OpenACC/OpenMP
the Easy Ways to GPU Programming

Giovanni Tedeschi Prades

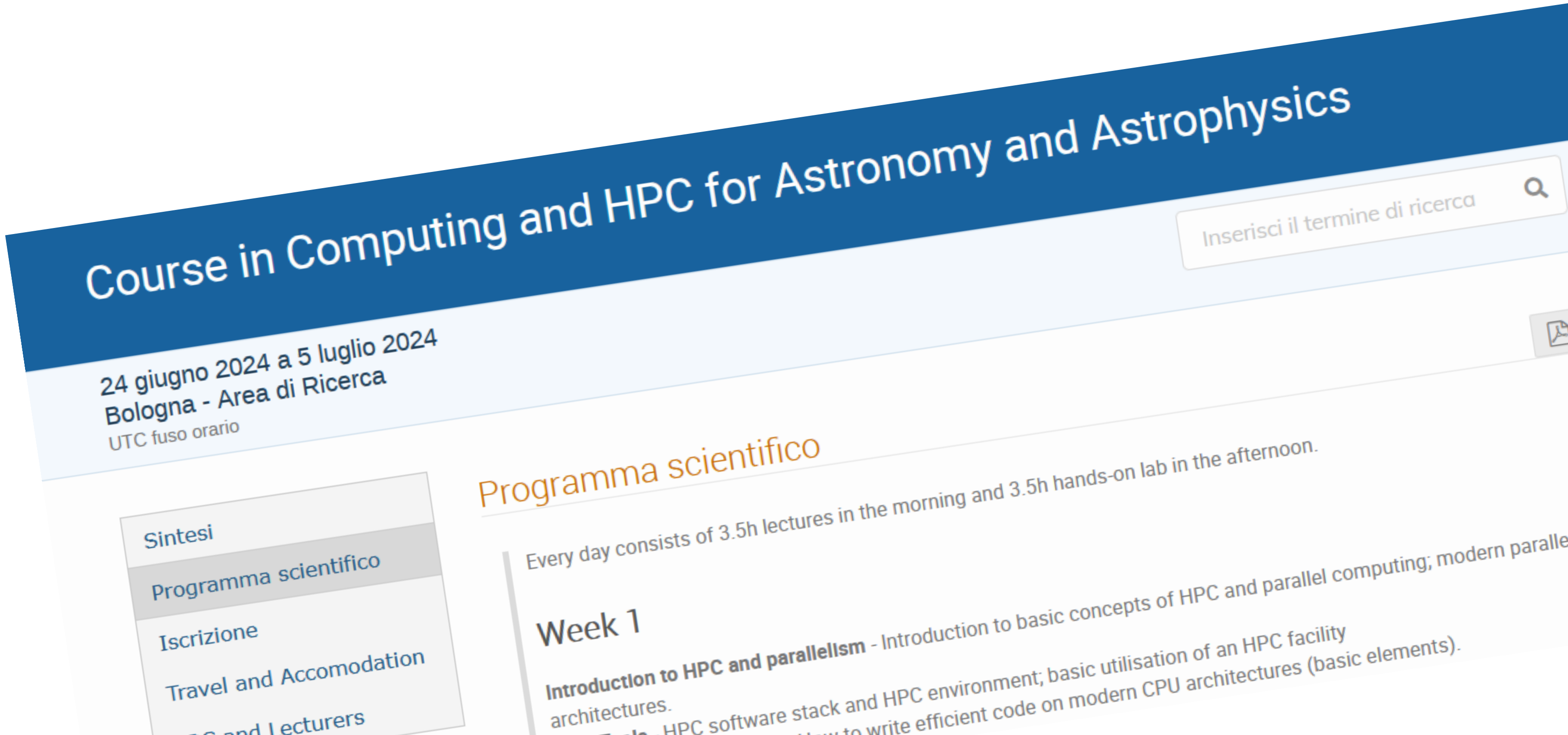
Code Coffee -25.11.2024

Outlook

- A basic overview of parallelization in CPUs and GPUs
- GPU resources at the Physics Cluster
- **First hands-on**: getting a GPU, open a parallel region, compile and run. You can code along if you want.
- The Mandelbrot set
- **Second hands-on**: parallelize a for loop
- Checking the performances

Disclaimers

- I am **not** an expert in GPU programming, I just attended a school
- Ask and interrupt, I'll do my best
- I'll be working in C



Course in Computing and HPC for Astronomy and Astrophysics

Inserisci il termine di ricerca

24 giugno 2024 a 5 luglio 2024
Bologna - Area di Ricerca
UTC fuso orario

Programma scientifico

Every day consists of 3.5h lectures in the morning and 3.5h hands-on lab in the afternoon.

Week 1

Introduction to HPC and parallelism - Introduction to basic concepts of HPC and parallel computing; modern paralle
architectures.
HPC software stack and HPC environment; basic utilisation of an HPC facility
how to write efficient code on modern CPU architectures (basic elements).

An incomplete map of parallelization

CPU

Shared-memory parallelization
OpenMP
Implemented in most compilers

An incomplete map of parallelization

CPU

Shared-memory parallelization
OpenMP
Implemented in most compilers

Distributed-memory parallelization
MPI (**M**essage **P**arsing **I**nterface)
Implemented in modules such as OpenMPI, MPICH...

