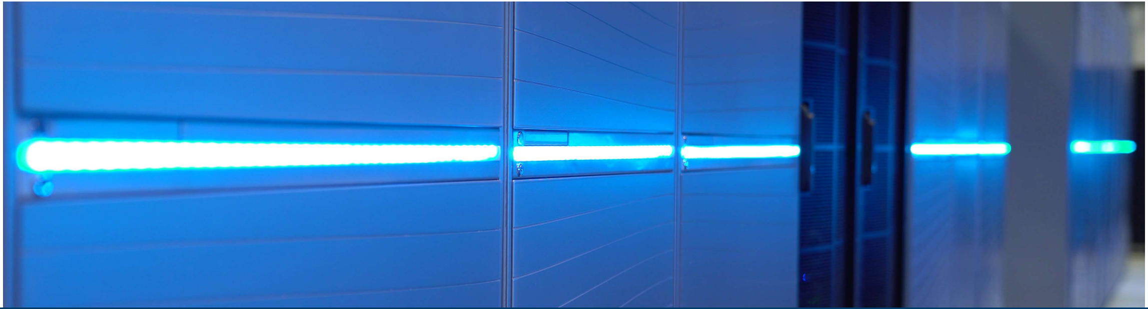


INTRODUCTION TO PARALLEL PROGRAMMING WITH MPI AND OPENMP

August 12-16 2024 | Junxian Chew, Michael Knobloch, Ilya Zhukov, Jolanta Zjupa | Jülich Supercomputing Centre

Member of the Helmholtz Association

JÜLICH
Forschungszentrum

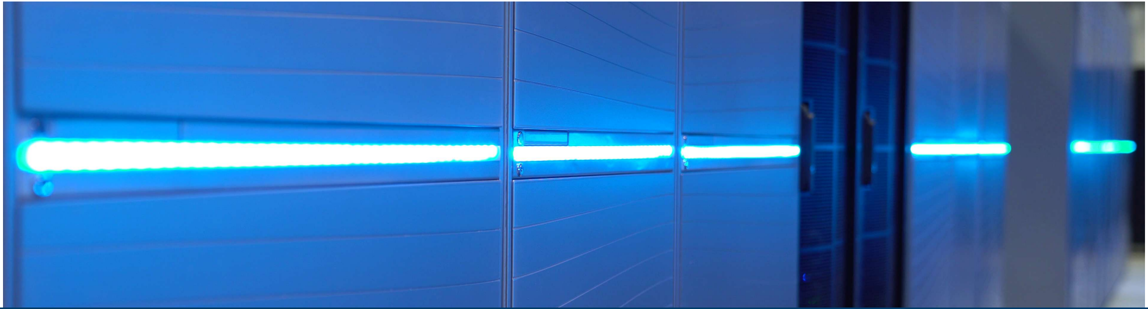
INTRODUCTION ROUND

Tell us about yourself!

1. Name
2. Education
3. Place of work
4. Motivation for coming
5. Prior knowledge
6. Programming language

TIMETABLE

	Day 1	Day 2	Day 3	Day 4	(Day 5)
09:00 10:30	Welcome and Setup	Introduction to MPI	Nonblocking Collective Communication	I/O	Hands-On Tutorial
☕					
11:00 12:30	Fundamentals of Parallel Computing	Blocking P2P Communication	Communicators	I/O	Hands-On Tutorial
🍴					
13:30 14:30	Introduction to OpenMP	Nonblocking P2P Communication	Derived Datatypes	Tools: MUST	Hands-On Tutorial
☕					
15:00 16:30	OpenMP	Blocking Collective Communication	Derived Datatypes	Hybrid programming	Hands-On Tutorial



Part I: Fundamentals of Parallel Computing

PARALLEL COMPUTING

Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. (Wikipedia[1])

[1] Wikipedia. Parallel computing — Wikipedia, The Free Encyclopedia. 2017. URL: https://en.wikipedia.org/w/index.php?title=Parallel_computing&oldid=787466585 (visited on 06/28/2017).

DEFINITIONS

CPU

Central processing unit

Core

Single processing unit within the CPU that can execute instructions.

Process

A sequentially executed instance of a computer program.

Thread

Smallest sequence of programmed instructions or an execution entity that can be managed independently by a scheduler (which is typically a part of the operating system).

Hyperthreading/Simultaneous Multithreading (SMT)

Presence of a/multiple virtual (logical) core/s per physical core which share workload by executing instructions in parallel, when possible.

