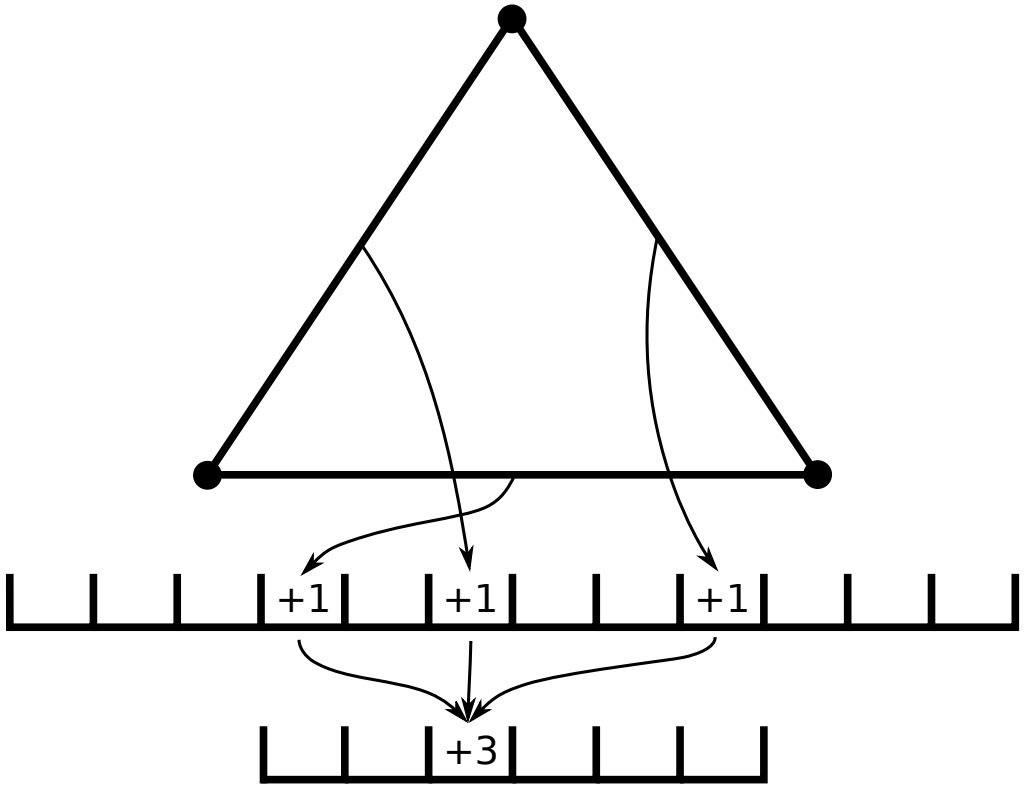
Thread-safety



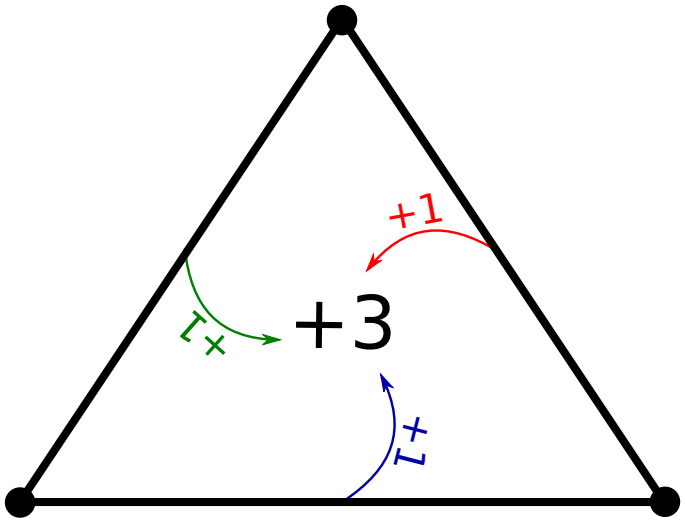
Thread-safety: atomics



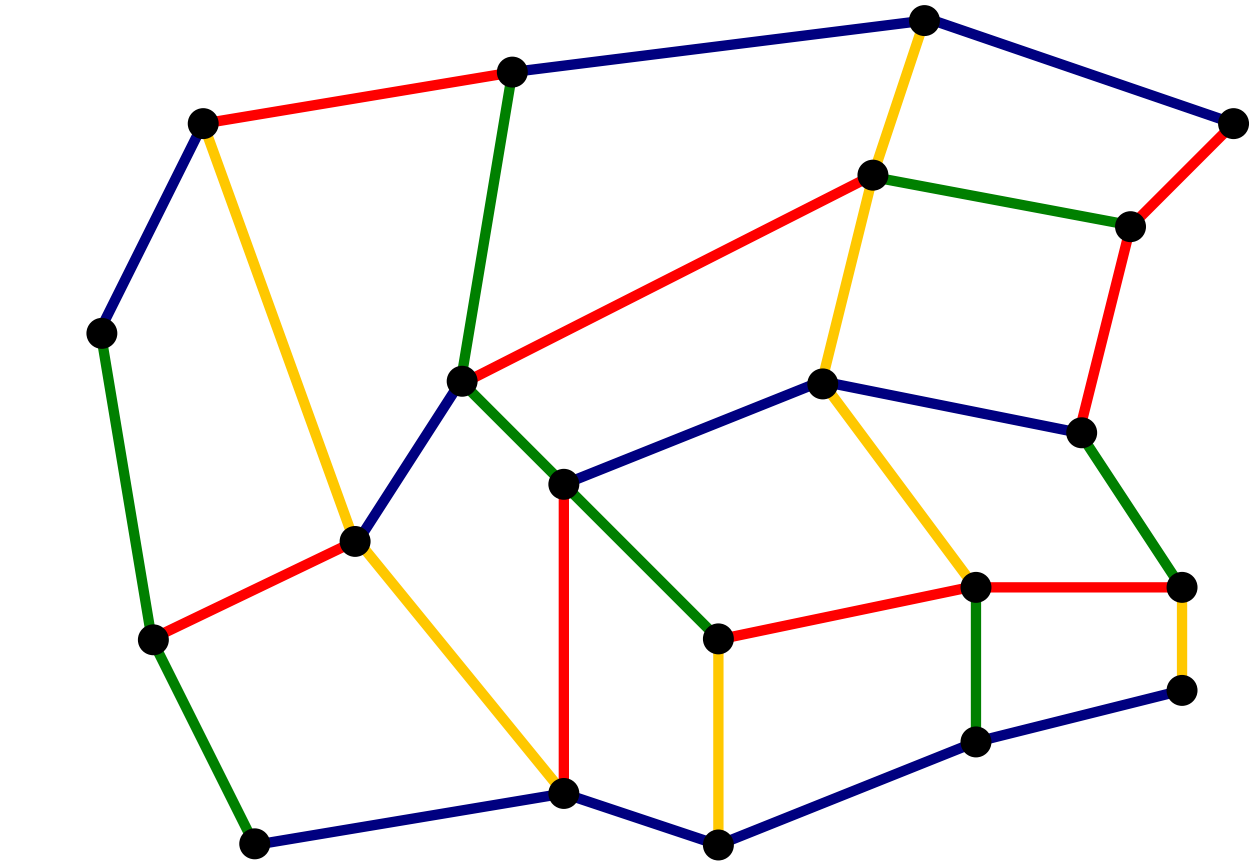
Thread-safety: helper arrays

