



Exercise 6 – GPUs

Sebastian Kuckuk

Erlangen National High Performance Computing Center (NHR@FAU)

Assignment 6 – Task 1

Performance: $P_{chip} = n_{core} \cdot n_{super}^{FP} \cdot n_{FMA} \cdot n_{SIMD} \cdot f$

The diagram shows the equation $P_{chip} = n_{core} \cdot n_{super}^{FP} \cdot n_{FMA} \cdot n_{SIMD} \cdot f$. Below each variable, there is a label with an arrow pointing to it: n_{core} is labeled "#Cores", n_{super}^{FP} is labeled "Super-scalarity", n_{FMA} is labeled "FMA factor", n_{SIMD} is labeled "SIMD factor", and f is labeled "Clock Speed".

■ Float

- $P_{CPU} = (2 \cdot 64) \cdot 1 \frac{inst}{cyc} \cdot 2 \frac{Flop}{FMA} \cdot 16 \frac{FMA}{inst} \cdot 2.4 GHz = 9.8 \frac{TFlop}{s}$
- $P_{GPU} = (168 \cdot 4) \cdot 16 \frac{inst}{cyc} \cdot 2 \frac{Flop}{FMA} \cdot 1 \frac{FMA}{inst} \cdot 1.5 GHz = 32.3 \frac{TFlop}{s}$
- $P_{node} = 4 \cdot P_{GPU} = 129.0 \frac{TFlop}{s}$
- $P_{cluster} = 192 \cdot P_{node} = 24.8 \frac{PFlop}{s}$

Assignment 6 – Task 1

Performance: $P_{chip} = n_{core} \cdot n_{super}^{FP} \cdot n_{FMA} \cdot n_{SIMD} \cdot f$

The diagram shows the equation $P_{chip} = n_{core} \cdot n_{super}^{FP} \cdot n_{FMA} \cdot n_{SIMD} \cdot f$ with five blue arrows pointing from labels below to variables above:

- From **#Cores** to n_{core}
- From **Super-scalarity** to n_{super}^{FP}
- From **FMA factor** to n_{FMA}
- From **SIMD factor** to n_{SIMD}
- From **Clock Speed** to f

■ Double

- $P_{CPU} = (2 \cdot 64) \cdot 1 \frac{inst}{cyc} \cdot 2 \frac{Flop}{FMA} \cdot 8 \frac{FMA}{inst} \cdot 2.4 GHz = 4.9 \frac{TFlop}{s}$
- $P_{GPU} = (168 \cdot 4) \cdot 1 \frac{inst}{cyc} \cdot 2 \frac{Flop}{FMA} \cdot 1 \frac{FMA}{inst} \cdot 1.5 GHz = 2.0 \frac{TFlop}{s}$
- $P_{node} = 4 \cdot P_{GPU} = 8.1 \frac{TFlop}{s}$
- $P_{cluster} = 192 \cdot P_{node} = 1.5 \frac{PFlop}{s}$

Assignment 6 – Task 2

- #FLOP for a **D**ouble **G**eneral **P**recision **M**atrix **M**atrix Multiplication (DGEMM) for matrices with dimensions $MK \times KN = MN$
- $N_{flops} = 2 \cdot MNK$
- Assume compute boundness in the computational part
- Assume no overlap between
 - Host-device data transfers
 - Device-host data transfers
 - Computation

Assignment 6 – Task 2

■ Float

$$■ T_{CPU} = \frac{N_{Flops}}{P_{CPU}} = \frac{2 \cdot 4096^3 \text{ Flop}}{9.8 \frac{\text{TFlop}}{s}} = 14.0 \text{ ms}$$

$$■ T_{GPU} = \frac{N_{Flops}}{P_{GPU}} = \frac{2 \cdot 4096^3 \text{ Flop}}{129.0 \frac{\text{TFlop}}{s}} = 1.1 \text{ ms}$$

$$■ T_{PCIe} = \frac{V}{B} = \frac{3 \cdot 4096^2 \cdot 4 \text{ B}}{2 \cdot 32.0 \frac{\text{GB}}{s}} = 3.1 \text{ ms}$$

