Recent Advances in Batch Linear Algebra Computation

Mawussi Zounon

joint work with

Samuel Relton

M\crNA
Manchester Numerical Analysis

Overview

Aim: Design of efficient batch linear algebra kernels

Target: batch of very small matrices (2×2 to 32×32)

Example: solve $C^{(i)} \leftarrow \alpha^{(i)} A^{(i)} B^{(i)} + \beta^{(i)} C^{(i)}$, i = 1:batch_count

Common Approach: `OpenMP for loop`

Challenge: keep CPU cores running at full efficiency

Manchester Numerical Analysis

Outline

M\crNA
Manchester Numerical Analysis

The OpenMP for loop approach

Example of batch DGEMM

- 1: **for** $i \leftarrow 1$ to *batch_count* **do**
  2: `#pragma omp parallel for`
  3: cblas_dgemm(...)
  4: **end for**

- Principle: Each core solve one problem at a time

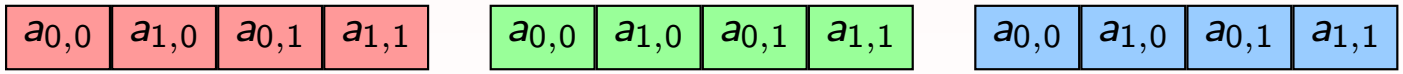
- Advantages:
  - Easy to implement
  - Use existing vendor optimized kernels

- Disadvantage:
  - Ineffective CPU cache exploitation
  - Fails to use vector units at full efficiency

M\crNA
Manchester Numerical Analysis

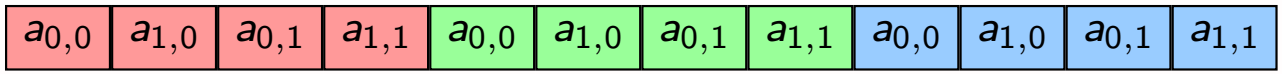
Batch of small matrices and CPU cache

Pointer to pointer layout:



- High memory access: thousands of matrices spread out in memory
- Ineffective cache use: matrices smaller than cache size

Stride layout:



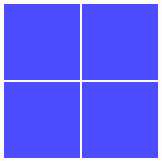
- Possible overhead: may require layout conversion
- Vectorization issue: may fail to use wide vector units at full efficiency

M\crNA
Manchester Numerical Analysis

Batch linear algebra and wide vector units

Issues of very small matrices

- Intel AVX-512: 8 double precision
- 2×2 matrix: 4 double precision
- Alternative: cross-matrix vectorization



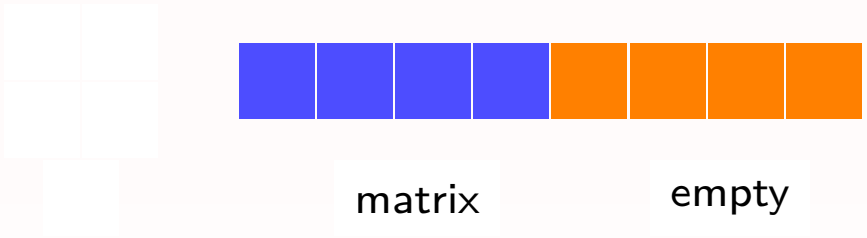
matrix

Intel AVX-512

Batch linear algebra and wide vector units

Issues of very small matrices

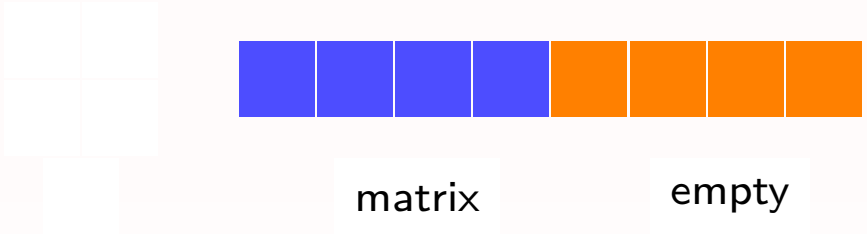
- Intel AVX-512: 8 double precision
- 2×2 matrix: 4 double precision
- Alternative: cross-matrix vectorization