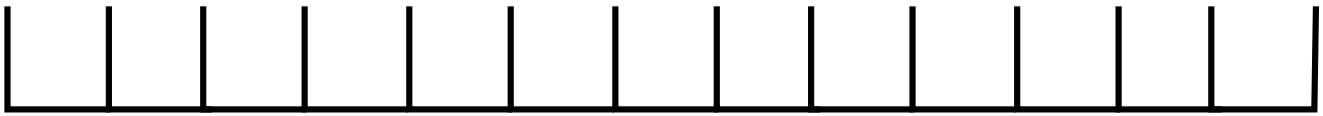


Thread-safety: colouring

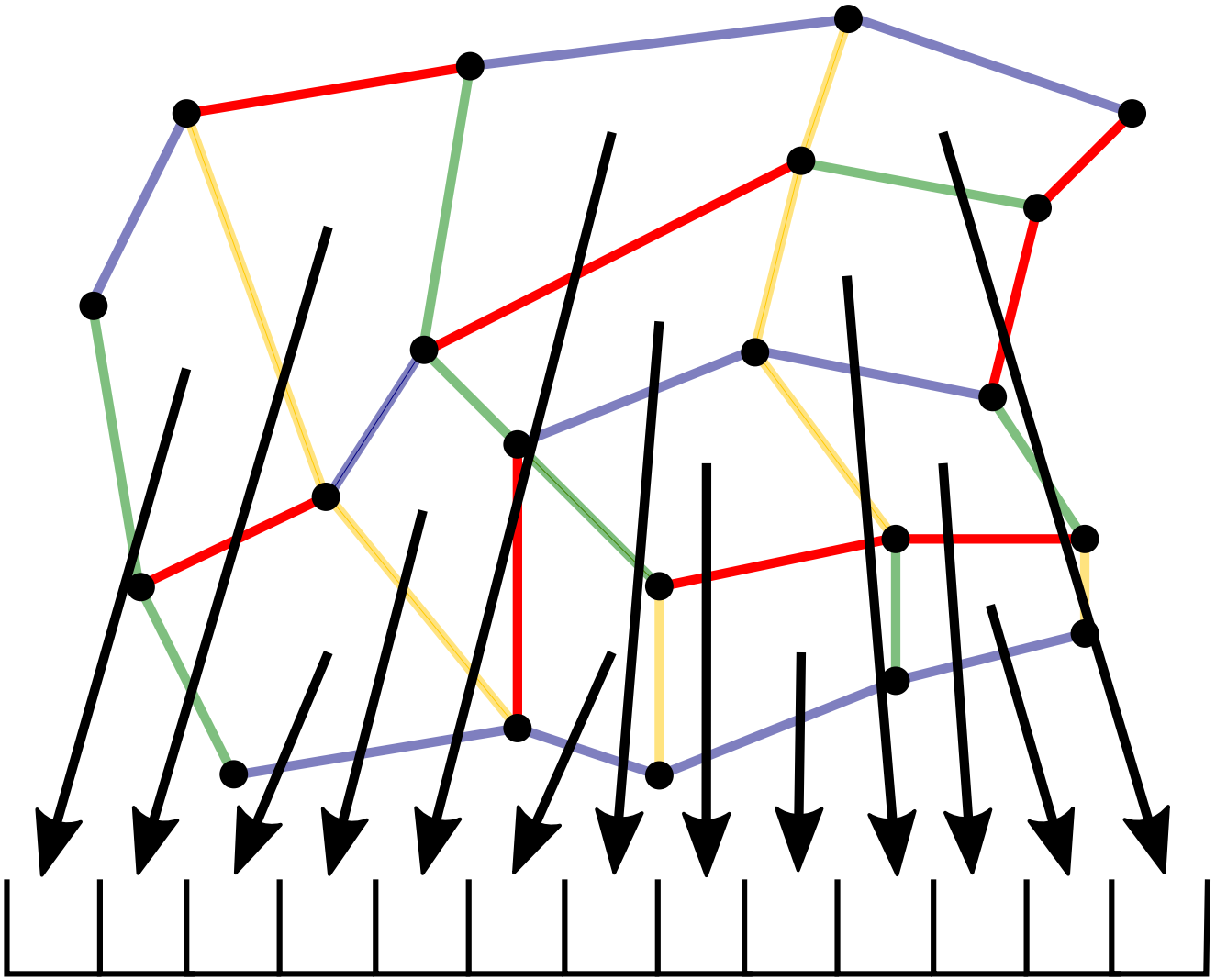


Global colouring

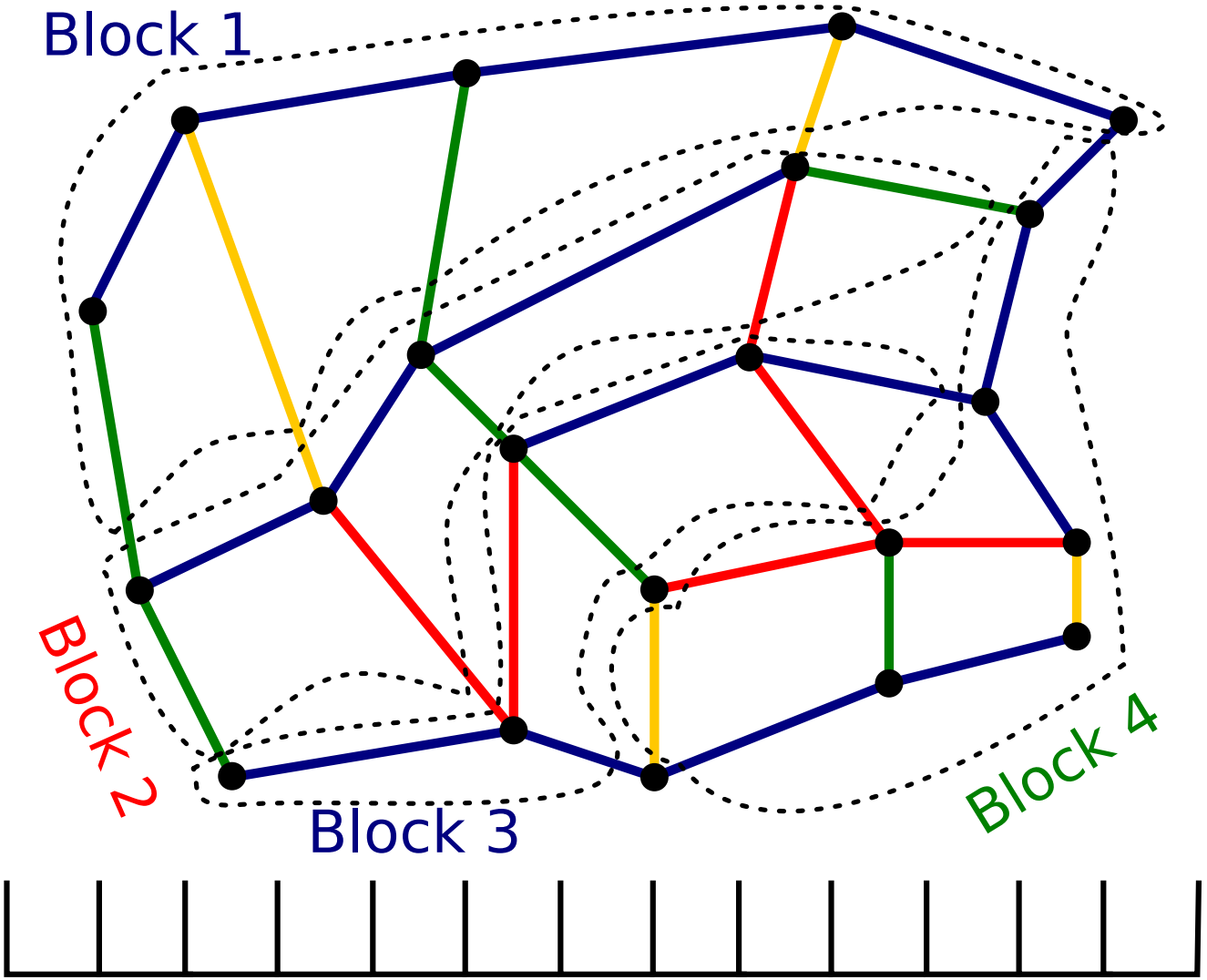




