GPU offloading in the High Level Trigger of the KOTO experiment

Measurement System Workshop

Kyushu University

Mario Gonzalez, Osaka University

Overview

Contents

- Why GPUs in KOTO's High Level Trigger
- How we have re-written CPU code for massive parallelism on GPUs
- Performance comparison

Goal:

- To encourage you to consider using GPUs in your experiment
- To show how KOTO is planning on taking advantage of them

HEP experiments

- **Thousands** of independent events are readout every second
- Online event processing does not require complex calculations

A CPU server

- **Tens** of independent computing cores that can perform very **complex tasks**

A single GPU

- **Millions** of independent computing threads that can perform **simple tasks**

Heterogeneous Computing:

- CPUs are used together with accelerators (GPUs, etc)
- Each device performs the tasks it is best suited for

the KOTO experiment aims to measure the Branching ratio (BR) of $K_L^0 \longrightarrow \pi^0 \nu \bar{\nu}$

Theoretically predicted to be very small:

$$\mathrm{BR}(K_L^0 \longrightarrow \pi^0 \nu \bar{\nu}) \sim 3 \times 10^{-11}$$

Any small fluctuation from this value due to physics Beyond the SM can be easily observed experimentally

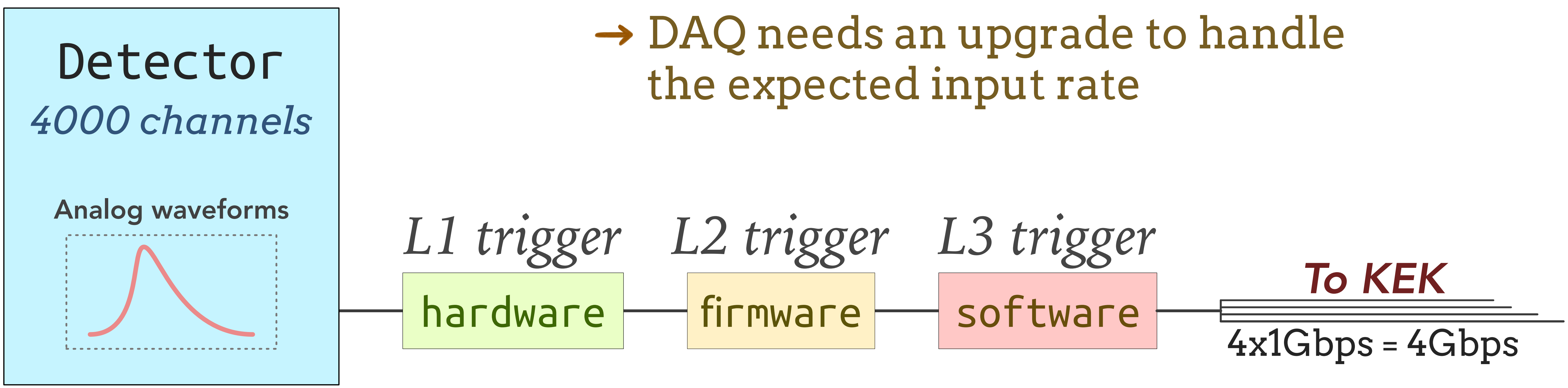
Ongoing beam intensity Upgrade at JPARC

64 kW→ 100 kW (near future)

Hoping to eventually reach 150 kW

→ Input data rate increases, KOTO's sensitivity improves faster

→ DAQ needs an upgrade to handle the expected input rate



The bandwidth to KEK is fixed and it is our main bottleneck

KOTO's data rates at the L3

Input from L2:

waveforms / second

2300*[trig/s]*100[kW]/64[1/kW]*4000[wfms/trig]

GiB / waveform

* 16*64[bits/wfm] / 8/2**30[GiB/bit]

= 1.8 GiB/s (2.6 GiB/s assuming a 150 kW beam)

Output to KEK:

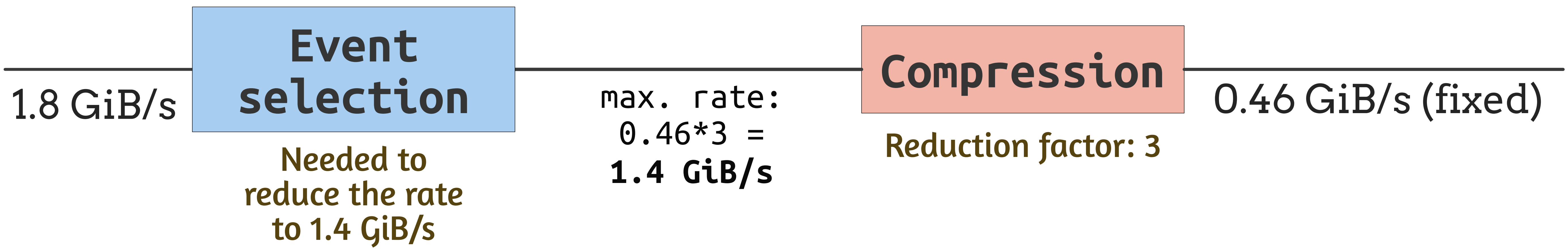
4[Gbps] = 0.46 GiB/s

Input/Output = 1.8/0.46 = 4

1 KiB = 1024 Bytes
2300 Triggers/s is the current rate

KOTO's data rates at the L3

- Data rate needs to be reduce by a factor of 4 at the L3
- We can get a factor of ~3 from waveform compression



