Transform Array Operations to Optimize Memory Access for GPGPU

Mads Ynddal
SJT402

Overview

1. Motivation
2. Access Patterns
3. Reduction
4. Benchmarks
5. Future Work
6. Conclusion

1. Motivation

107

Bh

Bohrium

[270]

Motivation

- Bohrium
  - Novice
  - Experts
- Performance Gap
  - Reductions
  - Memory Throughput

107

Bh

Bohrium

[270]

Motivation

- Bohrium
  - Novice
  - Experts
- Performance Gap
  - Reductions
  - Memory Throughput

NumPy Stencil

```
center = grid[1:-1, 1:-1]
north = grid[0:-2, 1:-1]
east = grid[1:-1, 2:]
west = grid[1:-1, 0:-2]
south = grid[2:, 1:-1]
work = 0.2 * (center + north +
    east + west + south)
delta = np.sum(np.abs(work-center))
center[:] = work
```

Motivation

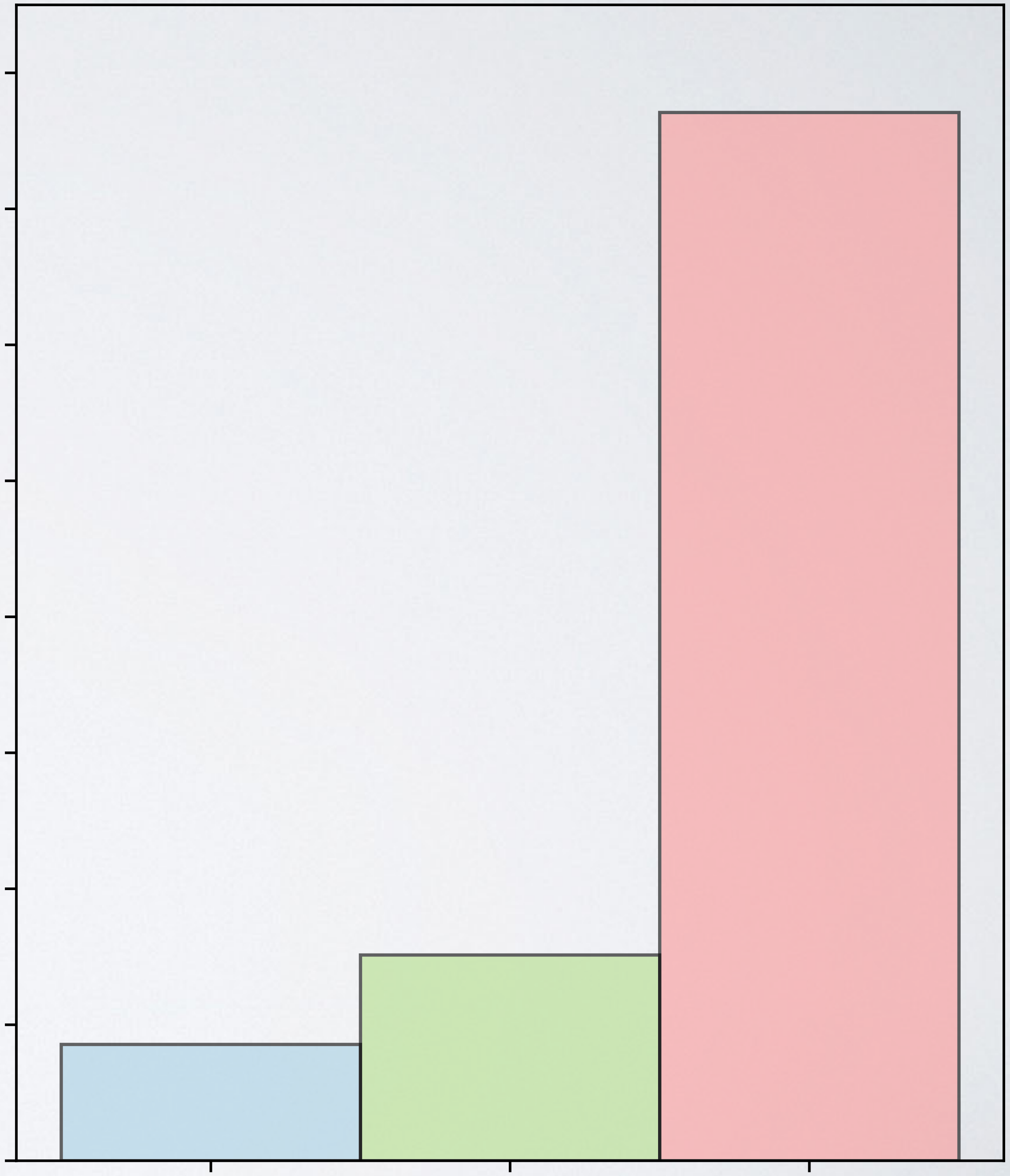
- Bohrium
  - Novice
  - Experts
- Performance Gap
  - Reductions
  - Memory Throughput

OpenCL Kernel Stencil

```
uint gid = get_global_id(0);
if (gid >= width)
   return;
uint w = width + 2;
uint h = height + 2;
DTYPE d = 0.0;
for (uint i = 0; i < height; ++i)
{
    uint offset = i*w;
    DTYPE up     = in[gid+1+offset];
    DTYPE left   = in[gid+w+offset];
    DTYPE right  = in[gid+w+2+offset];
    DTYPE down   = in[gid+1+w*2+offset];
    DTYPE center = in[gid+w+1+offset];
    DTYPE out_center =
(center + up + left + right + down) * 0.2;
    out[gid+w+1+offset] = out_center;
    d += fabs(out_center - center);
}
delta[gid] = d;
```

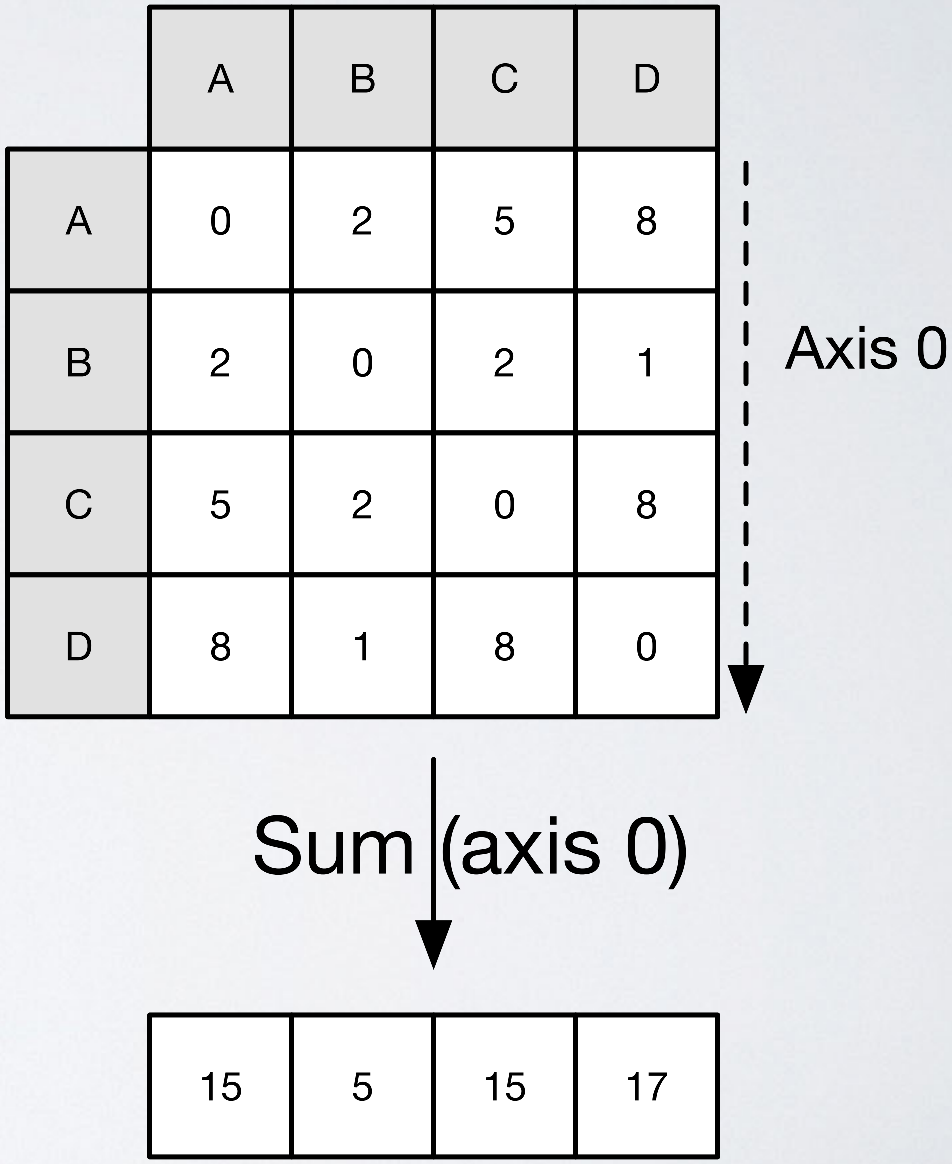
Motivation

- Bohrium
  - Novice
  - Experts
- Performance Gap
  - Reductions
  - Memory Throughput



Motivation

- Bohrium
  - Novice
  - Experts
- Performance Gap
  - Reductions
  - Memory Throughput



	A	B	C	D
A	0	2	5	8
B	2	0	2	1
C	5	2	0	8
D	8	1	8	0

Axis 0

Sum (axis 0)

15	5	15	17

Motivation

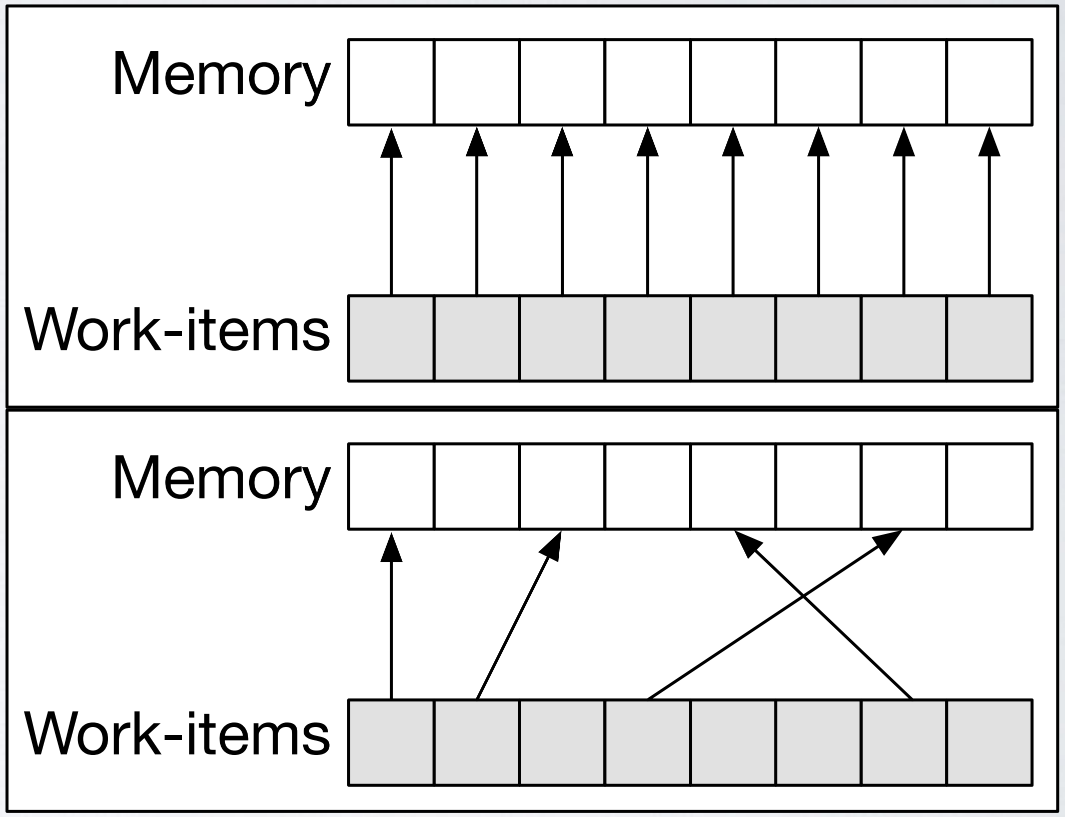
- Bohrium
  - Novice
  - Experts
- Performance Gap
  - Reductions
  - Memory Throughput

Nvidia Guidelines

> “Bandwidth is one of the most important gating factors for performance. Almost all changes to code should be made in the context of how they affect bandwidth”

