Programming Techniques for Supercomputers

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Exercise 6 – GPUs

S. Kuckuk, D. Ernst, G. Wellein, G. Hager

Erlangen National High Performance Computing Center (NHR@FAU)

Assignment 6 – Task 1

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

- For one Alex node, FP64
  - $P_{CPU} = 2 \cdot 64\ cores \cdot 2\frac{instruction}{cycle} \cdot 2\frac{FLop}{FMA} \cdot 4\frac{FMA}{instruction} \cdot 2\ Ghz = 4096\frac{GFlop}{s}$
  - $P_{GPU} = 108\ SM \cdot 1\frac{instruction}{cycle} \cdot 2\frac{FLop}{FMA} \cdot 32\frac{FMA}{SM} \cdot 1.41\ Ghz = 9746\frac{GFlop}{s}$
  - $P_{8-GPU} = 8 \cdot P_{GPU} = 78\frac{TLop}{s}$

Assignment 6 – Task 2

- #FLOP for a **D**ouble **G**eneral **P**recision **M**atrix **M**atrix Multiplication (DGEMM) for matrices with dimensions MK x KN = MN
  - $N_{flops} = 2 \cdot MNK$

- Assuming compute boundness
  - $T_{CPU} = \frac{N_{Flops}}{P_{CPU}} = \frac{2MNK}{P_{CPU}} = \frac{2 \cdot 2048^3}{4096\ GFlop/s} = 4194\ \mu s$
  - $T_{8-GPU} = \frac{N_{Flops}}{P_{8-GPU}} = \frac{2MNK}{P_{8-GPU}} = \frac{2 \cdot 2048^3}{78\ TFlop/s} = 220\ \mu s$

Assignment 6 – Task 2

- Assuming compute boundness
  - $T_{CPU} = \frac{N_{Flops}}{P_{CPU}} = \frac{2MNK}{P_{CPU}} = \frac{2 \cdot 2048^3}{4096\ GFlop/s} = 4194\ \mu s$
  - $T_{8-GPU} = \frac{N_{Flops}}{P_{8-GPU}} = \frac{2MNK}{P_{8-GPU}} = \frac{2 \cdot 2048^3}{78\ TFlop/s} = 220\ \mu s$

- Transfer times
  - Two shared PCIe links per node, each with 25GB/s per direction
  - Host-to-device: 2 input matrices, device-to-host: one result matrix
  - $T_{PCIe} = \frac{3 \cdot V_{matrix}}{2 \cdot B_{PCIe}} = \frac{3 \cdot M^2 \cdot 8\ B}{2 \cdot B_{PCIe}} = \frac{3 \cdot 2048^2 \cdot 8B}{2 \cdot 25 \frac{GB}{s}} = 2013\ \mu s$

Assignment 6 – Task 2

- Putting it together
  - $T_{CPU} = \frac{N_{Flops}}{P_{CPU}} = \frac{2MNK}{P_{CPU}} = \frac{2 \cdot 2048^3}{4096\ GFlop/s} = 4194\ \mu s$
  - $T_{8-GPU} = \frac{N_{Flops}}{P_{8-GPU}} = \frac{2MNK}{P_{8-GPU}} = \frac{2 \cdot 2048^3}{78\ TFlop/s} = 220\ \mu s$
  - $T_{PCIe} = \frac{3 \cdot V_{matrix}}{2 \cdot B_{PCIe}} = \frac{3 \cdot M^2 \cdot 8\ B}{2 \cdot B_{PCIe}} = \frac{3 \cdot 2048^2 \cdot 8B}{2 \cdot 25 \frac{GB}{s}} = 2013\ \mu s$
  - $S = \frac{T_{CPU}}{T_{8-GPU} + T_{PCIe}} = \frac{4194\ \mu s}{220\ \mu s + 2013\ \mu s} = 1.88$

Assignment 6 – Task 3

- For one Frontier node, FP64
  - $P_{CPU} = 64\ cores \cdot 2\frac{instruction}{cycle} \cdot 2\frac{FLop}{FMA} \cdot 4\frac{FMA}{instruction} \cdot 2\ Ghz = 2048\frac{GFlop}{s}$
  - $P_{GPU} = 110\ CU \cdot 1\frac{instruction}{cycle} \cdot 2\frac{FLop}{FMA} \cdot 64\frac{FMA}{CU} \cdot 1.7\ Ghz\ = 24\frac{TFlop}{s}$
  - $P_{8-GPU} = 8 \cdot P_{GPU} = 192\frac{TFlop}{s}$