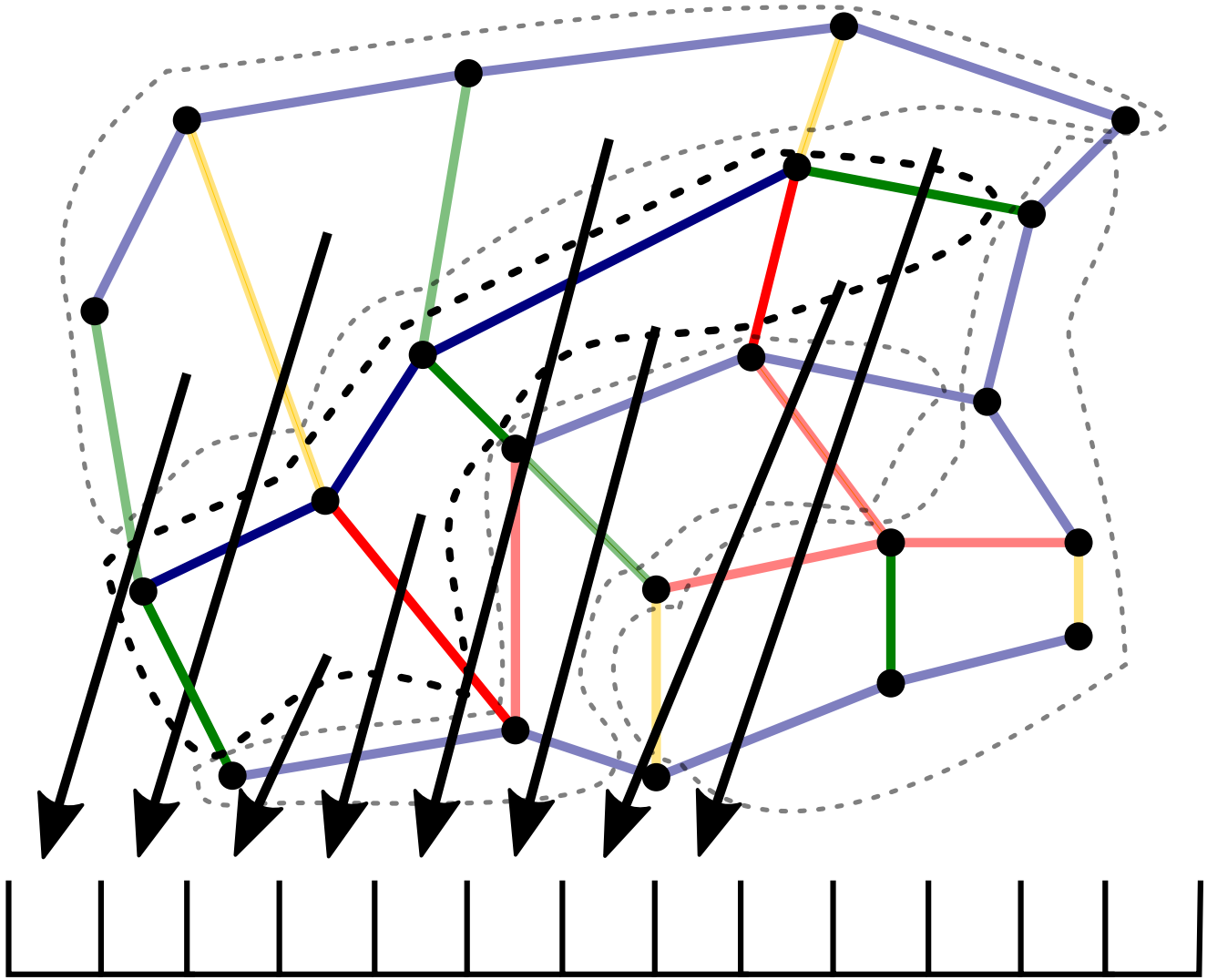
Global colouring



Hierarchical (two-layered) colouring



Hierarchical (two-layered) colouring

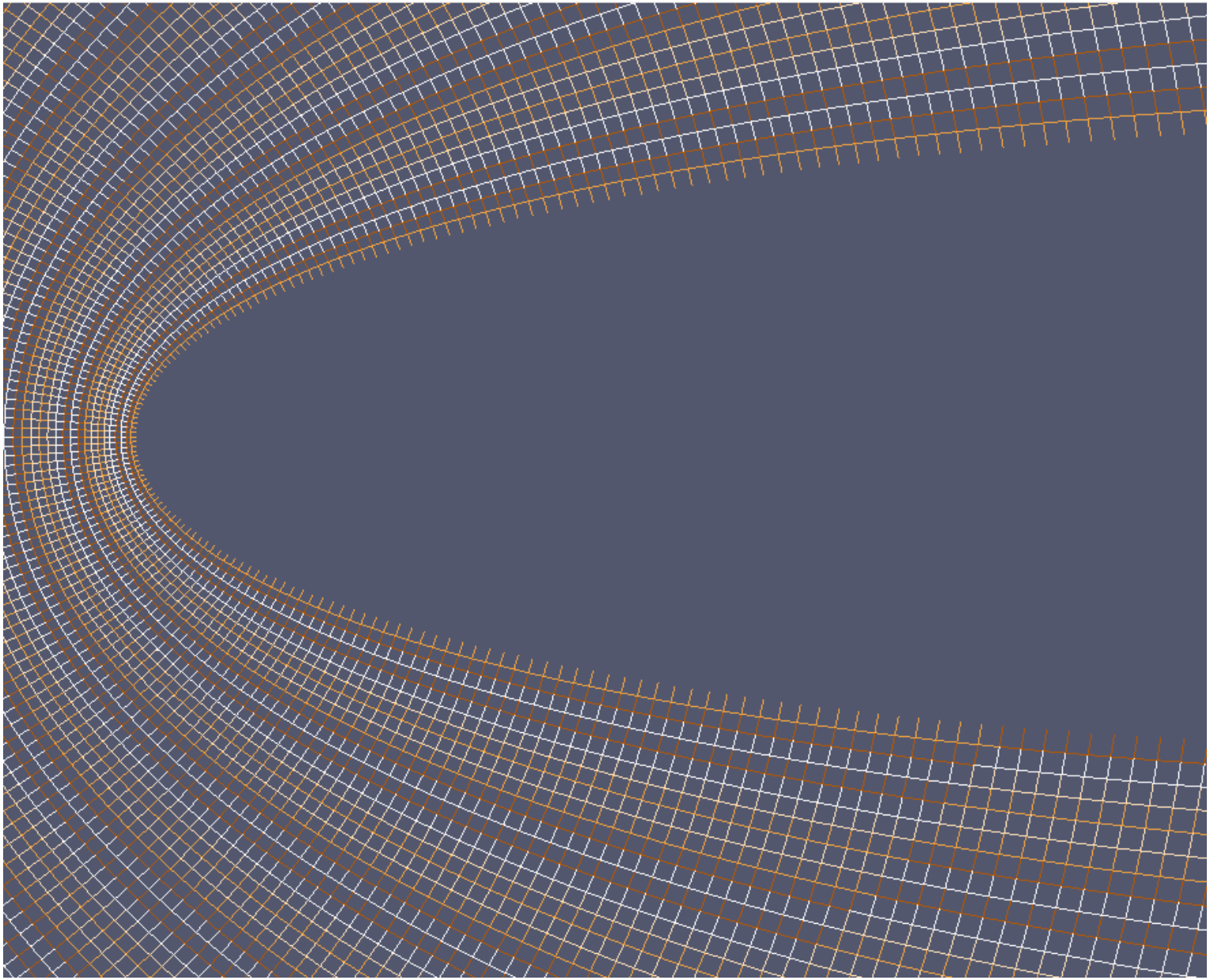


Improvement on global colouring