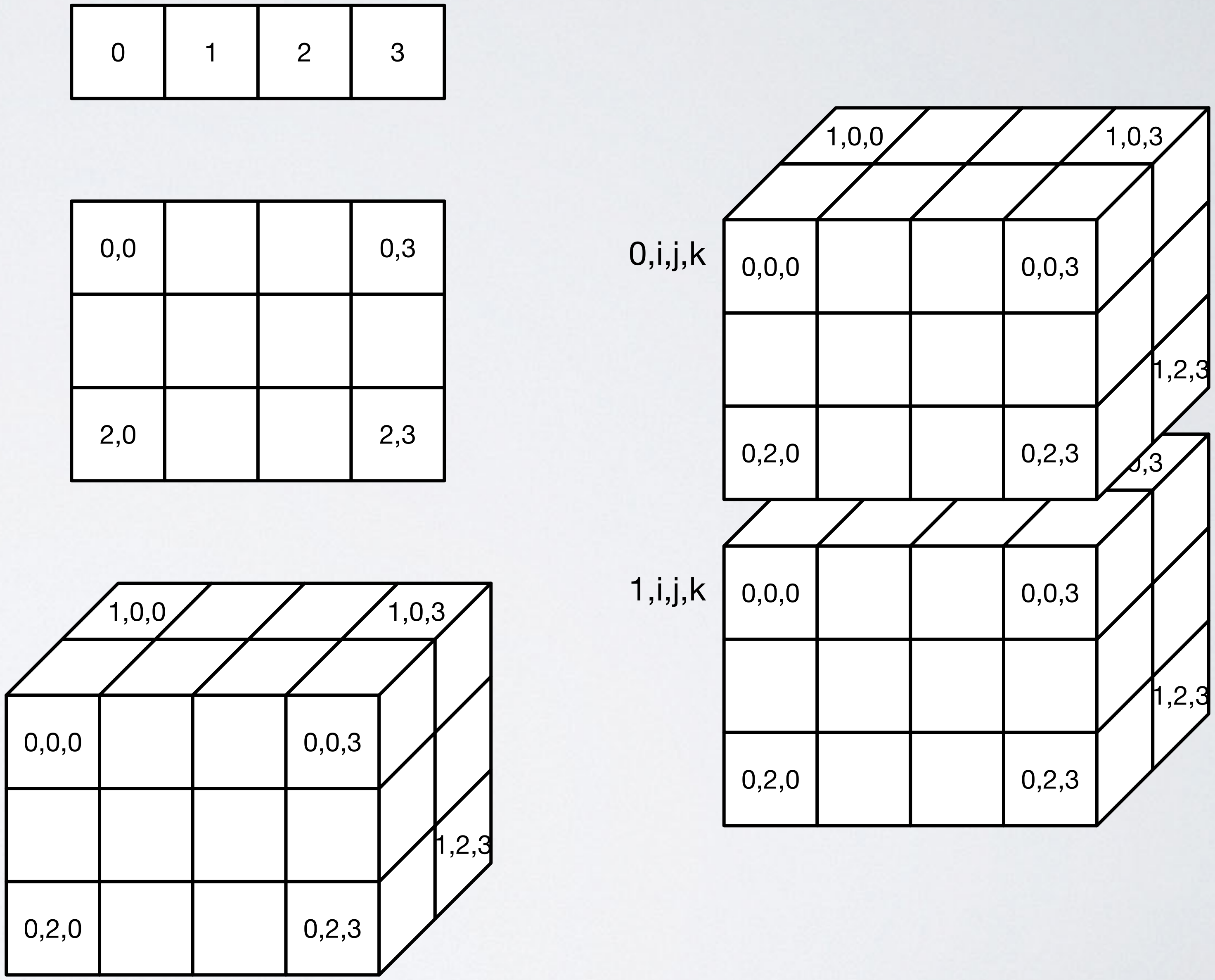
2. Access Patterns

- Tensors
- Inspection
- Changing Access Pattern



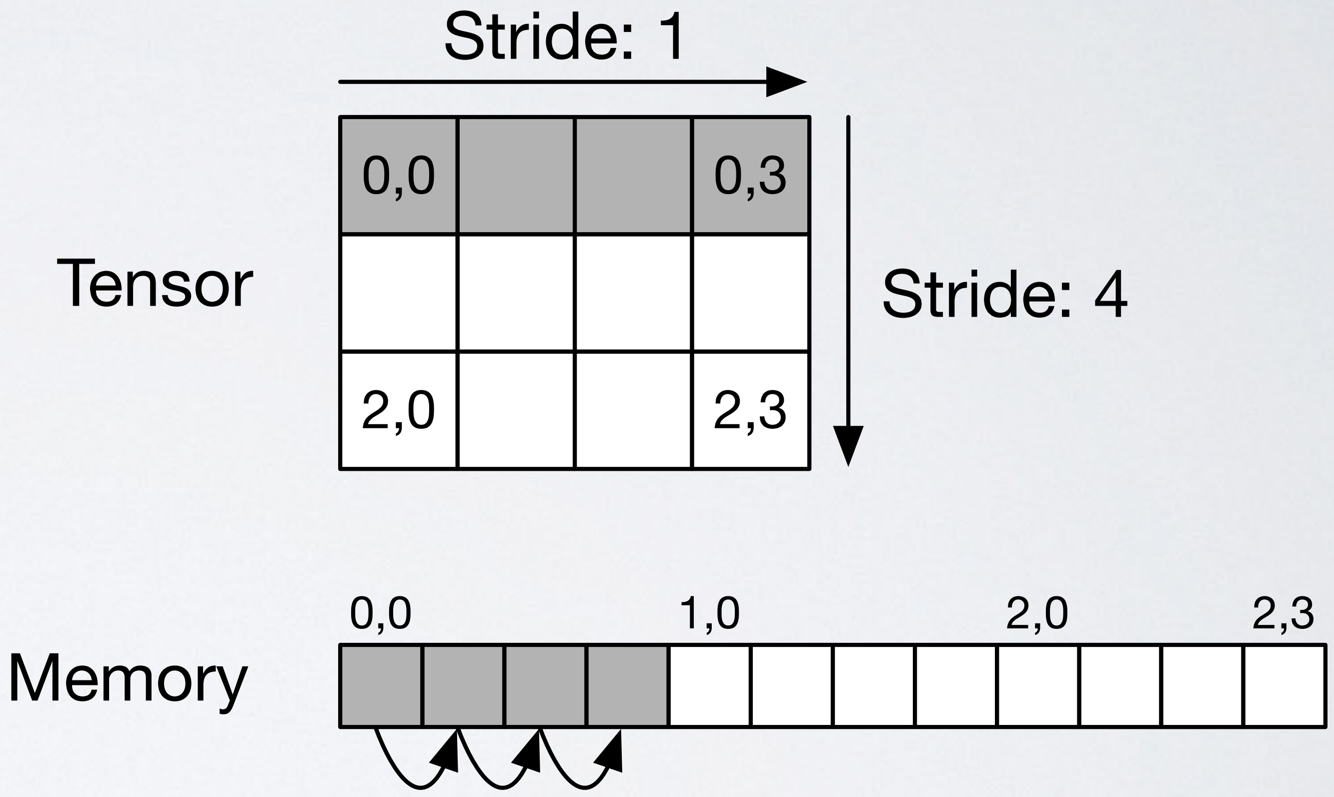
Tensors

- Rank
- Strides
- Operations



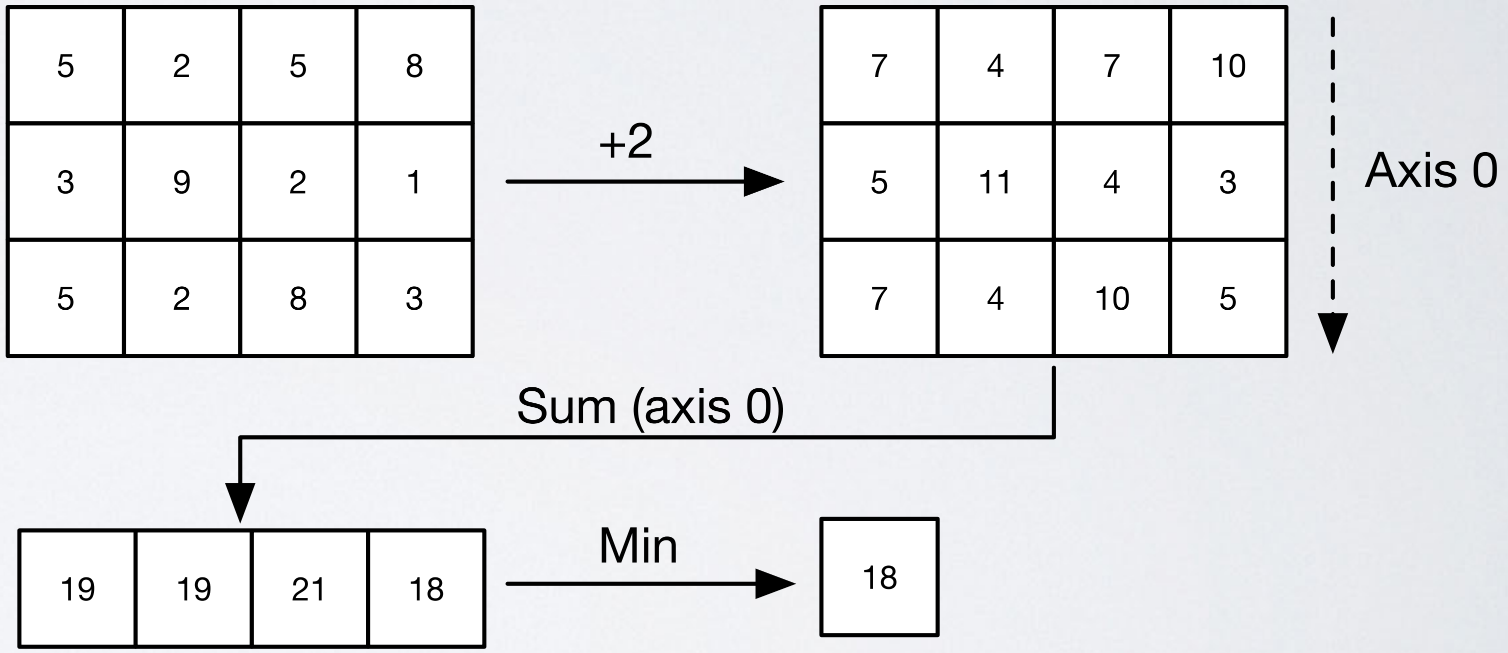
Tensors

- Rank
- Strides
- Operations



Tensors

- Rank
- Strides
- Operations



5	2	5	8
3	9	2	1
5	2	8	3

7	4	7	10
5	11	4	3
7	4	10	5

19	19	21	18

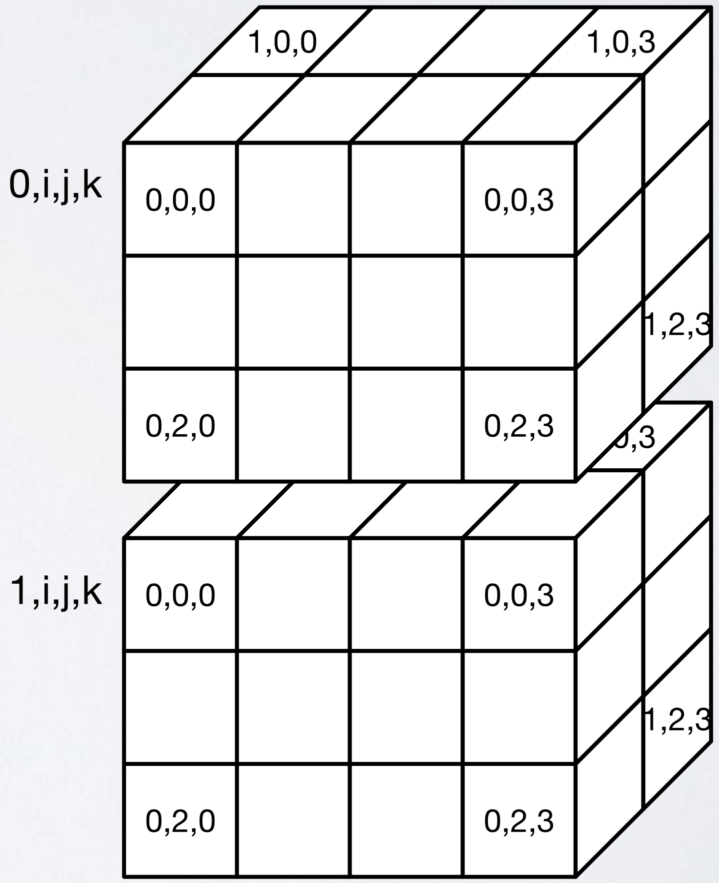
18

Inspection

- Tensor Example
- Generated Code
- Current Pattern
- Coalesced Pattern

Rank-4 tensor with dimensions:

$$50 \times 50 \times 50 \times 50$$

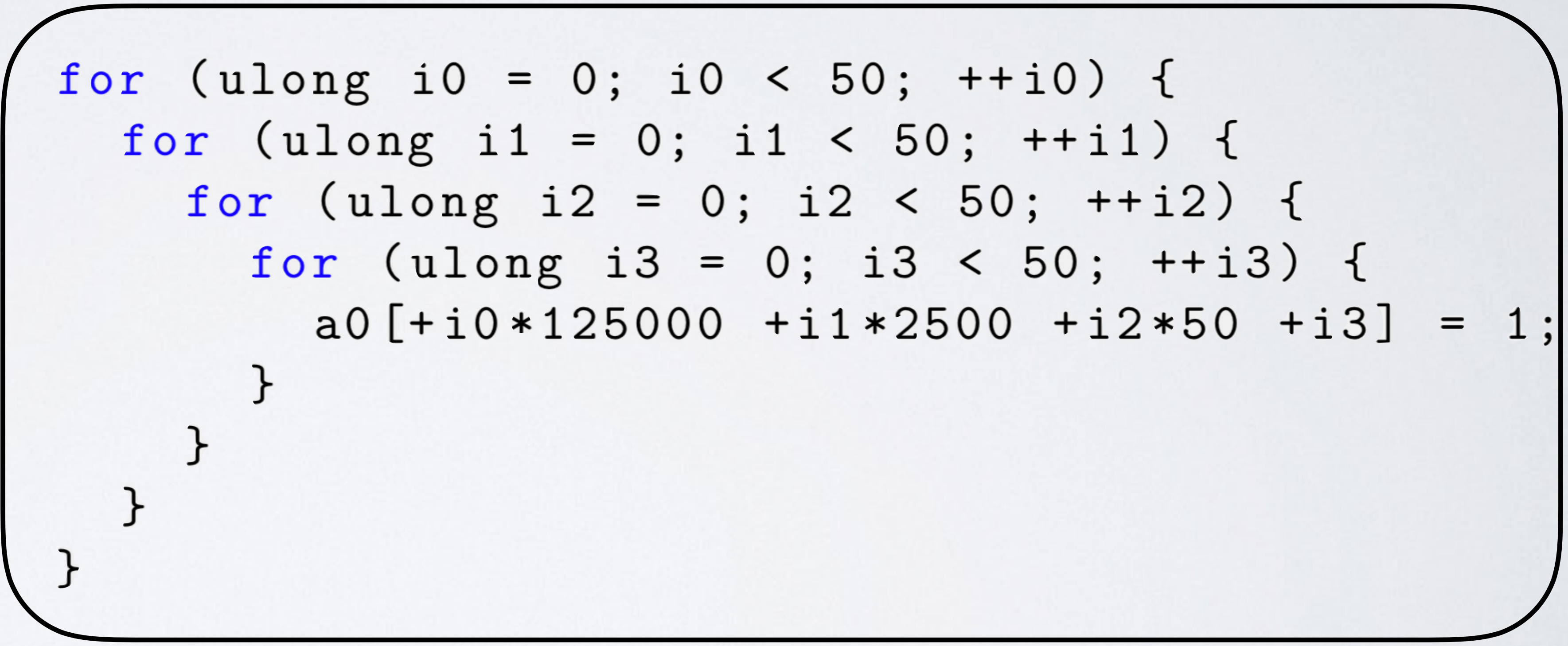


Inspection

Rank-4 tensor with dimensions:

$$50 \times 50 \times 50 \times 50$$

- Tensor Example
- Generated Code
- Current Pattern
- Coalesced Pattern



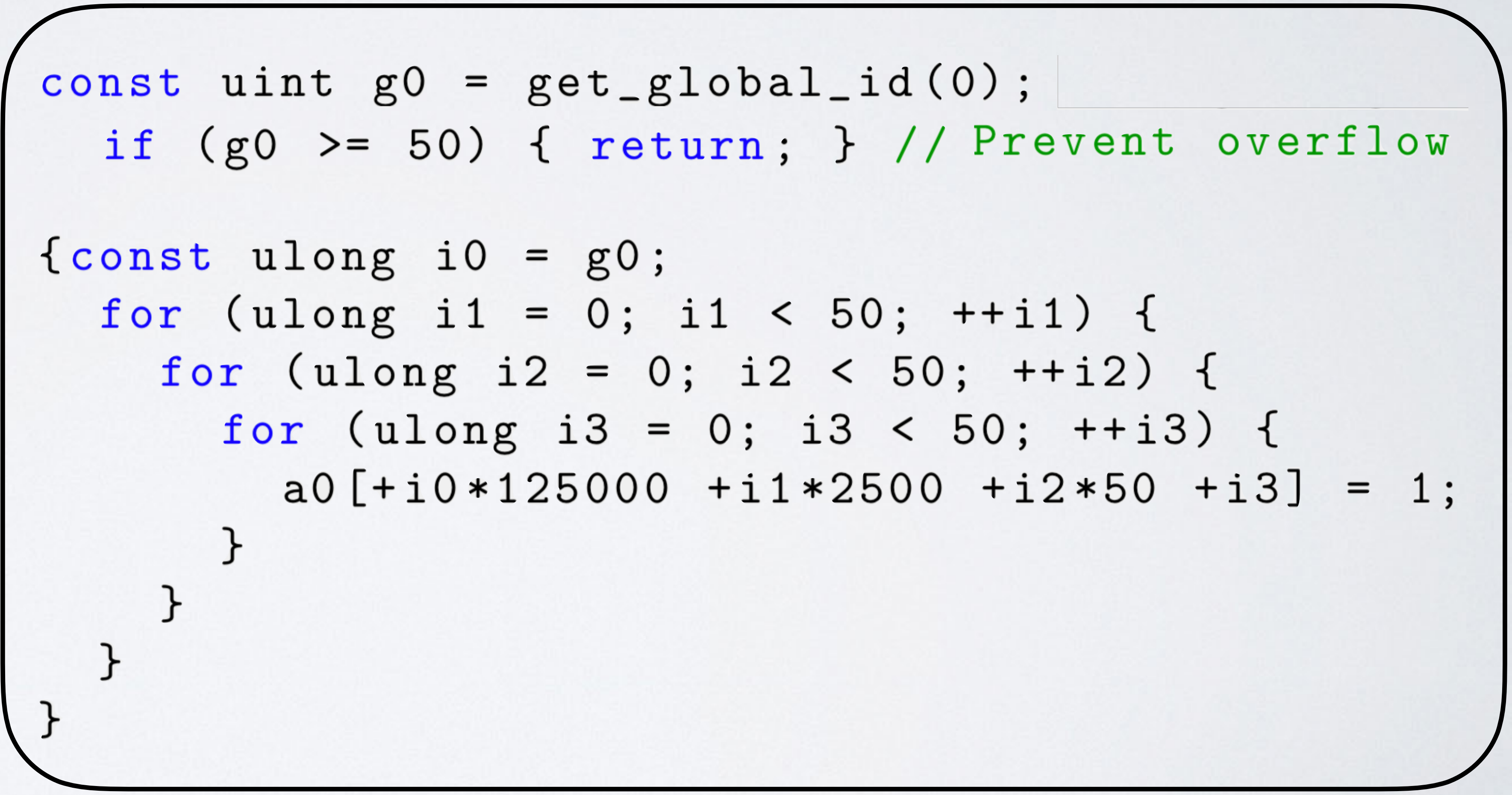
```
for (ulong i0 = 0; i0 < 50; ++i0) {
  for (ulong i1 = 0; i1 < 50; ++i1) {
    for (ulong i2 = 0; i2 < 50; ++i2) {
      for (ulong i3 = 0; i3 < 50; ++i3) {
        a0[+i0*125000 +i1*2500 +i2*50 +i3] = 1;
      }
    }
  }
}
```

