Optimising SpMV for FEM on FPGAs

Paul Grigoras, Pavel Burovskiy, Wayne Luk, Spencer Sherwin

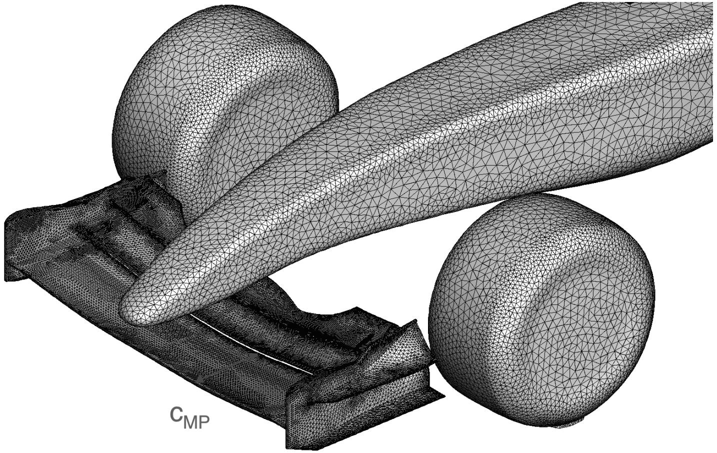

SWISS

Finite Element Methods - Solve PDEs over large, unstructured geometries

- PDEs: Incompressible Navier Stokes, Shallow Water etc.
- Applications: computational fluid dynamics, biomedicine, geoscience, etc.