Using Airfoil as an example:

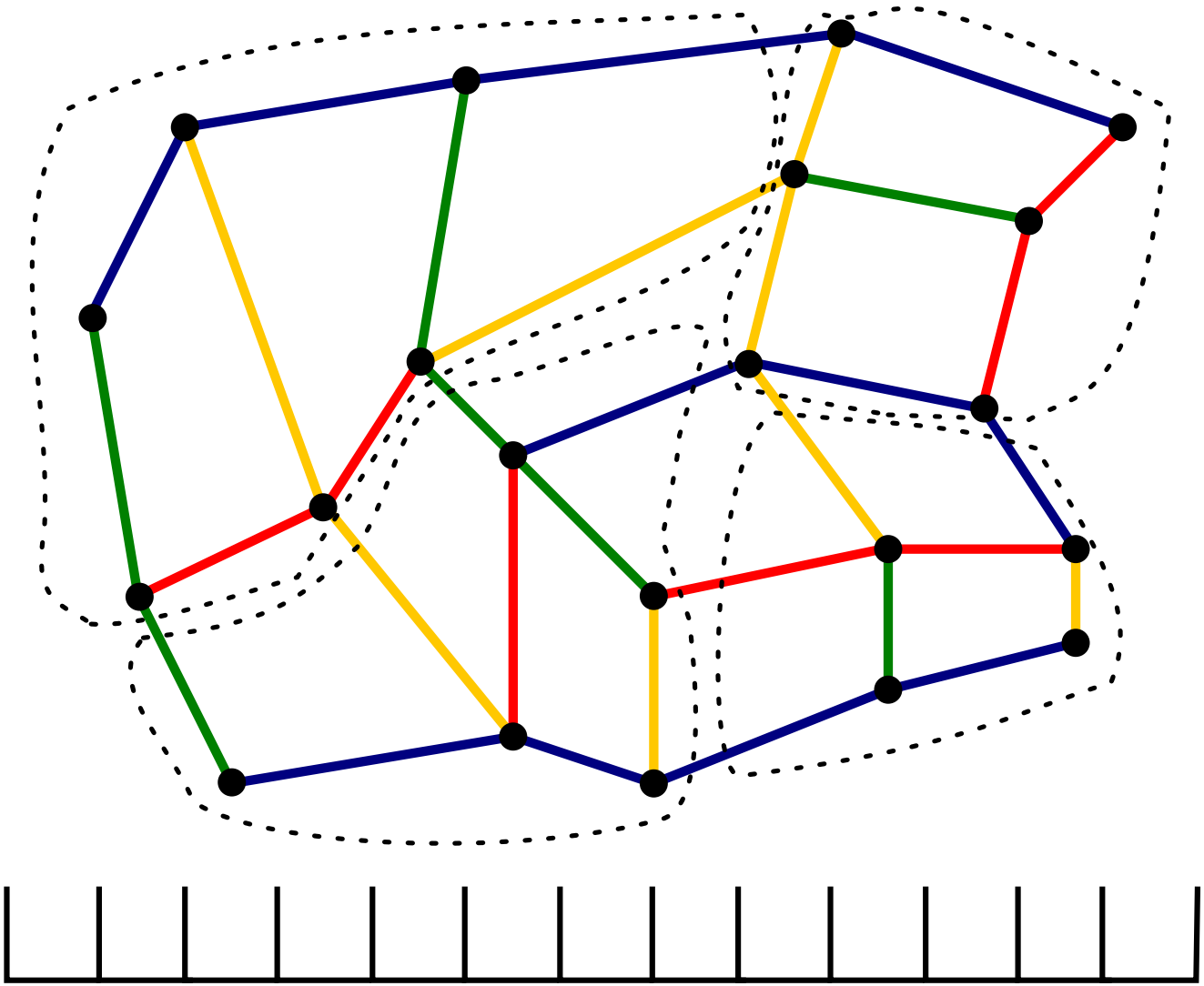
	Global	Hierarchical
Data reuse	1	2
Number of block colours	5	5
Number of thread colours	1	3
Bandwidth	71 GB/s	227 GB/s