Currently, compression is done in CPUs and there is little room for event selection

Option 1:

- Buy more CPUs

Option 2:

- Offload the compression stage to a GPU

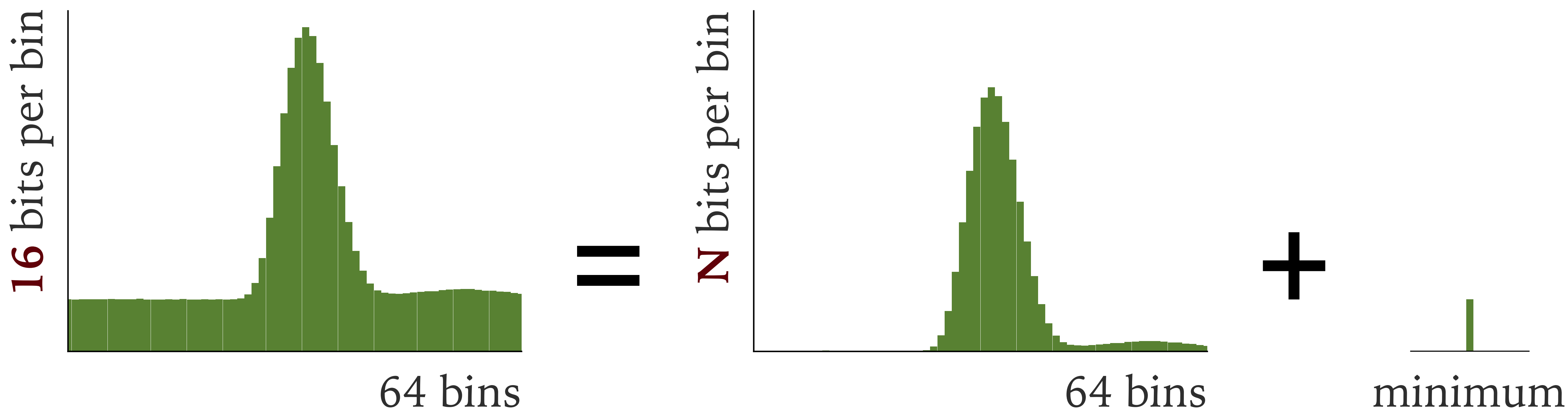
Requirements of the waveform compression

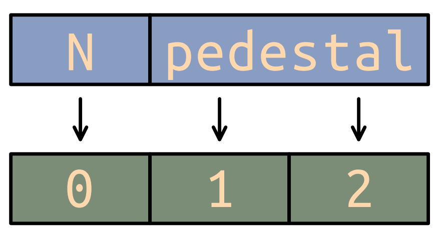
Maximum input rate: 1.4 GiB/s

This means we have:

- *1 s to compress 1.4 GiB*
- *1 s to copy 1.4 GiB to/from the GPU*

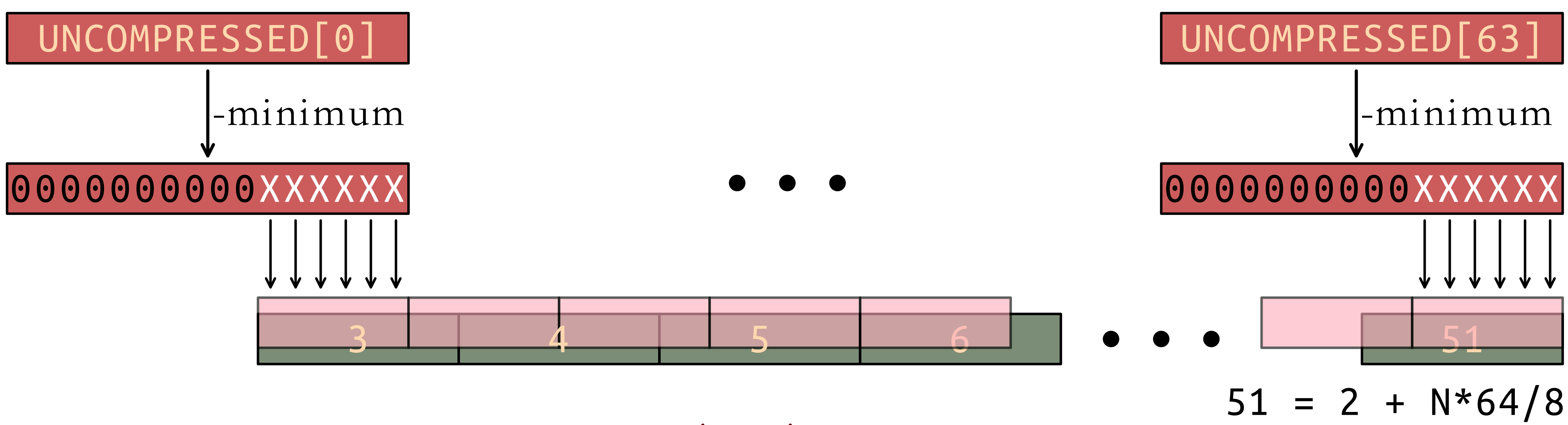
Waveform compression in the KOTO experiment





Uncompressed array (`uint16_t`)
Compressed array (`uint8_t`)