$$■ T_{Total} = T_{GPU} + T_{PCIe} = 1.1 + 3.1 \text{ ms} = 4.2 \text{ ms}$$

$$■ S = \frac{T_{CPU}}{T_{Total}} = 3.3$$

Assignment 6 – Task 2

■ Double

$$■ T_{CPU} = \frac{N_{Flops}}{P_{CPU}} = \frac{2 \cdot 4096^3 Flop}{4.9 \frac{TFlop}{s}} = 28.0 ms$$

$$■ T_{GPU} = \frac{N_{Flops}}{P_{GPU}} = \frac{2 \cdot 4096^3 Flop}{8.1 \frac{TFlop}{s}} = 17.0 ms$$

$$■ T_{PCIe} = \frac{V}{B} = \frac{3 \cdot 4096^2 \cdot 8 B}{2 \cdot 32.0 \frac{GB}{s}} = 6.3 ms$$

$$■ T_{Total} = T_{GPU} + T_{PCIe} = 17.0 + 6.3 ms = 23.3 ms$$

$$■ S = \frac{T_{CPU}}{T_{Total}} = 1.2$$

Assignment 6 – Task 3

- Little's Law

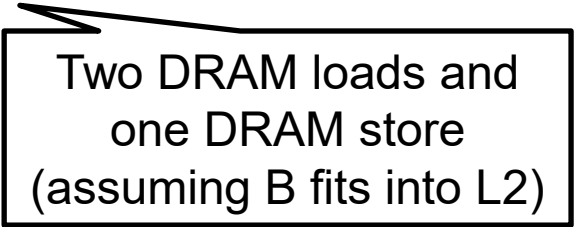
$$L = \frac{N}{b_s} \Leftrightarrow N = L \cdot b_s$$

The diagram illustrates Little's Law with the equation $L = \frac{N}{b_s} \Leftrightarrow N = L \cdot b_s$. Three callout boxes are present: one labeled '#Bytes in flight' pointing to the variable N in the second equation; one labeled 'Latency' pointing to the variable L in the second equation; and one labeled 'Bandwidth' pointing to the variable b_s in the second equation.

Assignment 6 – Task 3

- Kernel code

```
__global__ void kernel(double* A, double* B, double* C, int size) {  
    int tidx = threadIdx.x + blockDim.x * blockIdx.x;  
    int tidy = threadIdx.y + blockDim.y * blockIdx.y;  
  
    if (tidx >= size || tidy >= size) return;  
  
    C[tidx + tidy * size] += A[tidx + tidy * size] * B[tidy];  
}
```



Two DRAM loads and
one DRAM store
(assuming B fits into L2)

Assignment 6 – Task 3

■ Little's Law

- $L = \frac{N}{b_s} \leftrightarrow N = L \cdot b_s$

- $L_{seconds} = \frac{L_{cycles}}{F}$

- $N_{byte} = N_{threads} \cdot V_{thread}$

- $N_{byte} = 900.0 \frac{GB}{s} \cdot \frac{800}{1.5 GHz} = 480 KB$

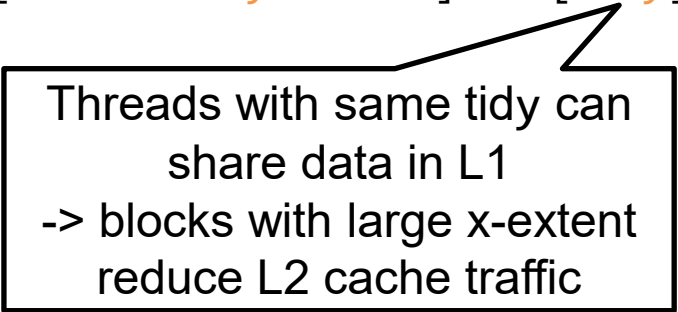
- $N_{thread} = \frac{V}{4 \cdot 3 \frac{B}{thread}} = \frac{480 KB}{12 \frac{B}{thread}} = 40000$