Batch linear algebra and wide vector units

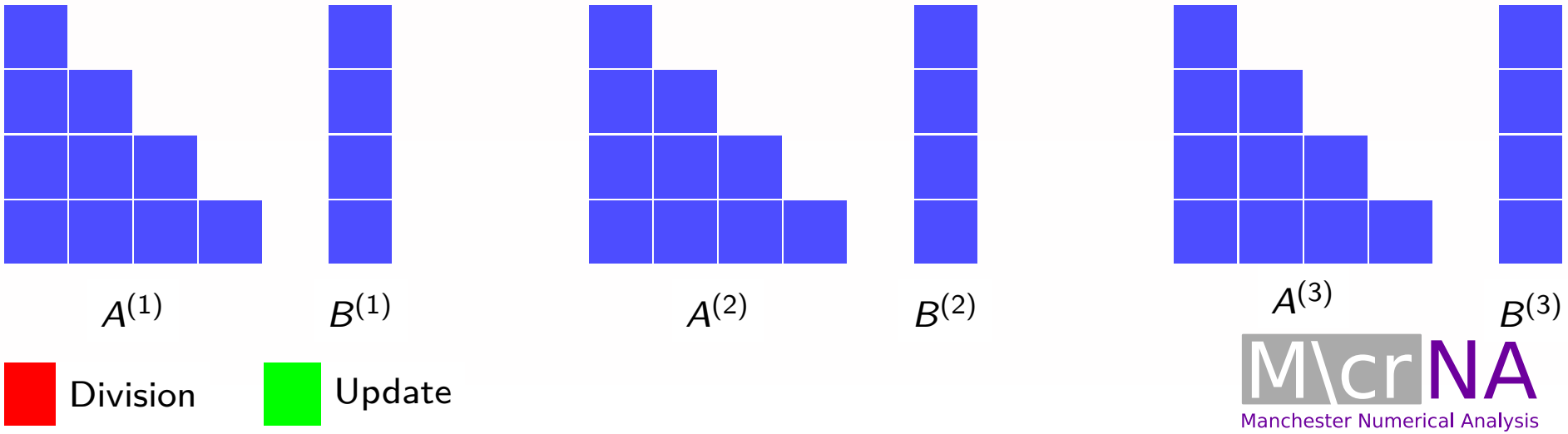
Issues of very small matrices

- Intel AVX-512: 8 double precision
- 2×2 matrix: 4 double precision
- Alternative: cross-matrix vectorization