Assuming N = 6,



The Challenge of doing compression on GPUs

```
for wfm in packet:
    for bin in wfm:
        # Find minimum and N

    for bin in wfm:
        # Write compressed wfm
```

~1e7 wfms in a 1.4 GiB packet

64 bins

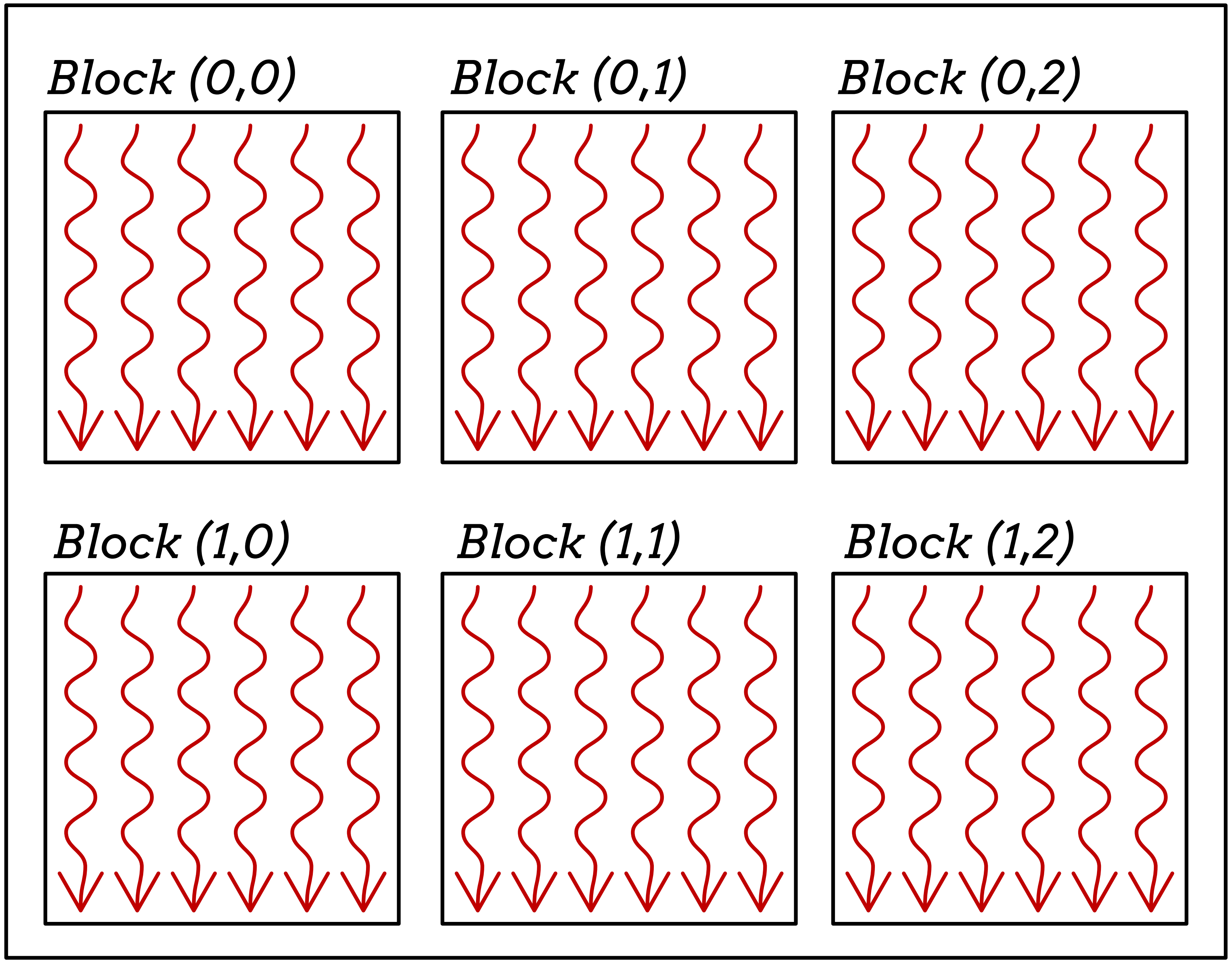
64 bins

Goal

Take advantage of the huge number of GPU threads to parallelize for loops and minimize the number of iterations