Finite Element Methods

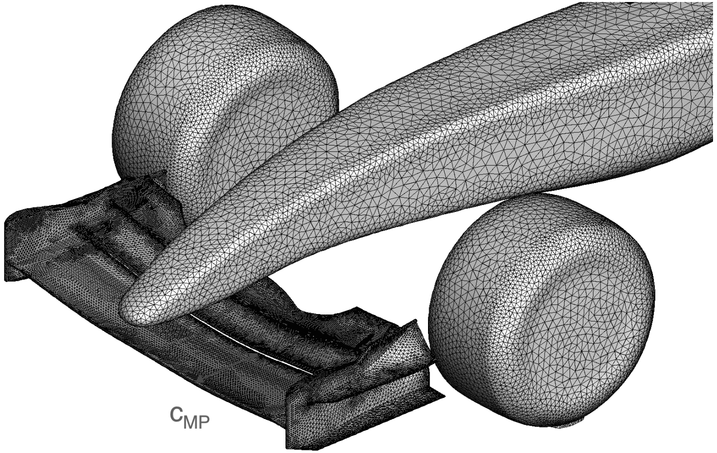
Source: www.nektar.info



Mesh over unstructured domain

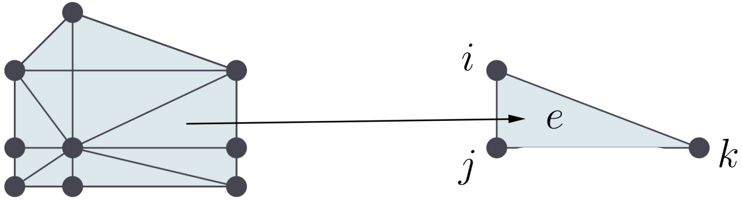
Finite Element Methods

Source: www.nektar.info



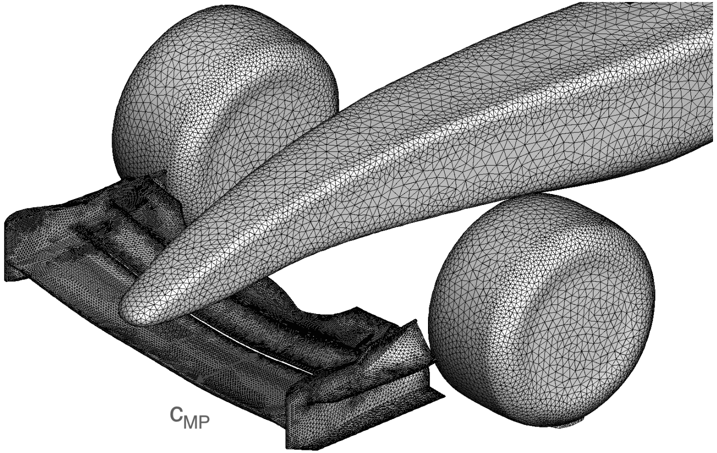
Mesh over unstructured domain

Mesh elements

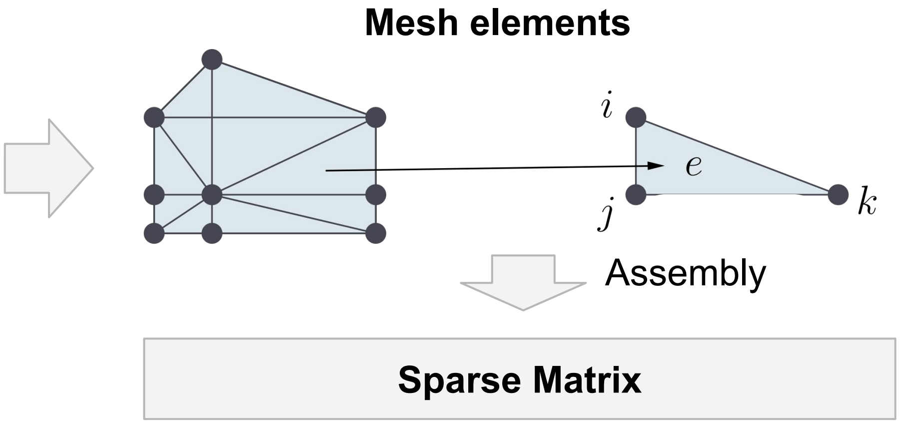


Finite Element Methods

Source: www.nektar.info

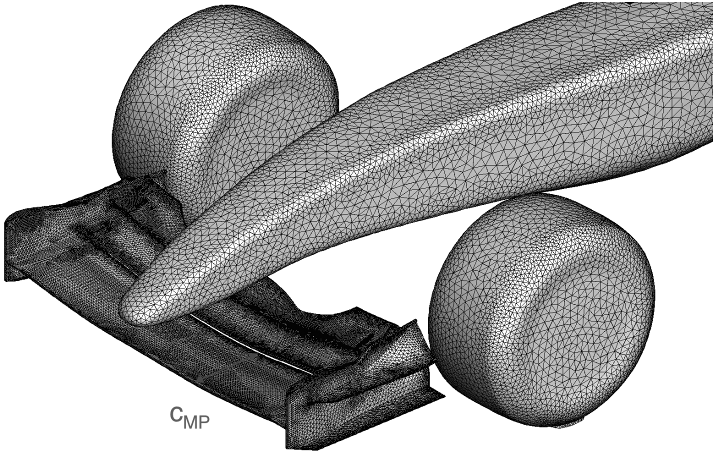


Mesh over unstructured domain



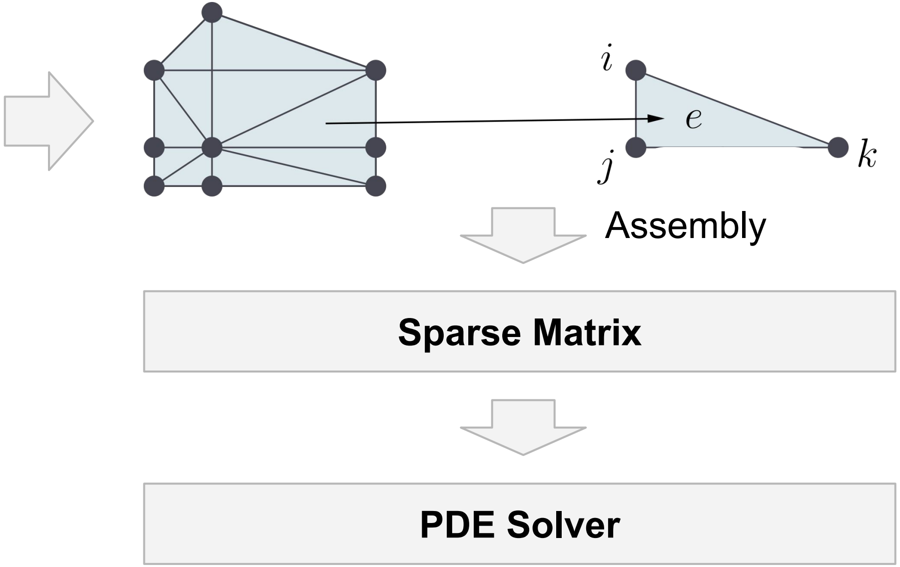
Finite Element Methods

Source: www.nektar.info



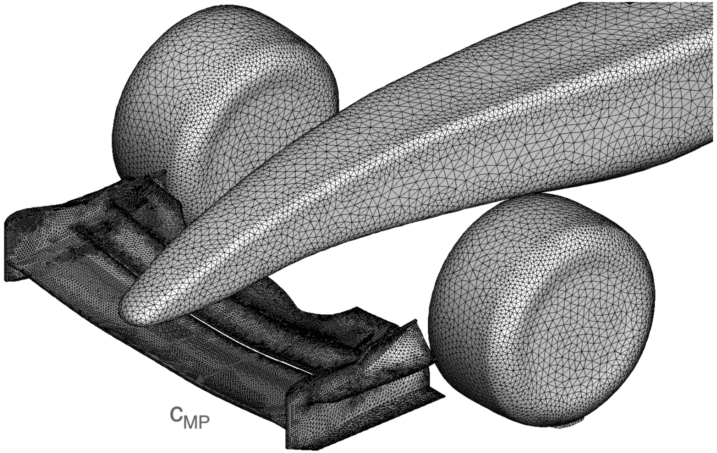
Mesh over unstructured domain

Mesh elements



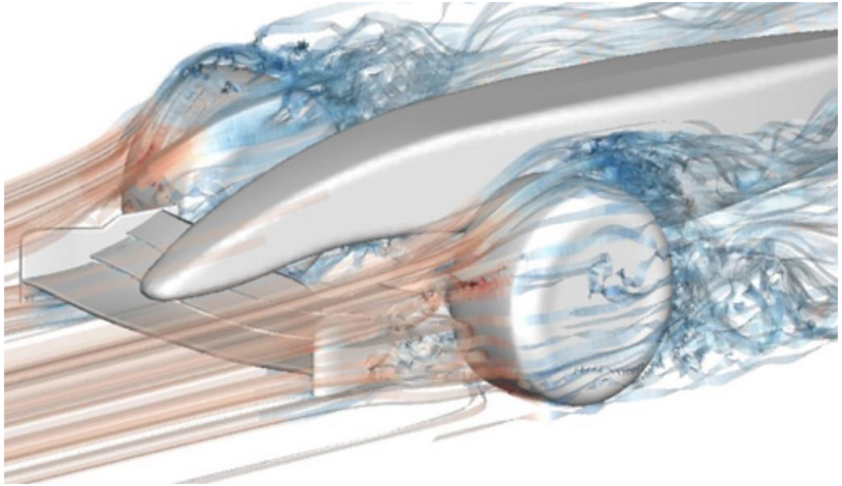
Finite Element Methods

Source: www.nektar.info



Mesh over unstructured domain

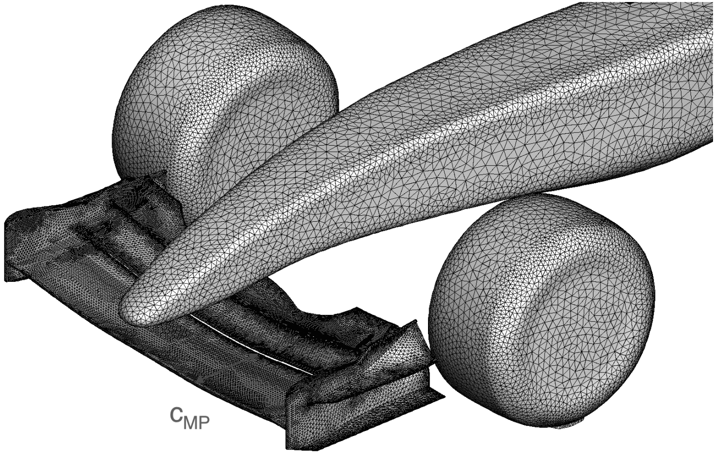
Source: www.nektar.info



CFD Simulation

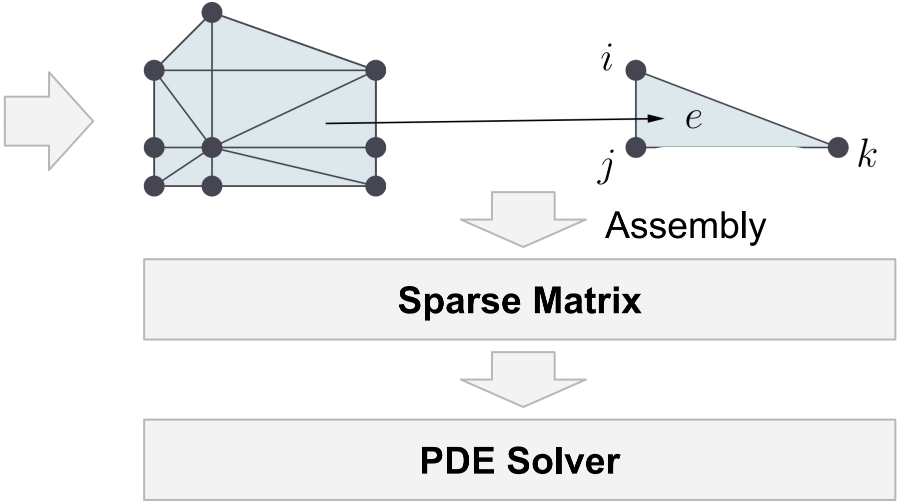
Finite Element Methods

Source: www.nektar.info



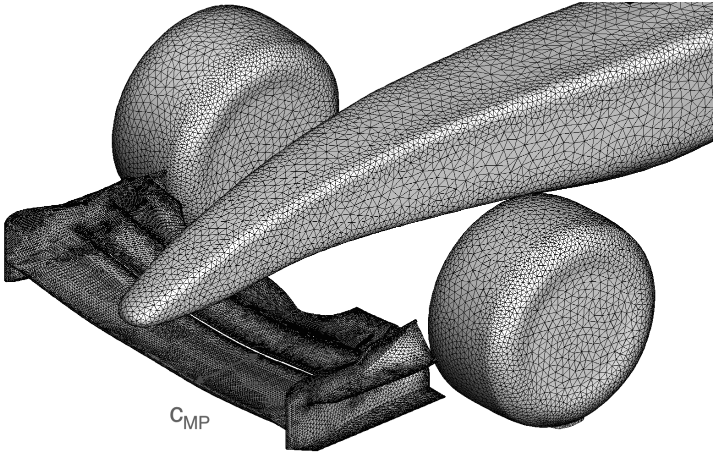
Mesh over unstructured domain

Mesh elements



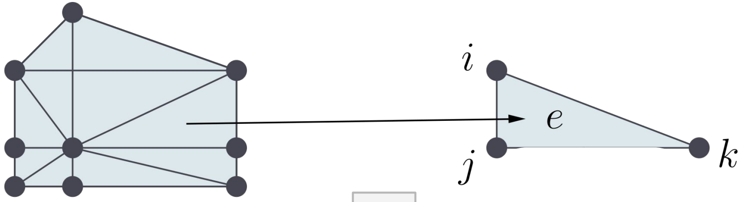
Finite Element Methods

Source: www.nektar.info



Mesh over unstructured domain

Mesh elements



Assembly

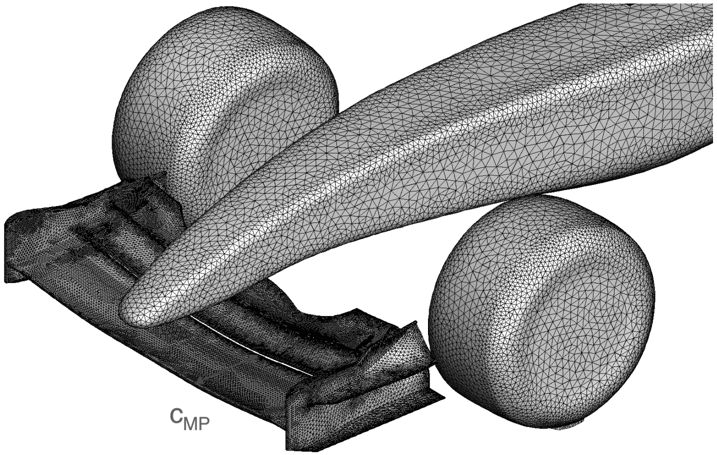
Sparse Matrix

PDE Solver

Linear Solver

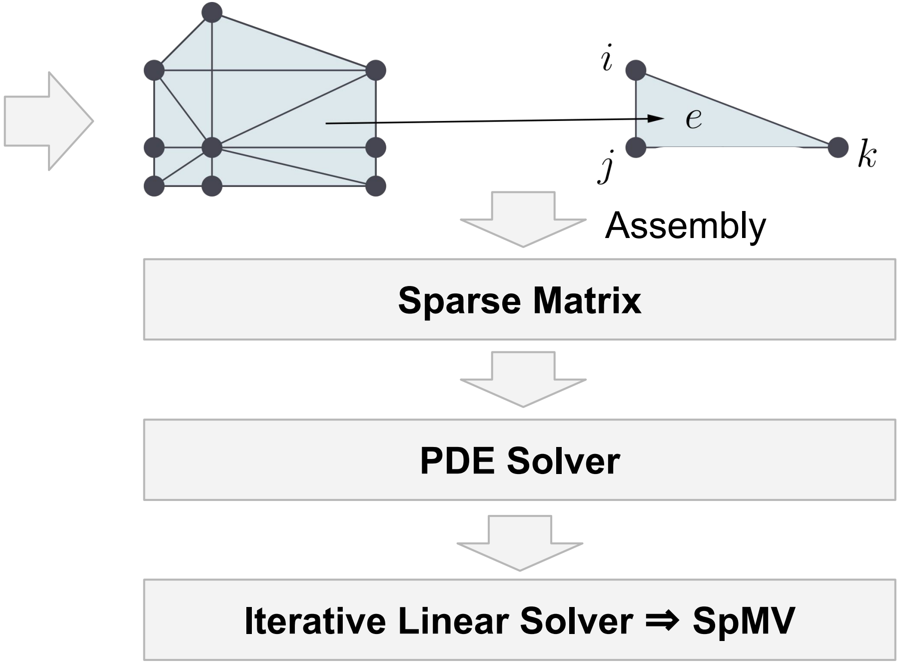
Finite Element Methods

Source: www.nektar.info



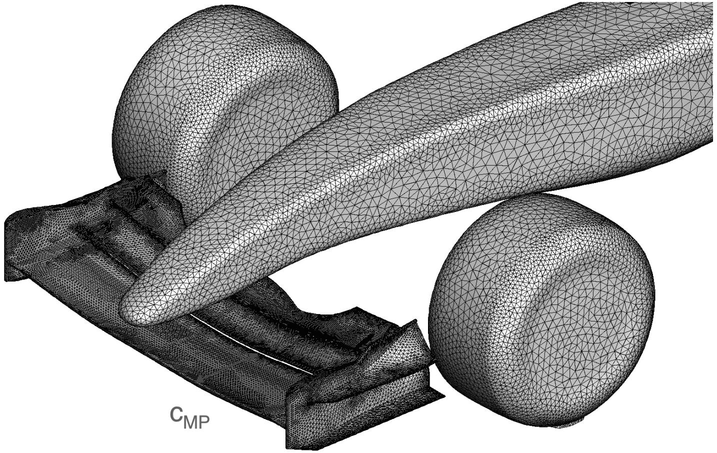
Mesh over unstructured domain

Mesh elements



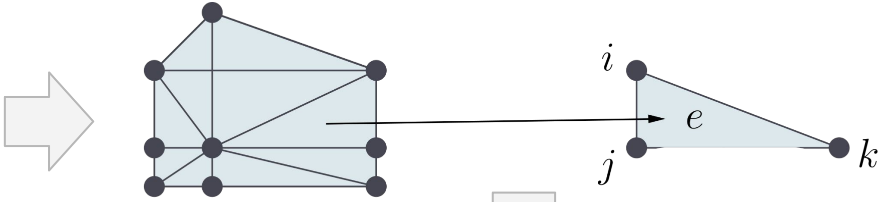
Finite Element Methods

Source: www.nektar.info



Mesh over unstructured domain

Mesh elements



Assembly

Sparse Matrix

PDE Solver

Vector Gather/Scatter [Burovskiy FPL15]

