Improving Locality of Unstructured Mesh Algorithms on GPUs

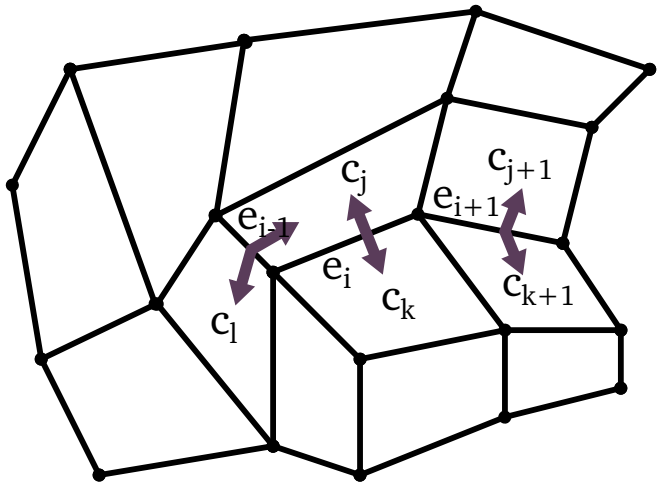
András Attila Sulyok[1] Gábor Dániel Balogh[1]
István Zoltán Reguly[1] Gihan R Mudalige[2]

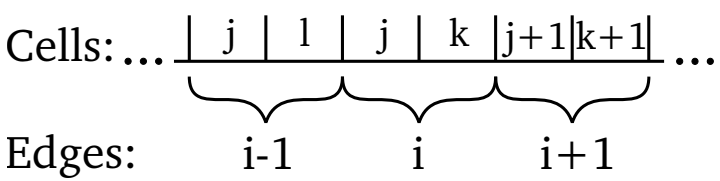
[1]Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

[2]The University of Warwick
Department of Computer Science

The problem

Memory access pattern on unstructured meshes

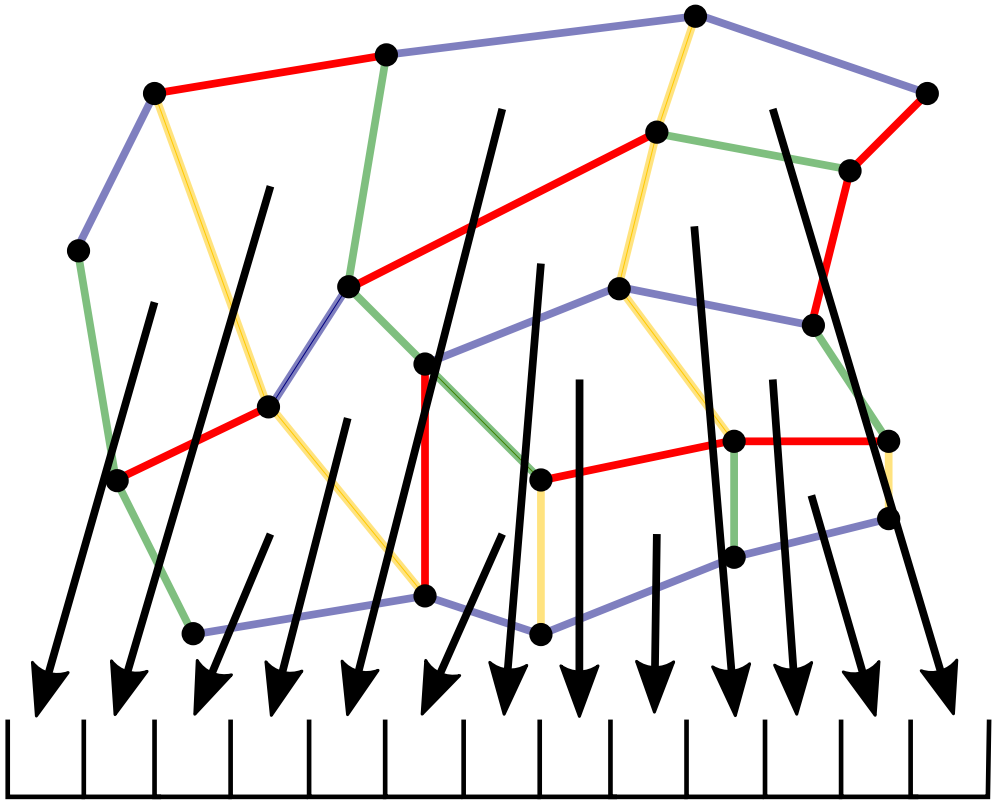




Unstructured mesh

- a collection of sets
  - nodes, edges, cells, etc.
- and the connections between them
  - the *mappings*
  - maps from one set to another
- mappings define the memory access pattern
  - only known at runtime
  - usually highly irregular
- colour the threads to avoid data races