An incomplete map of parallelization

CPU

Shared-memory parallelization
OpenMP
Implemented in most compilers

Distributed-memory parallelization
MPI (**M**essage **P**arsing **I**nterface)
Implemented in modules such as
OpenMPI, MPICH…

Not the same thing

An incomplete map of parallelization

CPU

Shared-memory parallelization
OpenMP
Implemented in most compilers

Distributed-memory parallelization
MPI (**M**essage **P**arsing **I**nterface)
Implemented in modules such as OpenMPI, MPICH…

GPU

CUDA-programming
A «programming model» to directly interact with GPUs
Available only for NVIDIA GPUs

An incomplete map of parallelization

CPU

Shared-memory parallelization
OpenMP
Implemented in most compilers

Distributed-memory parallelization
MPI (**M**essage **P**arsing **I**nterface)
Implemented in modules such as OpenMPI, MPICH…

GPU

CUDA-programming
A «programming model» to directly interact with GPUs
Available only for NVIDIA GPUs

OpenMP and **OpenACC**
Based on compiler directives which tell the compiler the sections of the code to run on the GPU

An incomplete map of parallelization

CPU

Shared-memory parallelization
OpenMP
Implemented in most compilers

Distributed-memory parallelization
MPI (**M**essage **P**arsing **I**nterface)
Implemented in modules such as OpenMPI, MPICH…

GPU

CUDA-programming
A «programming model» to directly interact with GPUs
Available only for NVIDIA GPUs

OpenMP and **OpenACC**
Based on compiler directives which tell the compiler the sections of the code to run on the GPU

Actually the same thing

An incomplete map of parallelization

CPU

Shared-memory parallelization
OpenMP
Implemented in most compilers

Distributed-memory parallelization
MPI (**M**essage **P**arsing **I**nterface)
Implemented in modules such as OpenMPI, MPICH…

GPU

CUDA-programming
A «programming model» to directly interact with GPUs
Available only for NVIDIA GPUs

OpenMP and **OpenACC**
Based on compiler directives which tell the compiler the sections of the code to run on the GPU

Kokkos (and friends)
High-level programming frameworks to make a code highly portable, including GPUs