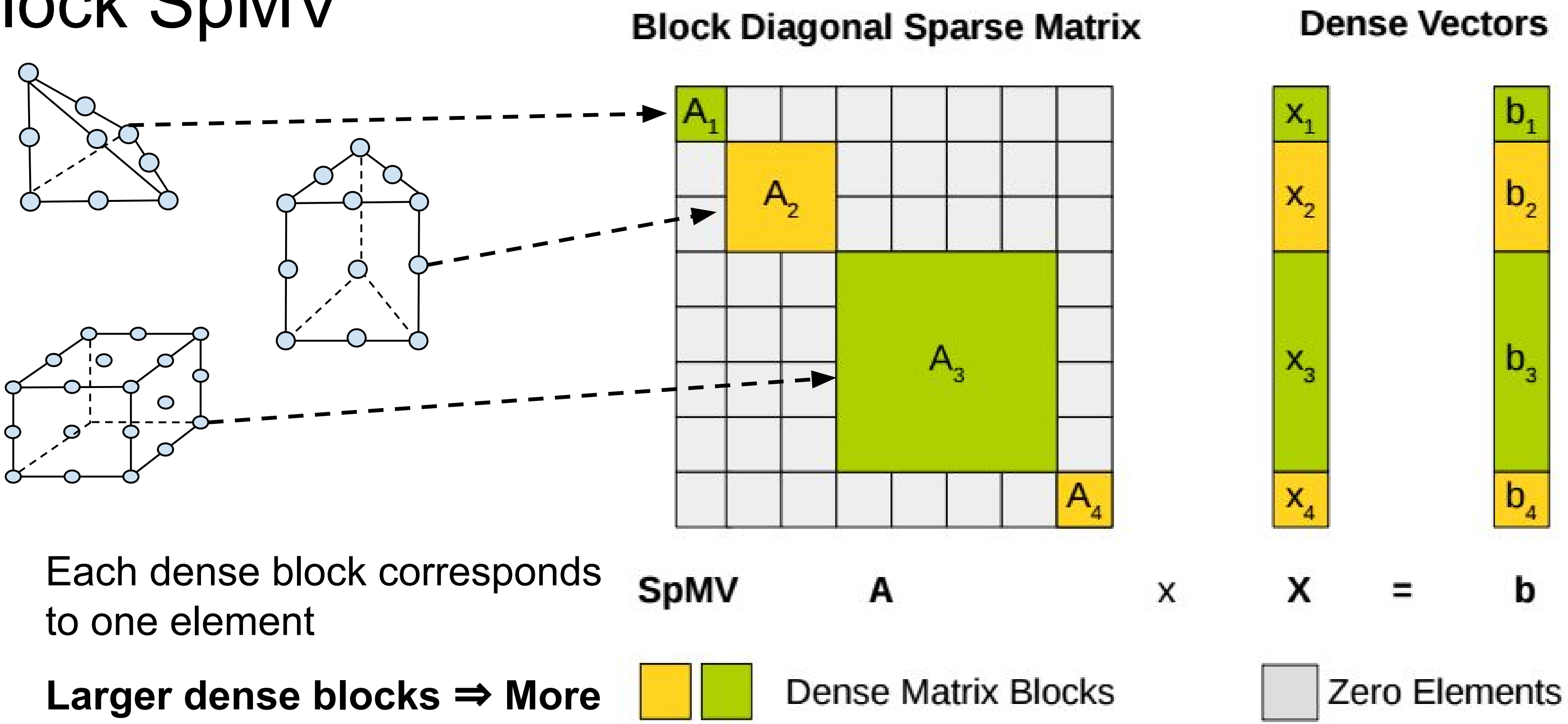
+

Block Diagonal SpMV (this work)

Overview

- *Point of departure*: focus on high order, spectral HP FEM, with local assembly
  - **block diagonal SpMV** (this work) vs **generic SpMV** (prior work)

Block SpMV



- Each dense block corresponds to one element
- **Larger dense blocks ⇒ More structured computation**

Overview

- *Point of departure*: focus on high order, spectral HP FEM, with local assembly
  - **block diagonal SpMV** (this work) vs **generic SpMV** (prior work)

Overview

- *Point of departure*: focus on high order, spectral HP FEM, with local assembly
  - **block diagonal SpMV** (this work) vs **generic SpMV** (prior work)

- *Contributions*:
  - Optimised architecture and implementation for block diagonal SpMV
  - Resource constrained performance model for the proposed architecture
  - Automated method to customise the architecture based on mesh parameters

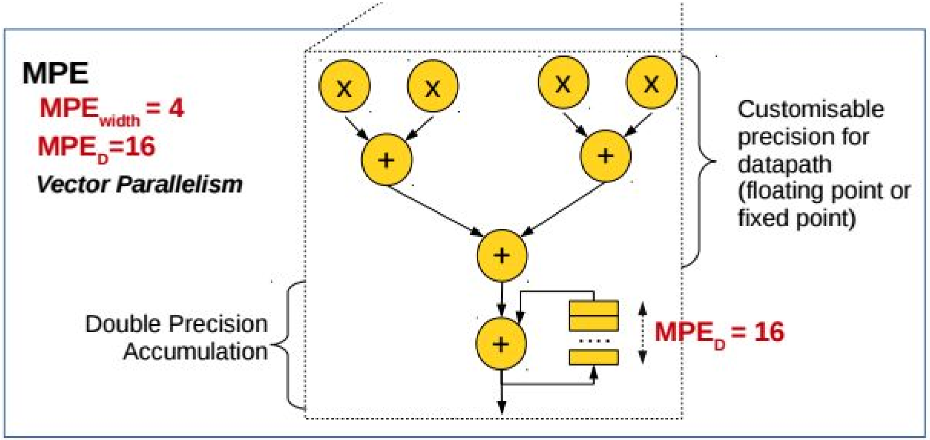
Overview

- *Point of departure*: focus on high order, spectral HP FEM, with local assembly
  - **block diagonal SpMV** (this work) vs **generic SpMV** (prior work)

- *Contributions*:
  - Optimised architecture and implementation for block diagonal SpMV
  - Resource constrained performance model for the proposed architecture
  - Automated method to customise the architecture based on mesh parameters

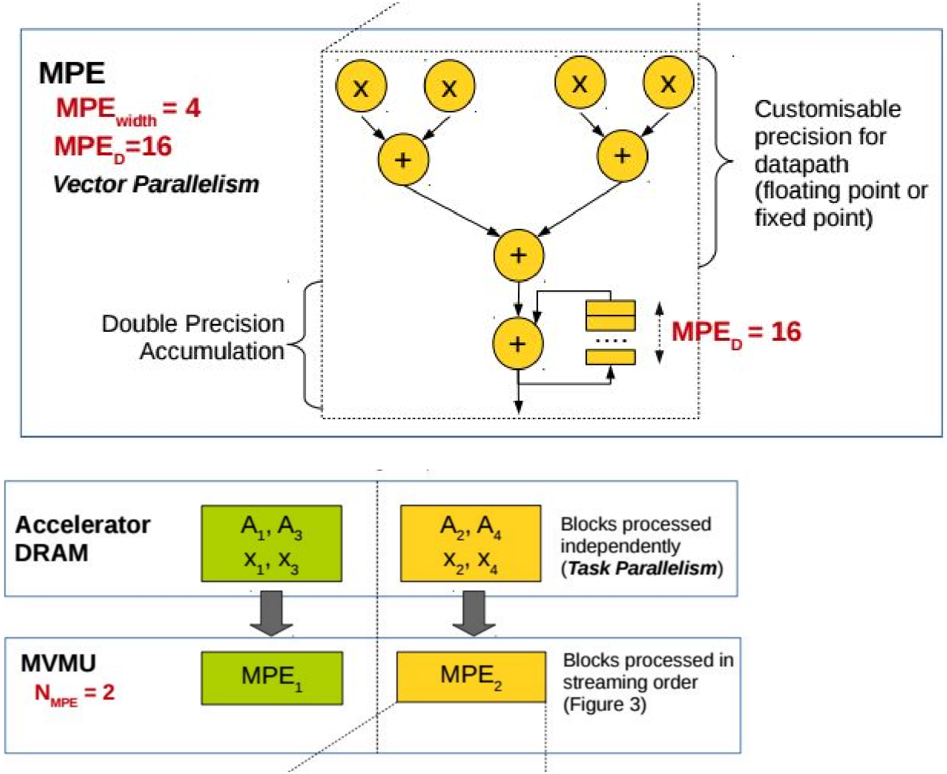
- *Result*: a custom, mesh-specific architecture generator
  - Maximise throughput/area ⇒ fit larger meshes & improve performance

Architecture



- Each MPE has
  - Independent memory channel
  - Customisable precision datapath
  - Variable depth FIFO - support block variations at runtime

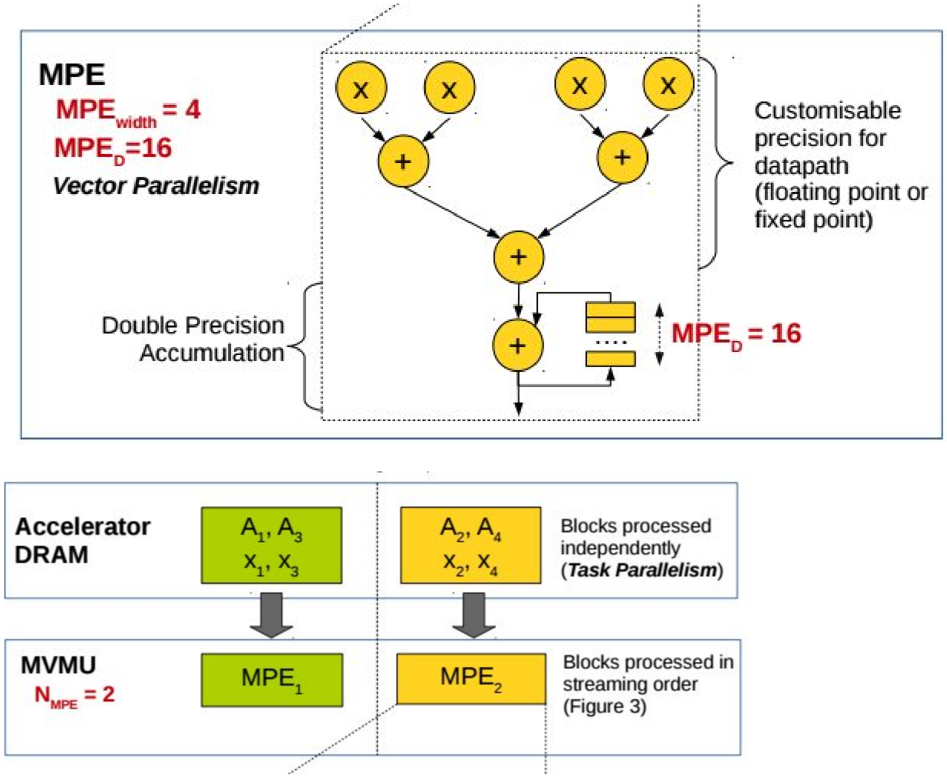
Architecture



- Each MPE has
  - Independent memory channel
  - Customisable precision datapath
  - Variable depth FIFO - support block variations at runtime

- Design:
  - Parametric: NMPEs, MPEwidth
  - Task vs Data Parellelism tradeoff
  - ⇒ Mesh specific optimal config.