Hierarchical colouring – original Airfoil mesh



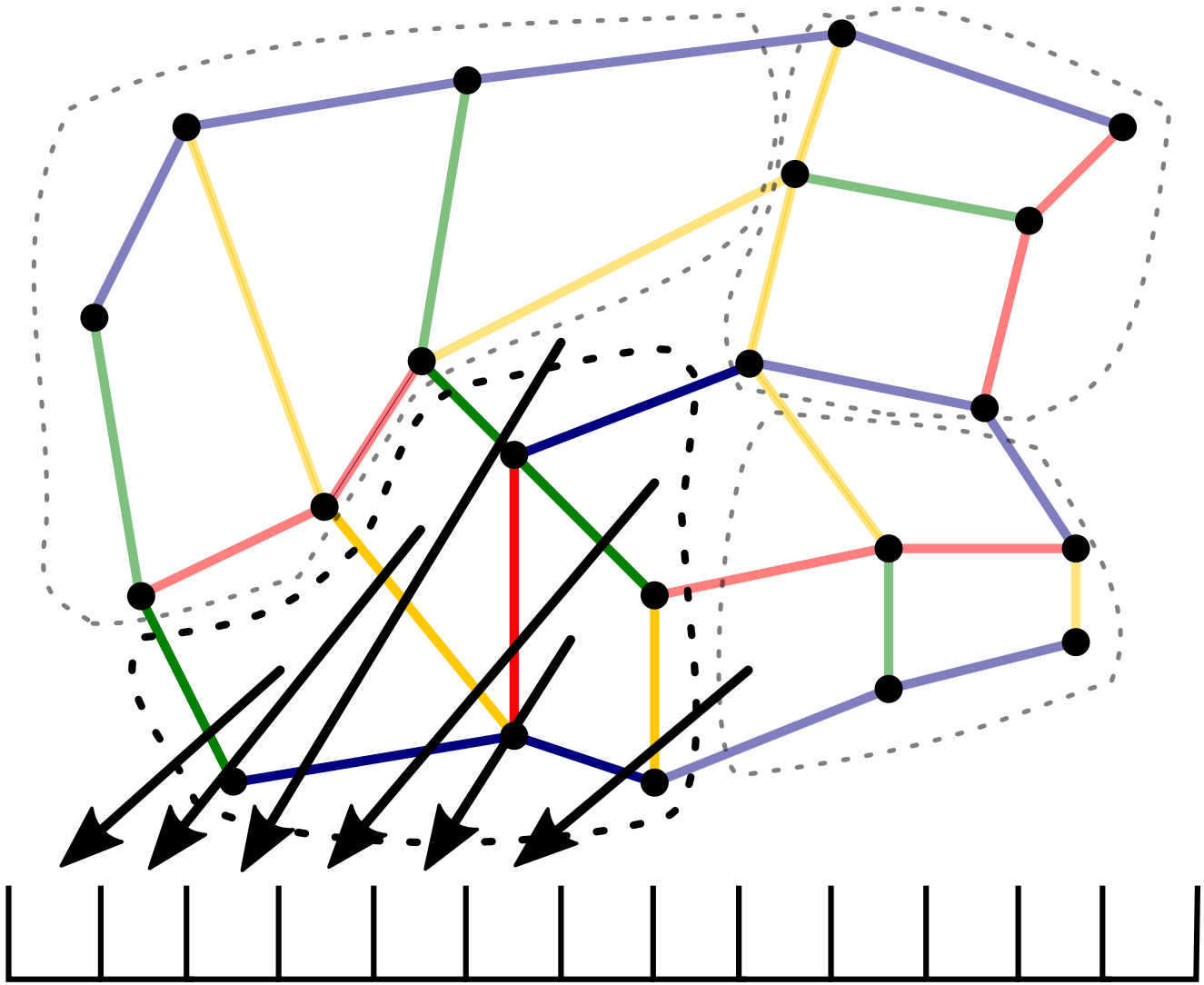
Hierarchical colouring – reordering

Aiming for high data reuse

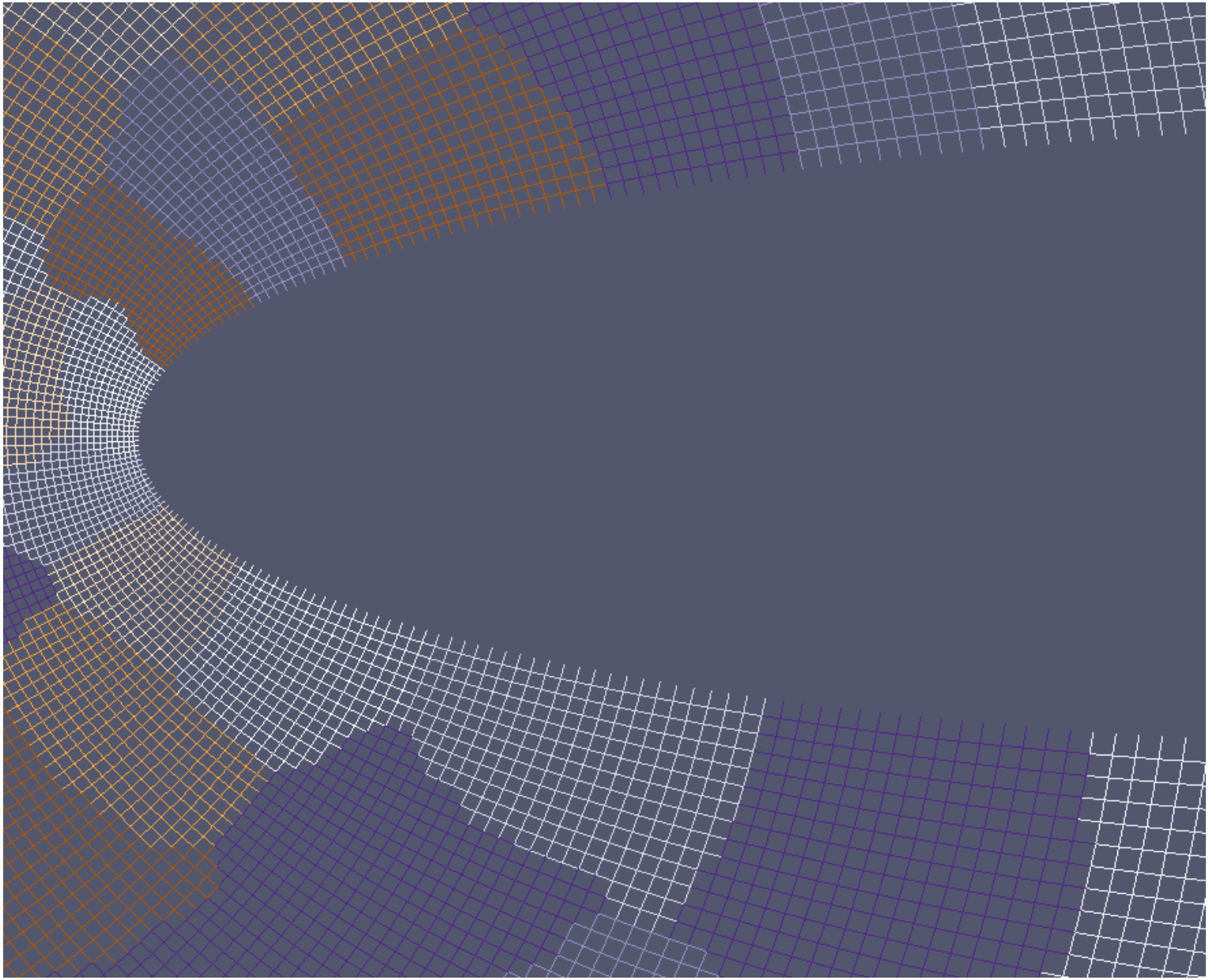


Hierarchical colouring – reordering

Aiming for high data reuse



Reordered Airfoil mesh

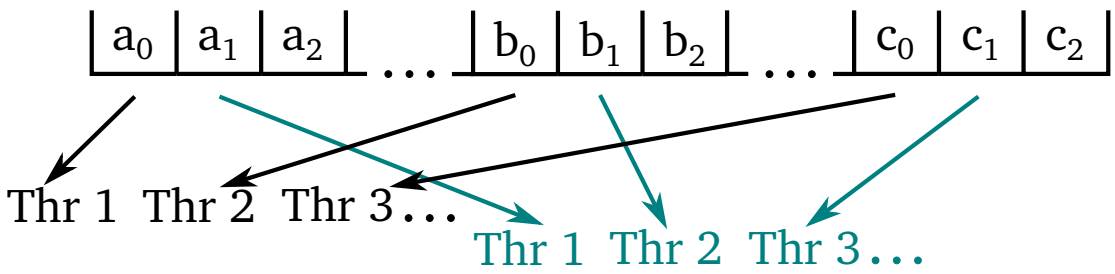


Effect of partitioning