An incomplete map of parallelization

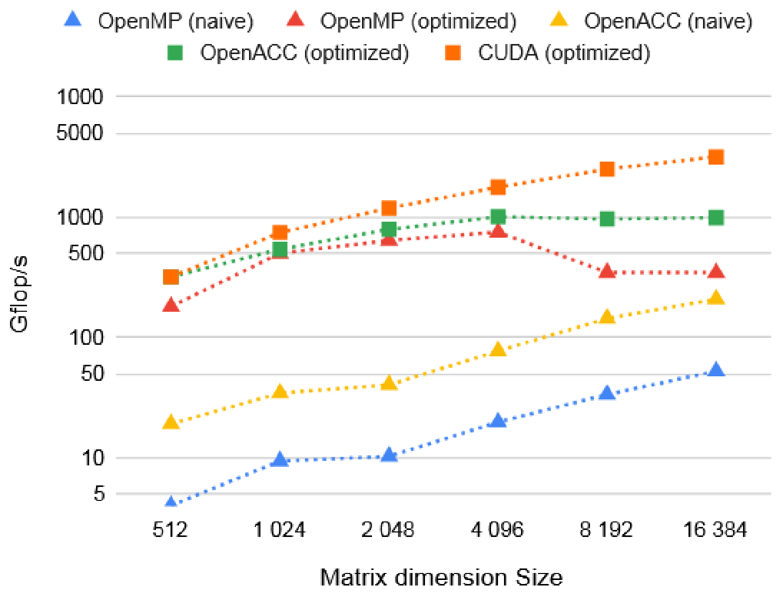
CUDA	**OpenMP/OpenACC**	**Kokkos**

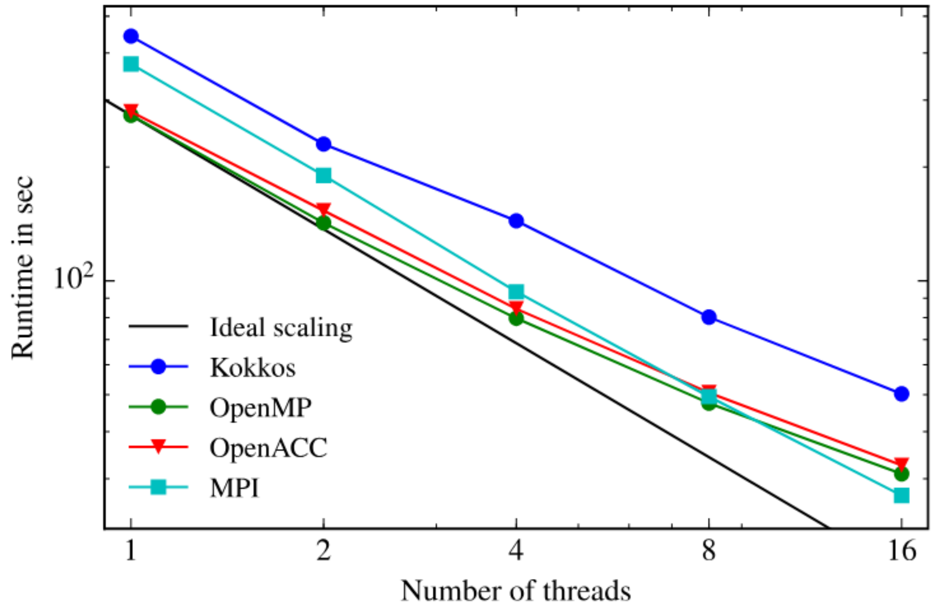
An incomplete map of parallelization

Take this with a grain of salt

CUDA **OpenMP/OpenACC** **Kokkos**

Performance

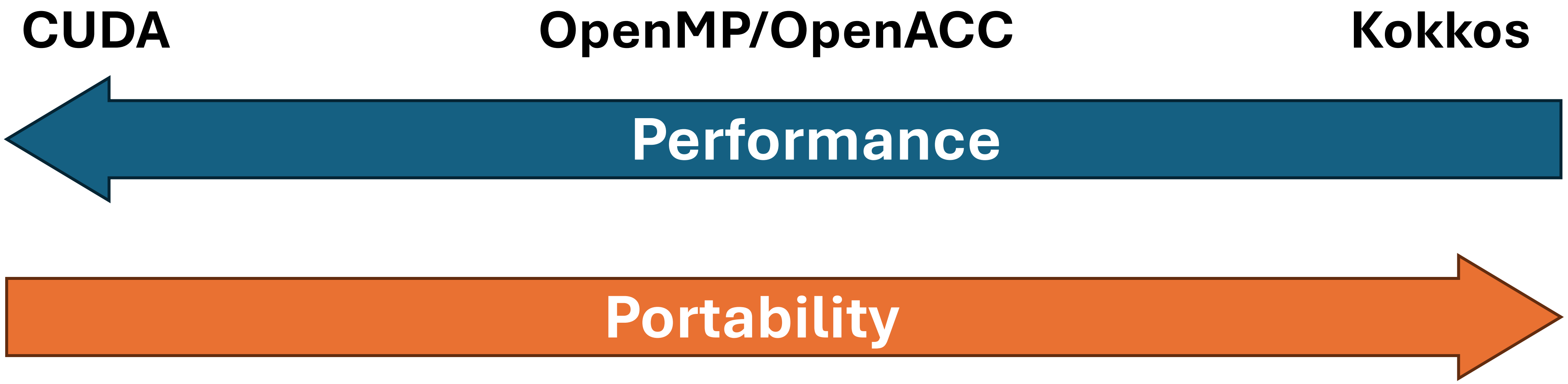




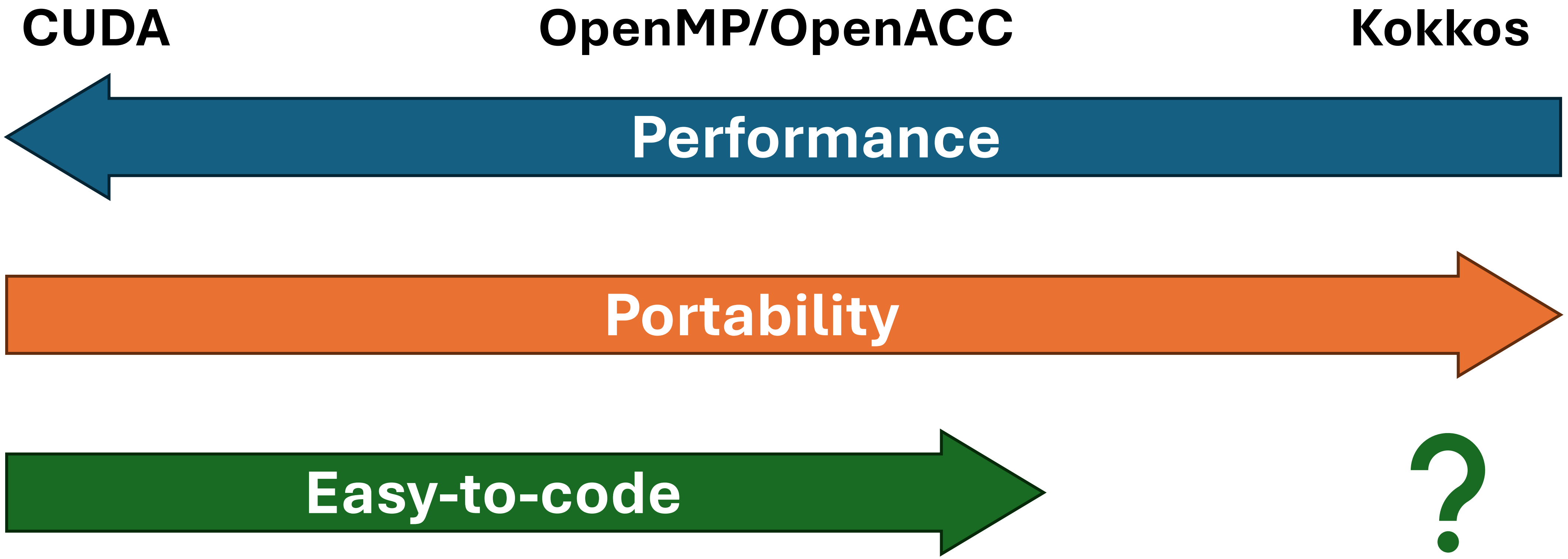
Eichstädt et al. (2020) https://doi.org/10.1016/j.cpc.2020.107245

Khalilov et al. (2021) https://doi.org/10.1088/1742-6596/1740/1/012056

An incomplete map of parallelization



An incomplete map of parallelization



GPUs at the Physics Cluster

- 121 total GPUs available at the cluster
    - 28x P2000
    - 4x P5000
    - 8x V100
    - 7x H100
    - 66x A40
    - 2x A100
    - 4x RTX2080Ti
    - 2x TitanXP
- All NVIDIA GPUs

GPUs at the Physics Cluster

- 121 total GPUs available at the cluster
  - 28x P2000
  - 4x P5000
  - 8x V100
  - 7x H100
  - 66x A40 ← 49 of which availble at the CIP nodes
  - 2x A100
  - 4x RTX2080Ti
  - 2x TitanXP

- All NVIDIA GPUs

GPUs at the Physics Cluster

- 121 total GPUs available at the cluster
  - 28x P2000
  - 4x P5000
  - 8x V100
  - 7x H100
  - 66x A40
  - 2x A100
  - 4x RTX2080Ti
  - 2x TitanXP

Not compatible with OpenMP

- All NVIDIA GPUs

To the terminal!