Architecture



- Each MPE has
  - Independent memory channel
  - Customisable precision datapath
  - Variable depth FIFO - support block variations at runtime
- Design:
  - Parametric: NMPEs, MPEwidth
  - Task vs Data Parellelism tradeoff
  - ⇒ Mesh specific optimal config.
- Block SpMV advantages:

  ⇒ Simplified control (format decoding)

  ⇒ Reduced metadata

  ⇒ Simplified reduction circuit

Parameter Extraction

- Assume matrix is block diagonal
- Extract mesh parameters: size & number of blocks for each element
- In DSE: find and synthesise optimal architectures
- At runtime: select the appropriate architecture

Algorithm 2 Extract BS_i, NB_i for the sparse matrix A

```
function ExtractBlockMatrixParams(A)
    SortEntries(A)                          ▷ Sort by row and column index
    occ ← {}                                ▷ Dictionary of BS_i and NB_i
    for i ∈ 0 ... A.nrows do
        (first, last) ← RowSpan(i)
        bs ← last − first                   ▷ Assume new block size
        startPosition ← i                   ▷ Upper left corner of block
        while i − startPosition < bs do
            if last > startPosition + bs then
                ERROR! Not a block diagonal matrix
            else if first < startPosition then
                ERROR! Not a block diagonal matrix
            end if
            i ← i + 1
            (first, last) ← RowSpan(i)
        end while
        occ[bs] ← occ[bs] + 1               ▷ update block count
    end for
    return occ                              ▷ return the frequency map of block sizes
end function
```

Performance Model

- Mesh parameters ⇒ optimal architecture parameters

- Performance:

$$N_{Cycles} = \sum_i CyclesPerBlock(B_i) \times Tasks(B_i) \quad (3)$$

$$= \sum_i \left(\left\lceil \frac{BS_i}{MPE_{width}} \right\rceil \times BS_i \right) \times \left\lceil \frac{NB_i}{N_{MPE}} \right\rceil \quad (4)$$

- Resource usage:

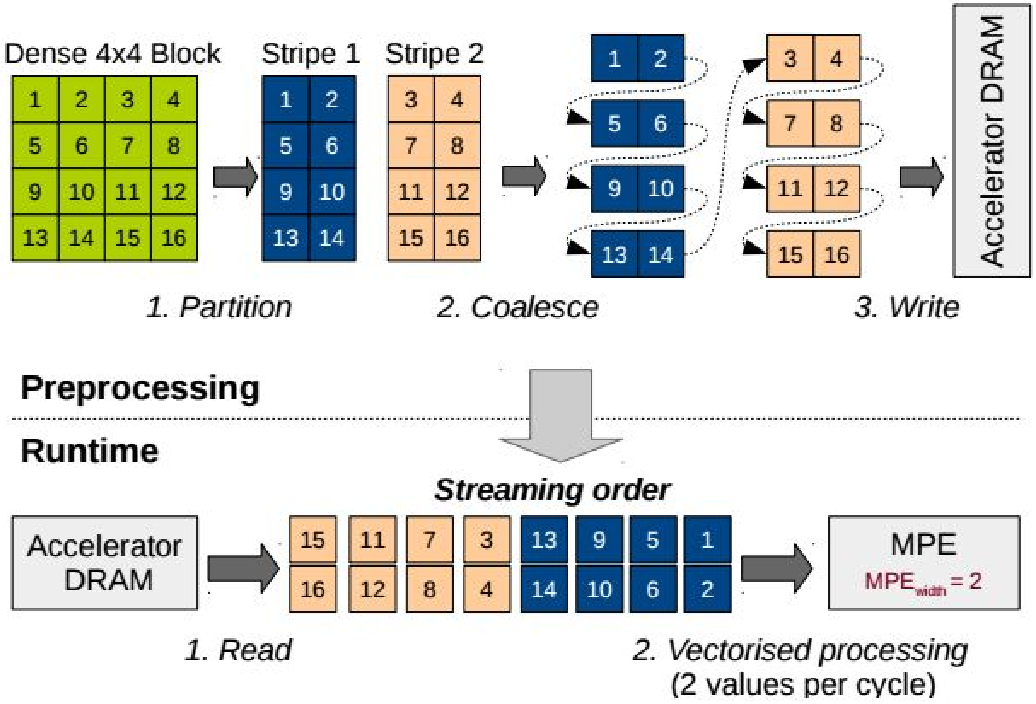
$$R = R_{Static} + \quad (5)$$
$$N_{MPE} \times (MPE_{width} \times (R_{+} + R_X) + R_B + R_{DRAM})$$

- Functional, hardware constraints ⇒ See paper for details

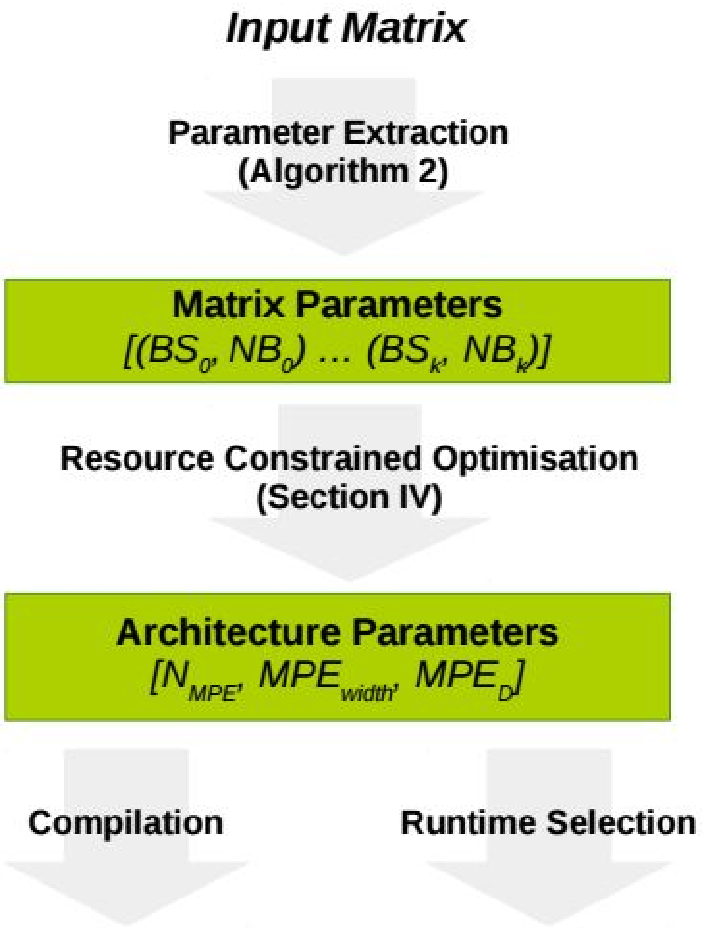
Runtime

- Software layer - can be integrated in existing FEM software packages

- Reorder to ***enforce linear access pattern in DRAM***
  - Maximise throughput
  - Minimise control logic

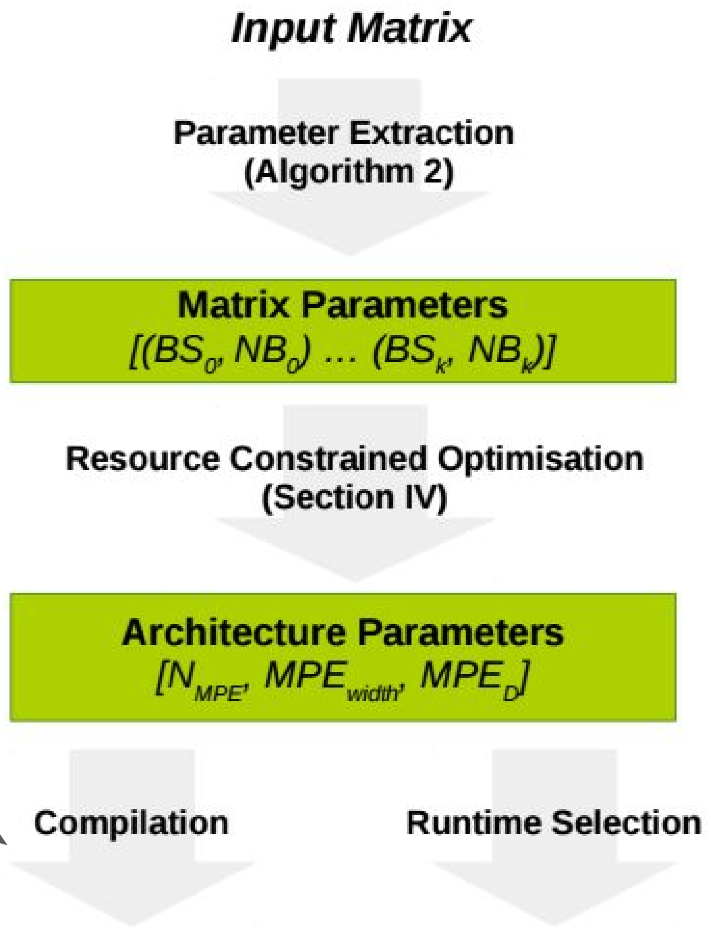


Putting it Together



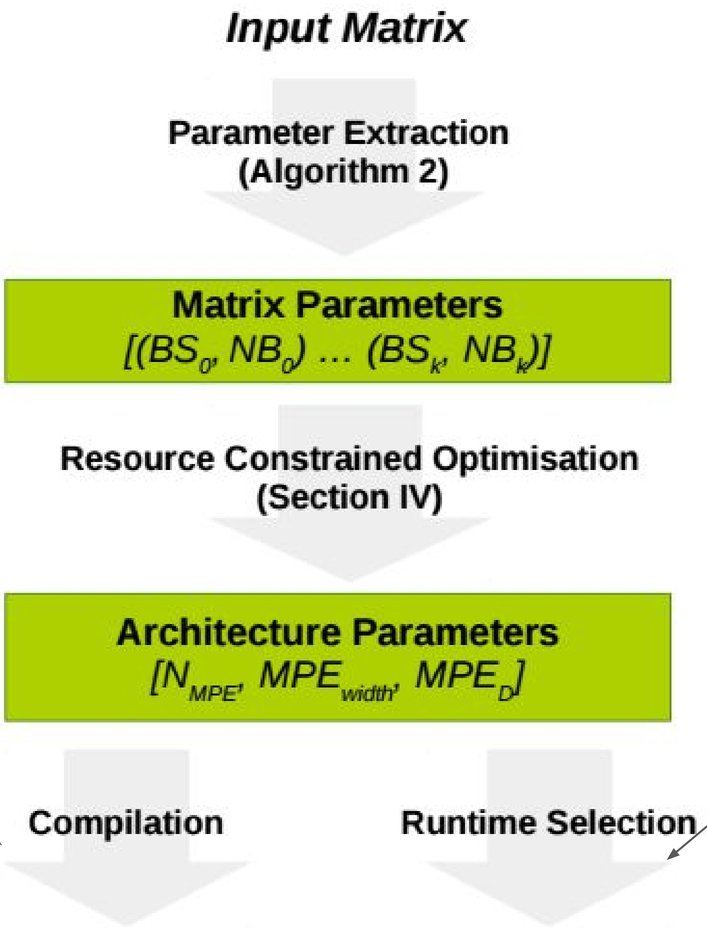
Putting it Together

Offline tuning: build a repository of customised architectures from a set of mesh instances