Issues of sequential algorithms

- Synchronization point
- One division: Only 1 DP flop over 8

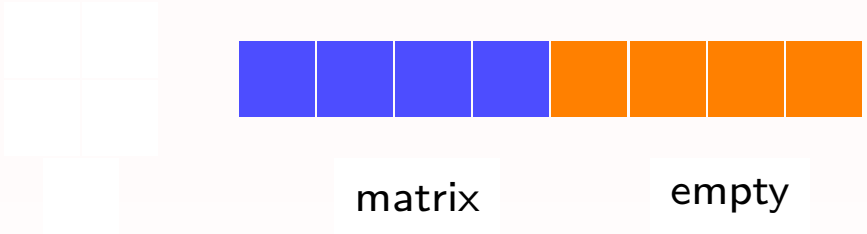


M\cr NA
Manchester Numerical Analysis

Batch linear algebra and wide vector units

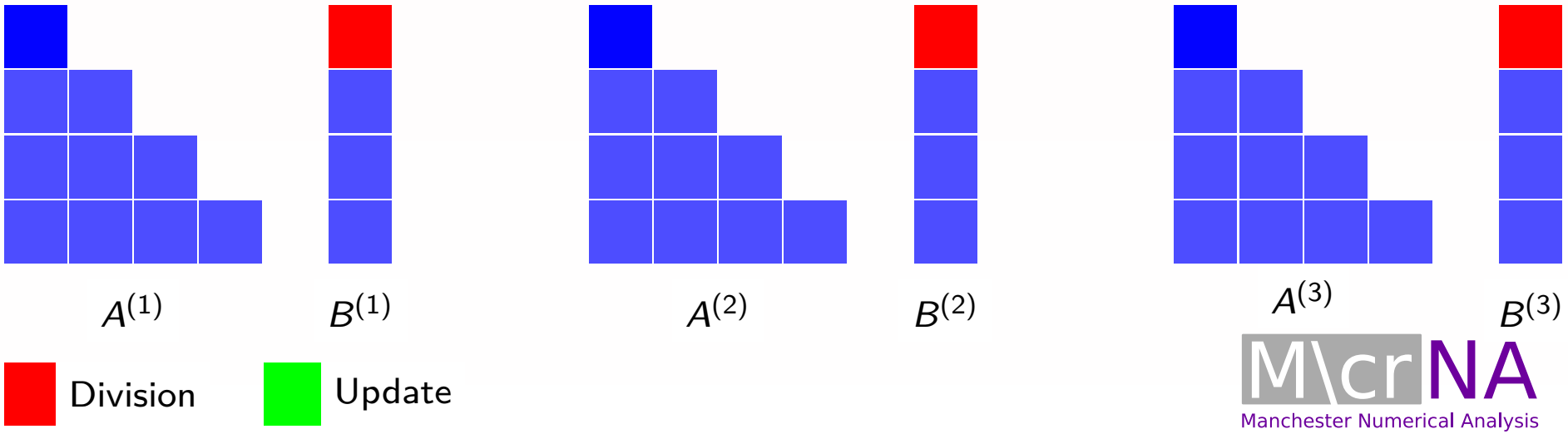
Issues of very small matrices

- Intel AVX-512: 8 double precision
- 2×2 matrix: 4 double precision
- Alternative: cross-matrix vectorization



Issues of sequential algorithms

- Synchronization point
- One division: Only 1 DP flop over 8

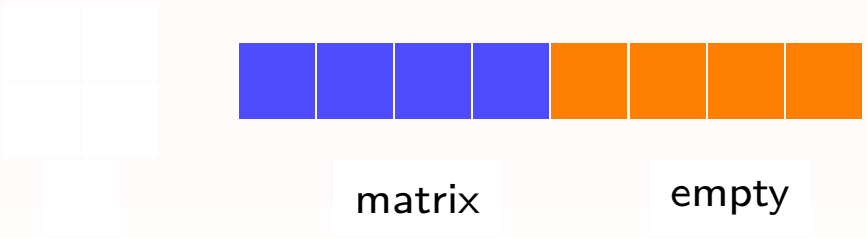


M\cr NA
Manchester Numerical Analysis

Batch linear algebra and wide vector units

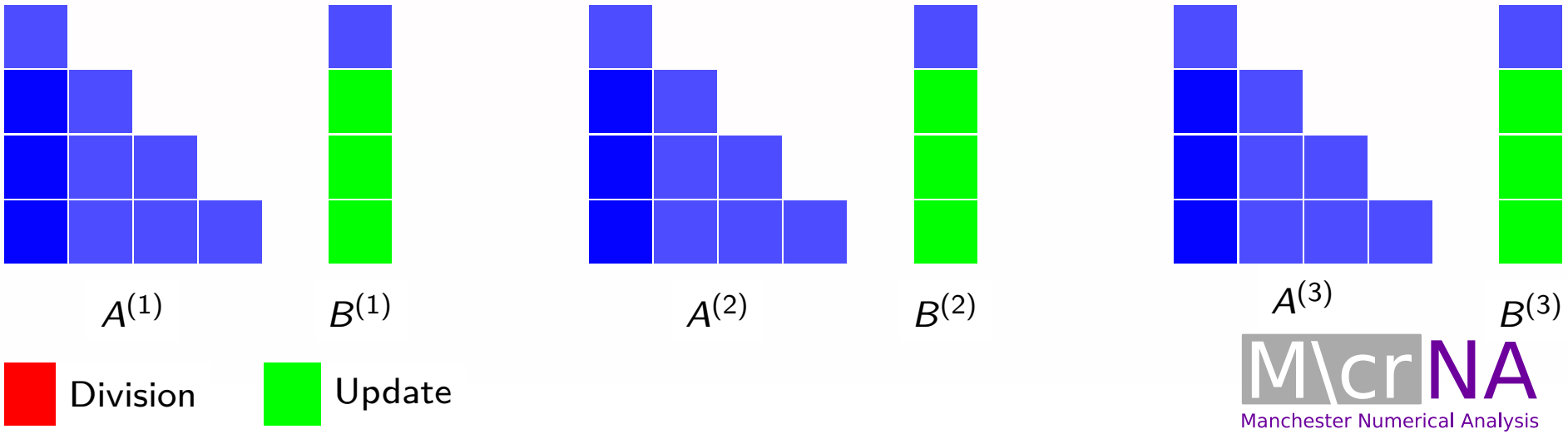
Issues of very small matrices

- Intel AVX-512: 8 double precision
- 2×2 matrix: 4 double precision
- Alternative: cross-matrix vectorization