Inspection

Rank-4 tensor with dimensions:

$$50 \times 50 \times 50 \times 50$$

- Tensor Example
- Generated Code
- Current Pattern
- Coalesced Pattern



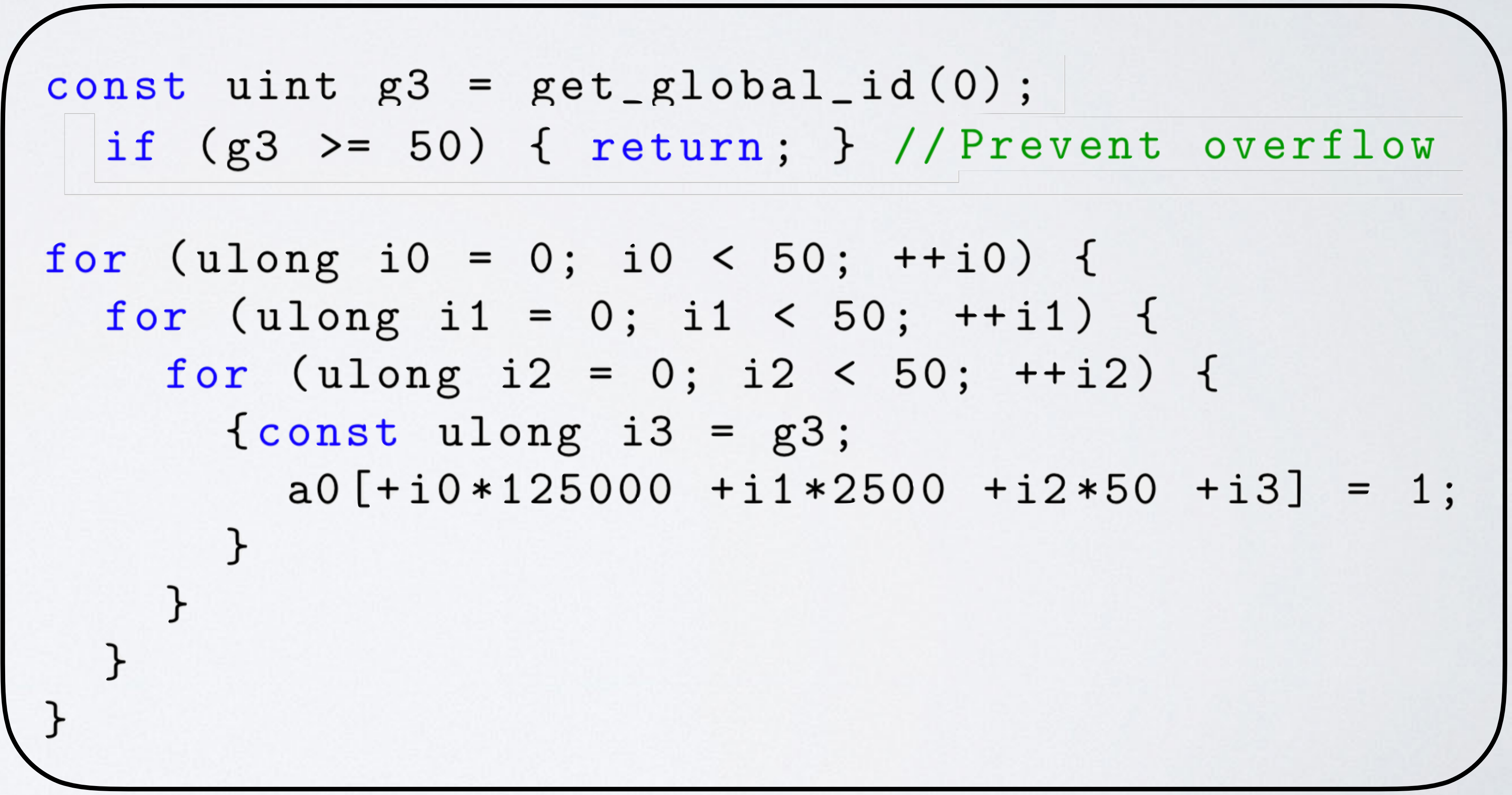
```
const uint g0 = get_global_id(0);
  if (g0 >= 50) { return; } // Prevent overflow

{const ulong i0 = g0;
  for (ulong i1 = 0; i1 < 50; ++i1) {
    for (ulong i2 = 0; i2 < 50; ++i2) {
      for (ulong i3 = 0; i3 < 50; ++i3) {
        a0[+i0*125000 +i1*2500 +i2*50 +i3] = 1;
      }
    }
  }
}
```

Inspection

Rank-4 tensor with dimensions:

$$50 \times 50 \times 50 \times 50$$

- Tensor Example
- Generated Code
- Current Pattern
- **Coalesced Pattern**



```
const uint g3 = get_global_id(0);
  if (g3 >= 50) { return; } //Prevent overflow

for (ulong i0 = 0; i0 < 50; ++i0) {
  for (ulong i1 = 0; i1 < 50; ++i1) {
    for (ulong i2 = 0; i2 < 50; ++i2) {
      {const ulong i3 = g3;
        a0[+i0*125000 +i1*2500 +i2*50 +i3] = 1;
      }
    }
  }
}
```

Changing Access Pattern

- Decide on Parallel Axis
- Implementation
- Results

Truncated Block Structure of Kernel

```
rank: 0, size: 75, sweeps: {}
  rank: 1, size: 75, sweeps: {}
    rank: 2, size: 75, sweeps: {
       BH_ADD_REDUCE a0 a1 3}
      rank: 3, size: 75, sweeps: {
         BH_ADD_REDUCE a2 a3 4}
```

Changing Access Pattern

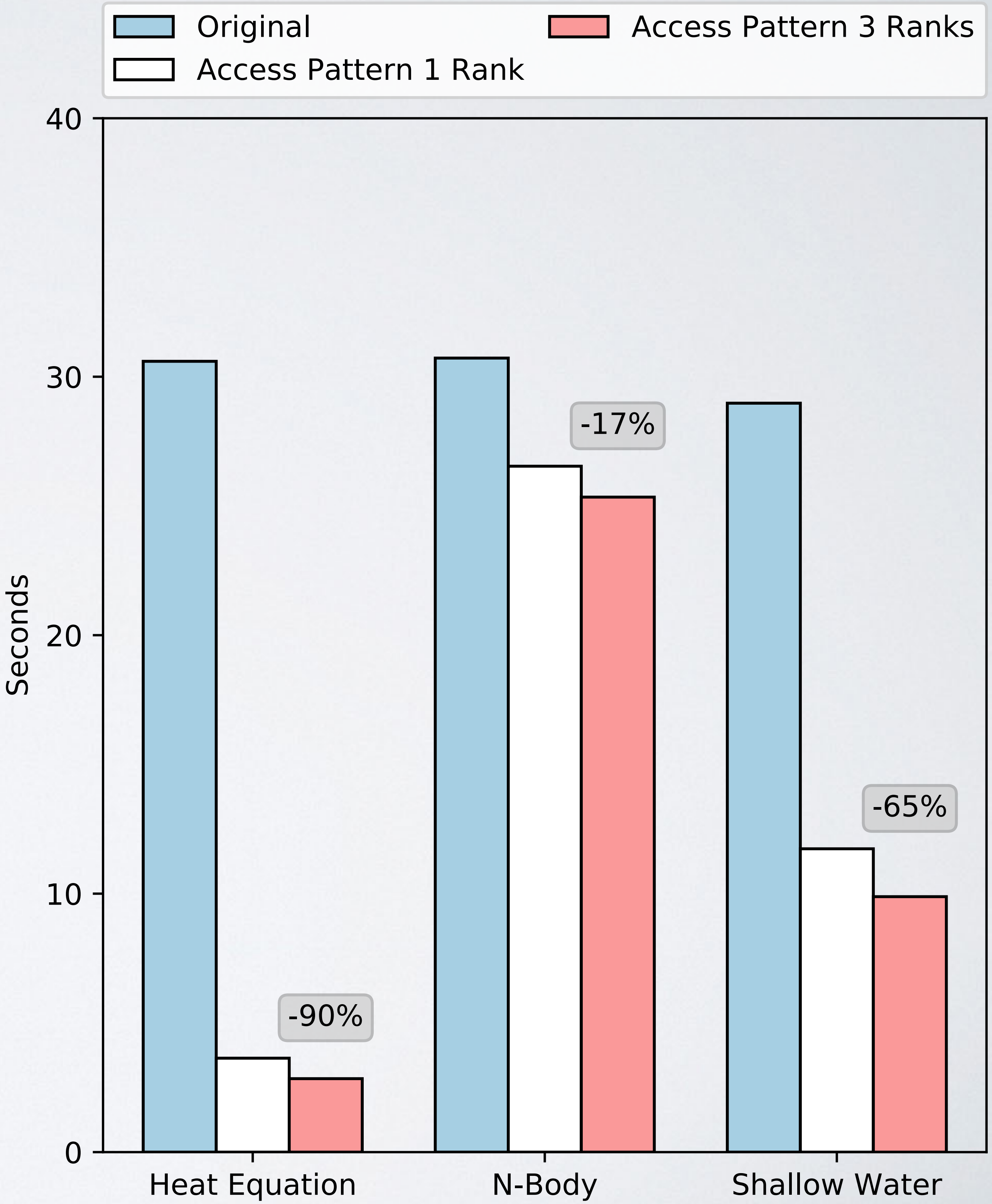
- Decide on Parallel Axis
- Implementation
- Results

Remapped Iterators

```
uint g2 = get_global_id(0);
uint g1 = get_global_id(1);
uint g0 = get_global_id(2);
// ...
i0 = g0
i1 = g1
i2 = g2
```