Assignment 6 – Task 4

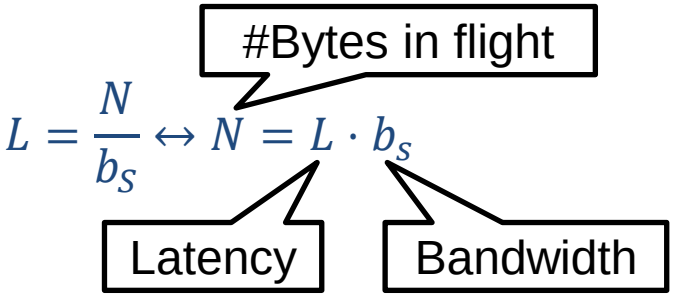
- DGEMM on Frontier
  - $T_{CPU} = \frac{N_{Flops}}{P_{CPU}} = \frac{2MNK}{P_{CPU}} = \frac{2 \cdot 2048^3}{2048\ GFlop/s} = 8389\ \mu s$
  - $T_{8-GPU} = \frac{N_{Flops}}{P_{8-GPU}} = \frac{2MNK}{P_{8-GPU}} = \frac{2 \cdot 2048^3}{192\ TFlop/s} = 89.5\ \mu s$
  - $T_{Link} = \frac{3 \cdot V_{matrix}}{8 \cdot B_{Link}} = \frac{3 \cdot M^2 \cdot 8\ B}{8 \cdot B_{Link}} = \frac{3 \cdot 2048^2 \cdot 8B}{8 \cdot 36 \frac{GB}{s}} = 350\ \mu s$
- Speed-up CPU vs GPU
  - $S = \frac{T_{CPU}}{T_{8-GPU} + T_{Link}} = \frac{8389\ \mu s}{89.5\ \mu s + 350\ \mu s} = 19$

Assignment 6 – Task 4

- Speed-up Alex vs Frontier
  - $S = \frac{T_{Alex}}{T_{Frontier}} = \frac{2233\ \mu s}{440\ \mu s} = 5$

Assignment 6 – Task 5

- Little’s Law



Assignment 6 – Task 5

- Kernel code (slightly modified)

```
__global__ void kernel(double* A, double* B, double* C, int size) {
  int x = threadIdx.x + blockDim.x * blockIdx.x;
  int y = threadIdx.y + blockDim.y * blockIdx.y;

  if (x >= size || y >= size) return;

  C[x + y * size] += A[x + y * size] * B[y];
}
```

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

Assignment 6 – Task 5

- Little's Law
  - $L = \frac{N}{b_S} \leftrightarrow N = L \cdot b_S$
  - $L_{seconds} = \frac{L_{cycles}}{F}$
  - $N_{bytes} = N_{threads} \cdot V_{thread}$
  - $V_{thread} = 3 \cdot 8\ B = 24\ B$
- Put together
  - $N_{threads} = \frac{670\ cycles}{1.41\ GHz} \cdot \frac{2039\ \frac{GB}{s}}{24\ \frac{B}{thread}} = 40370\ threads$

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

Assignment 6 – Task 5

- Kernel code (slightly modified)

```
__global__ void kernel(double* A, double* B, double* C, int size) {
  int x = threadIdx.x + blockDim.x * blockIdx.x;
  int y = threadIdx.y + blockDim.y * blockIdx.y;

  if (x >= size || y >= size) return;

  C[x + y * size] += A[x + y * size] * B[y];
}
```

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

Assignment 6 – Task 6

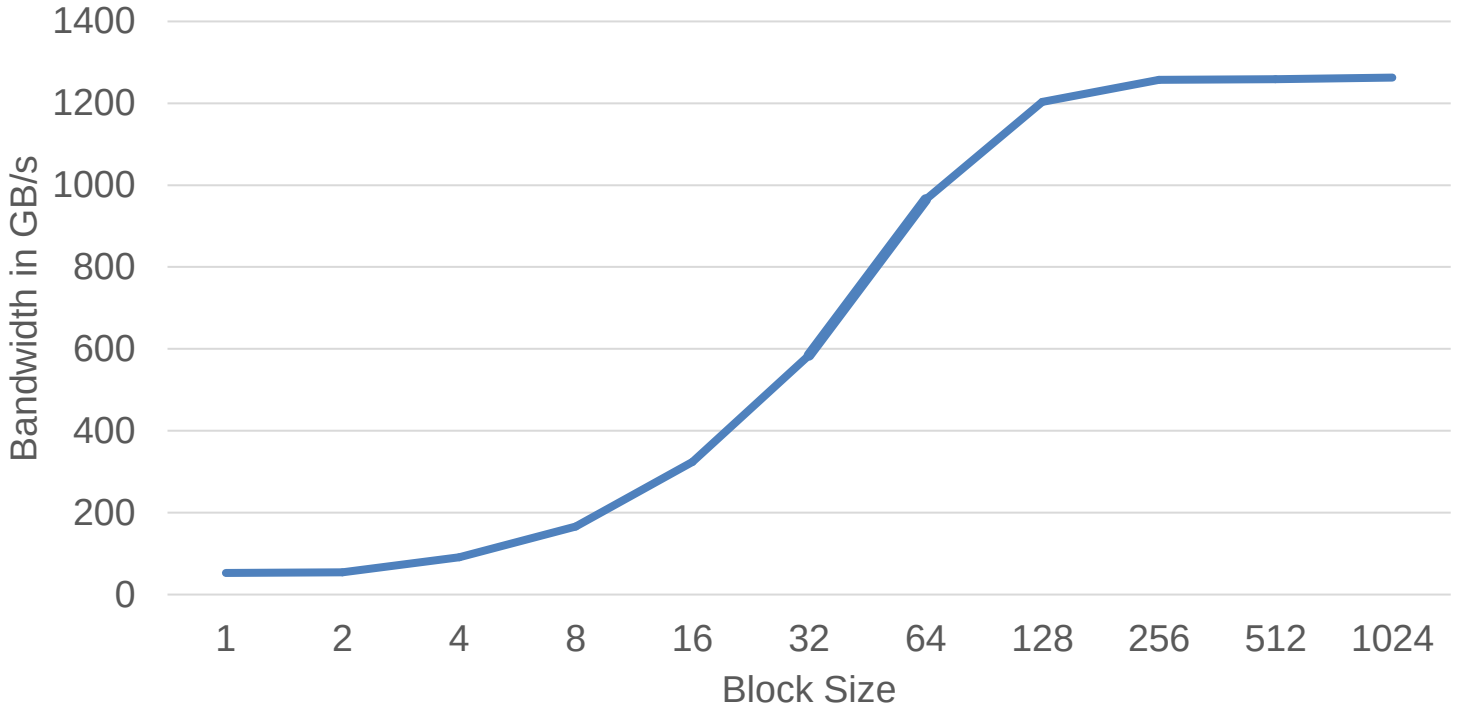
- Kernel code for `a[i] = a[i] + 1.0`

```
__global__ void kernel(double* a, int size) {
  int x = threadIdx.x + blockDim.x * blockIdx.x;
  int stride = gridDim.x * blockDim.x;

  for ( ; x <= size; x += stride)
    a[x] += 1;
}
```