Putting it Together

Offline tuning: build a repository of customised architectures from a set of mesh instances



Runtime: select the optimal architecture for an input mesh instance

Evaluation

Evaluation

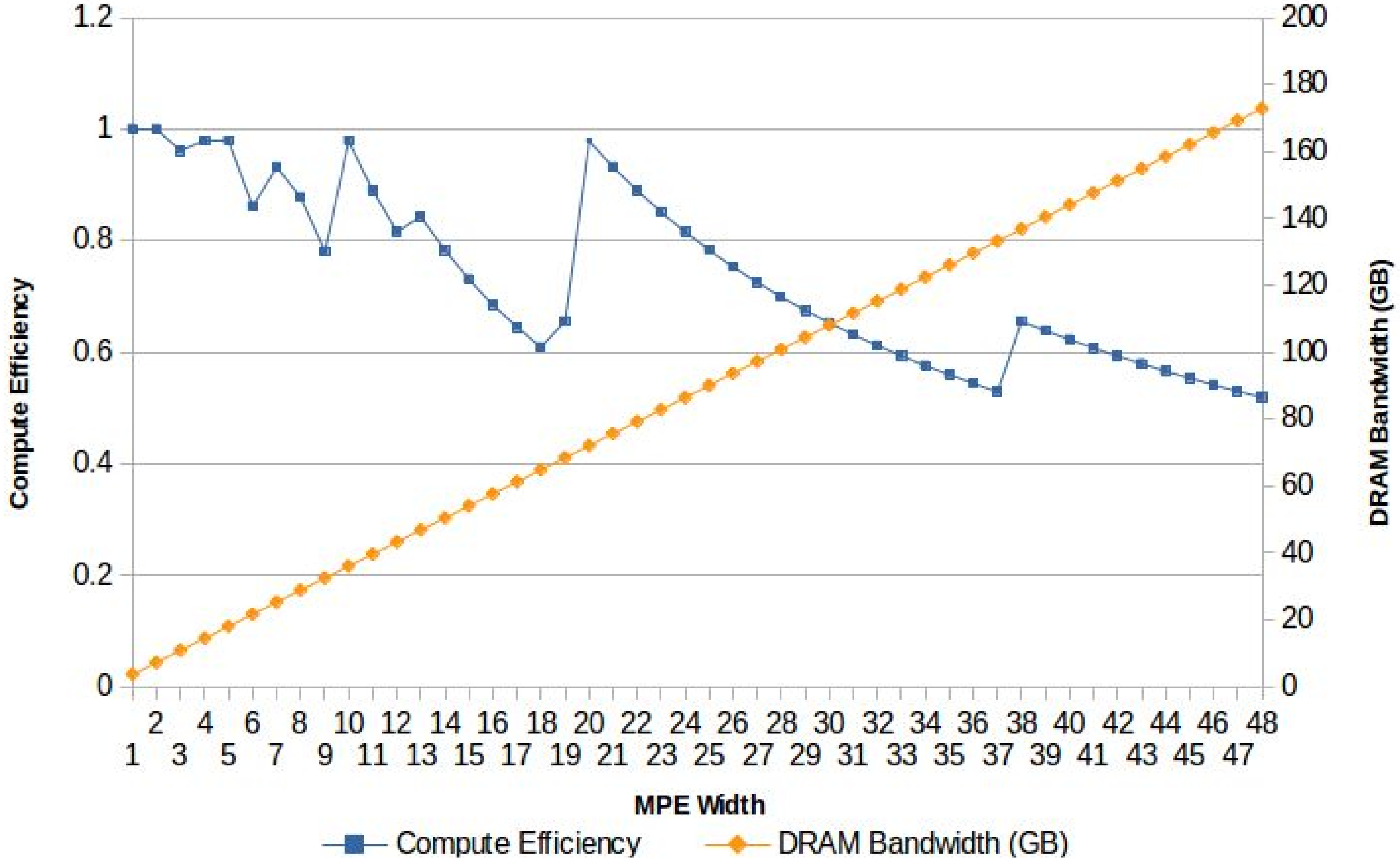
- **Implementation**
  - Design: MaxComplier + MaxJ dataflow language
  - FPGA Server: Maxeler Max 4 Maia (Stratix VSG, 48GB DRAM, per board)
  - Software: C++14, G++ 5.2
  - CPU Server: Dual Intel Xeon E5-2640, 64GB DRAM, Infiniband QSFP
  - Place and route with Altera Quartus 14.1
  - Available as extension to the CASK framework [Grigoras et al, FPGA 16]:
    - http://caskorg.github.io/cask/

- **Reference software -** Nektar++ FEM Package, http://www.nektar.info/

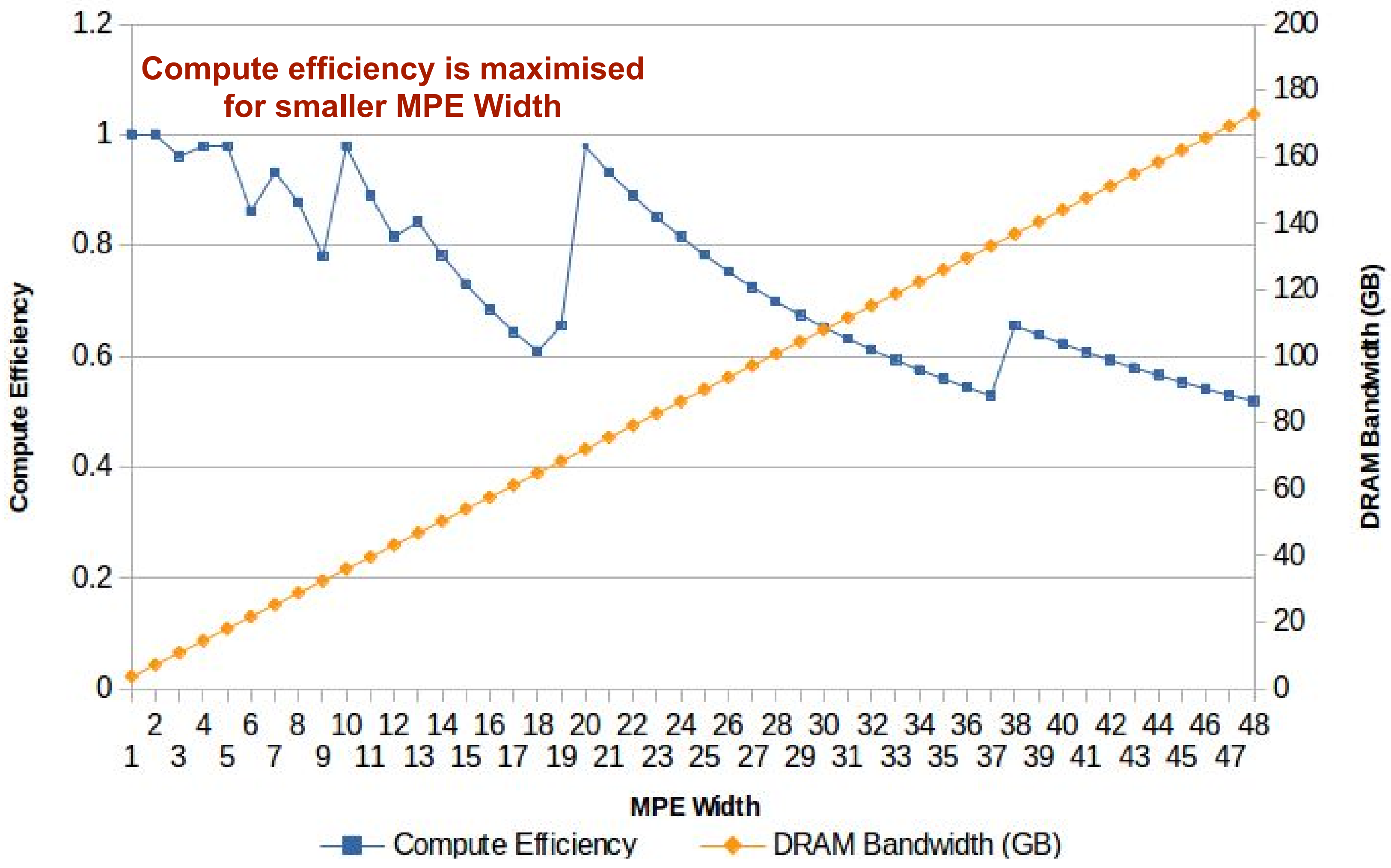
- **Reference hardware**
  - [Burovskiy et al, FPL 15], Nektar++ Accelerated FEM

Experiments

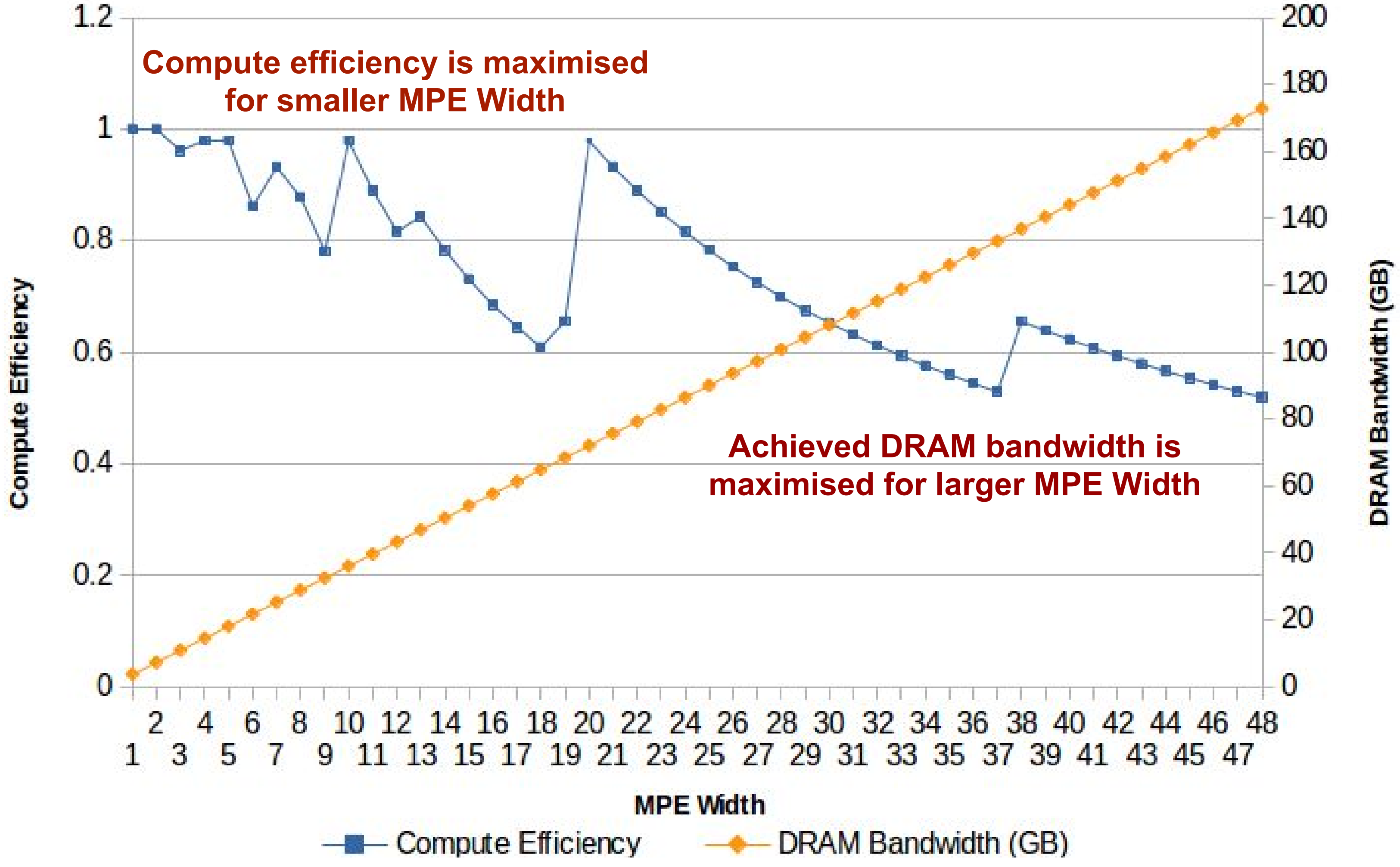
1. *What is the benefit of tuning architecture based on mesh properties*?
    a. Fixed mesh (NACA 1L, [Burovskiy et al, FPL 2015]) - optimal architecture