The GPU architecture

Grid 1



Threads within the same block share very fast memory

Grid size and block size are configurable within some limits

First step

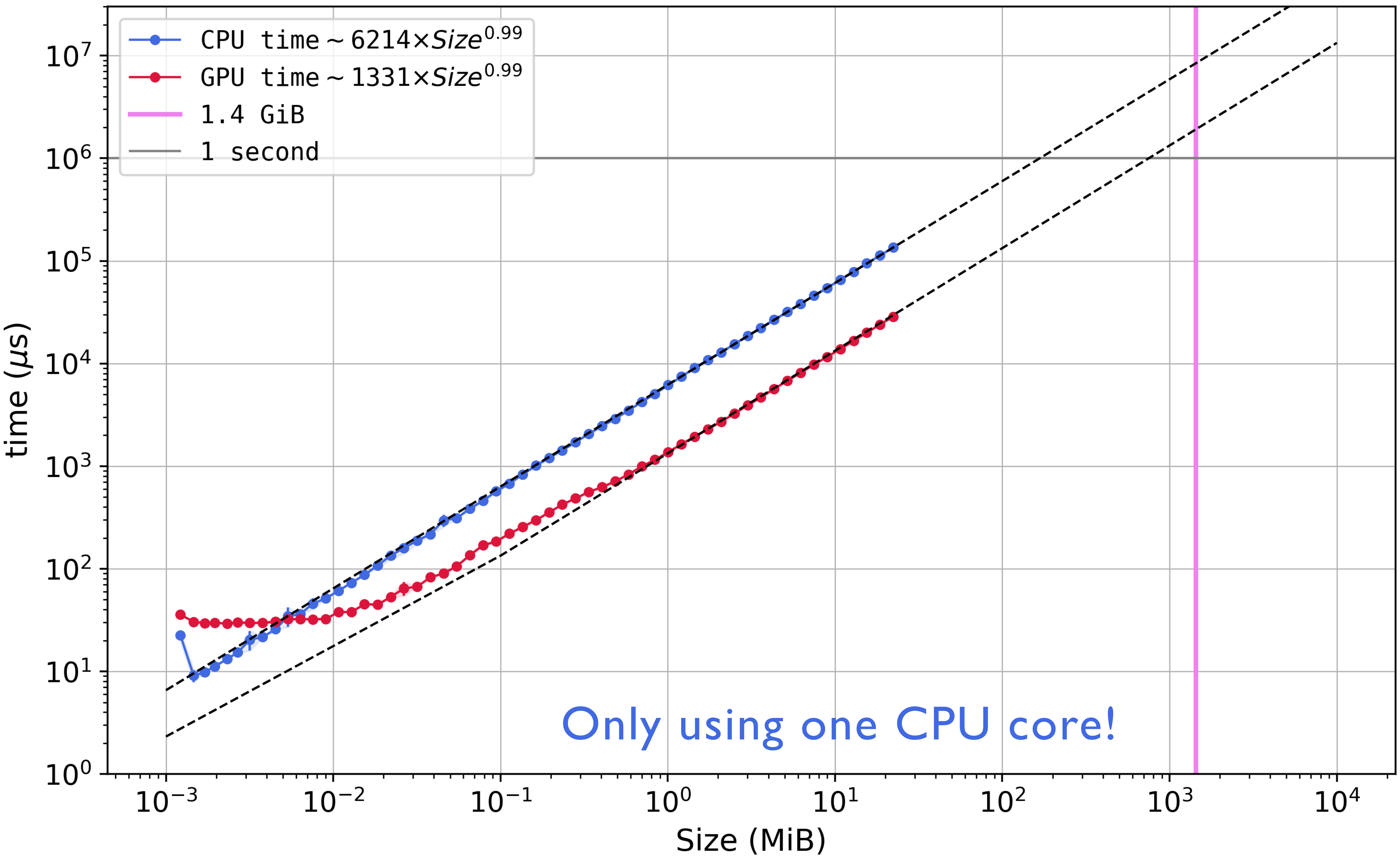
```
for wfm in packet:
    for bin in wfm:
        # Find minimum and N

    for bin in wfm:
        # Write compressed wfm
```

~1e7 wfms per packet

- Get rid of this loop by processing one waveform per thread
- Most of the code doesn't change

Results

At least a factor of 2 improvement needed on GPU
At least a factor of 10 improvement needed on CPU



Second step

Start re-designing the code for massive parallelism

- One block per waveform
- 64 threads per block

```
~~for wfm in packet:~~

for bin in wfm: (next slide)
    # Find minimum and N

for bin in wfm:  (later)
    # Write compressed wfm
```

Re-write "Find maximum in an array" on GPUs

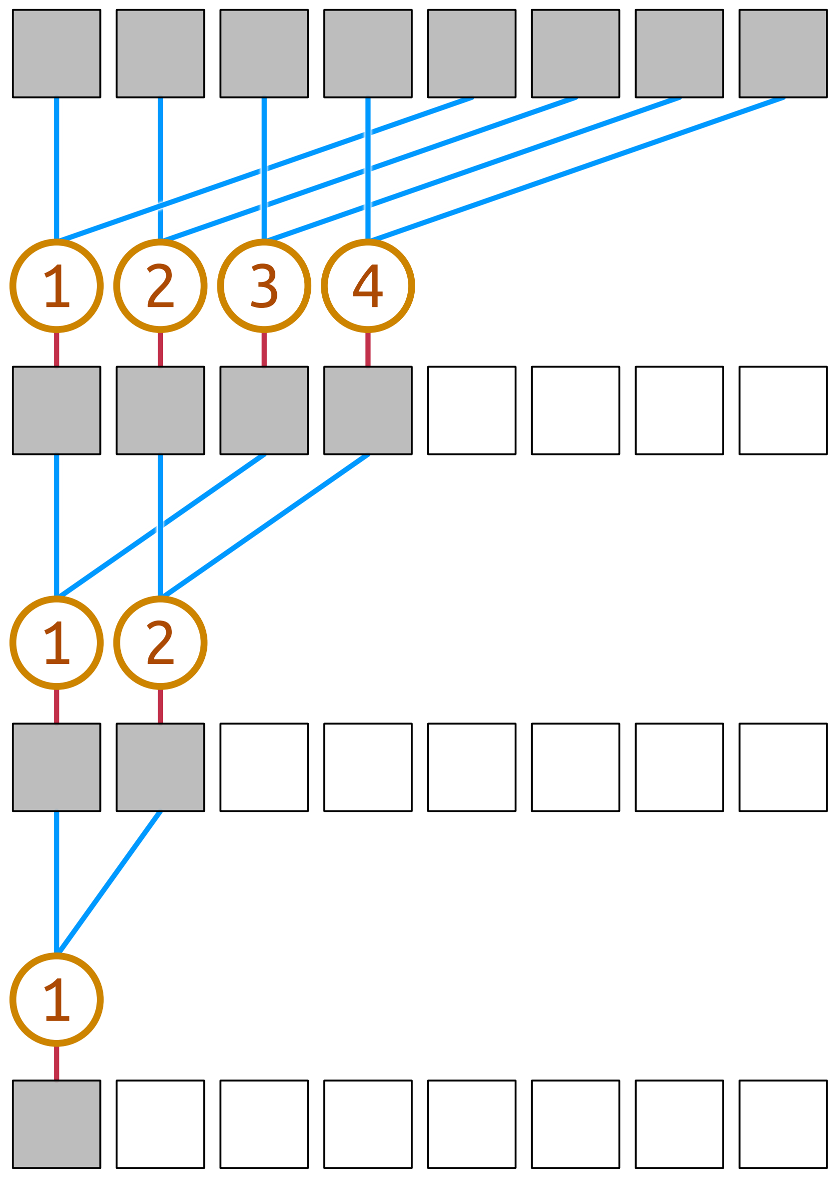
How to find the max/min of an array of integers