- **How to start an interactive job with a GPU**

Simple version: `intjob --gres=gpu:1`

To specify which GPU you want (in this case an A40, available only at the `inter` partition):

```
intjob --gres=gpu:a40:1 --partition=inter
```

- **Compilers and commands**

Compile with OpenACC: `nvc -acc test.c`

Compile with OpenMC: `nvc -mp=gpu test.c`

The nvc compiler is availble in the nvhpc module at the cluster

To the terminal!

- **Open a parallel region with OpenACC and OpenMP**

OpenACC: `#pragma acc parallel`

OpenMP: `#pragma omp target`

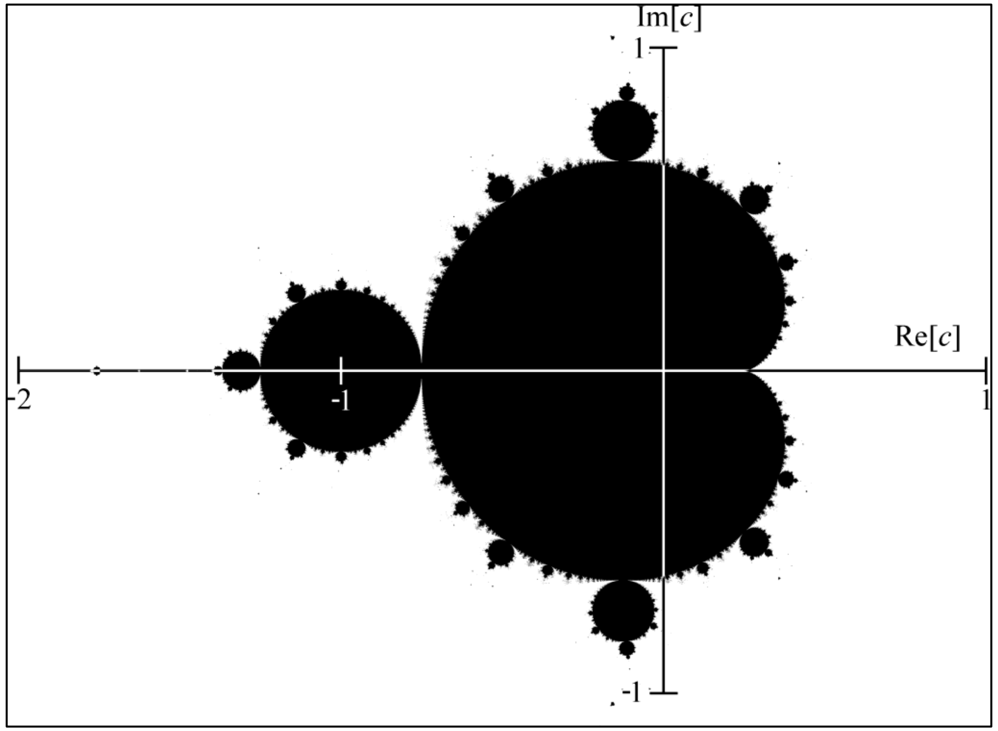
- **Compile, run and check you are actually using a GPU**

Use the command `nvidia-smi` to check the status of the GPU.

Useful to put it in a `watch -n 0.1 nvidia-smi` to see it changing live.