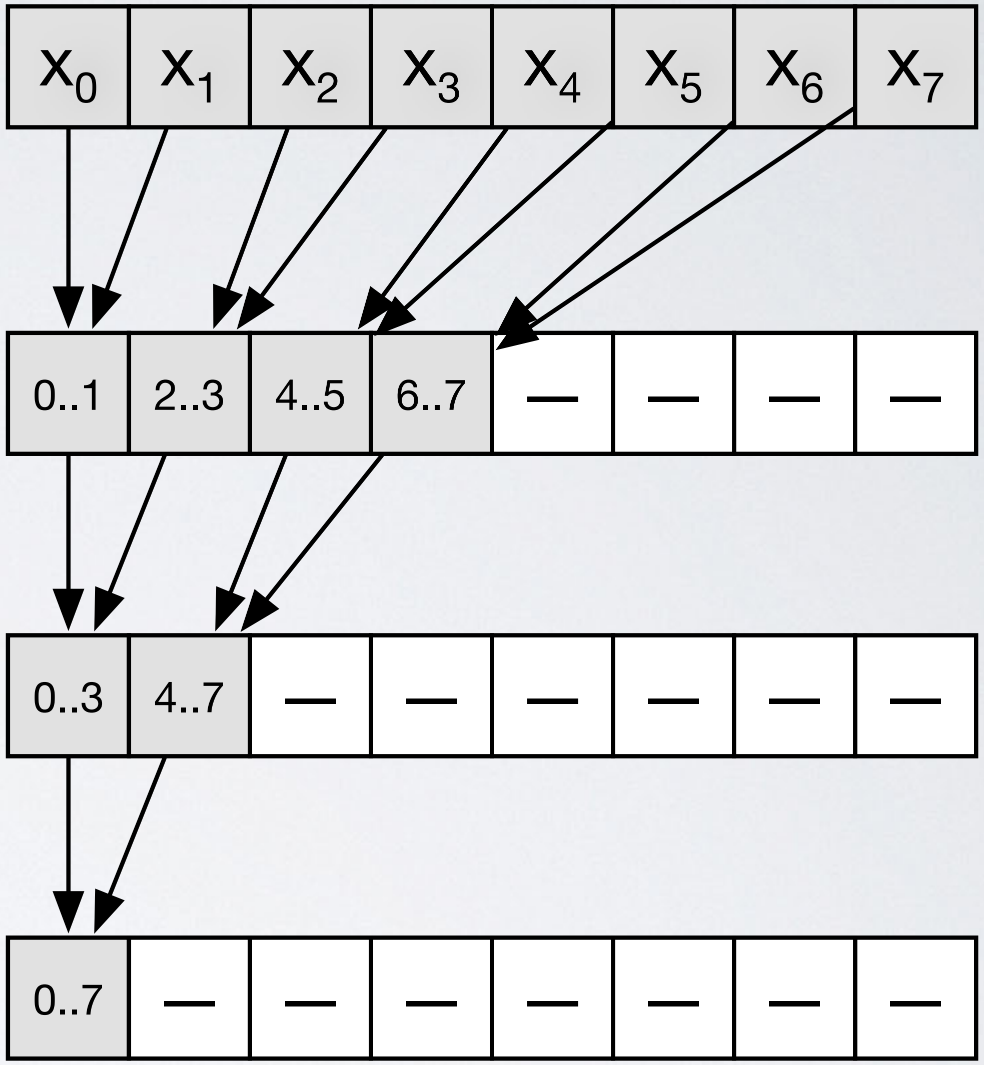
Changing Access Pattern

- Decide on Parallel Axis
- Implementation
- Results



3. Reduction

- Hardware Constraints
- Access Pattern in Reductions



x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7

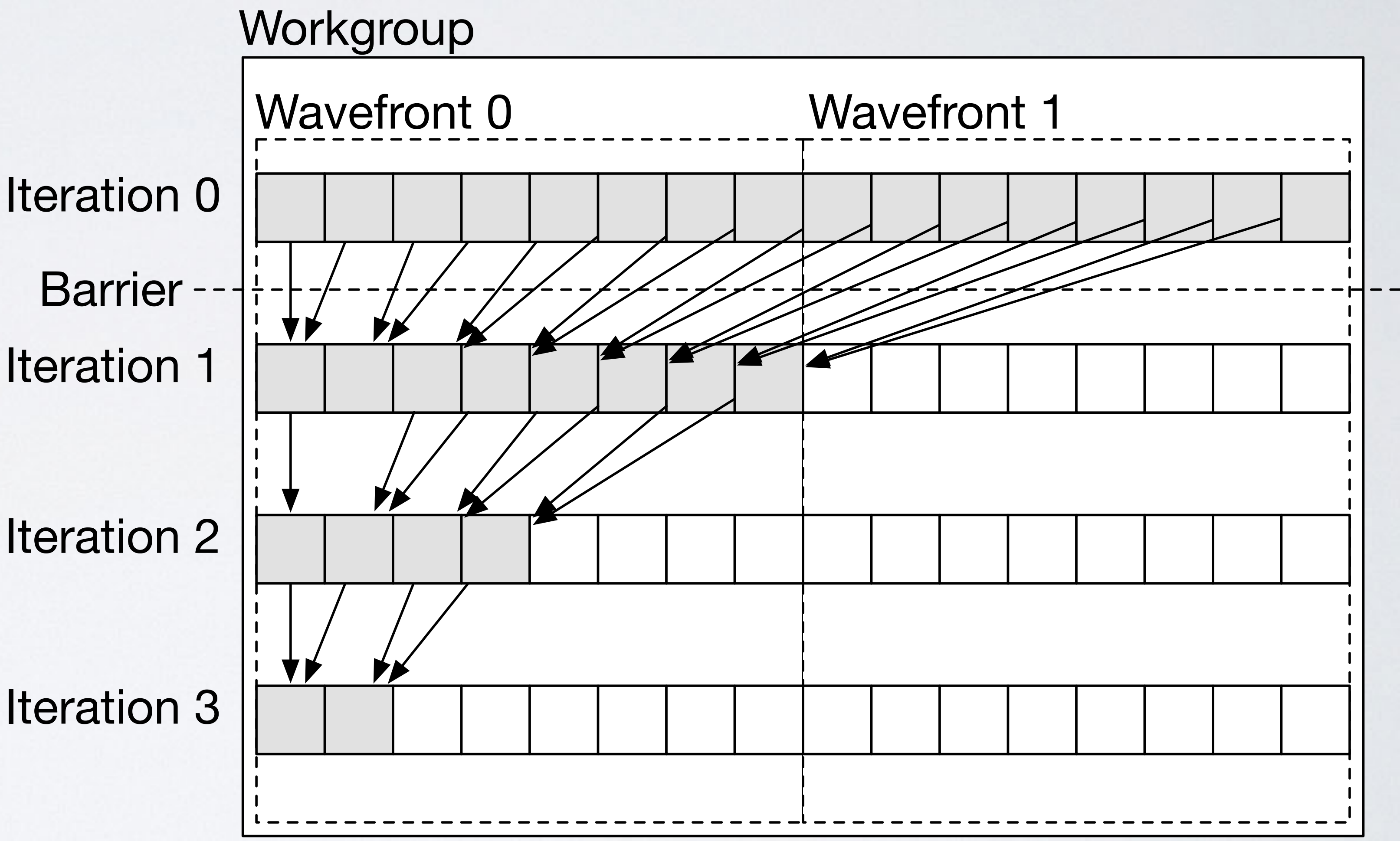
0..1 2..3 4..5 6..7 — — — —

0..3 4..7 — — — — — —

0..7 — — — — — — —

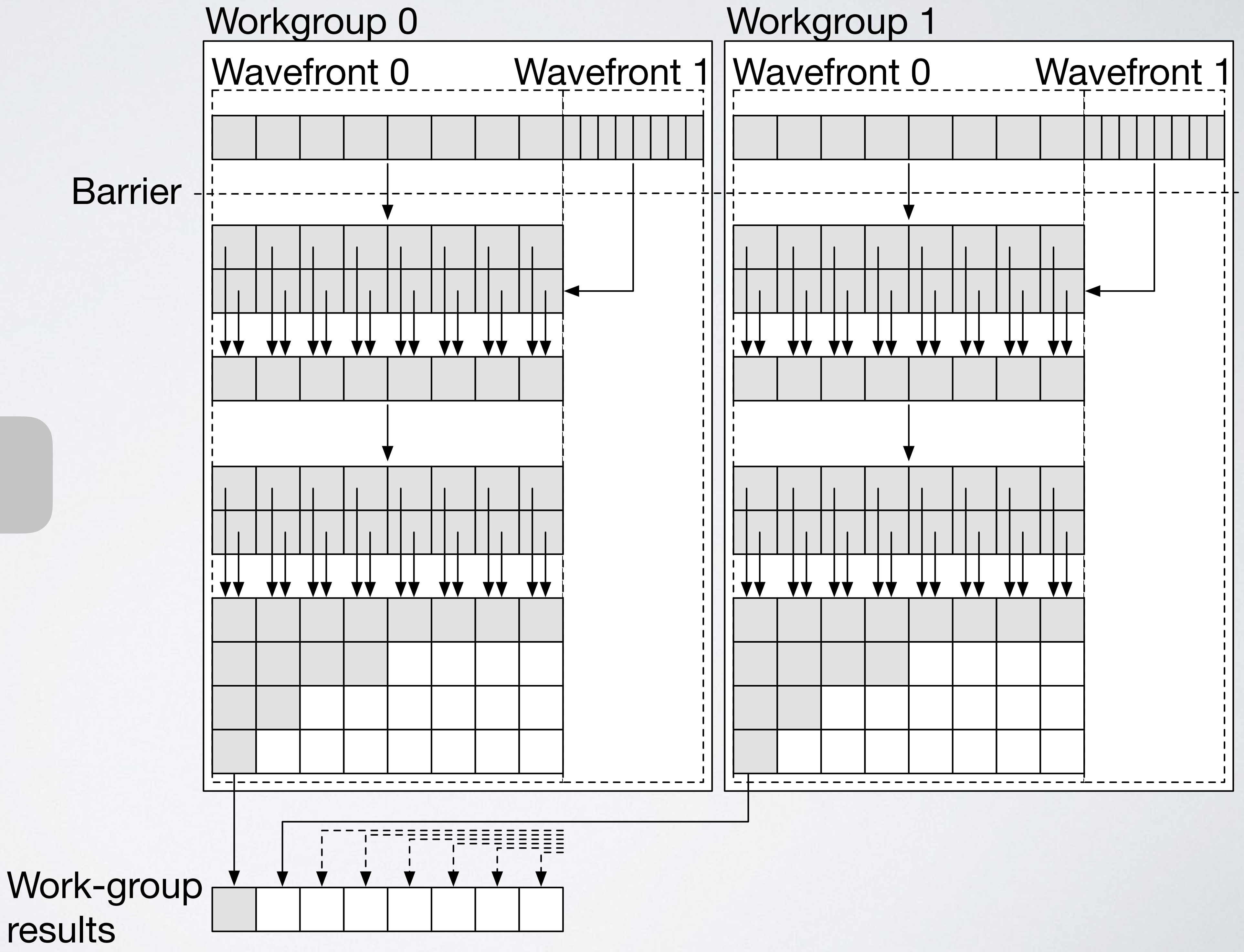
Hardware Constraints

- Wavefronts
- Work-groups
- Bohrium



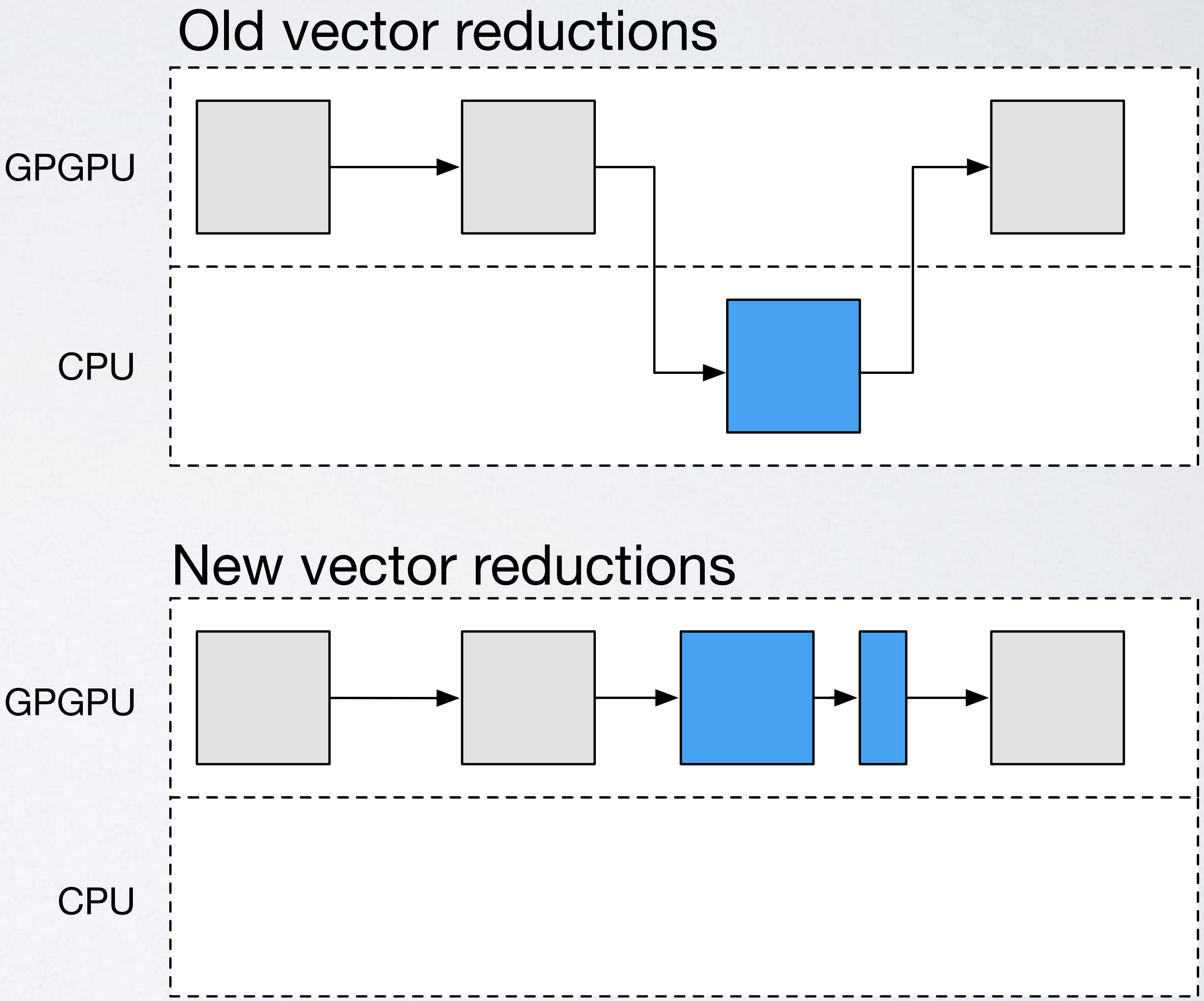
Hardware Constraints

- Wavefronts
- Work-groups
- Bohrium



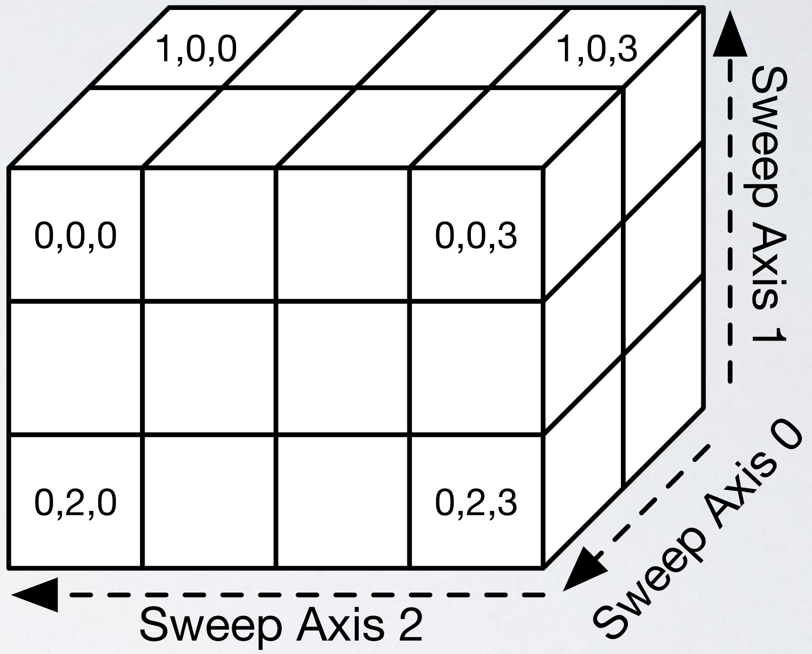
Hardware Constraints

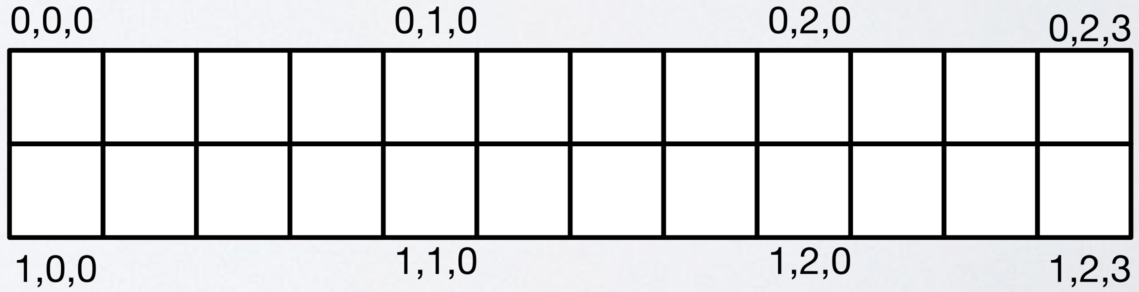
- Wavefronts
- Work-groups
- Bohrium



Access Pattern in Reductions

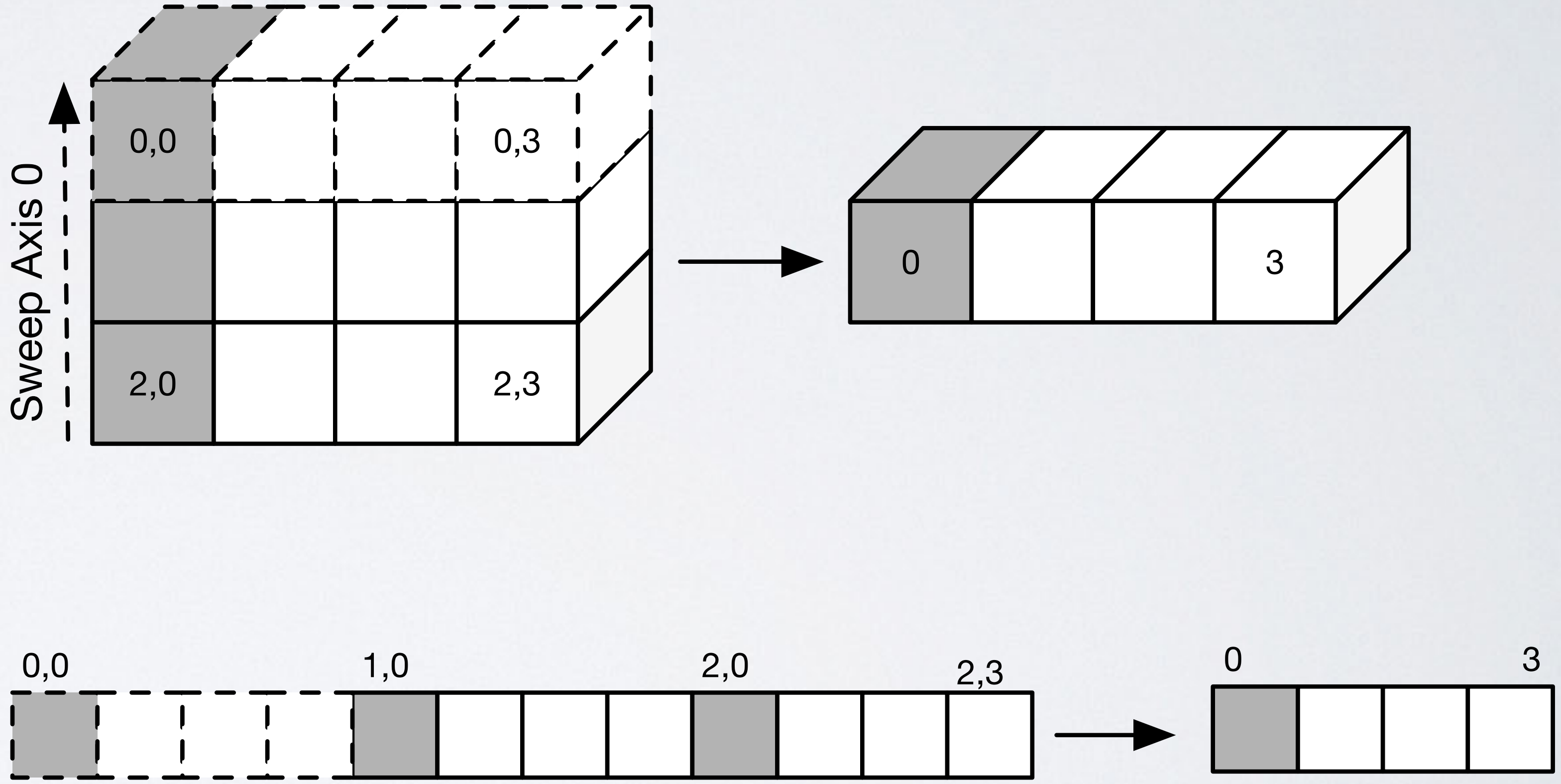
- Axis
- Column-wise
- Row-wise
- Transposing