...on a GPU

Parallel reduction

Computation is fast

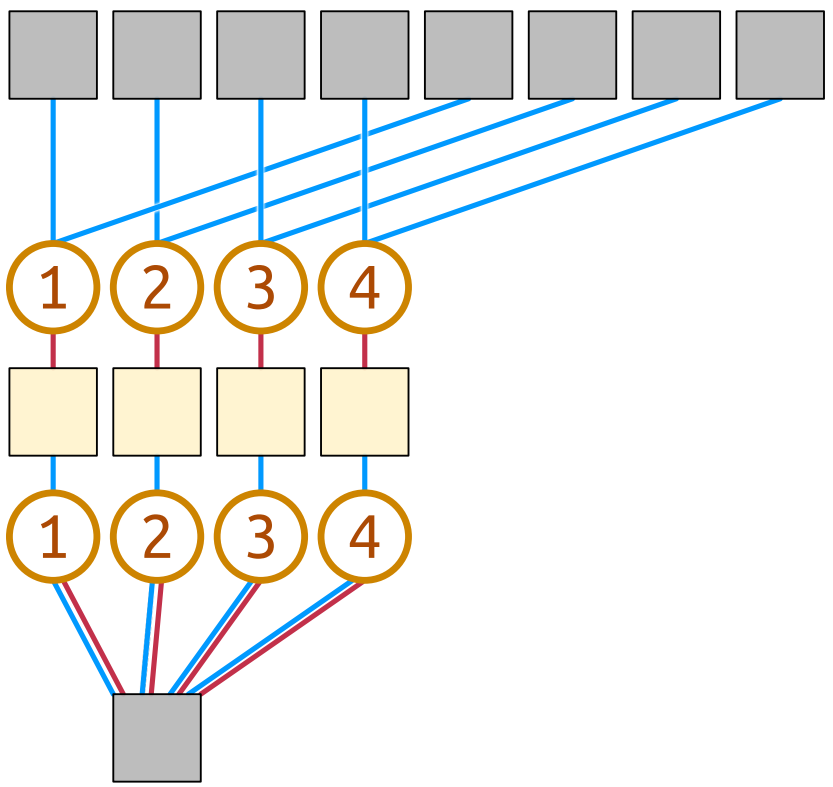
Needs to allocate extra memory



Method 2

No extra memory required

Multiple threads write to the same address, so needs atomic operations



Global Memory

Thread Memory

① Thread (performs max() / min())

Input

Output

Parallelizing the compression

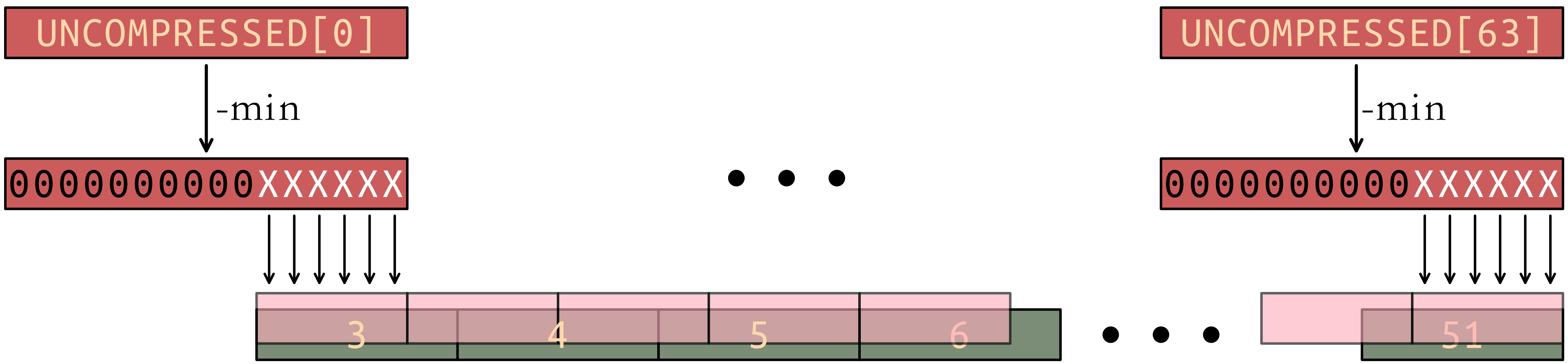
```
for wfm in packet:

    for bin in wfm:
        # Find N

    for bin in wfm:
        # Write compressed array
```