First approach
Global colouring

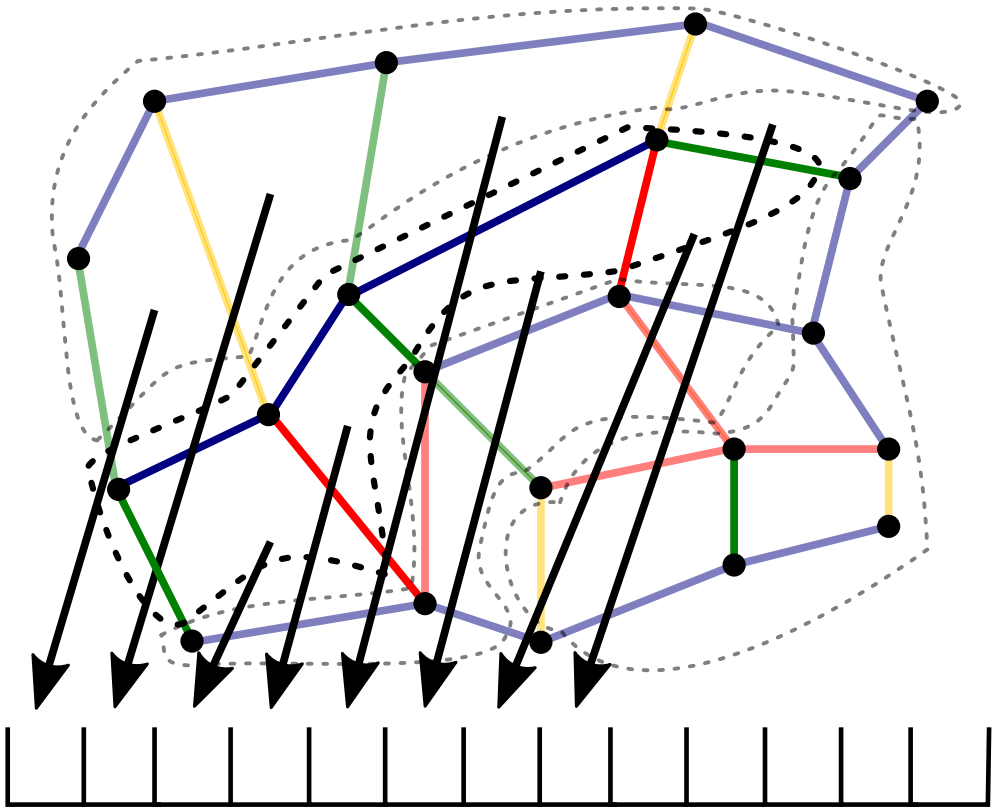


Lots of data points to load from global memory
$\Rightarrow$ application is slow (memory bound)

Second approach
Use the shared memory

Load indirect data into shared memory
→ do computation
→ write answer back

$\Rightarrow$ data points are loaded only once per block



Need to colour the CUDA blocks and also the threads within the blocks.