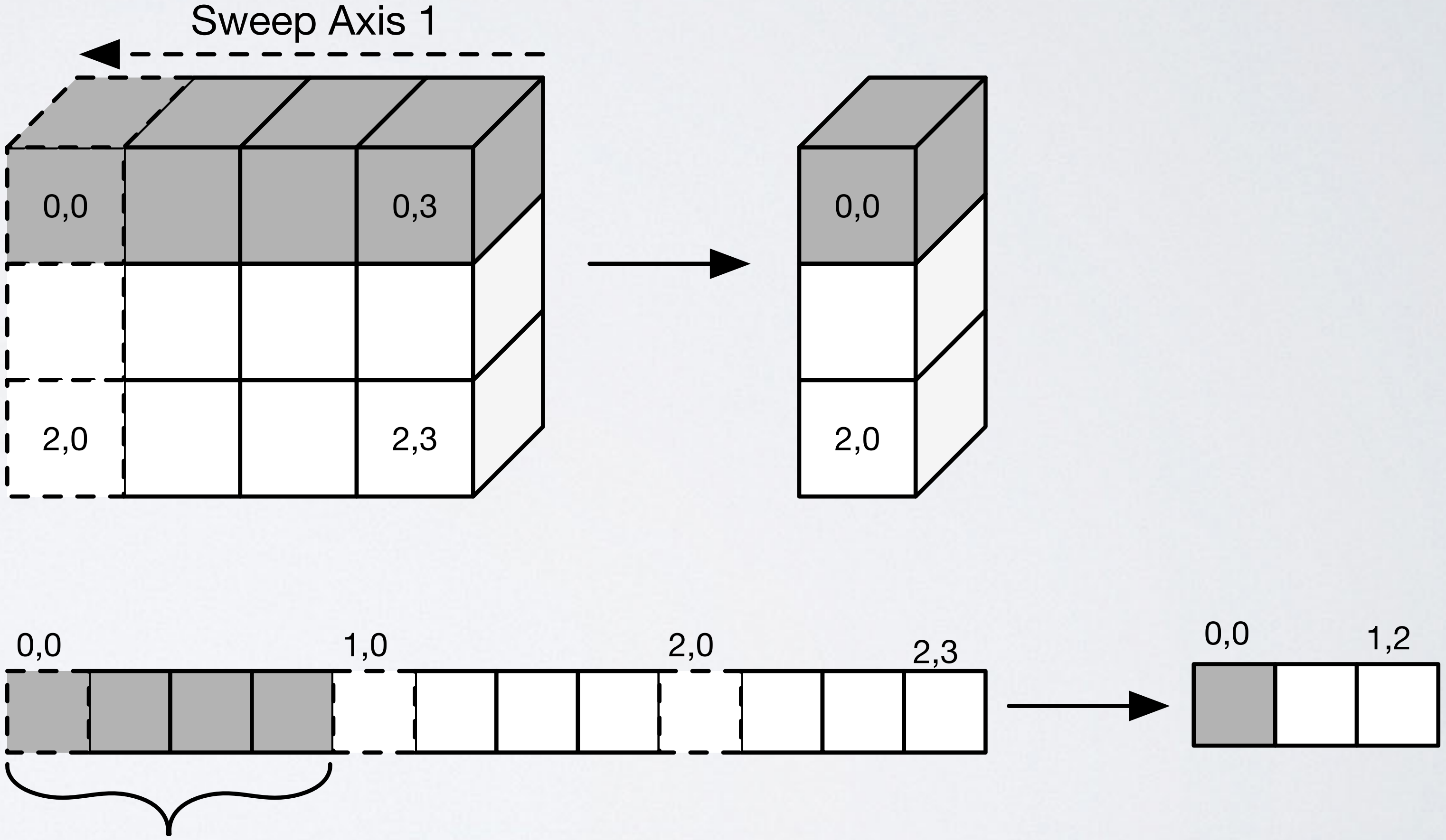
Access Pattern in Reductions

- Axis
- Column-wise
- Row-wise
- Transposing



Access Pattern in Reductions

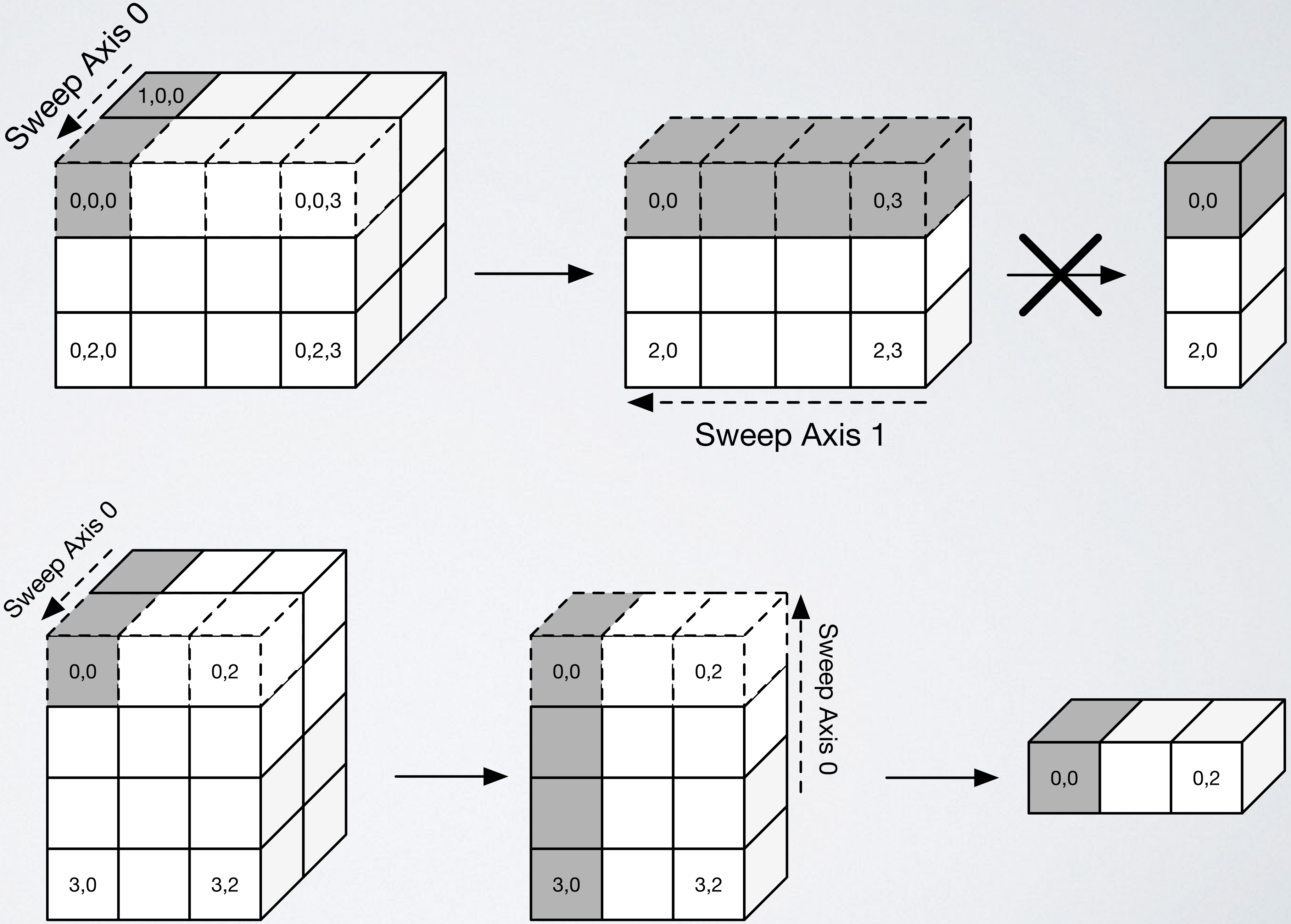
- Axis
- Column-wise
- Row-wise
- Transposing