Minimum number of threads required to saturate the DRAM bandwidth, actual number is usually higher

Assignment 6 – Task 3

- Kernel code

```
__global__ void kernel(double* A, double* B, double* C, int size) {  
    int tidx = threadIdx.x + blockDim.x * blockIdx.x;  
    int tidy = threadIdx.y + blockDim.y * blockIdx.y;  
  
    if (tidx >= size || tidy >= size) return;  
  
    C[tidx + tidy * size] += A[tidx + tidy * size] * B[tidy];  
}
```



Threads with same tidy can
share data in L1
-> blocks with large x-extent
reduce L2 cache traffic

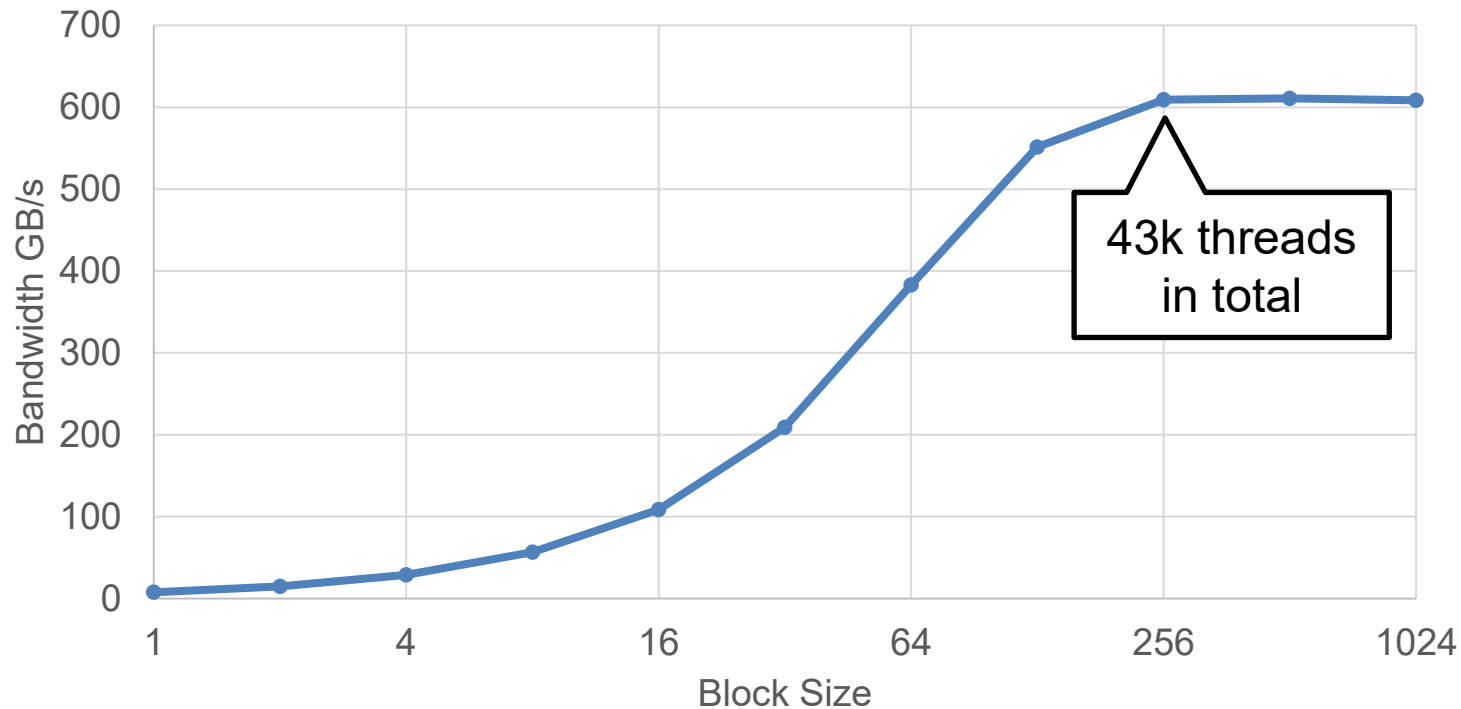
Assignment 6 – Task 4

- Kernel code for $a[i] = a[i] * 2.0$

```
__global__ void kernel(double* a, int size) {  
    int x = threadIdx.x + blockDim.x * blockIdx.x;  
    int stride = blockDim.x * blockDim.x;  
  
    for ( ; x <= size; x += stride)  
        a[x] *= 2;  
}
```