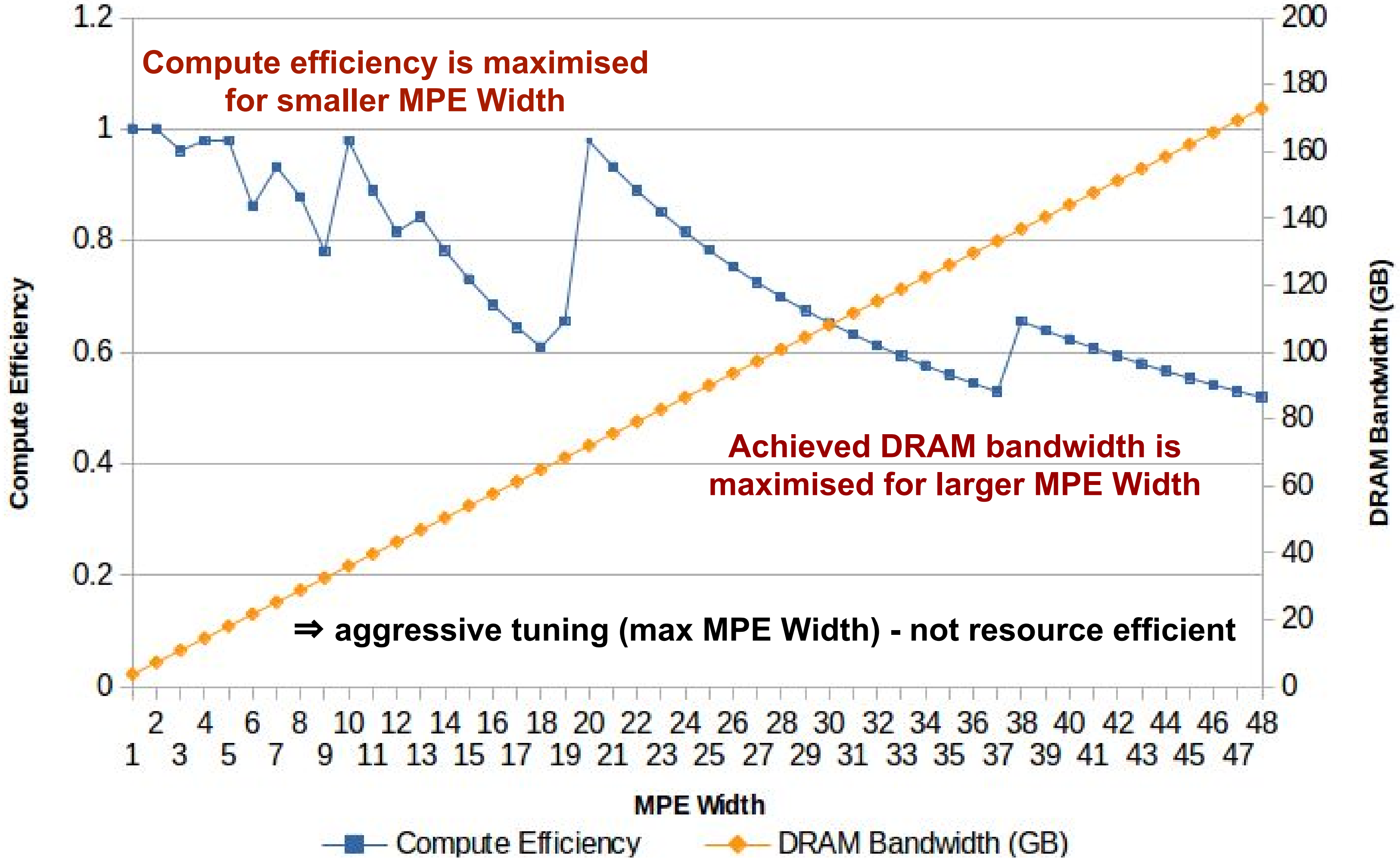


Compute Efficiency
DRAM Bandwidth (GB)
MPE Width
1.2
1
0.8
0.6
0.4
0.2
0
200
180
160
140
120
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
Compute Efficiency
DRAM Bandwidth (GB)



Compute efficiency is maximised
for smaller MPE Width
Compute Efficiency
DRAM Bandwidth (GB)
MPE Width
Compute Efficiency
DRAM Bandwidth (GB)


Compute efficiency is maximised
for smaller MPE Width
Achieved DRAM bandwidth is
maximised for larger MPE Width
Compute Efficiency
DRAM Bandwidth (GB)
MPE Width
1.2
1
0.8
0.6
0.4
0.2
0
200
180
160
140
120
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48


Compute efficiency is maximised
for smaller MPE Width
Achieved DRAM bandwidth is
maximised for larger MPE Width
⇒ aggressive tuning (max MPE Width) - not resource efficient
Compute Efficiency
DRAM Bandwidth (GB)
MPE Width
Compute Efficiency
DRAM Bandwidth (GB)

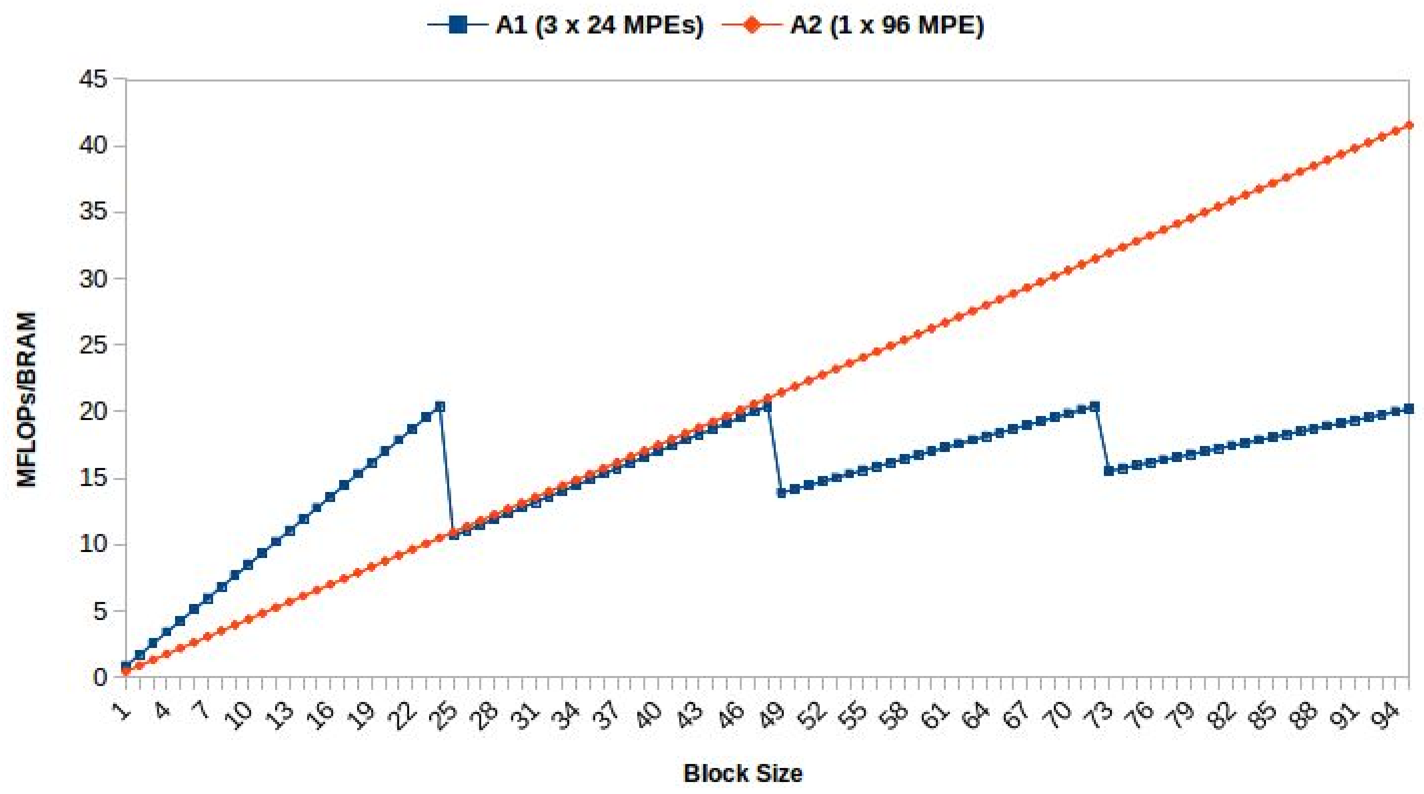
Experiments

1. *What is the benefit of tuning architecture based on mesh properties*?
   a. Fixed mesh (NACA 1L, [Burovskiy et al, FPL 2015]) - optimal architecture
      ⇒ find architecture with good efficiency
      ⇒ improve performance s.t. resource usage