Segmented Reduction Preferred

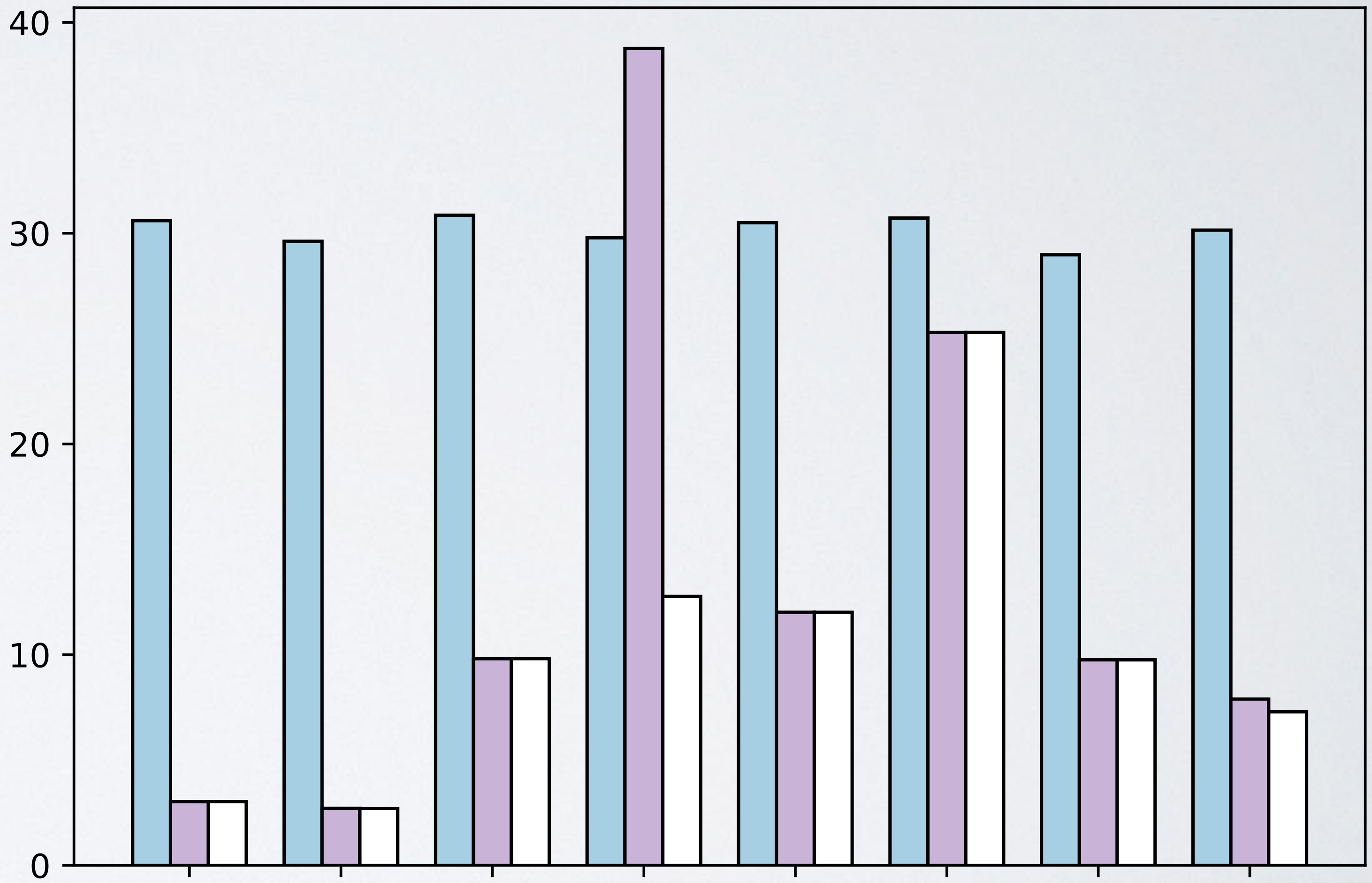
Access Pattern in Reductions

- Axis
- Column-wise
- Row-wise
- Transposing



4. Benchmarks

- Benchmark Suite
- Handwritten vs. Bohrium

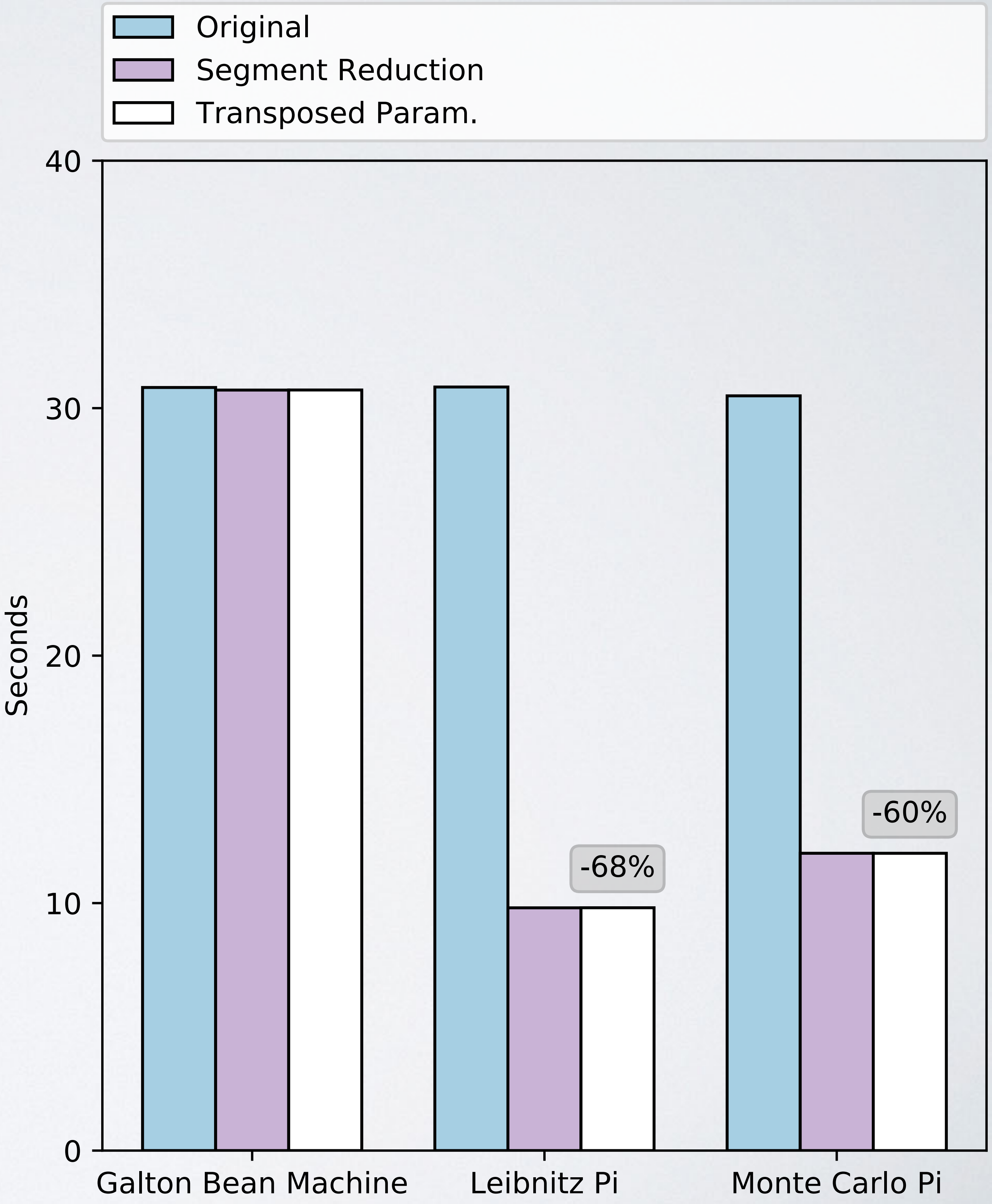


Benchmark Suite

- Embarrassingly Parallel
- Finite Difference
- Streaming/Reduction

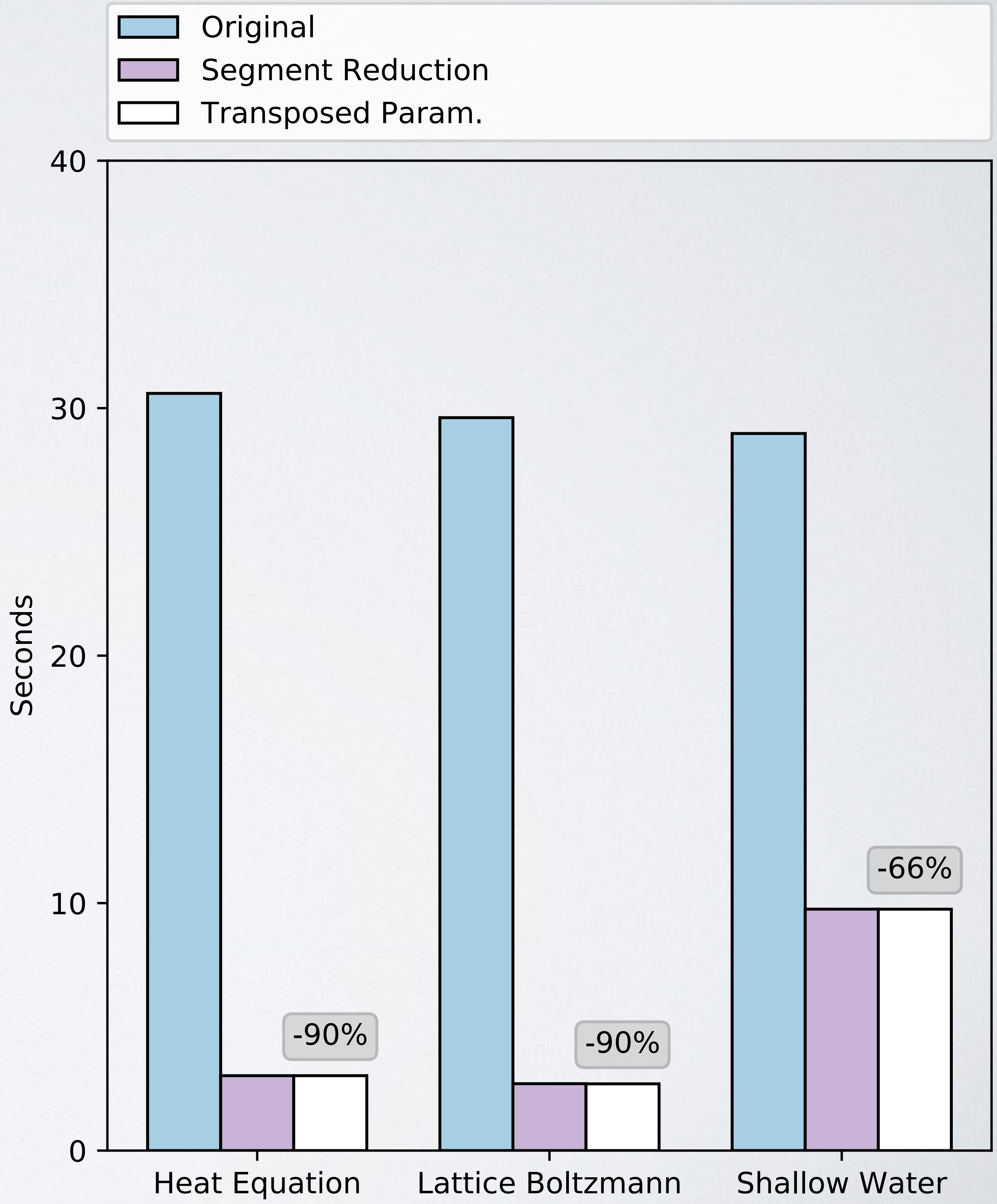
Benchmark Suite

- Embarrassingly Parallel
  I. Galton Bean Machine
  II. Leibnitz Pi
  III. Monte Carlo Pi
- Finite Difference
- Streaming/Reduction



Benchmark Suite

- Embarrassingly Parallel
- Finite Difference
    I. Heat Equation
    II. Lattice Boltzmann
    III. Shallow Water
- Streaming/Reduction



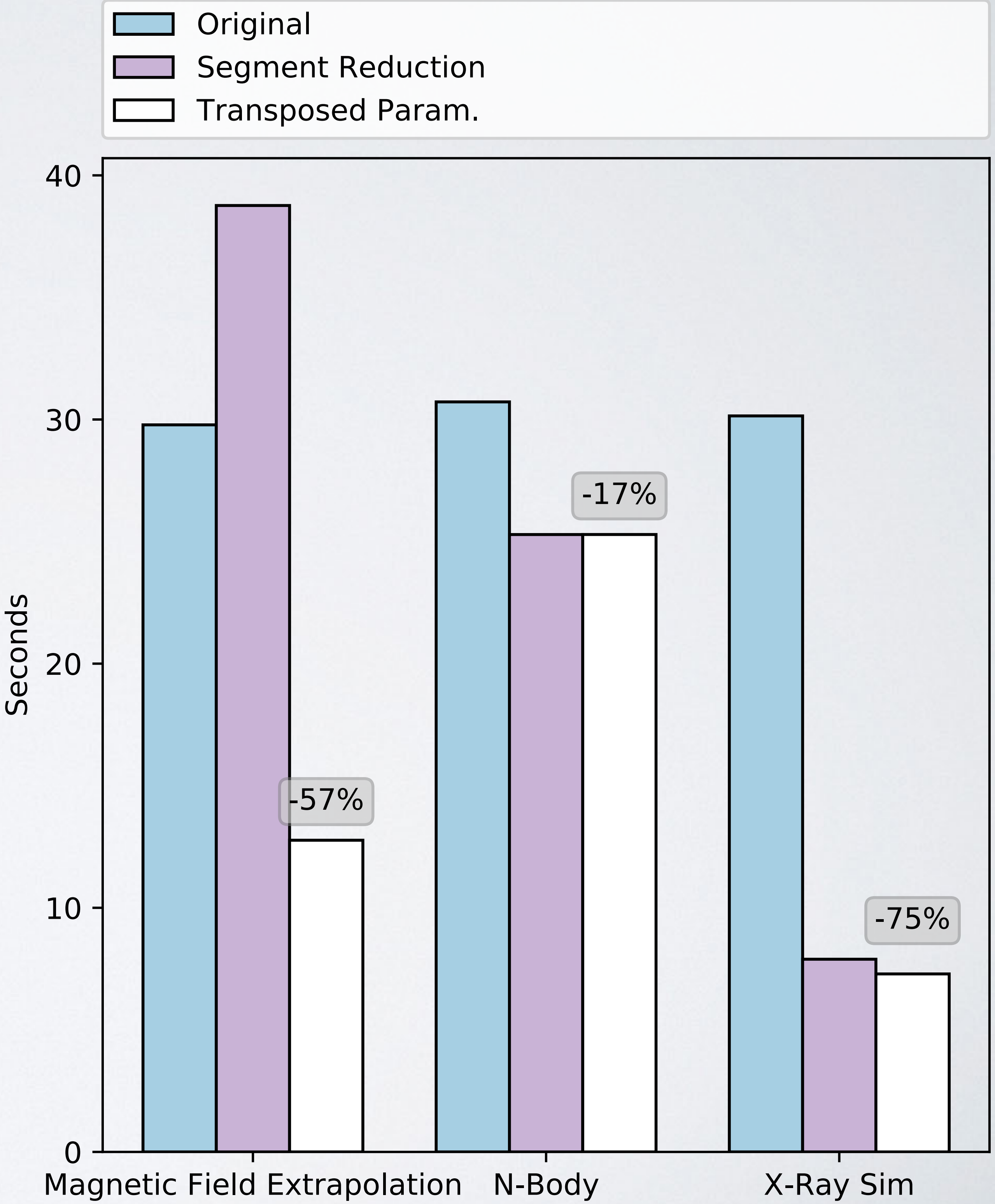
Benchmark Suite

- Embarrassingly Parallel
- Finite Difference
- Streaming/Reduction

I. Magnetic Field Extrapolation

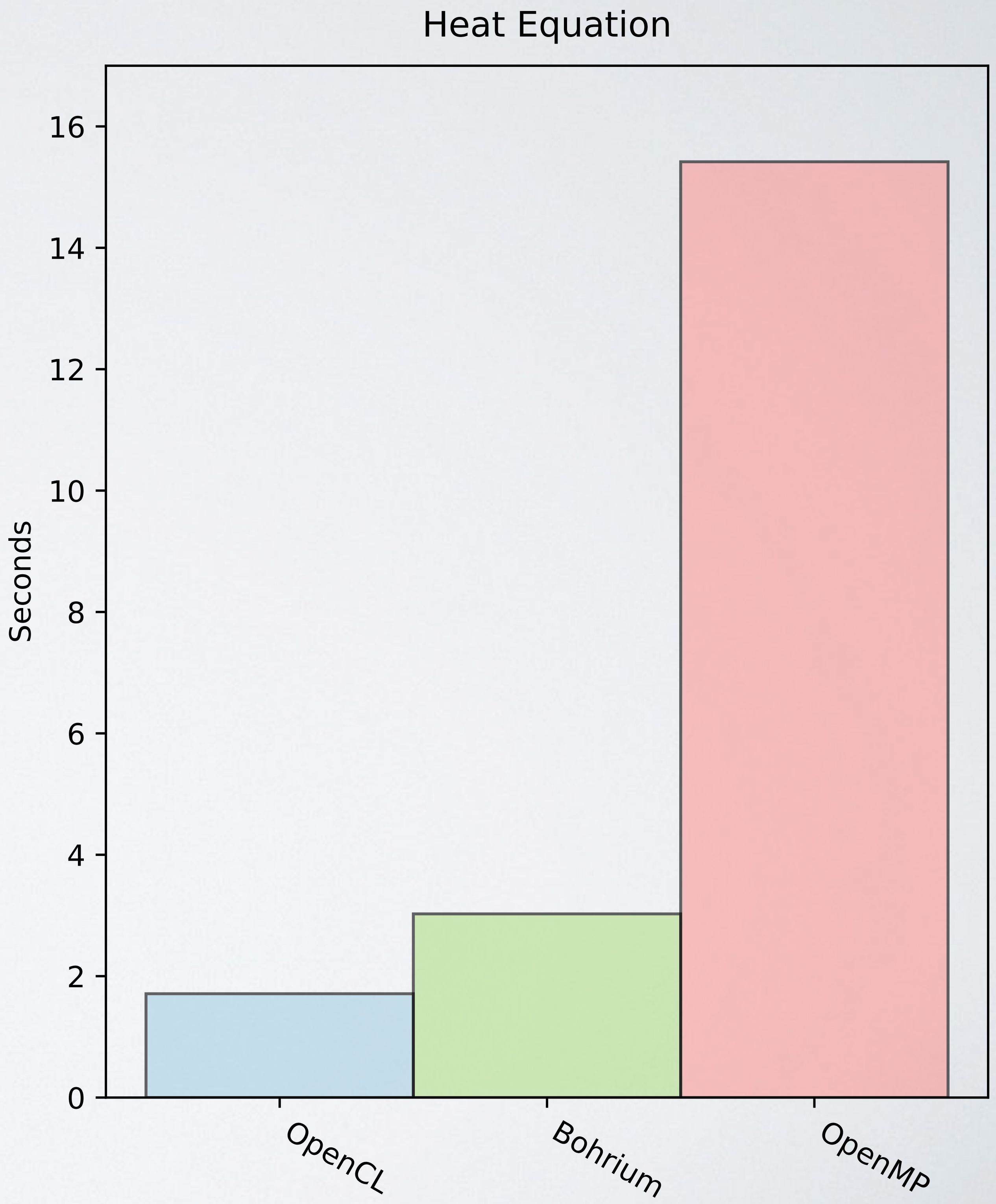
II. N-body

III. X-ray Sim



Handwritten vs. Bohrium

- Handwritten version
- NumPy version
- Complexity of simulations



5. Future Work

v2.0

- Problem with MFE
- Throughput
- Next Focus Points

Problem with MFE

- Access Pattern
- Handling the Access Pattern

MFE Kernel Stripped of Constant Strides
Param. 128 × 1 × 1

```
s8_8 = a8[ +i3*75 +i4];
t6 = a7[ +i2*5625 +i3*75 +i4]
     * s8_8;
```