Write the compressed array in parallel

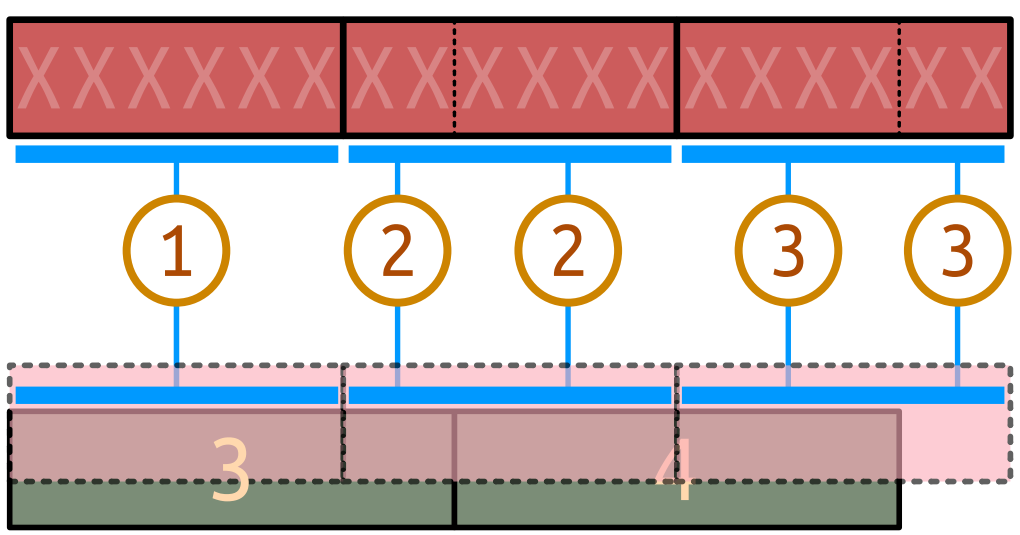
Parallelizing the compression

Uncompressed array (uint16_t)
Compressed array (uint8_t)

Assuming N = 6,



Option 1: As many threads as waveform bins (red squares): 64



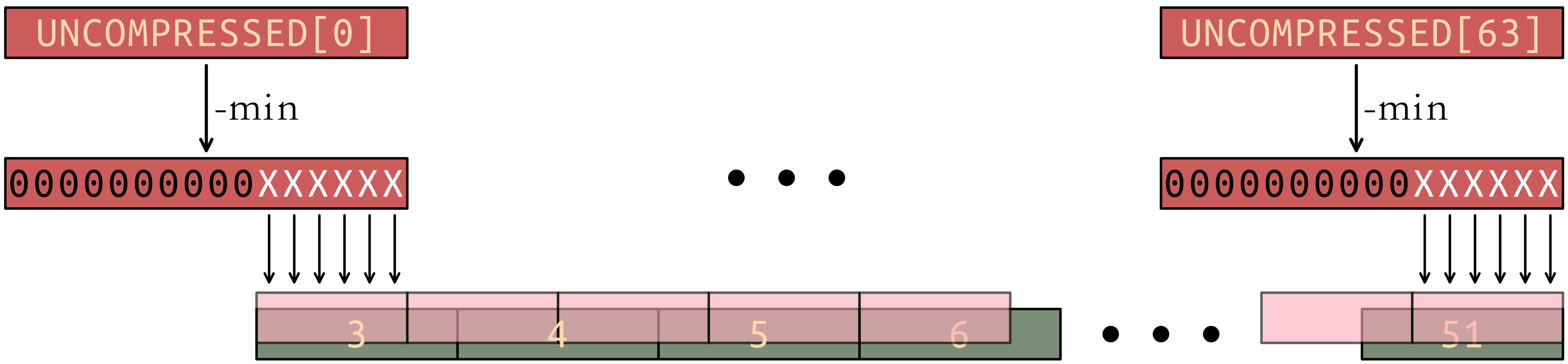
Every thread subtracts the minimum from one input element

Multiple threads write to the same address simultaneously, so atomic operations are needed