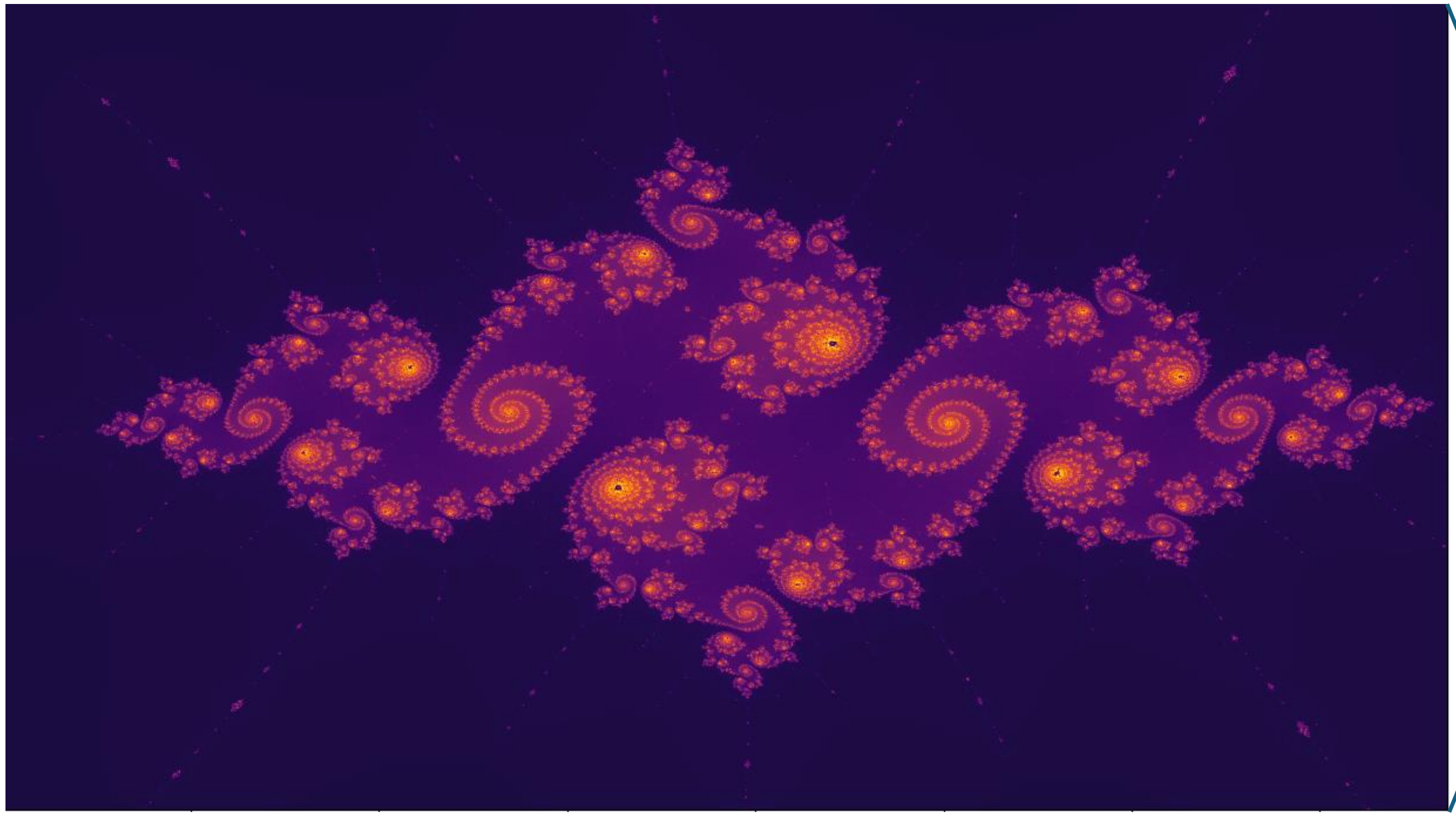
The Mandelbrot set

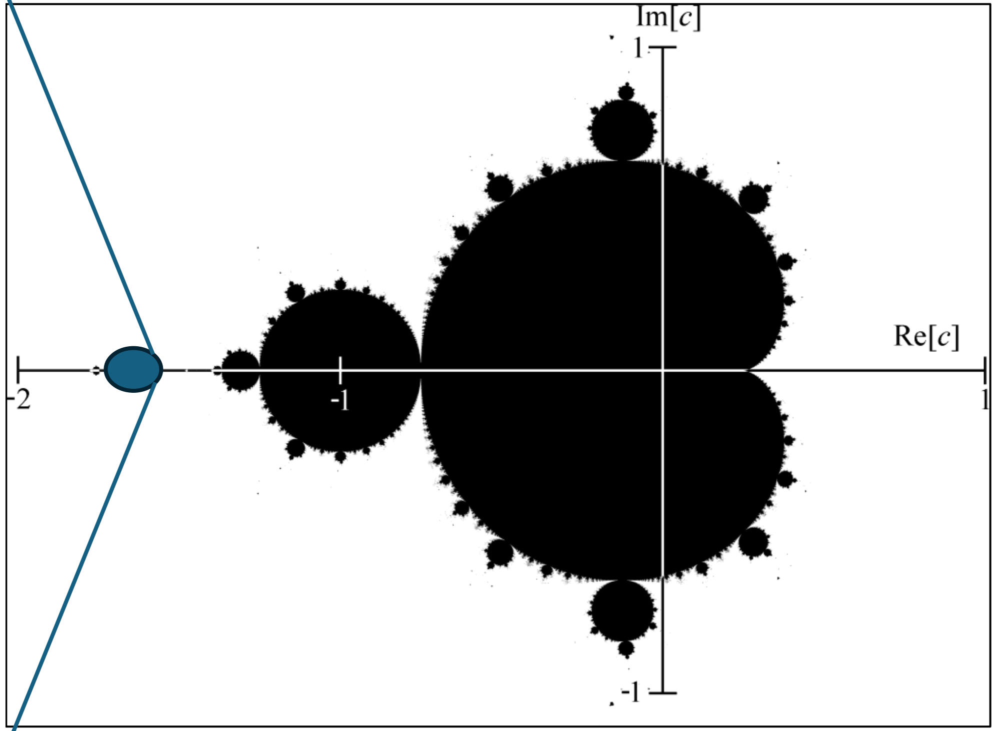
- Defined as the set of points z_0 in the complex plane for which the sequence $z_{n+1} = z_n^2 + z_0$ converges.
- Computationally demanding at the edges, where a large number of iterations are required
- Embarassingly parallel: each point can be evaluated indipendently to the others
- Double for loop to check all the points inside a certain domain



The Mandelbrot set

- We will focus somewhere here





To the terminal!