Issues of sequential algorithms

- Synchronization point
- One division: Only 1 DP flop over 8

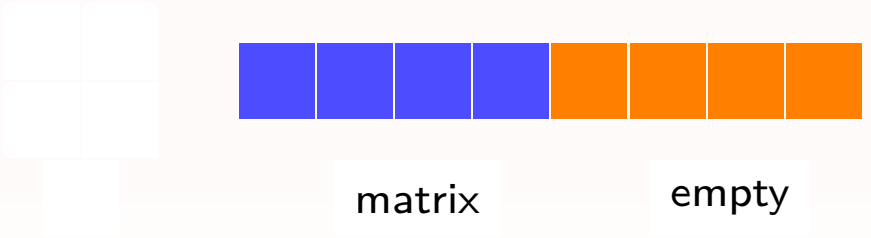


M\crNA
Manchester Numerical Analysis

Batch linear algebra and wide vector units

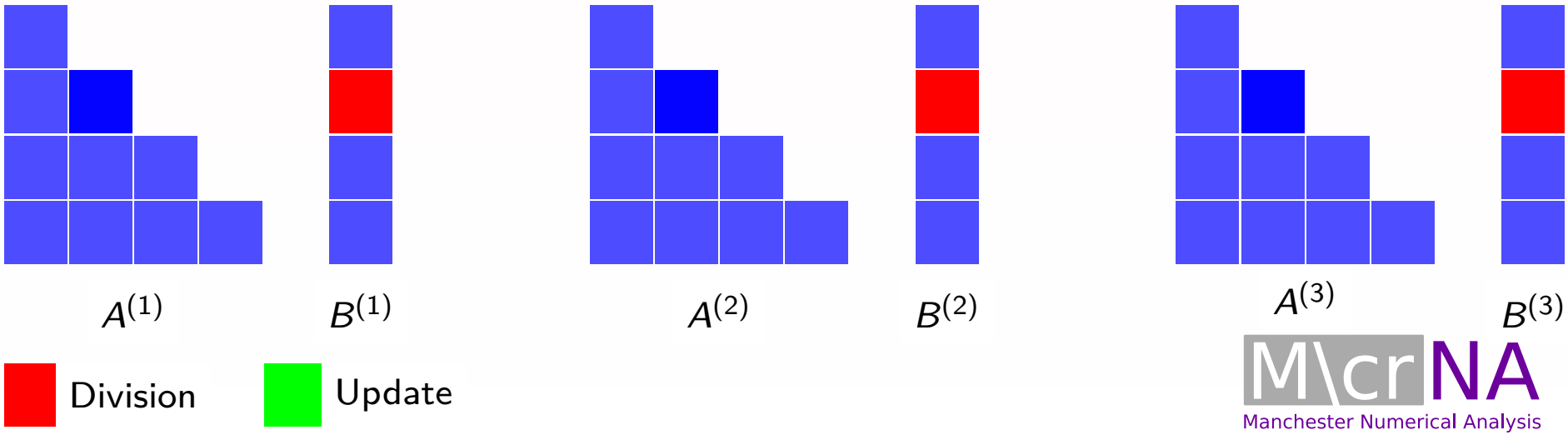
Issues of very small matrices

- Intel AVX-512: 8 double precision
- 2×2 matrix: 4 double precision
- Alternative: cross-matrix vectorization



Issues of sequential algorithms

- Synchronization point
- One division: Only 1 DP flop over 8
- Alternative: cross-matrix vectorization
- Investigate novel memory layouts



M\cr NA
Manchester Numerical Analysis

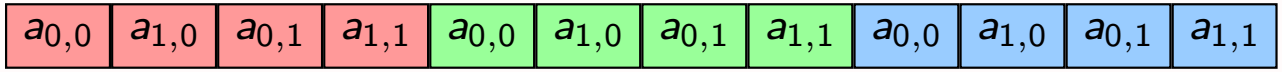
Outline

1. The `OpenMP for loop` approach
2. The interleaved memory layout approach
3. Experimental results
4. Concluding remarks