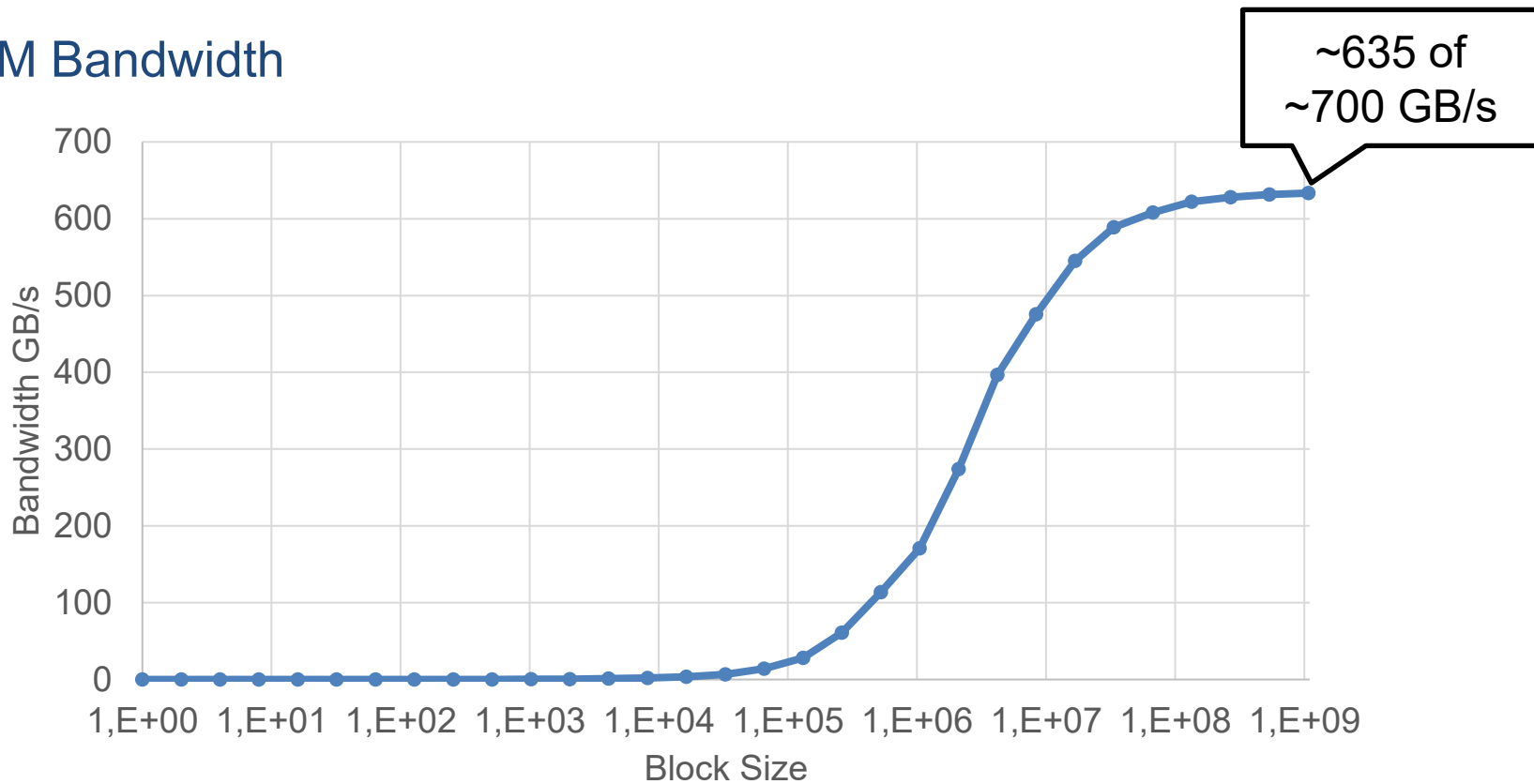
Assignment 6 – Task 5

- Block size investigation



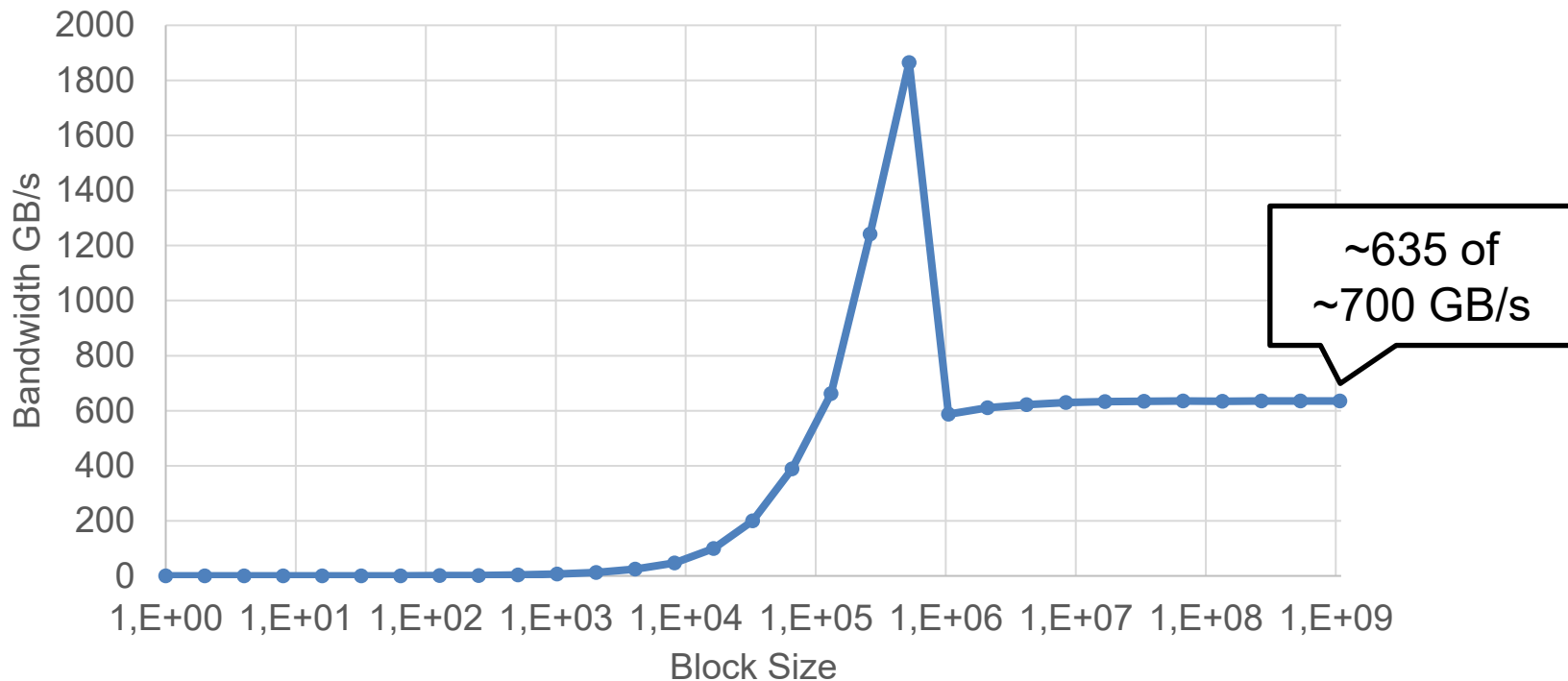
Assignment 6 – Task 6

- DRAM Bandwidth



Assignment 6 – Task 6

- DRAM Bandwidth (with one warm-up iteration, not part of exercise)



Assignment 6 – Task 6

■ PCIe Bandwidth