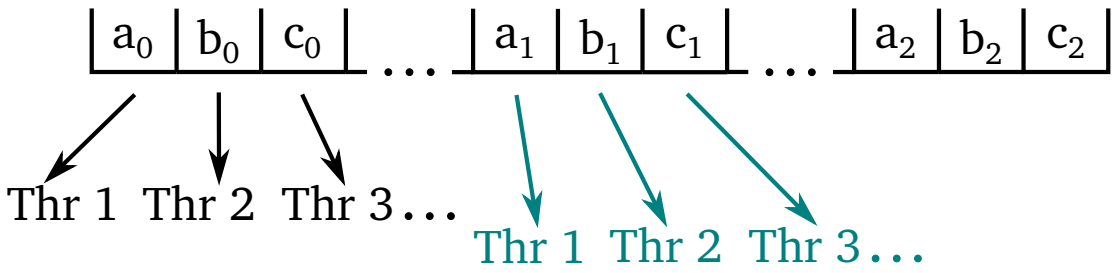
On Airfoil:

	Original	Partitioned
Data reuse	2	3.6
Number of block colours	5	8
Number of thread colours	3	4
Bandwidth	227 GB/s	270 GB/s

Array of Structs layout

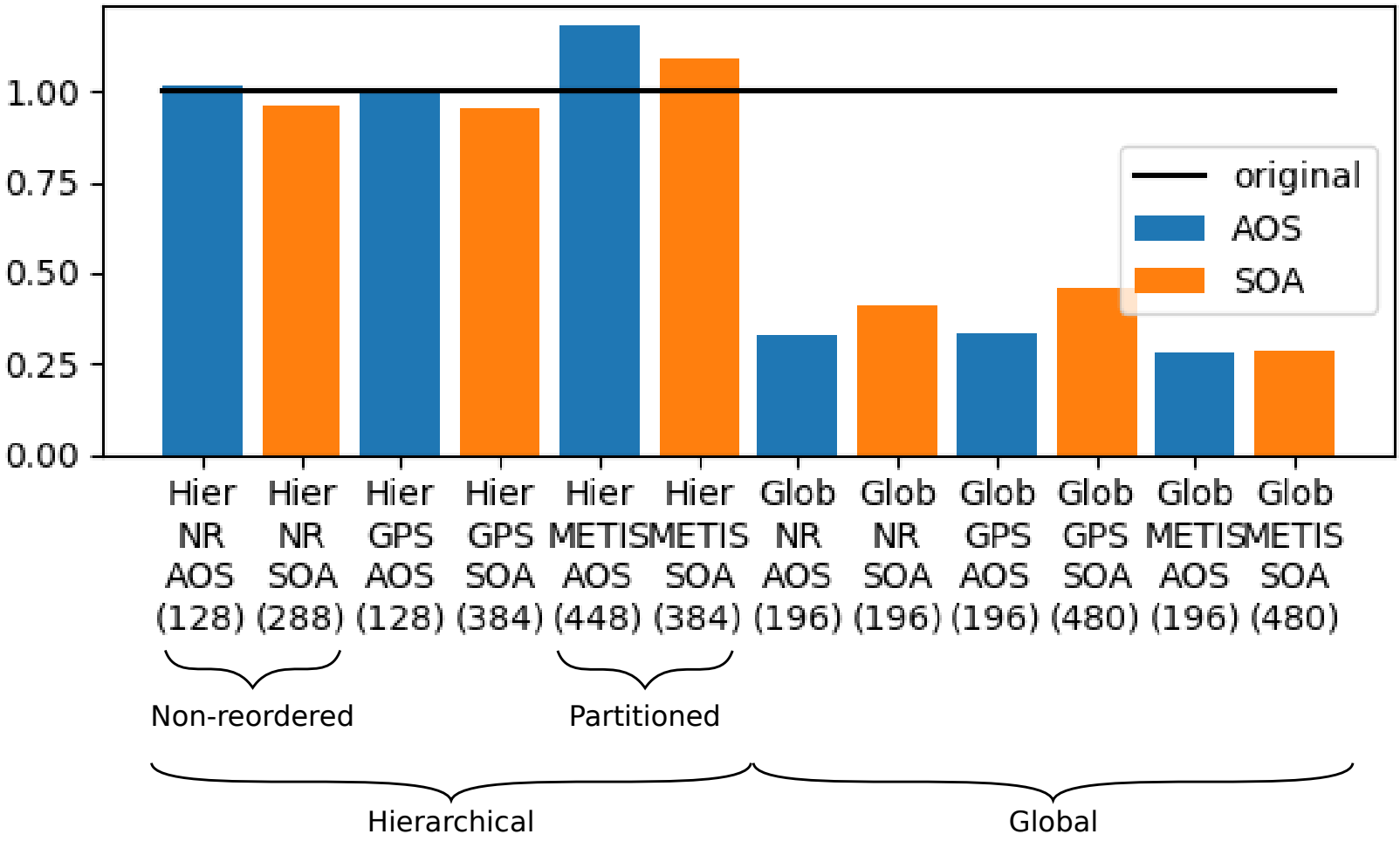


Struct of Arrays layout



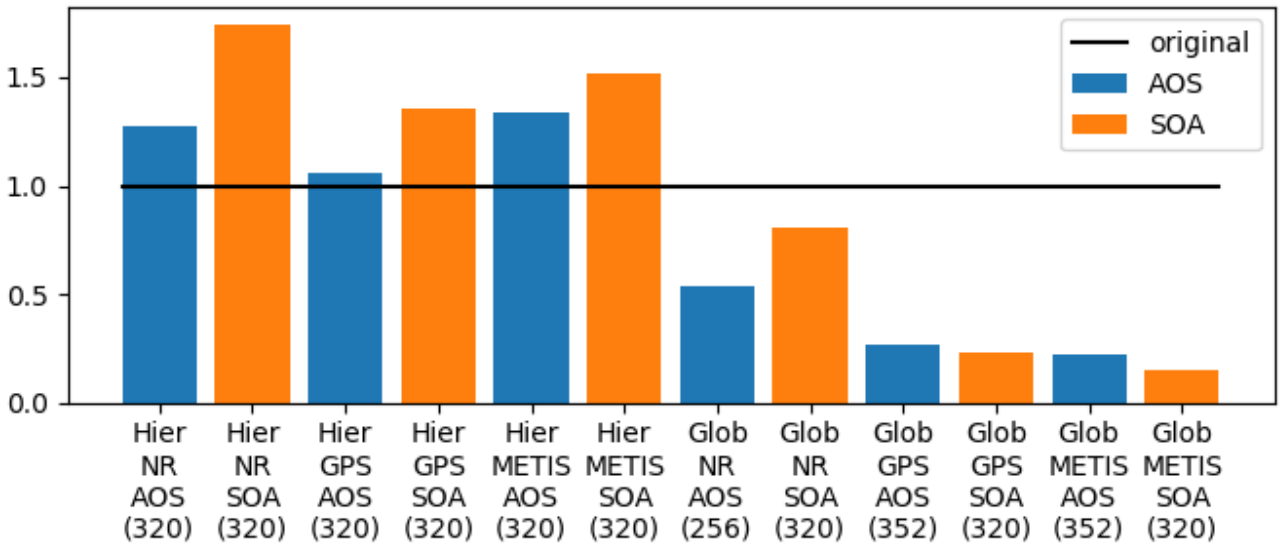
Effect of different parameters on Airfoil:

Improvement on the original code:



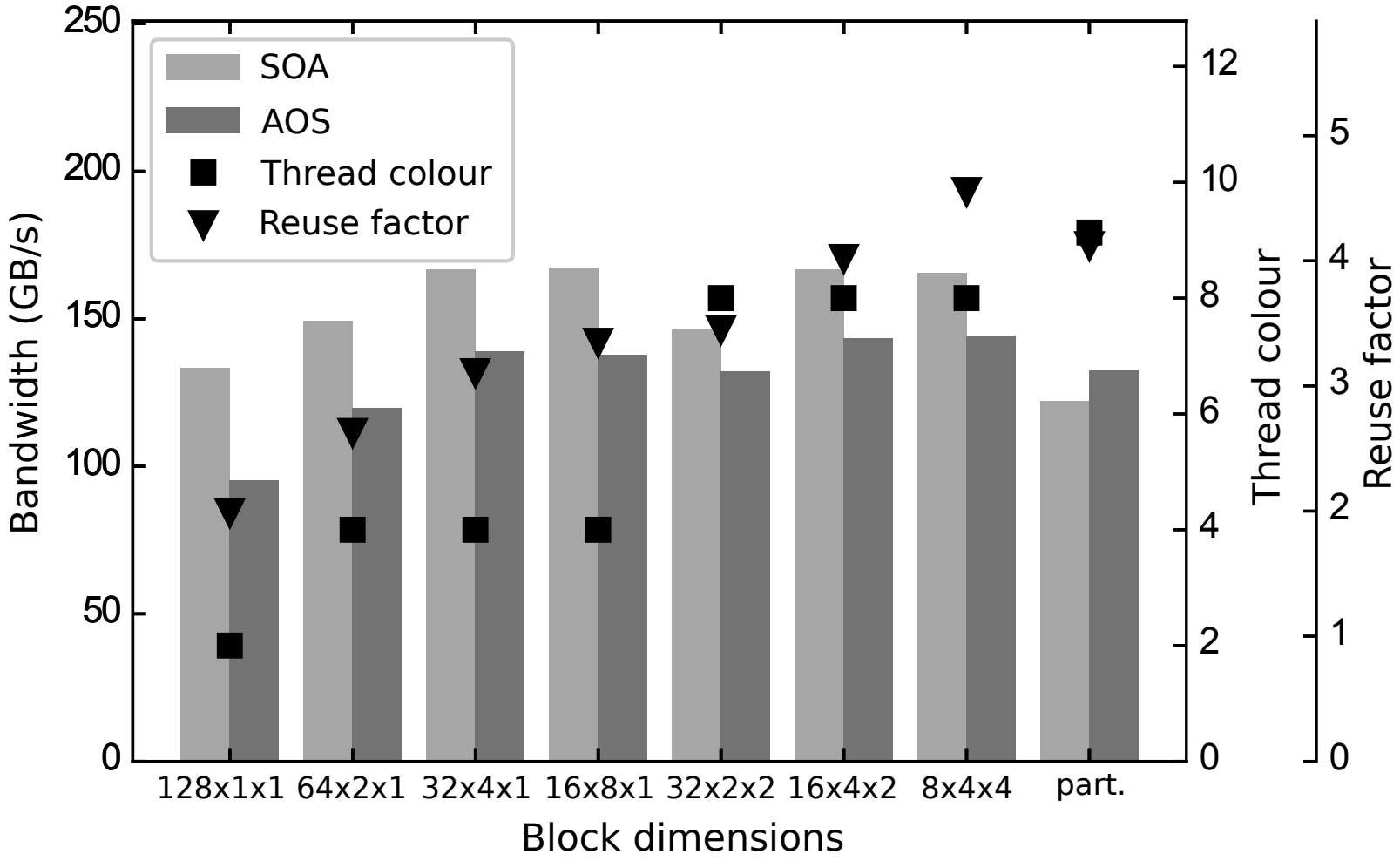
Effect of different parameters on LULESH

Improvement on the original code:

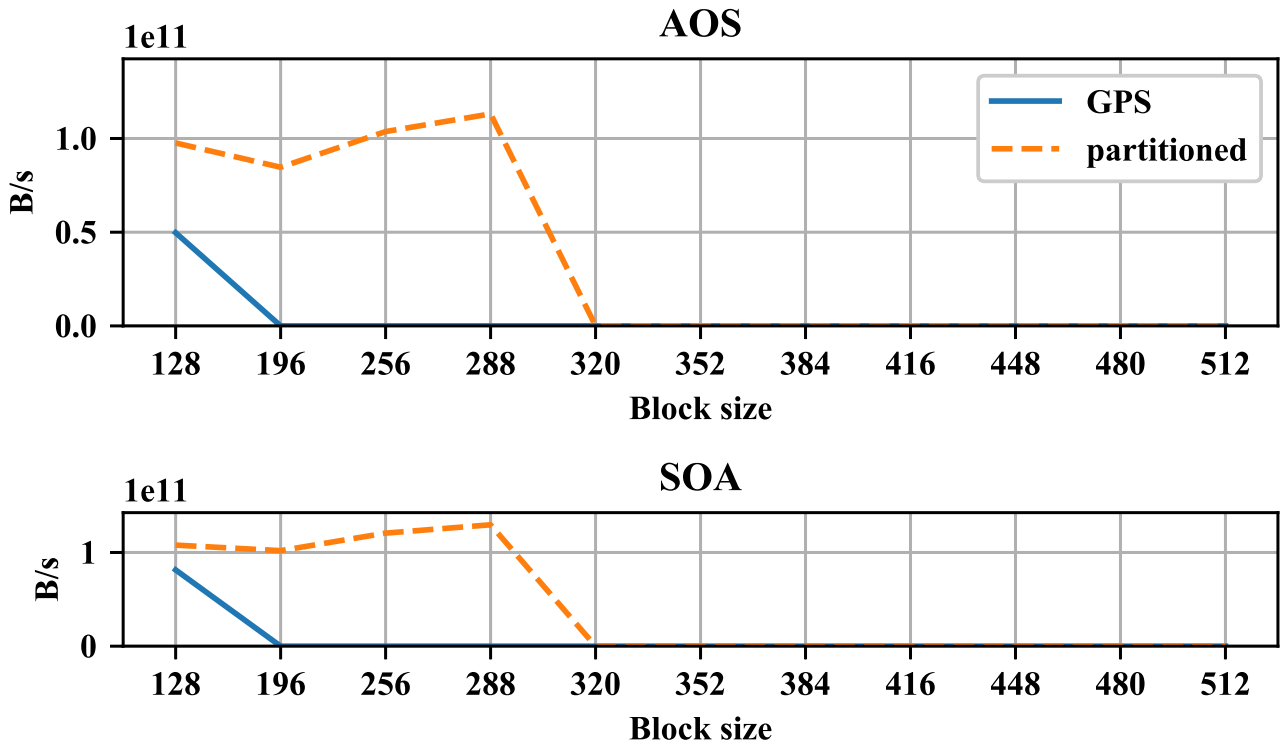


	Original	Partitioned
Data reuse	2.6	4.8
Number of block colours	4	15
Number of thread colours	4	11.6
Bandwidth	168 GB/s	147 GB/s