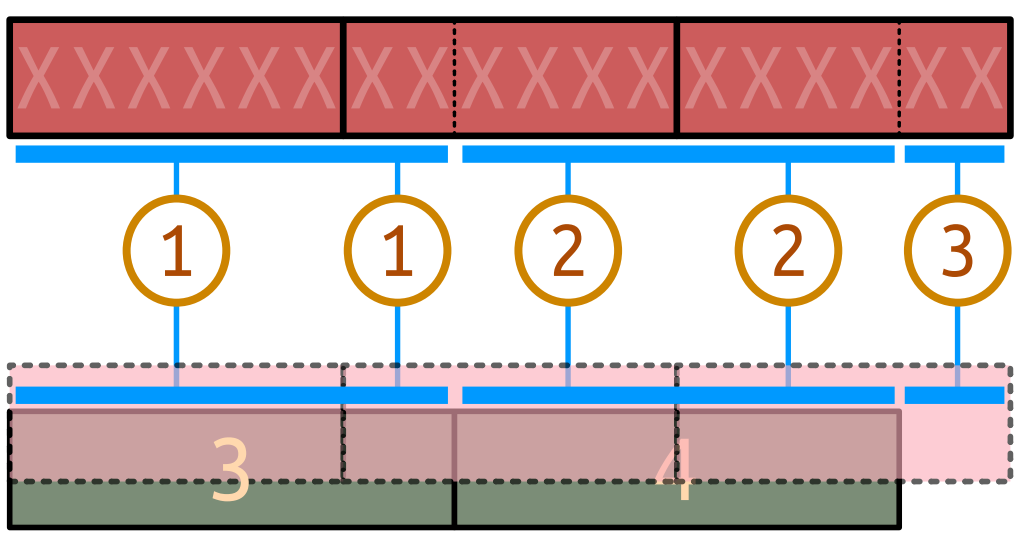
Parallelizing the compression

Uncompressed array (`uint16_t`)
Compressed array (`uint8_t`)

Assuming N = 6,



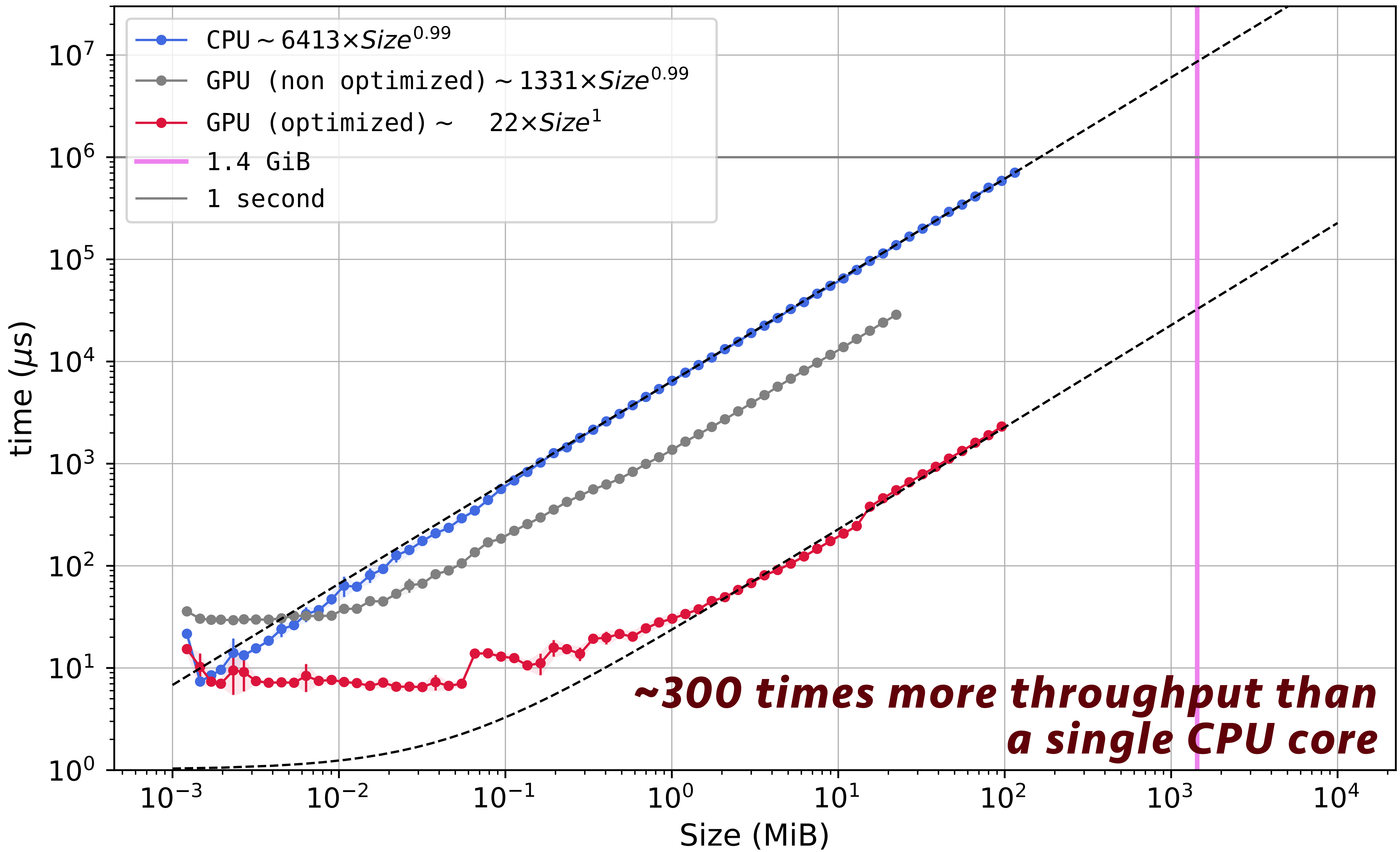
Option 2: As many threads as compressed bytes (green squares)



Fewer number of threads, since the compressed array has less bytes than the original

