Heidelberg University
Institute of Computer Engineering
Computer Engineering Group

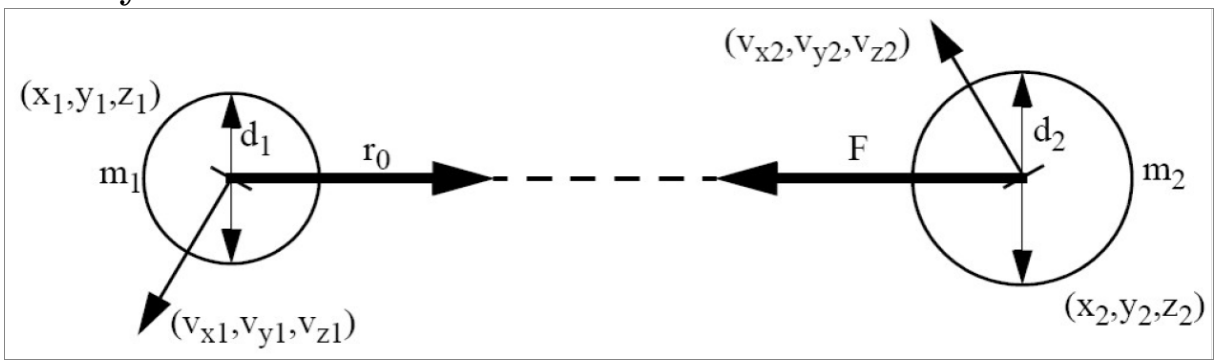
Holger Fröning
Bálint Soproni

GPU Computing - Architecture and Programming
Winter Term 2023/2024

Exercise 7

- **Hand in via Moodle until Monday, Jan 08, 2024, 09:00**
- **Include all names on the top sheet. Hand in a single PDF.**
- **Please compress your results into a single archive (`.zip` or `.tar.gz`).**
- **Please employ the following naming convention `hpc<XX>_ex<N>.{zip,tar.gz}`, where `XX` is your group number, and `N` is the number of the current exercise.**
- **A maximum of four students is allowed to collaborate on the exercises.**
- **In case an exercise requires programming:**
  - **include clean and documented code**
  - **include a Makefile for compiling**

n-Body Problem



Each object has the following data:

- Position p = (x,y,z) [m]
- Velocity v = (vx,vy,vz) [m/s]
- Mass m [kg]

Note that calculations are performed in 3D space. These are the most important formulas:

$$Force: F = -\gamma * \frac{m_1 \cdot m_2}{r^2} \cdot \overrightarrow{r_0}[N] \qquad \gamma = 6.673 \cdot 10^-11 \frac{Nm^2}{kg^2}$$

$$Acceleration: \overrightarrow{a} = \frac{\overrightarrow{F}}{m}[\frac{N}{kg}]$$

$$Velocity: \overrightarrow{v} = \overrightarrow{a} \cdot \Delta t[\frac{m}{s}]$$

$$Distance: \overrightarrow{s} = \overrightarrow{v} \cdot \Delta t[m]$$

With r the distance between particle 1 and 2. And $\overrightarrow{r_0}$ the unit vector between particle 1 and 2, denoting the relative direction of particle 2 to particle 1.
Ensure the following:

1. The program accepts the number of objects as command line parameter.
2. Use single precision floating points for all object data.

Holger Fröning
Bálint Soproni

3. Define the length of a time step with the constant TIMESTEP (Δt).
4. Initialize positions and masses with random values, velocity with zero.
5. Perform a reasonable amount of time steps (iterations), for instance 100. Each iteration updates the velocity and position of all objects.

7.1 Naïve GPU implementation

Implement a naïve GPU version that primarily serves for testing the compute kernel. Use AOS (arrays of structures) with two packed values, as shown in the template. Do not use shared memory yet. We will later report speedups based on this baseline performance. Do not include data movement over the PCIe interface in your measurements.

- Report performance numbers for an increasing body count and discuss!

(15 points)

7.2 Optimized GPU implementation

Based on the previous implementation, perform optimizations to maximize performance for this n-body computation. In particular:

- Consider if you prefer packed SOA (structure of arrays) or AOS (arrays of structures) data structures
- Make use of tiling to optimize for shared memory use
- Optional: investigate if loop unrolling is beneficial

Report performance numbers and discuss your optimizations. For which reason(s) is a certain optimization increasing performance?
(25 points)

7.3 n-Body GPU computations and streaming

Start with the optimized GPU implementation of the n-Body problem from the last task. In this exercise we will implement a streaming version of this code, meaning that the GPU is artificially limited in terms of device memory and has to spill to host memory. In this sense, the host memory is acting as swap space. As such spilling has to be manually controlled by the user, your task is to implement a streamed version of the optimized n-Body code. Use the guidelines from the lecture, and use different streams to maximize the utilization of the GPU.
As the GPU memory capacity would result in too many bodies for reasonable run times, we artificially limit the amount of device memory to **4 MB**. Ensure in your code that you are not using more device memory! Opposed to this, run your code with a body count of **512 k**. As each body consumes about 7x 32bit = 28B, better say 32B, you can see that device memory is not sufficient to completely store the problem.
Discuss the following:

- Compare your optimized streaming implementation to the non-streamed implementation (using only the default stream #0) and interpret!
- Similarly, compare against the optimized kernel from last exercise (without PCIe time) and interpret!
- Vary the segment size for the streams. Discuss the effects you are seeing!
- How many streams are required for an optimized implementation? Discuss why!

(26 points)

Holger Fröning
Bálint Soproni

7.4 Bonus: Simple convolution performance estimation

In this exercise you will make some estimates about the expected performance of a simplified convolution calculation. Assume a simplified convolution computation on a 2D square grid G with the dimensions of $n \times n$. For each element of this grid a convolution window W of $m \times m$ is applied. For the purposes of this exercise we disregard the boundary condition and assume that the convolution window is applied n times in each direction, such that:

$$C_{ij} = \sum_{k=1}^{m} \sum_{l=1}^{m} N_{i+k,j+l} \cdot W_{kl}$$

Where C is the resulting $n \times n$ grid. Furthermore assume that all values are single precision floating point values.

- Calculate the compute intensity for the above computation for $m = 3, 5, 7, 11$. Assume a perfect caching, therefore only consider unique memory accesses.
- Look up the whitepaper of the RTX2080 GPU and based on the relevant information provide an estimate for the maximum performance for the above values of m.
- For what values of m will be the problem compute bound and memory bound. Explain your reasoning.
- What is the minimum cache bandwidth required in order to reach peak floating point performance of the GPU? Assume that the windows is stored in registers, therefore the cache only need to serve the accesses to the grid.

(Bonus 20 points)

Holger Fröning
Bálint Soproni

7.5 Willingness to present

Please declare whether you are willing to present any of the previous exercises.
The declaration can be made on a per-exercise basis. Each declaration corresponds to 50% of the exercise points. You can only declare your willingness to present exercises for which you also hand in a solution. If no willingness to present is declared you may still be required to present an exercise for which your group has handed in a solution. This may happen if as example nobody else has declared their willingness to present.

- Naïve GPU implementation
- Optimized GPU implementation
- n-Body GPU computations and streaming
- Bonus: Simple convolution performance estimation

(33 points + 10 bonus)

Total: 99 points + 30 bonus