Heidelberg University
Institute of Computer Engineering
Computer Engineering Group

Holger Fröning
Bálint Soproni

Introduction to High-Performance and Distributed Computing
Winter Term 2023/2024

Exercise 6

- **Hand in via Moodle until 23:59 on Sunday 11 December, 2022**
- **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 three students is allowed to collaborate on the exercises.**
- **In case an exercise requires programming:**
  - **include clean and documented code**
  - **include a Makefile for compiling**

6.1 Reading

Read the following two papers and provide reviews as explained in the first lecture (see slides):

- Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM 52, 4* (April 2009), 65-76.
- David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. 1993. LogP: towards a realistic model of parallel computation. *SIGPLAN Not. 28, 7* (July 1993), 1-12.

(20 points)

6.2 n-Body Problem – Partitioning/Communication Design

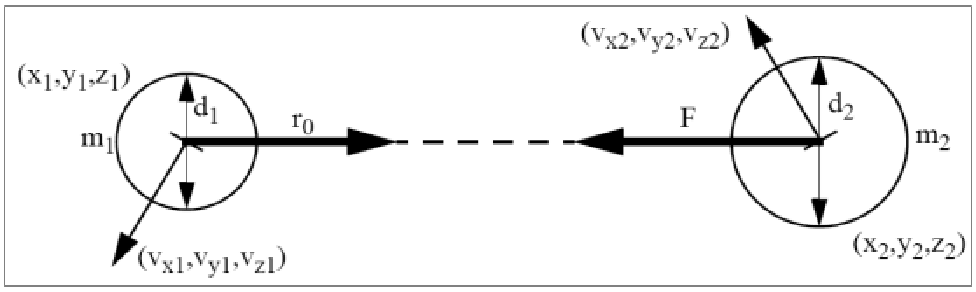

Figure 1: Two elements and acting forces

$$Position : p = f(x,y,z) = [m] \tag{1}$$

$$Velocity : v = f(v_x, v_y, v_z) = [\frac{m}{s}] \tag{2}$$

$$Mass : m = [kg] \tag{3}$$

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

$$Force : F = -\gamma \cdot \frac{m_1 \cdot m_2}{r^2} \cdot \overrightarrow{r_0} = [N] \tag{4}$$

$$\gamma = 6.673 \cdot 10^{-11} \frac{Nm^2}{kg^2} \tag{5}$$

$$Acceleration : \overrightarrow{a} = \frac{\overrightarrow{F}}{m} = [\frac{N}{kg}] \quad (6)$$

$$Velocity : \overrightarrow{v} = \overrightarrow{a} \cdot \Delta t = [\frac{m}{s}] \quad (7)$$

$$Distance : \overrightarrow{s} = \overrightarrow{v} \cdot \Delta t = [m] \quad (8)$$

Consider the following, **but do not implement the MPI program yet, this is part of the next exercise**:

- The program accepts the number of objects as command line parameter.
- Use double precision floating points for all object data.
- Define the length of a time step with the constant *DELTA_T*.
- Initialize positions and masses with random values, velocity with zero.
- Perform a reasonable amount of time steps (iterations), for instance 100. Each iteration, update the velocity and position of all objects.

Design a suitable partitioning and communication model which ensures the following:

- Equally distribute the objects among the processes. Due to the all-to-all communication scheme the mapping is not crucial for this exercise.
- When performing one iteration, ensure that you optimize overall performance by leveraging overlap as much as possible. Thus, try to send out object data in a non-blocking way while performing calculations on objects whose associated data is already available.
- Try to avoid collective calls, as they are blocking and do not allow any overlap at all. However, you can use collective calls for initialization and finalization.
- It might be helpful to arrange the processes in a ring structure, so that object data (a collection of a certain number of objects) is circularly shifted around. Each time it is located at a certain process, this process uses this object data to calculate the corresponding contribution to the overall force on the objects which this process is responsible for.
- Make yourself familiar with MPI derived data types, which might help simplifying communication struct-like datatypes. However, it is up to you if you want to use MPI derived data types or not.

Explain your partitioning and communication model using explanations and graphs. In particular address how you plan to maximize the overlap between computation and communication.

(20 points)

40 points