Third approach

Partition the threads to increase data reuse

We want: even less data loaded

⇓

higher data reuse in a block

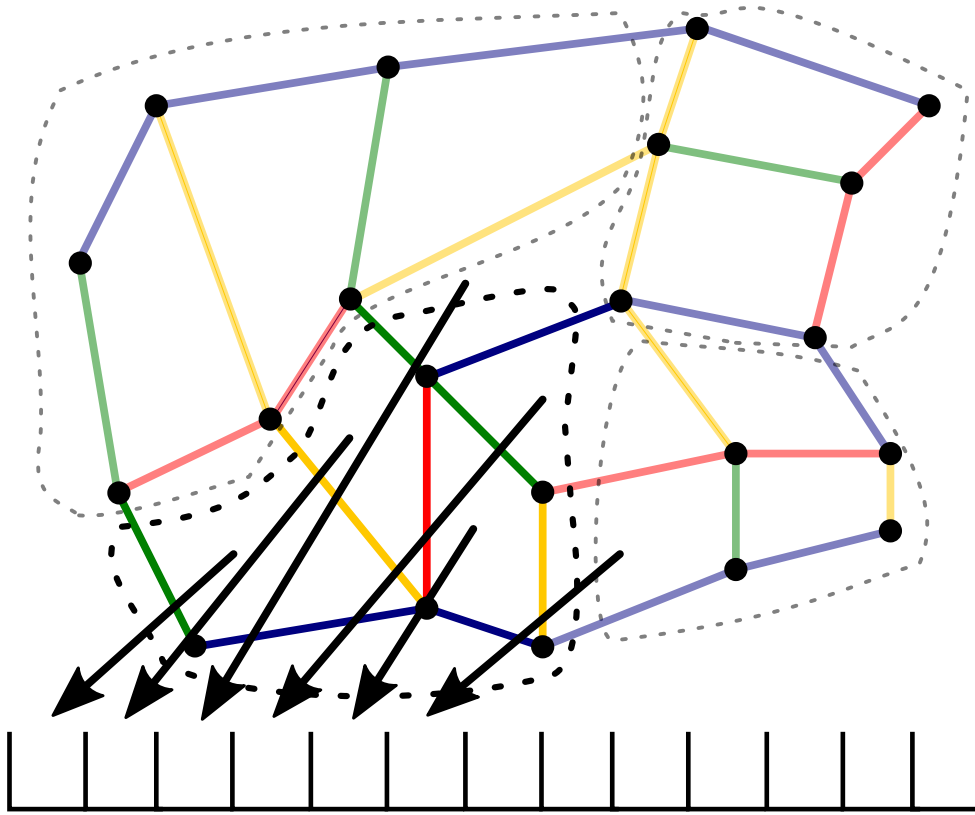
⇓

lower data reuse *between* the blocks

⇓

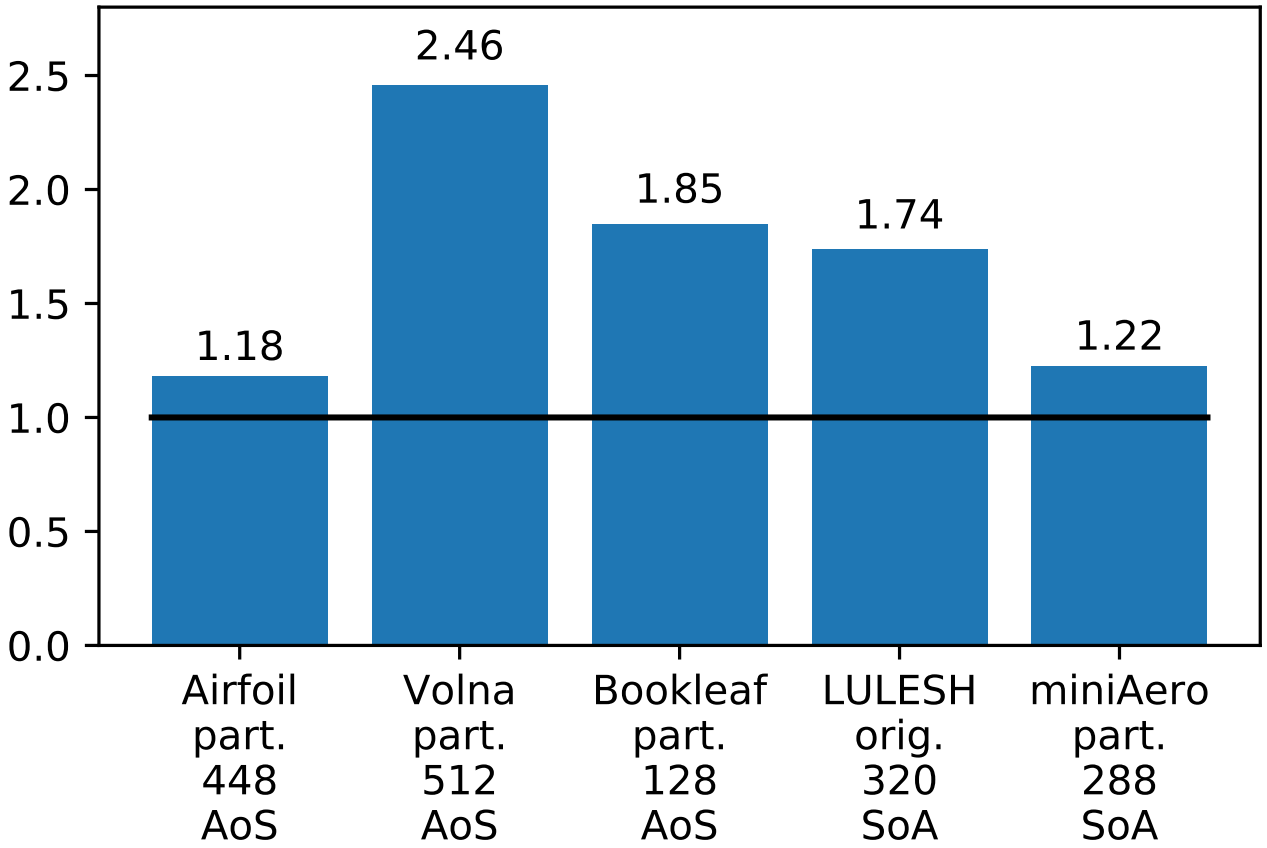
use partitioning to form the blocks

(e.g. the K-way partitioning algorithm in METIS)



Summary of speed improvements

On 5 different applications, compared to the original codes (black line):



Bandwidth is the ratio of the amount of useful data to the time spent in calculating the kernel.