- **How to parallelize for loops with GPUs**

OpenACC: `#pragma acc parallel loop`

OpenMP: `#pragma omp target parallel for`

- **The basics of data exchange**

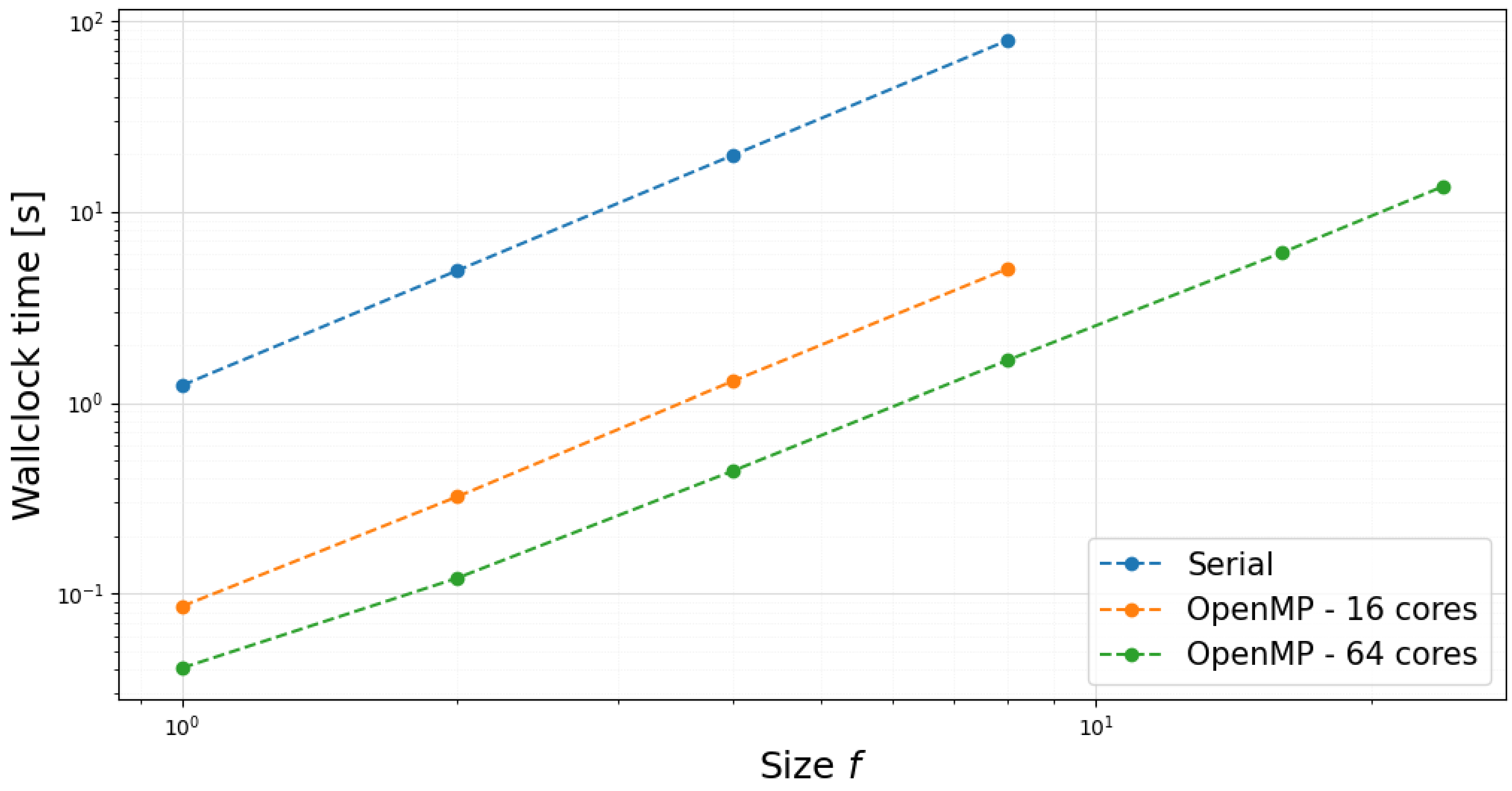
OpenACC: `copyin(<var_name>) copyout(<var_name>)`