MFE Kernel Stripped of Constant Strides
Param. 1 × 2 × 64

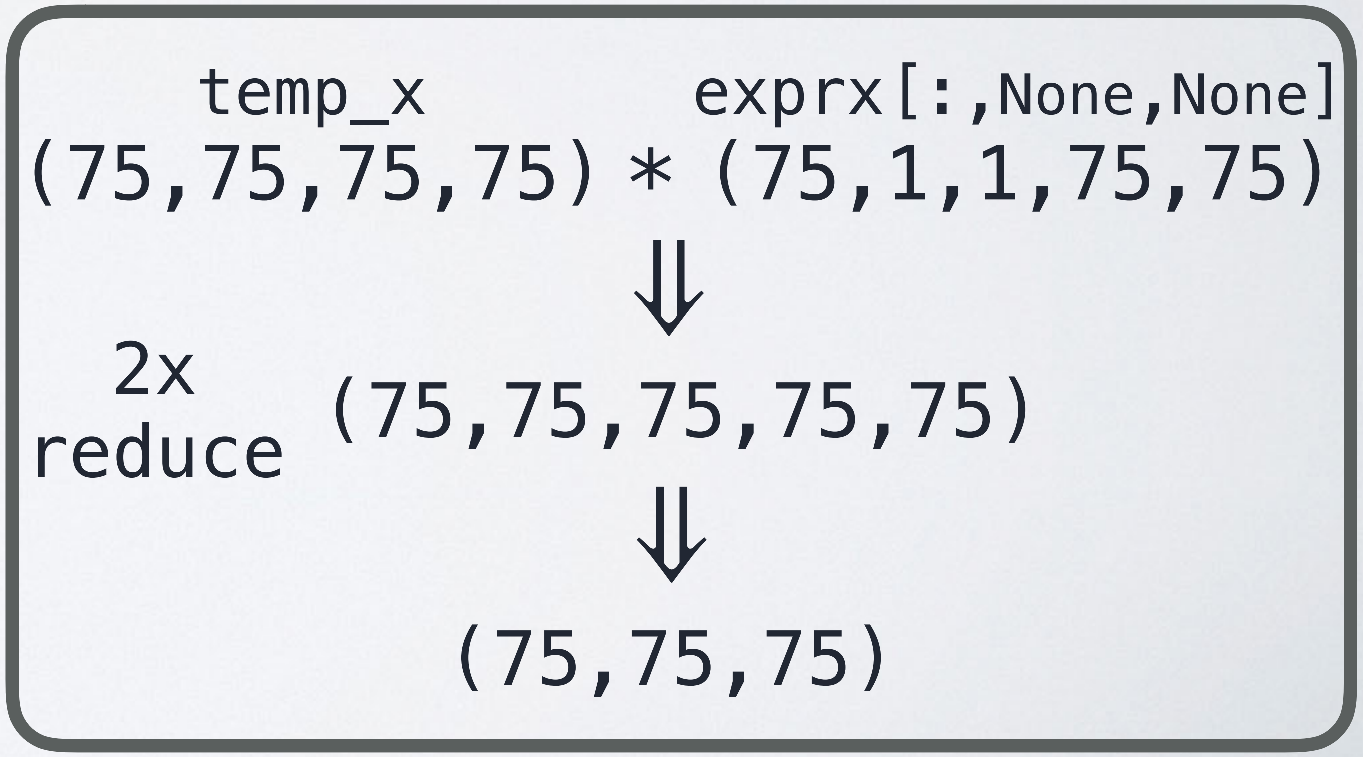
```
s8_8 = a8[ +i0*5625 +i3*75 +i4];
t6 = a7[ +i3*75 +i4] * s8_8;
```

Problem with MFE

- Access Pattern
- Handling the Access Pattern

MFE NumPy Code

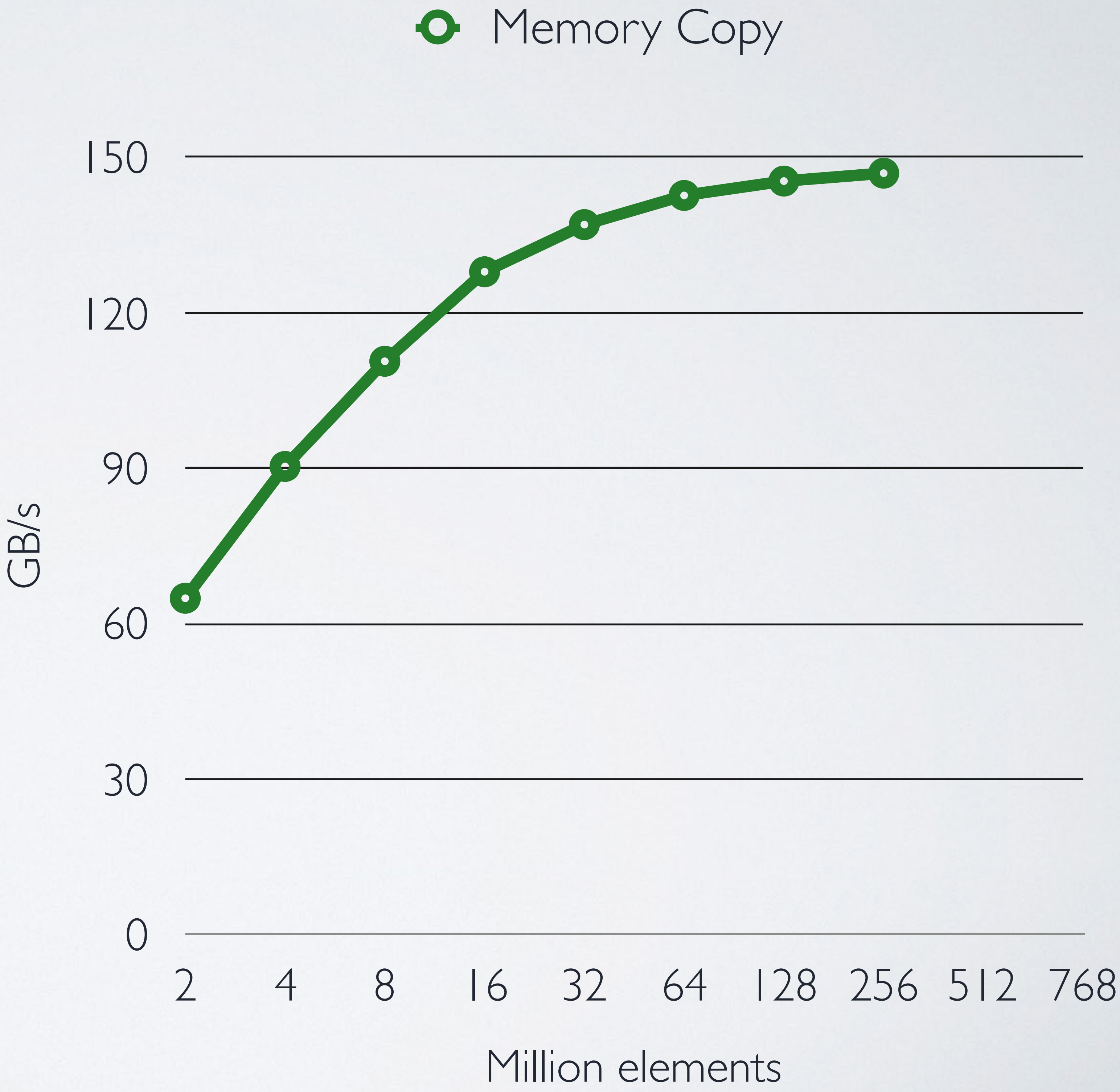
```
np.sum(
 np.sum(
  temp_x * exprx[:, None, None],
  -1),
 -1)
```



```
    temp_x        exprx[:,None,None]
(75,75,75,75) * (75,1,1,75,75)
                  ⇓
  2x
reduce (75,75,75,75,75)
                  ⇓
           (75,75,75)
```