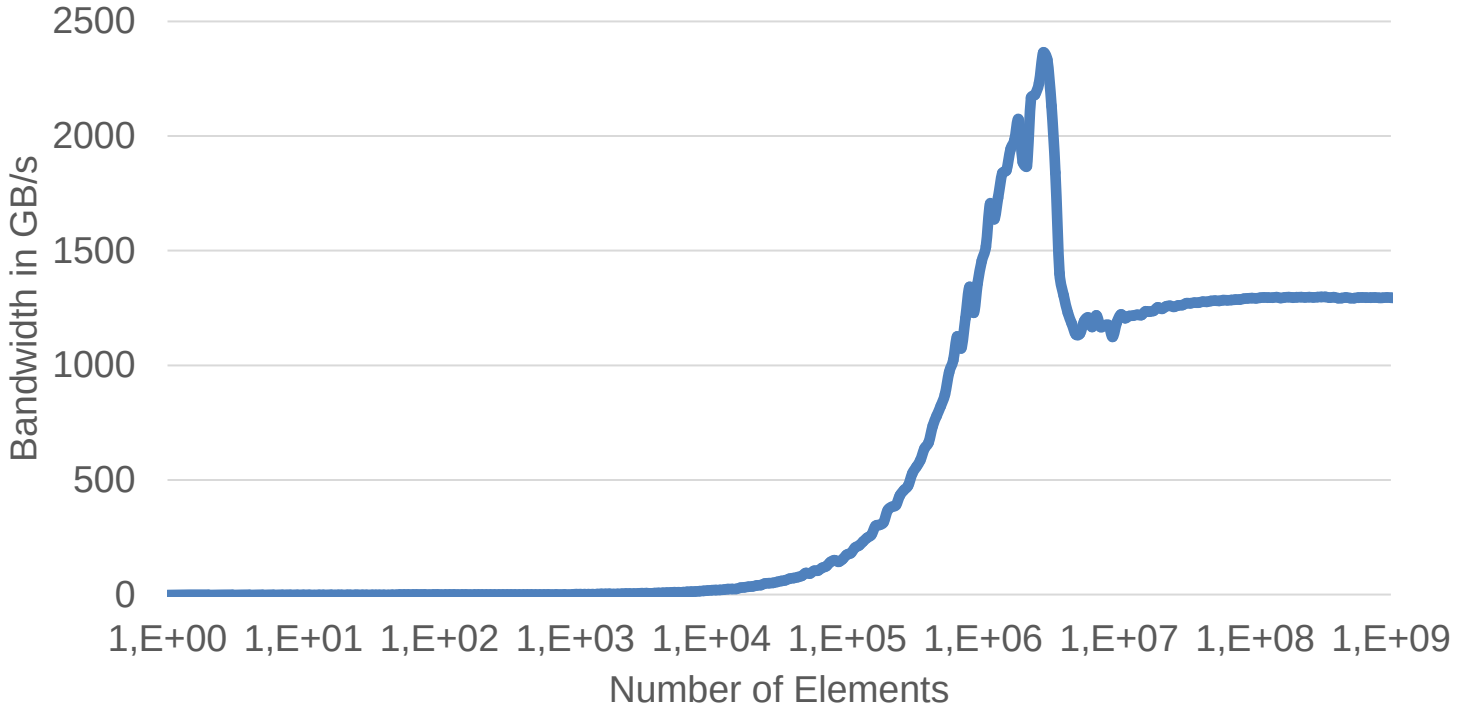
Assignment 6 – Task 6

- Block size investigation



Assignment 6 – Task 6

- DRAM Bandwidth



Assignment 6 – Task 6

- PCIe Bandwidth