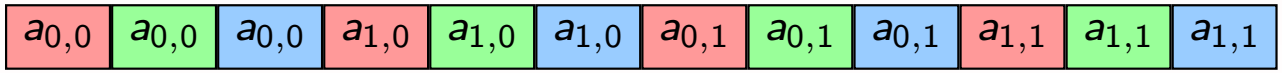
M\crNA
Manchester Numerical Analysis

Interleaved memory layout

Stride layout:



Interleaved memory layout:



Less prone to cache misses

Maximizes use of vector units

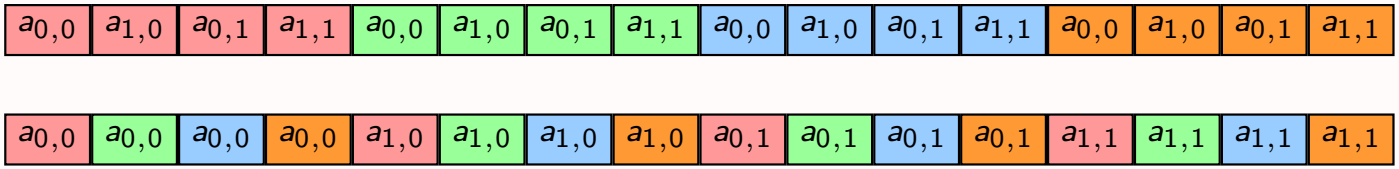
Requires layout conversion

Requires appropriate algorithms

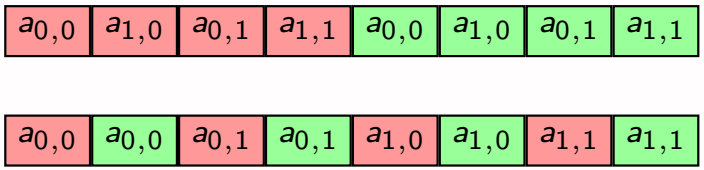
M\crNA
Manchester Numerical Analysis

Algorithm design for interleaved layout

Full interleave of a large batch: high cache misses



Block interleaved



- Divide the batch in blocks
- Interleave each block

- Block big enough to fill vector units
- Block small enough to fit in cache

M\crNA
Manchester Numerical Analysis

Algorithm design for interleaved layout

Basic GEMM algorithm

```
for i ← 1 to m do
|    for j ← 1 to n do
|    |    c_{i,j} ← 0
|    |    for k ← 1 to K do
|    |    |    c_{i,j} ← c_{i,j} + a_{i,k} × b_{k,j}
|    |    end for
|    end for
end for
```

M\crNA
Manchester Numerical Analysis

Algorithm design for interleaved layout

Basic GEMM algorithm

```
for i ← 1 to m do
    for j ← 1 to n do
        c_{i,j} ← 0
        for k ← 1 to K do
            c_{i,j} ← c_{i,j} + a_{i,k} × b_{k,j}
        end for
    end for
end for
```

- Extra inner `for loop`
- Vectorization transformations

Interleaved basic GEMM algorithm

```
for i ← 1 to m do
    for j ← 1 to n do
        for k ← 1 to K do
            #pragma simd
            for idx ← 1 to BlockSize do
                if k == 1 then
                    c_{i,j}^{(idx)} ← 0
                end if
                c_{i,j}^{(idx)} ← c_{i,j}^{(idx)} + a_{i,k}^{(idx)} × b_{k,j}^{(idx)}
            end for
        end for
    end for
end for
```

- Kernel for a single block
- Parallelize blocks over cores

Algorithm design for interleaved layout

- Auto-tuning: set the optimal block size
- Conversion: translate each block to the interleaved layout
- Batch computation: call block interleaved kernels
- Conversion: translate results back to the user layout