OpenMP: `map(to:<var_name>) map(from:<var_name>)`

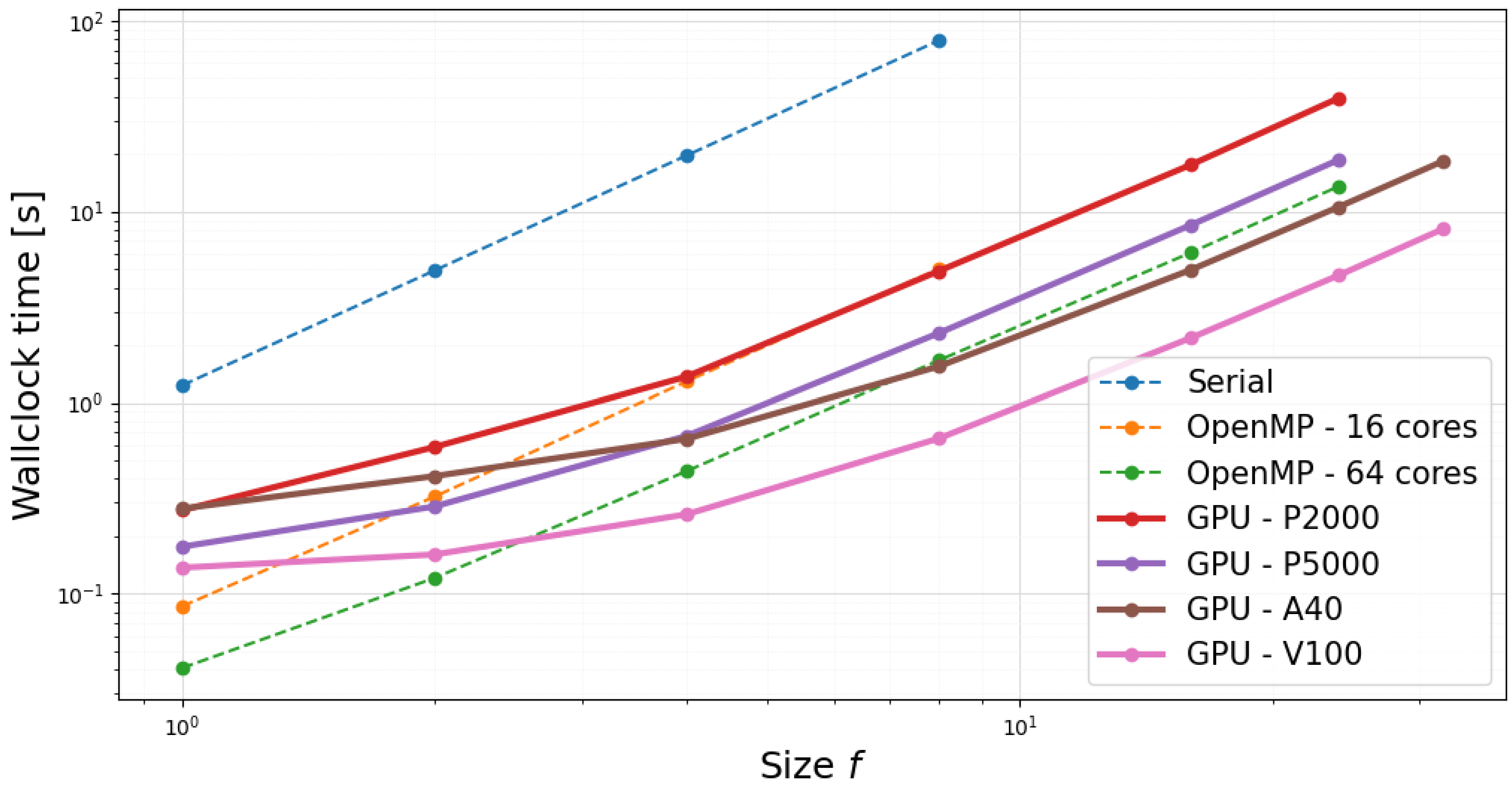
Needed for OpenMP, otherwise it will crash!