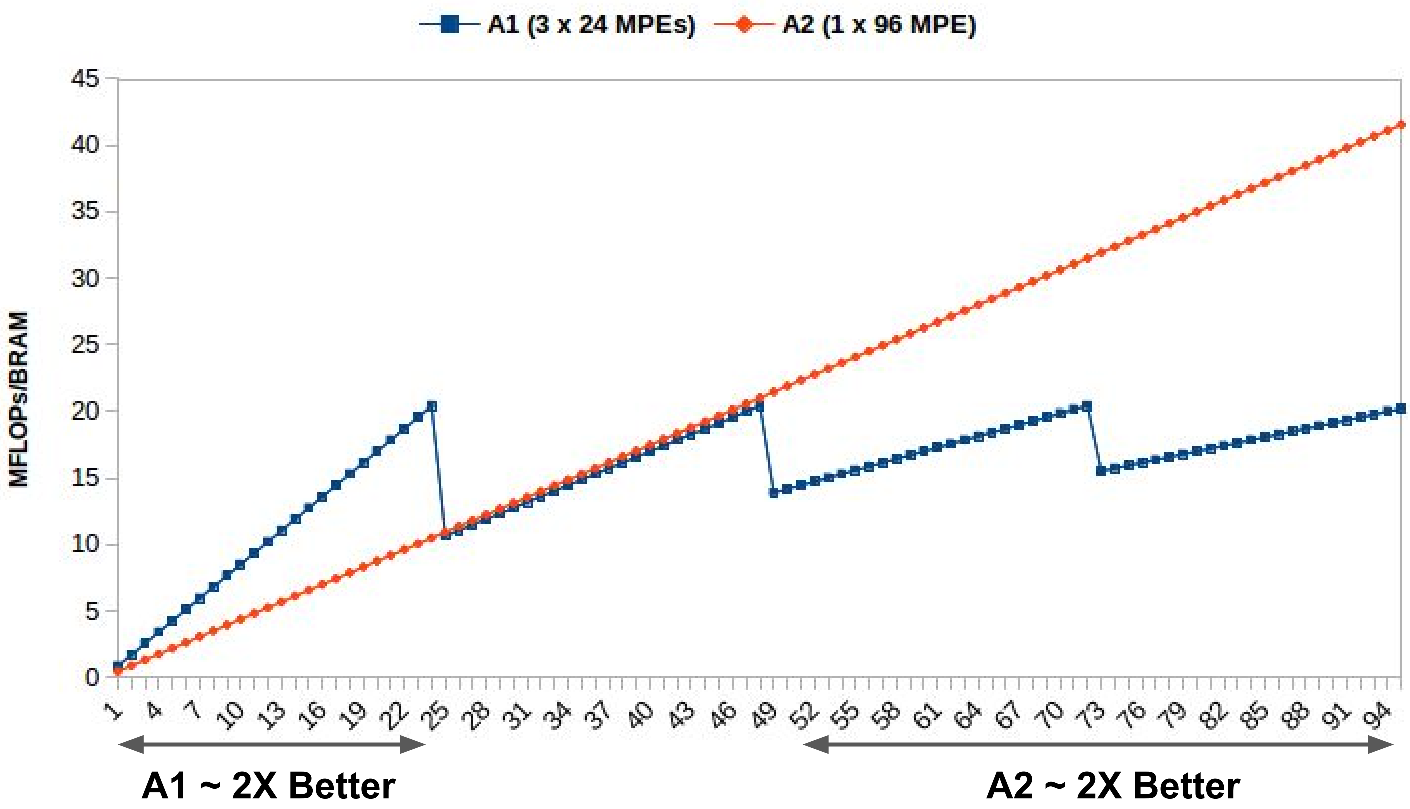
Experiments

1. *What is the benefit of tuning architecture based on mesh properties*?
   a. Fixed mesh (NACA 1L, [Burovskiy et al, FPL 2015]) - optimal architecture
      ⇒ find architecture with good efficiency
      ⇒ improve performance s.t. resource usage

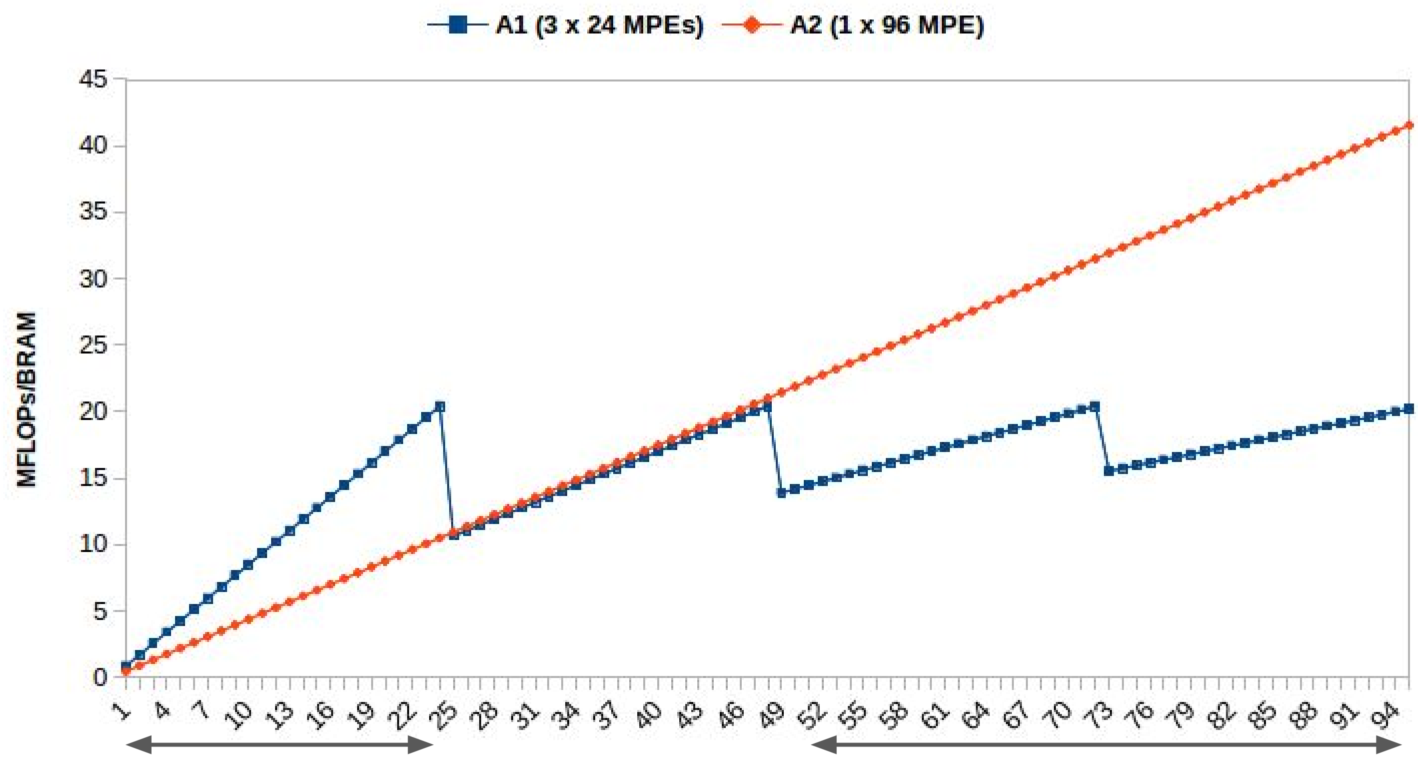
   b. Fixed architecture, variable mesh, data parallel vs task parallel



A1 (3 x 24 MPEs)
A2 (1 x 96 MPE)
MFLOPs/BRAM
45
40
35
30
25
20
15
10
5
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94
Block Size

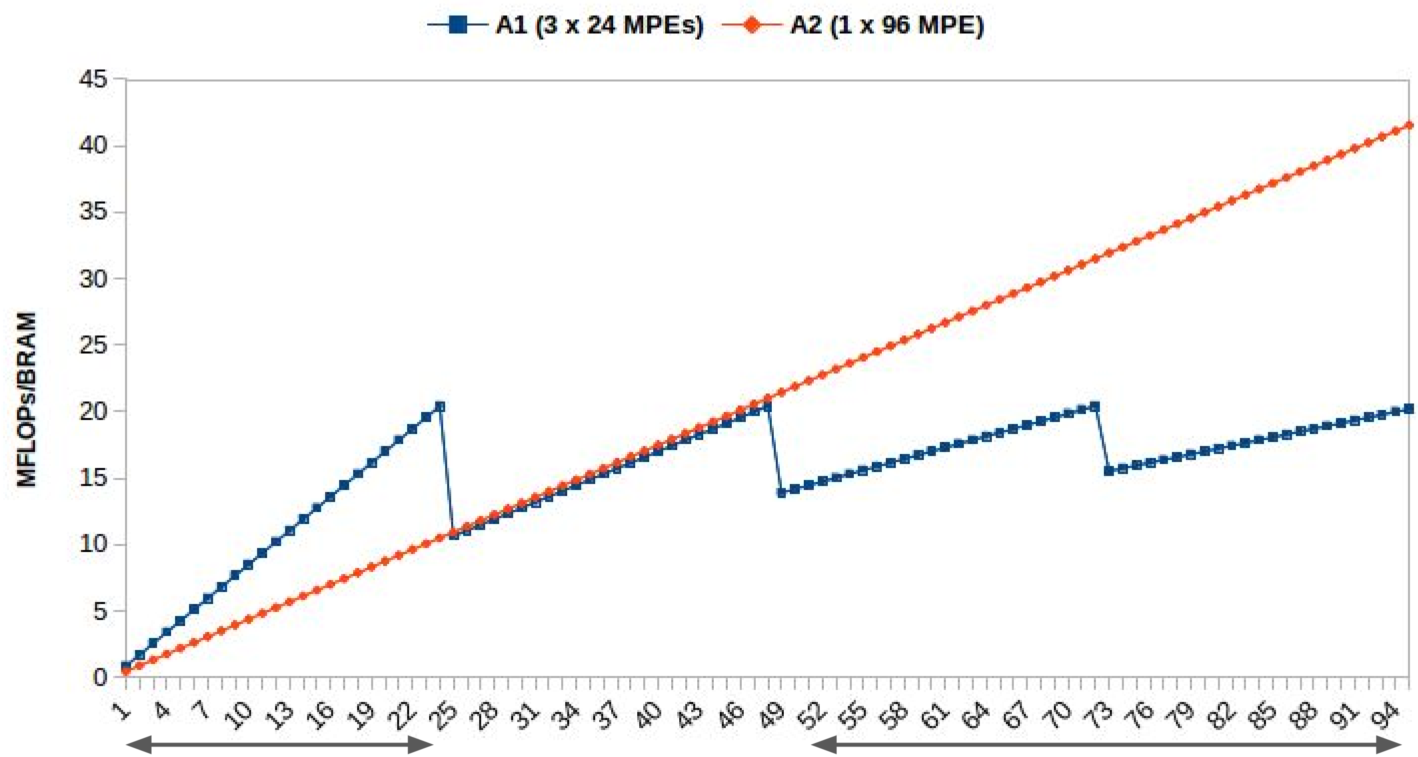


A1 (3 x 24 MPEs)
A2 (1 x 96 MPE)
MFLOPs/BRAM
45
40
35
30
25
20
15
10
5
0
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
67
70
73
76
79
82
85
88
91
94
A1 ~ 2X Better
A2 ~ 2X Better