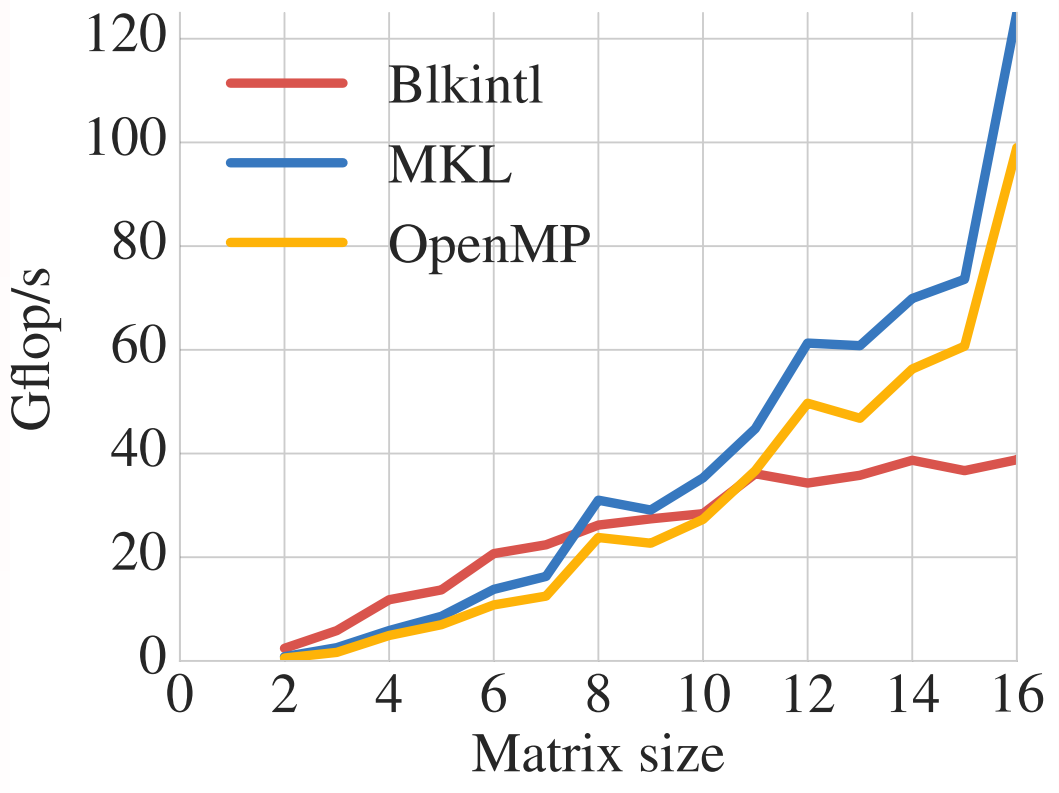
M\crNA
Manchester Numerical Analysis

Outline

1. The `OpenMP for loop` approach
2. The interleaved memory layout approach
3. Experimental results
4. Concluding remarks

M\crNA
Manchester Numerical Analysis

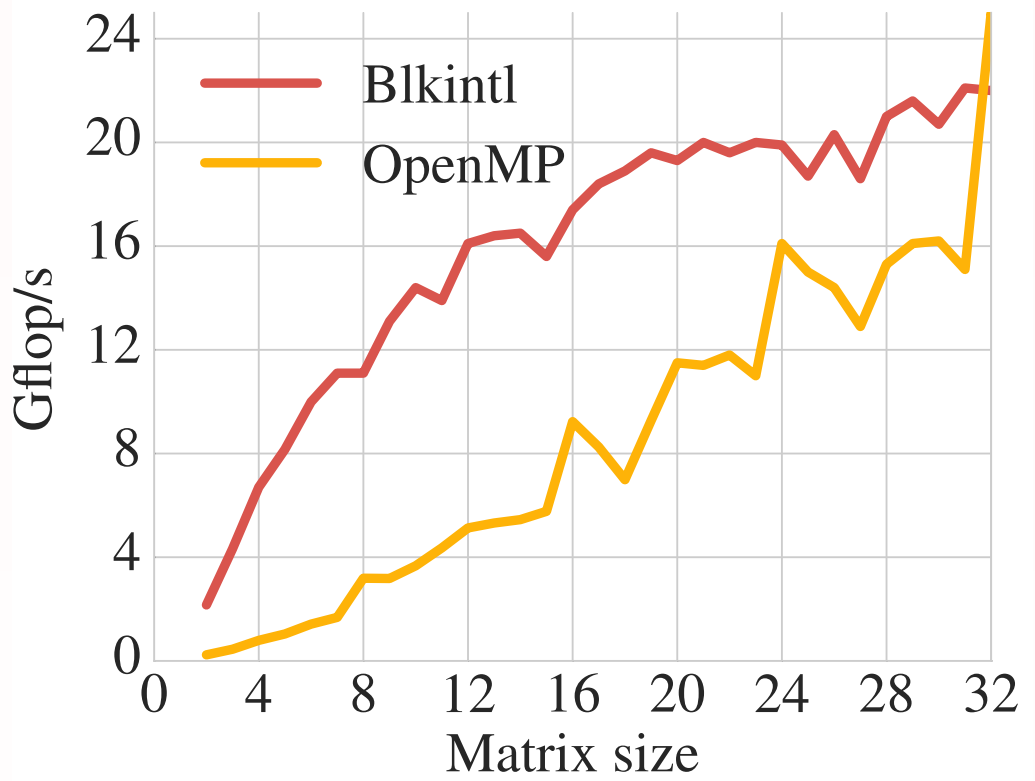
Interleaved memory layout - GEMM (Intel KNL)