Manual partitioning of LULESH

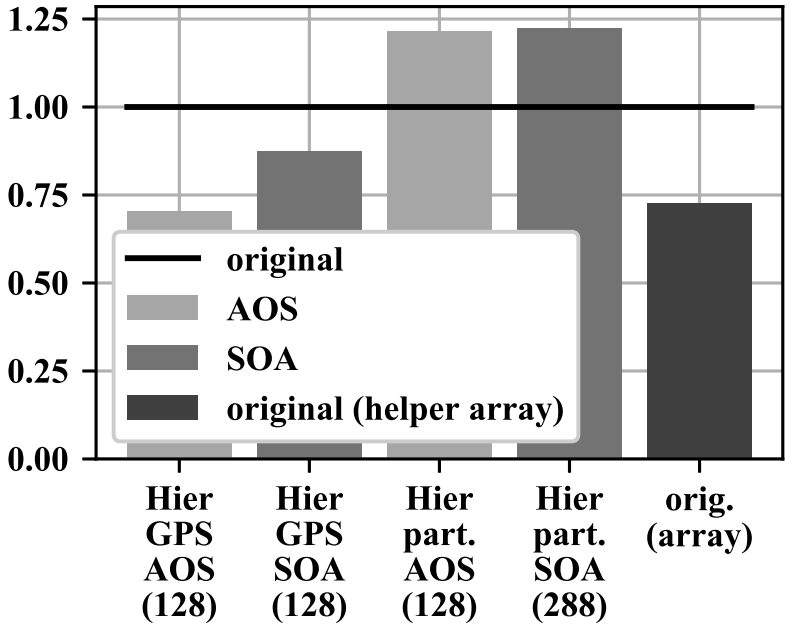


miniAero – hierarchical colouring



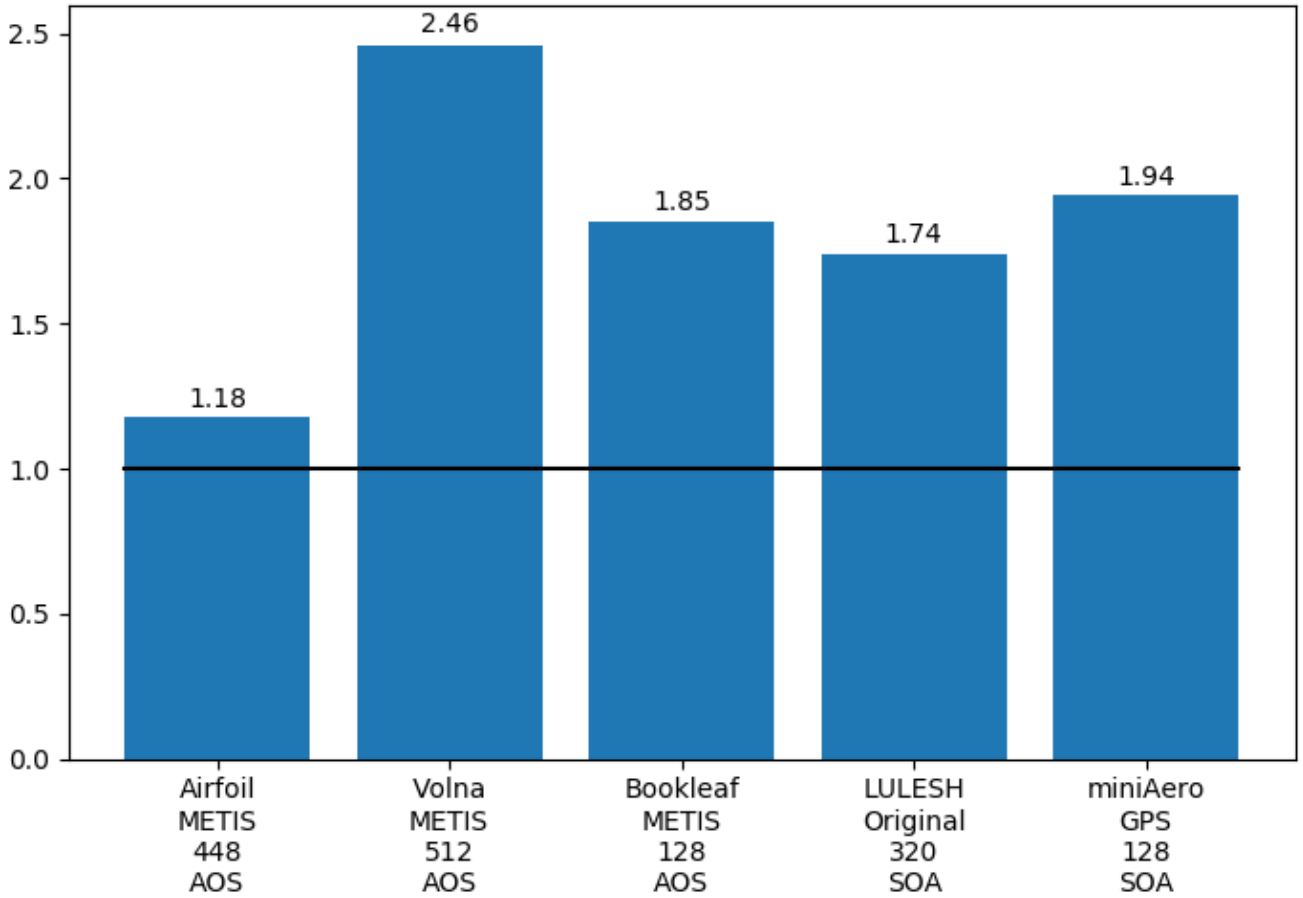
Performance of miniAero

Improvement on the original code:



	GPS reordered	Partitioned
Data reuse	2.2	3.9
Number of block colours	18	15
Number of thread colours	3	6
Bandwidth	84 GB/s	117 GB/s

Summary of speed improvements



Summary

- Our optimisations:
  - Cache-in to shared memory
  - Use hierarchical colouring
    - this has not been done before in the case of LULESH and miniAero
  - New reordering algorithm based on partitioning
  - Higher data reuse leads to higher performance
    - When the amount of synchronisation is low
    - Achieved improvements in performence between 10% (Airfoil) and 140% (Volna) compared to the original codes
- Directions of further research
  - Better partitioning algorithm that considers the number of thread colours
  - Efficient partitioning code, e.g. a parallel one