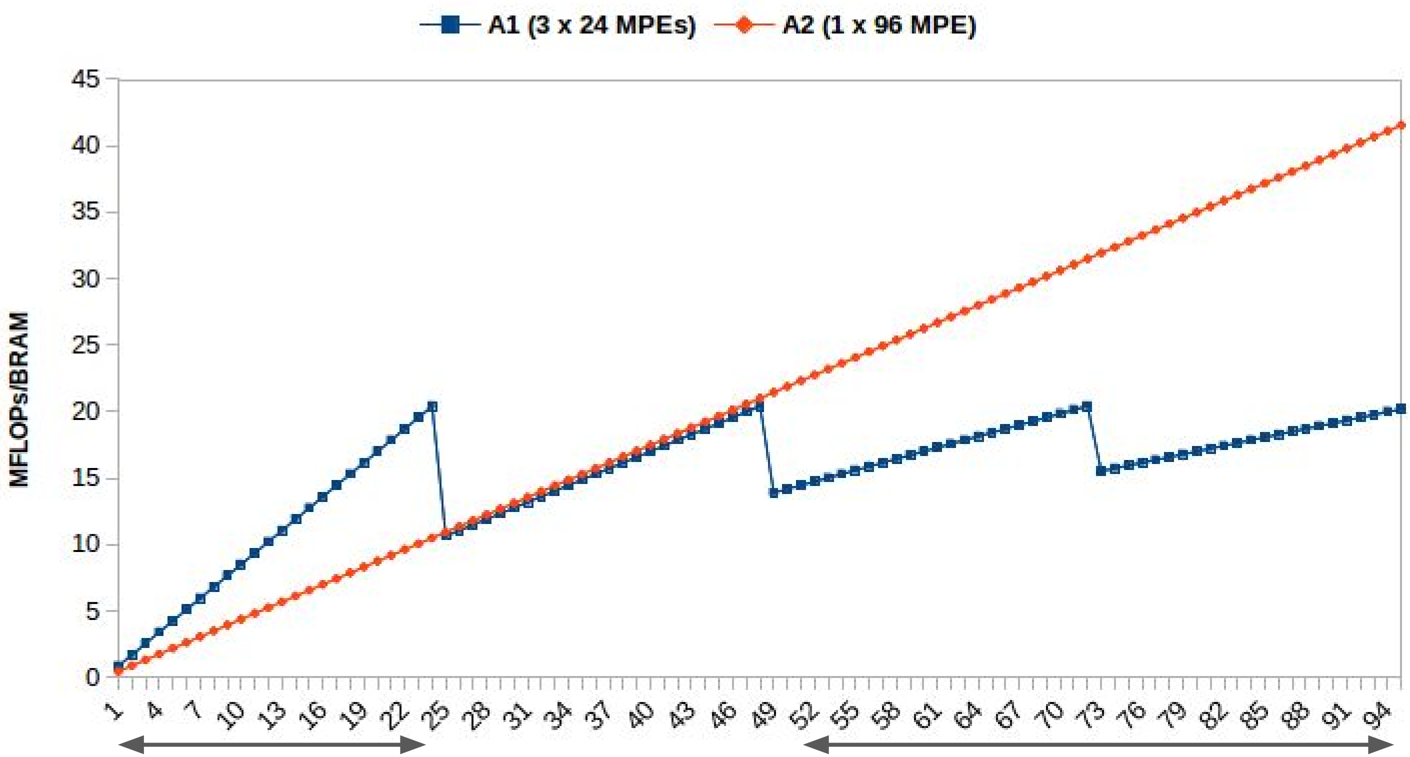
A1 ~ 2X Better
+Mem. Chan., -Vector lanes, 1058 BRAMs

A2 ~ 2X Better
- Mem. Chan., + Vector lanes, 686 BRAMs



A1 ~ 2X Better
+Mem. Chan., -Vector lanes, 1058 BRAMs
⇒ Good for small blocks

A2 ~ 2X Better
- Mem. Chan., + Vector lanes, 686 BRAMs
⇒ Good for large blocks



A1 ~ 2X Better
+Mem. Chan., -Vector lanes, 1058 BRAMs
⇒ Good for small blocks

A2 ~ 2X Better
- Mem. Chan., + Vector lanes, 686 BRAMs
⇒ Good for large blocks

Experiments

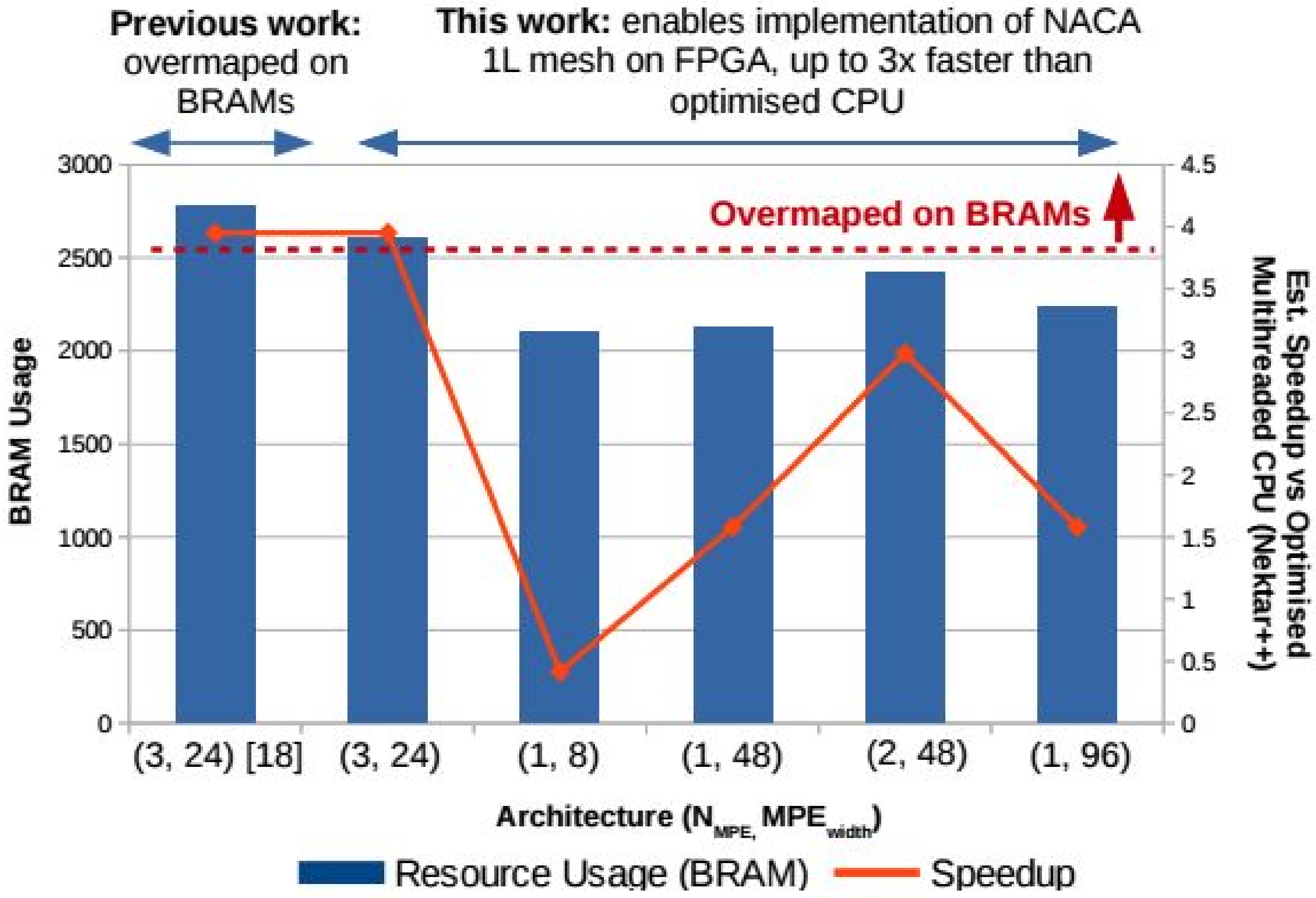
1. *What is the benefit of tuning architecture based on mesh properties*?
   a. Fixed mesh (NACA 1L, [Burovskiy et al, FPL 2015]) - optimal architecture
      ⇒ find architecture with good efficiency
      ⇒ improve performance s.t. resource usage
   b. Fixed architecture, variable mesh, data parallel vs task parallel
      ⇒ select optimal MPE Width and N Mpe for given mesh
      ⇒ improve performance, reduce resource usage

Experiments

1. *What is the benefit of tuning architecture based on mesh properties*?
   a. Fixed mesh (NACA 1L, [Burovskiy et al, FPL 2015]) - optimal architecture
      ⇒ find architecture with good efficiency
      ⇒ improve performance s.t. resource usage

   b. Fixed architecture, variable mesh, data parallel vs task parallel
      ⇒ select optimal MPE Width and N Mpe for given mesh
      ⇒ improve performance, reduce resource usage

2. *What is the expected benefit for a full FEM implementation?*
   a. Baseline, Nektar++ implementation from [Burovskiy et al, FPL 2015]



Previous work: overmaped on BRAMs
This work: enables implementation of NACA 1L mesh on FPGA, up to 3x faster than optimised CPU
Overmaped on BRAMs
BRAM Usage
3000
2500
2000
1500
1000
500
0
Est. Speedup vs Optimised Multihreaded CPU (Nektar++)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
(3, 24) [18]
(3, 24)
(1, 8)
(1, 48)
(2, 48)
(1, 96)
Architecture (N_{MPE}, MPE_{width})
Resource Usage (BRAM)
Speedup

Experiments

1. *What is the benefit of tuning architecture based on mesh properties*?
   a. Fixed mesh (NACA 1L, [Burovskiy et al, FPL 2015]) - optimal architecture
      ⇒ find architecture with good efficiency
      ⇒ improve performance s.t. resource usage

   b. Fixed architecture, variable mesh, data parallel vs task parallel
      ⇒ select optimal MPE Width and N Mpe for given mesh
      ⇒ improve performance, reduce resource usage

2. *What is the expected benefit for a full FEM implementation?*
   a. Baseline, Nektar++ implementation from [Burovskiy et al, FPL 2015]
      ⇒ enabling larger problem sizes, not supported by previous work.
      ⇒ enable a good proportion of the projected speedup (3X over CPU)

Conclusion

1. *Proposed*:
   a. FPGA architecture optimised for variable-size block diagonal SpMV
   b. method to extract customisation parameters directly from mesh instance
   c. software to integrate with existing FEM package, Nektar++

2. *Achieved:*
   a. Fit larger FEM problems on a single FPGA
   b. 3X speedup over optimised CPU

3. *Future*: exploration of additional trade-offs & parameters

That’s it folks! Thank you!