- Batch of 10,000 DGEMM
- 68 threads on 68-core Intel KNL
- Flush the cache
- Include conversion time

Manchester Numerical Analysis

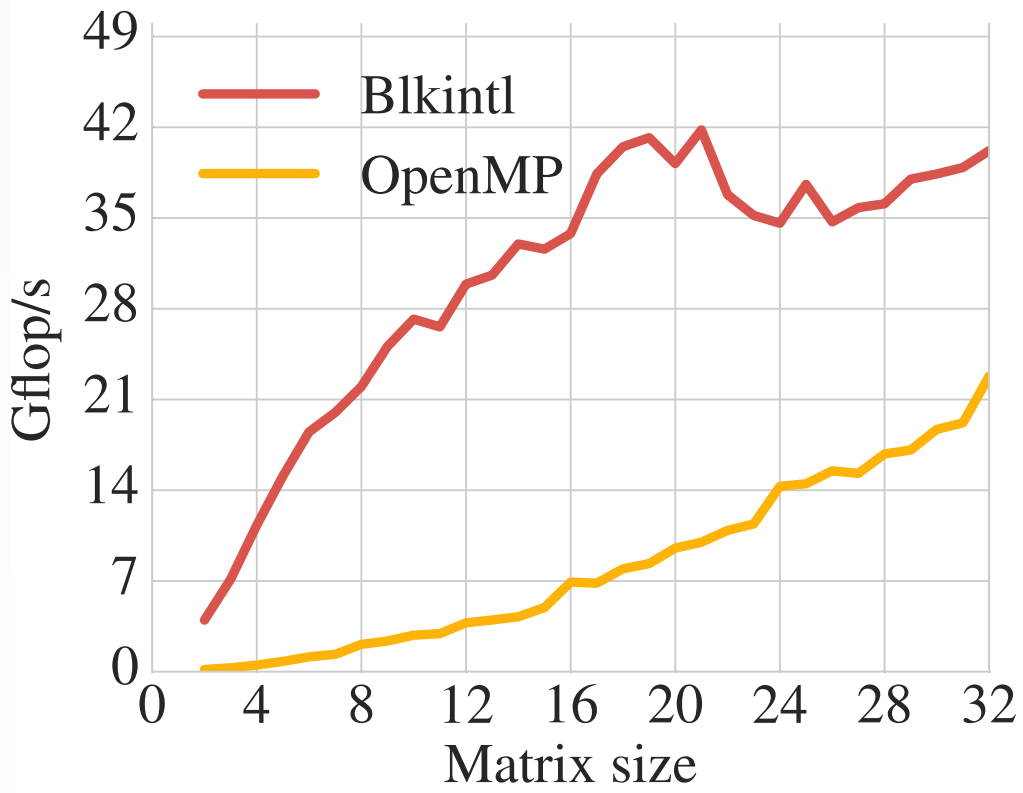
Interleaved memory layout - TRSM (Intel KNL)



- Batch of 10,000 DTRSM (4rhs)
- 68 threads on 68-core Intel KNL
- Flush the cache
- Include conversion time

Manchester Numerical Analysis

Interleaved memory layout - DPOSV (Intel KNL)



- Batch of 10,000 POSV (4rhs)
- 68 threads on 68-core Intel KNL
- Flush the cache
- Include conversion time

M\crNA
Manchester Numerical Analysis

Outline

1. The `OpenMP for loop` approach
2. The interleaved memory layout approach
3. Experimental results
4. Concluding remarks

Manchester Numerical Analysis

Concluding remarks

Remarks

- Keep `OpenMP for loop` for medium size matrices
- `Block interleaved` for very small matrices

Perspective

- Extend interleave to other kernels
- Interleaved LU factorization

Manchester Numerical Analysis

A Batch Of Thanks