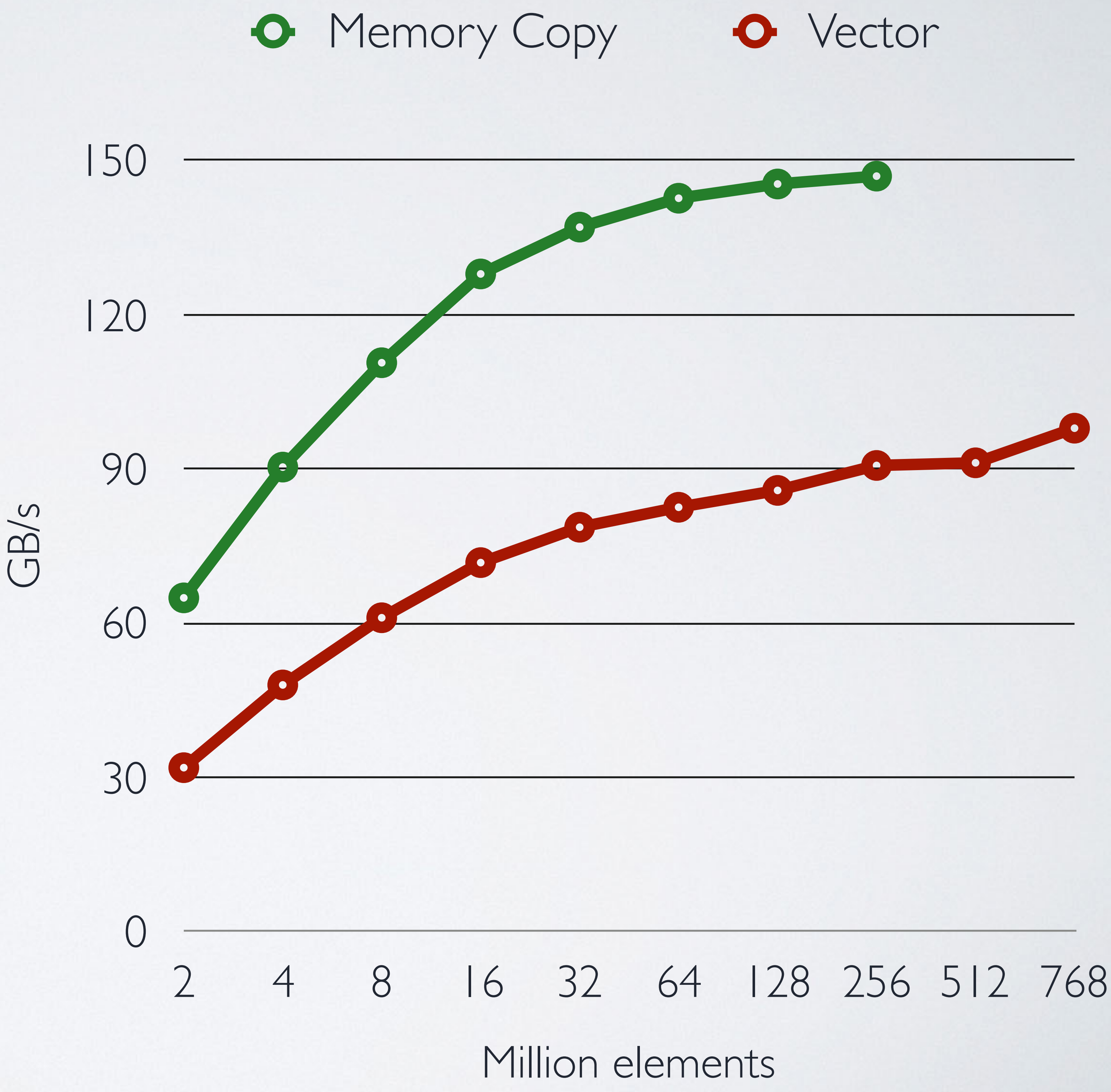
Throughput

- Memory Copy
- Vector Reduction
- Segmented Reduction



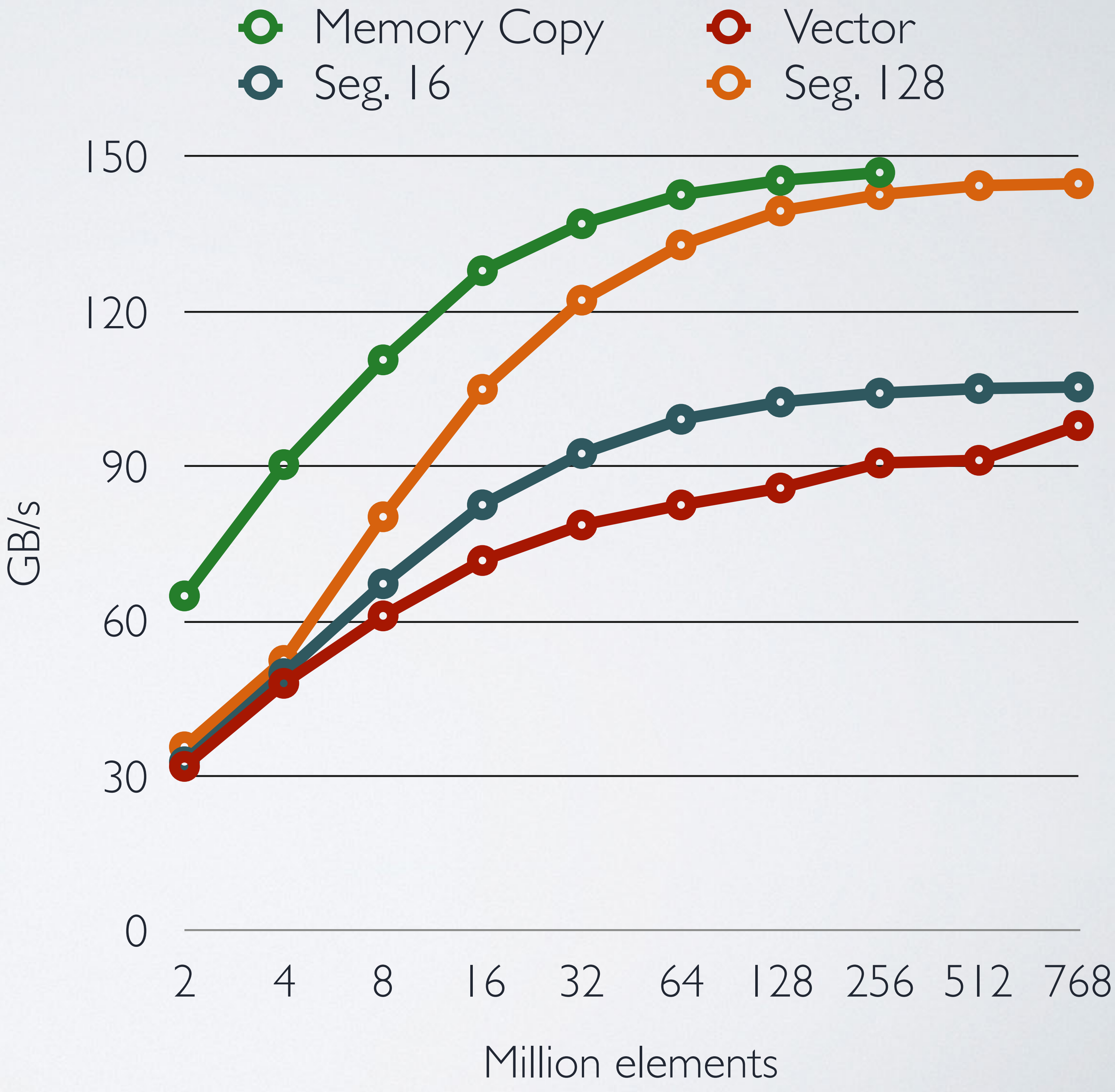
Throughput

- Memory Copy
- Vector Reduction
- Segmented Reduction



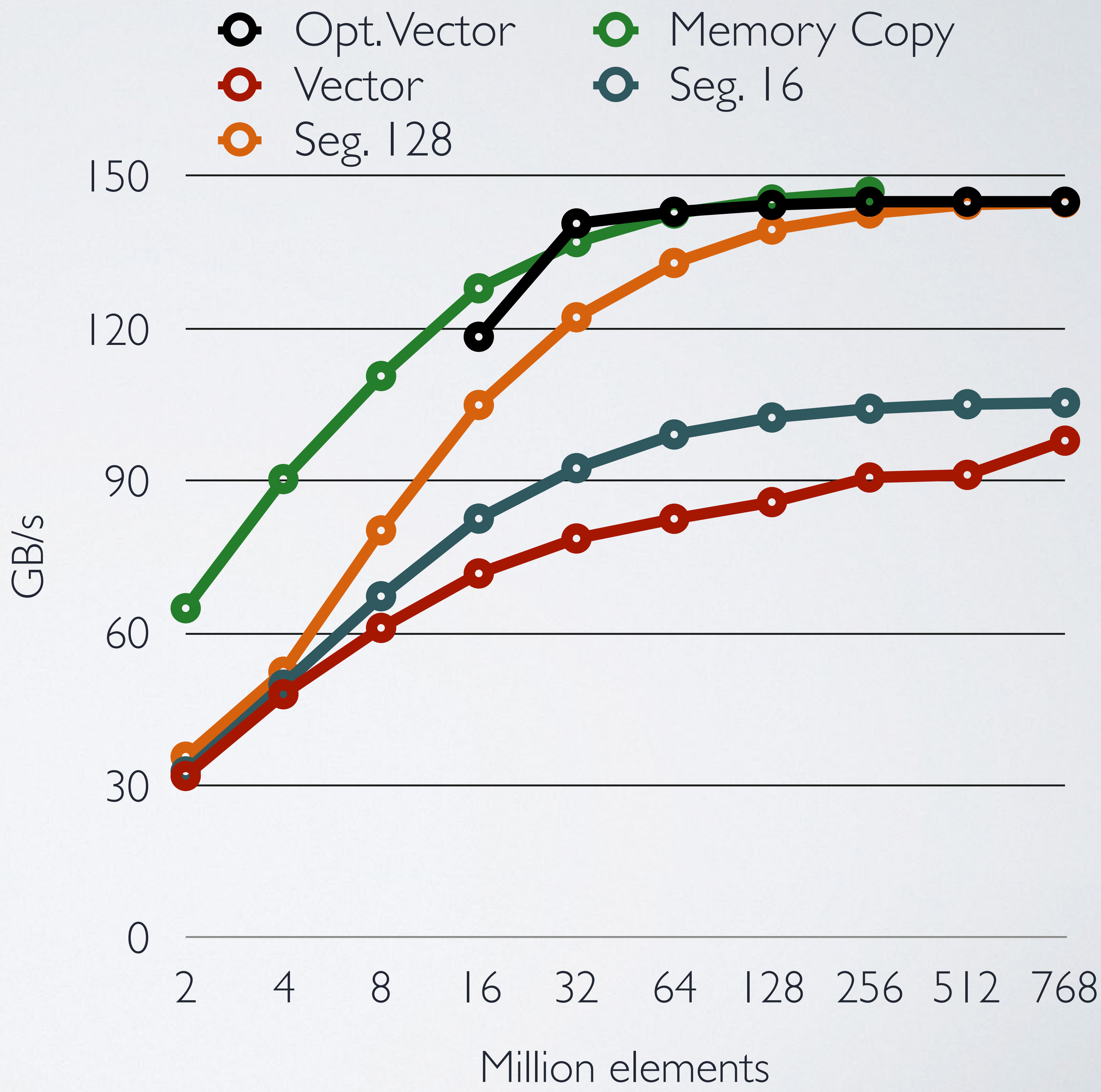
Throughput

- Memory Copy
- Vector Reduction
- Segmented Reduction



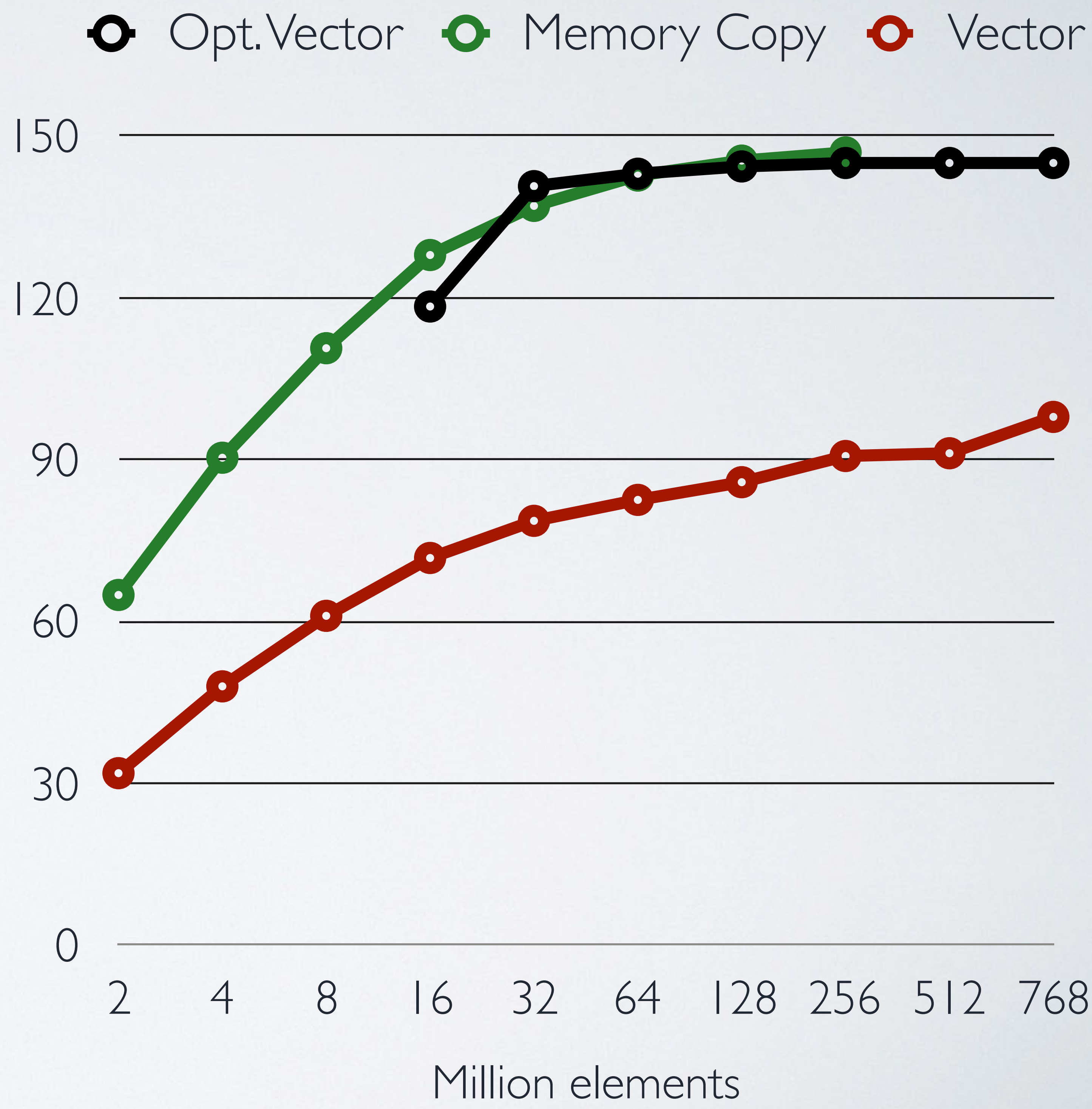
Throughput

- Memory Copy
- Vector Reduction
- Segmented Reduction



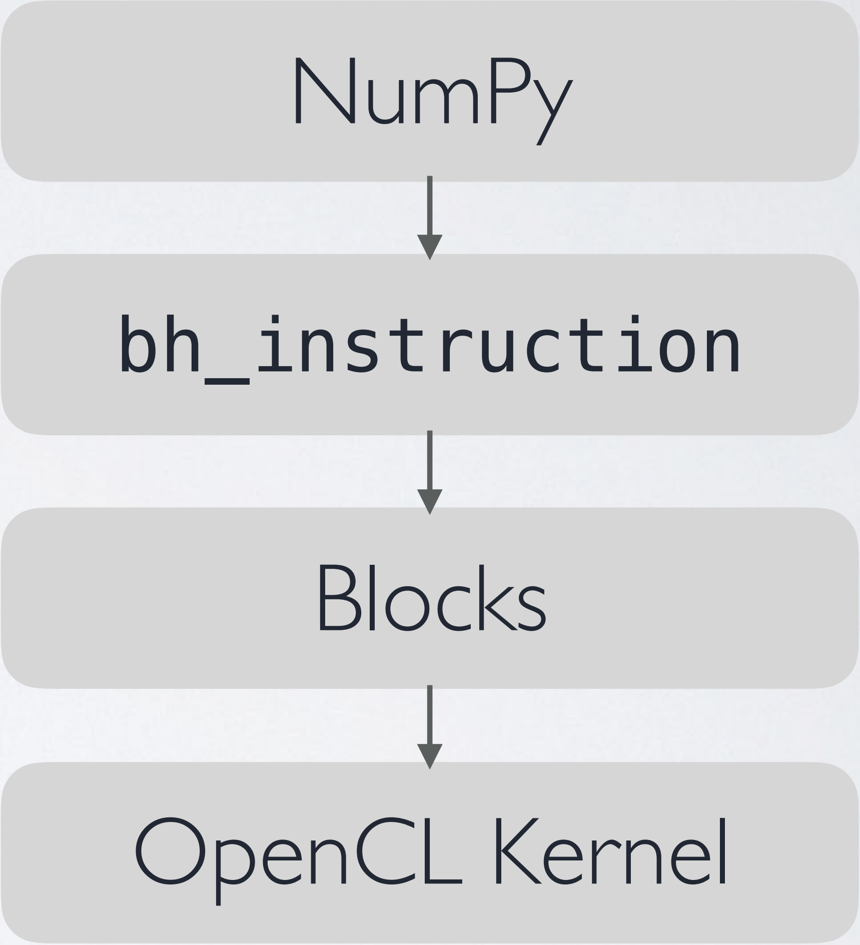
Next Focus Points

- Memory Throughput
- "Chicken or the egg" Issues
- Kernel-based Parameters
- Reshape and Transform



Next Focus Points

- Memory Throughput
- “Chicken or the egg” Issues
- Kernel-based Parameters
- Reshape and Transform


NumPy

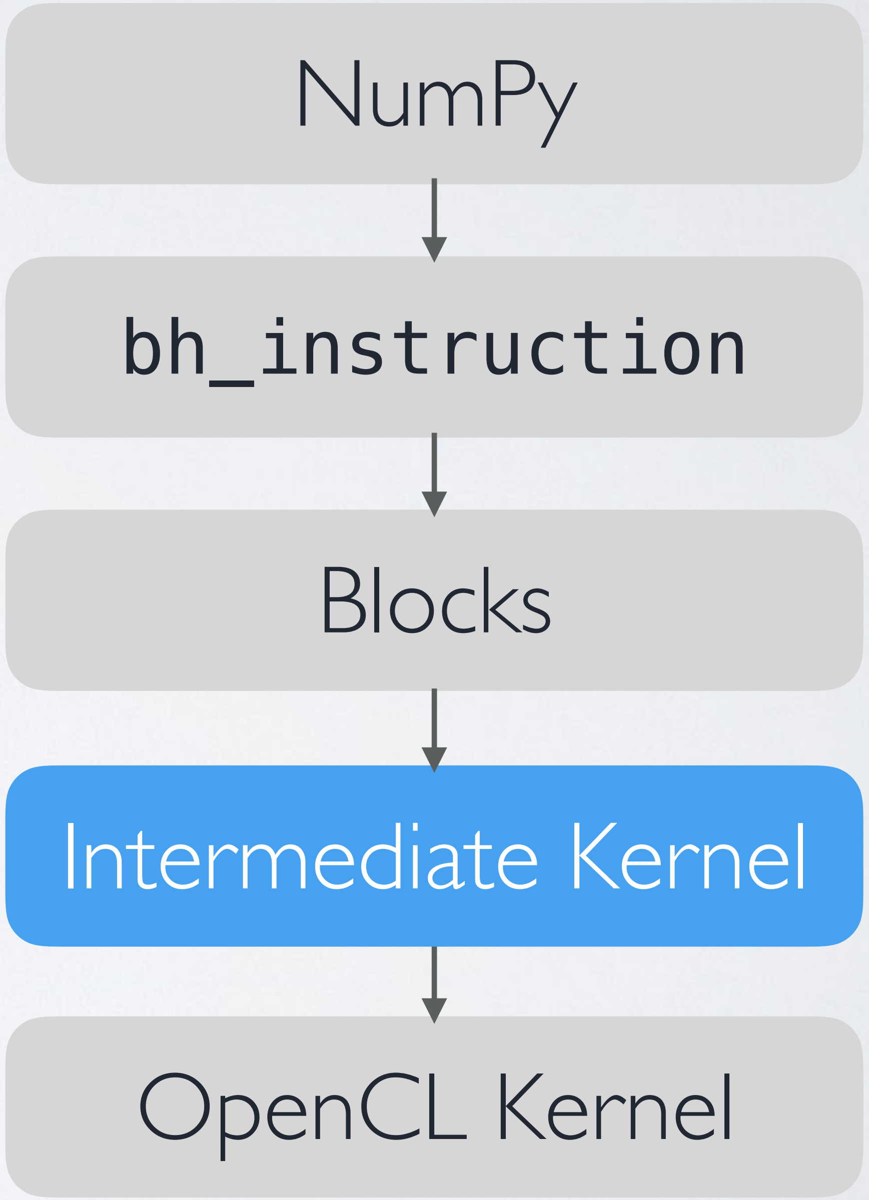
`bh_instruction`

Blocks

OpenCL Kernel

Next Focus Points

- Memory Throughput
- “Chicken or the egg” Issues
- Kernel-based Parameters
- Reshape and Transform



NumPy

`bh_instruction`

Blocks

Intermediate Kernel

OpenCL Kernel

Next Focus Points

- Memory Throughput
- “Chicken or the egg” Issues
- Kernel-based Parameters
- Reshape and Transform

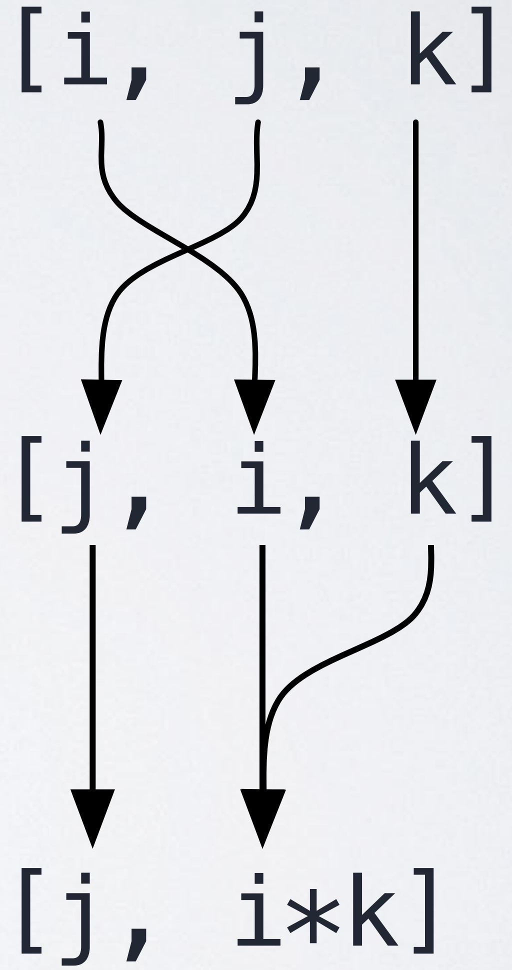
```
exprx[:,None,None]
(75,1,1,75,75)
```

⇓

```
uint g2 = get_global_id(0);
uint g1 = get_global_id(1);
uint g0 = get_global_id(2);
```

Next Focus Points

- Memory Throughput
- “Chicken or the egg” Issues
- Kernel-based Parameters
- Reshape and Transform



6. Conclusion

- Vector Reduction
- Access Pattern
- Performance Gap
- Reflection

6. Conclusion

- Vector Reduction
- Access Pattern
- Performance Gap
- Reflection

6. Conclusion

- Vector Reduction
- Access Pattern
- Performance Gap
- Reflection

6. Conclusion

- Vector Reduction
- Access Pattern
- Performance Gap
- Reflection