No write conflicts between threads, no atomic operations needed

Results

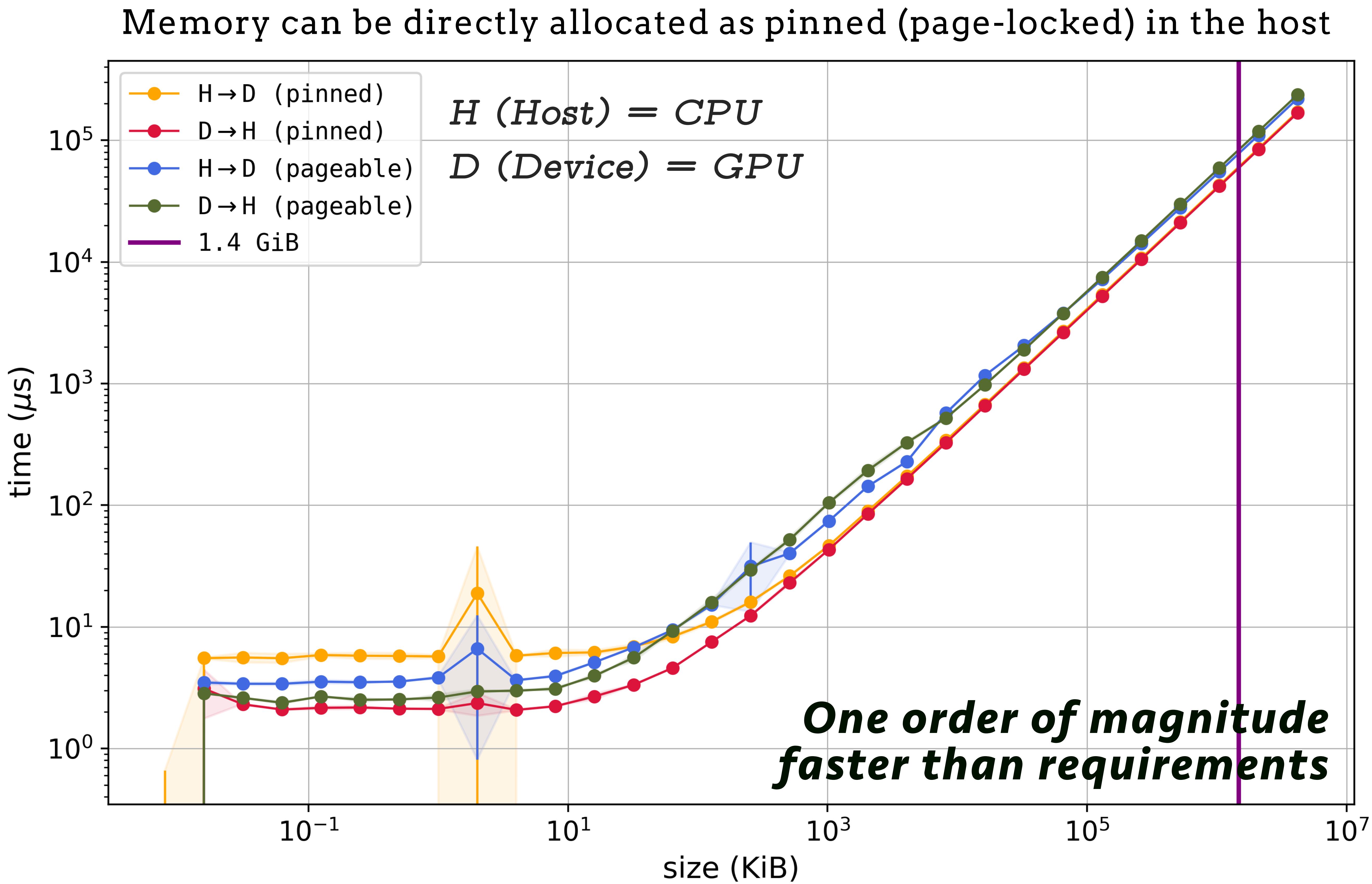

CPU ~ 6413×Size^0.99
GPU (non optimized) ~ 1331×Size^0.99
GPU (optimized) ~ 22×Size^1
1.4 GiB
1 second
time (μs)
Size (MiB)
~300 times more throughput than
a single CPU core

Checklist update

✓ - *1 s to compress 1.4 GiB*

? - *1 s to copy 1.4 GiB to/from the GPU*

CPU-GPU data rates



Checklist update

- ✓ 1 s to compress 1.4 GiB
- ✓ 1 s to copy 1.4 GiB to/from the GPU

Conclusions

GPUs have a great potential in HEP triggers and will be part of the future of KOTO

- *One GPU can perform faster than a whole CPU server*
- *They are a lot smaller, cheaper and energetically efficient*


BACKUP

Hardware

```
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce RTX 3060 Ti"
  CUDA Driver Version / Runtime Version          11.4 / 11.4
  CUDA Capability Major/Minor version number:    8.6
  Total amount of global memory:                 7980 MBytes (8367439872 bytes)
  (038) Multiprocessors, (128) CUDA Cores/MP:    4864 CUDA Cores
  GPU Max Clock rate:                            1695 MHz (1.70 GHz)
  Memory Clock rate:                             7001 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 3145728 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        102400 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1536
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):     Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.4, CUDA Runtime Version = 11.4, NumDevs = 1
Result = PASS
```

GPU

NVIDIA GeForce
RTX 3060 Ti

CPU

Intel(R) Core(TM)
i7-4770 CPU @ 3.40GHz