- **What kind of speedup is possible?**

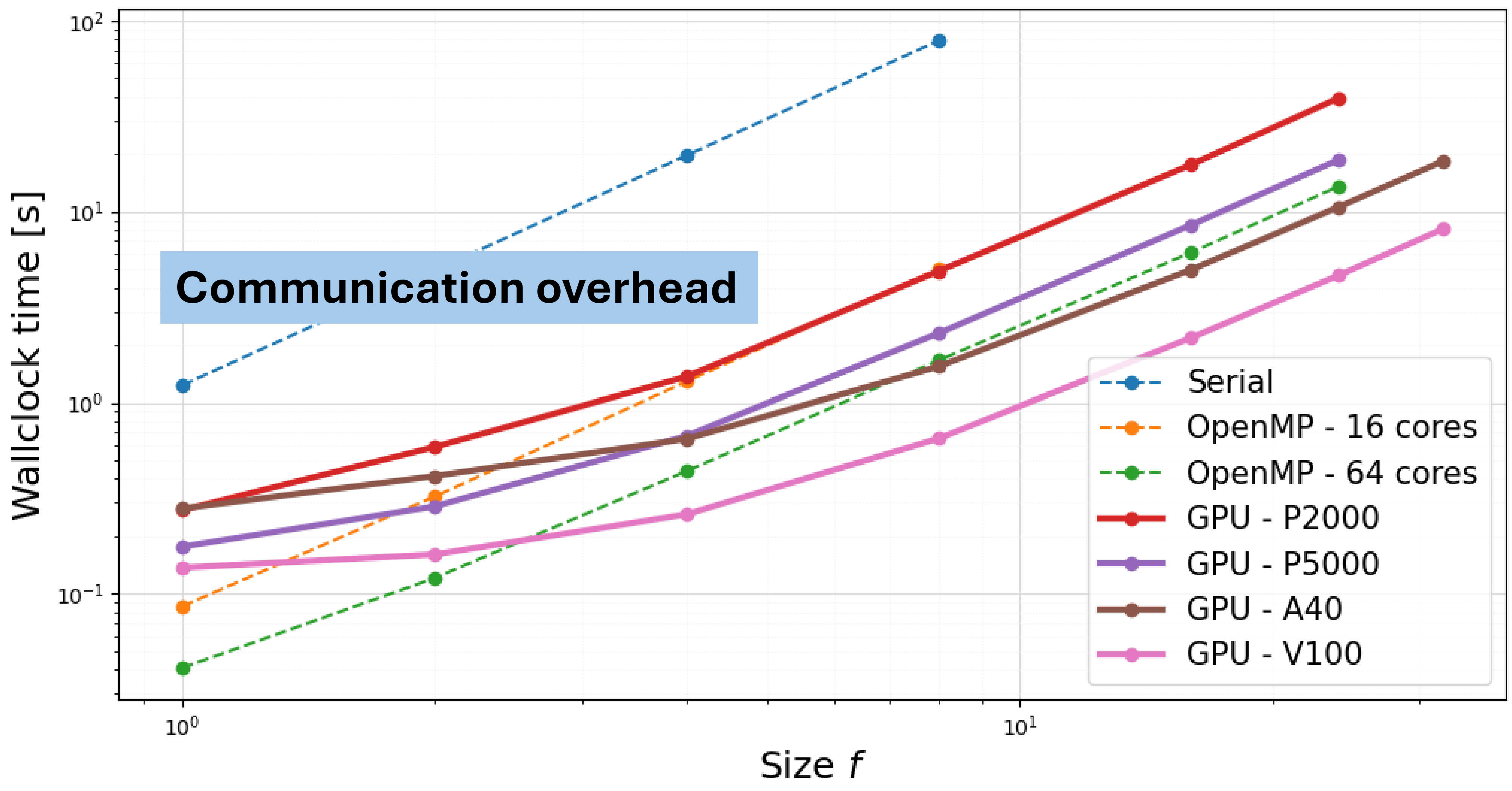
Performance study



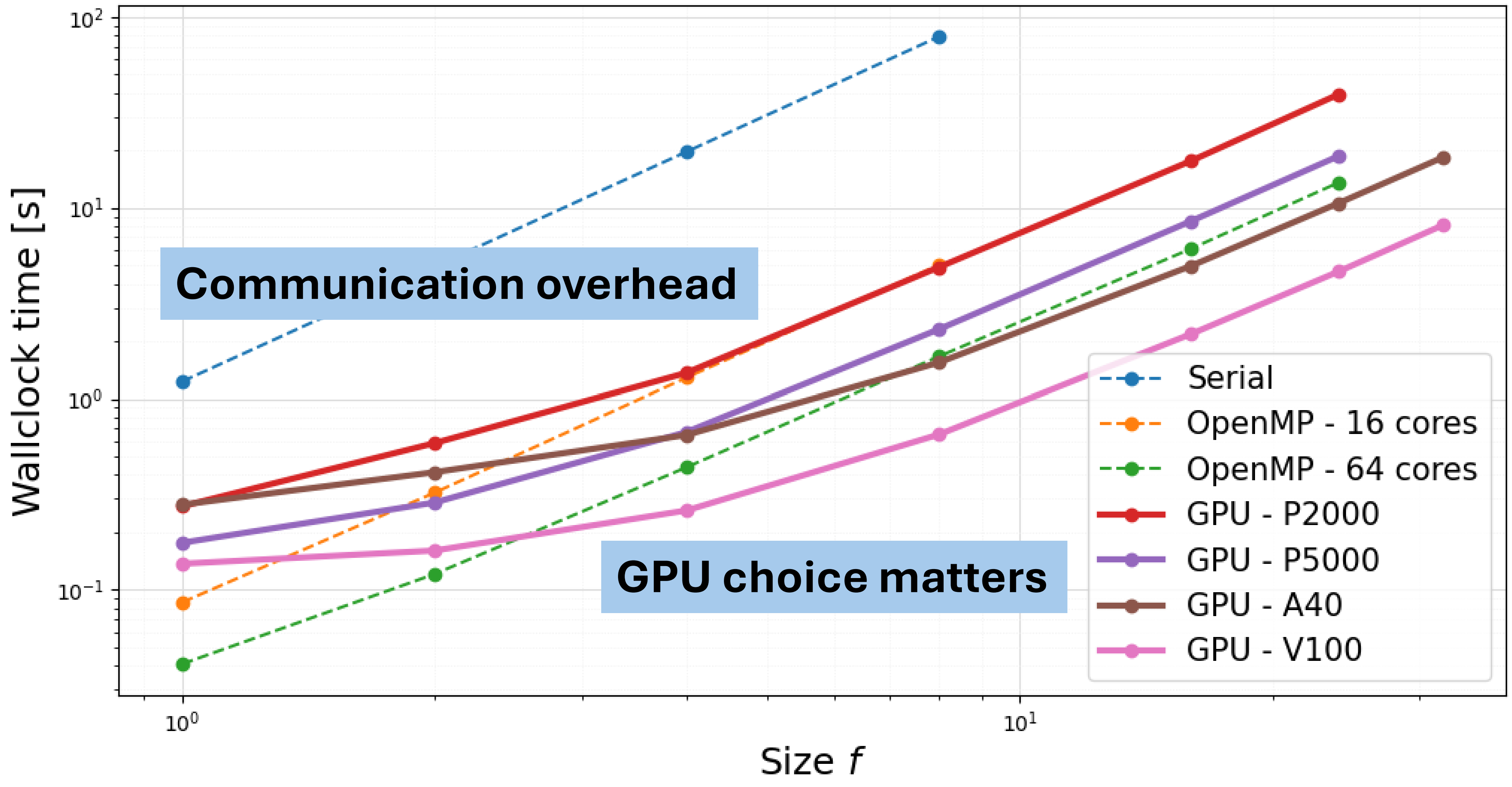
Performance study



Performance study



Performance study



Communication overhead

GPU choice matters

#pragma summary

OpenACC

```
#pragma acc parallel

#pragma acc parallel loop
copyin(<var_name>)
copyout(<var_name>)
```

OpenMP

```
#pragma omp target


#pragma omp target parallel for
map(to:<var_name>)
map(from:<var_name>)
map(tofrom:<var_name>)
```

Conclusions

- OpenMP and OpenACC offer an easy way into GPU programming
- Most of the heavy work is done by the compiler
- Compilers matter: `nvc` is the best choice for NVIDIA GPUs
- Many `#pragma` directives offered, syntax differs between the two
- Mind the communication overhead
- The choice of which GPU to use matters

- A useful resource: OpenACC programming and Best Practices Guide