QUIZ

How many CPU cores does a stationary personal computer or laptop have? (order of magnitude)

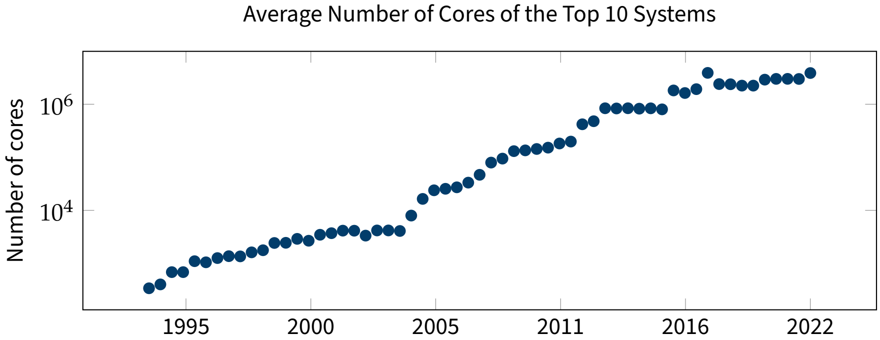
1. one
2. ten
3. one hundred
4. one thousand

QUIZ

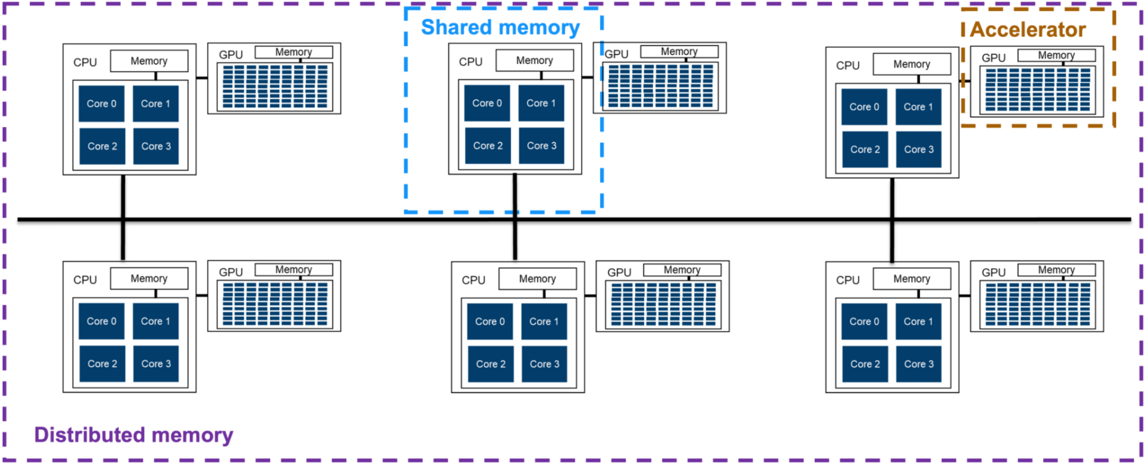
How many CPU cores does a top ten supercomputer have? (order of magnitude)

1. ten thousand
2. one hundred thousand
3. one million
4. ten million

PARALLELISM IN THE TOP 500 LIST



MEMORY DOMAINS



MEMORY DOMAINS

Node

- ‘Individual computer’ that is the fundamental building block of an HPC cluster. Typically a **multiprocessor**: computer system with two or more CPUs sharing the same memory.
- **Non-uniform memory access (NUMA)**: Shared memory architecture used in multiprocessing, where the memory access time depends on the memory location relative to the processor (CPU).
- **Uniform memory access (UMA)**: Shared memory architecture used in multiprocessing, where the memory access time is *independent* of which processor makes the request or where in memory the data is located.

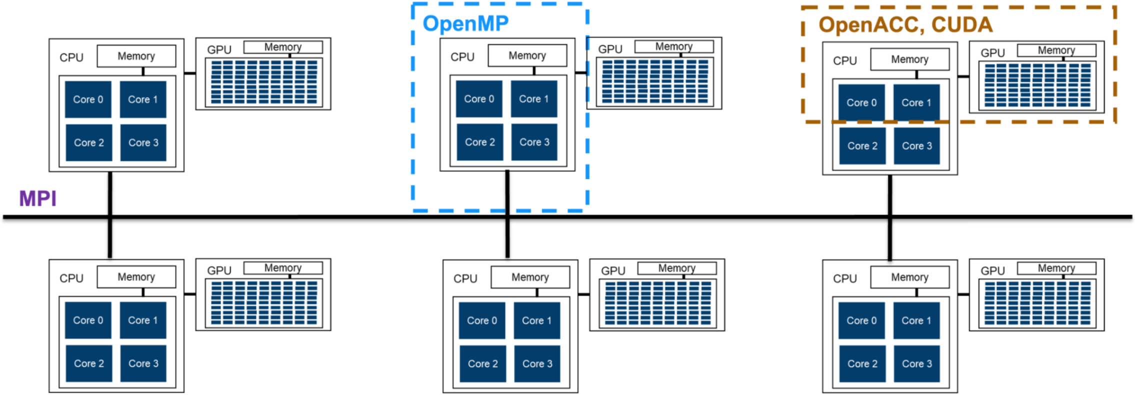
Shared Memory

- All memory is directly accessible by the parallel computational units
- Single address space (programmer might have to synchronize access)

Distributed Memory

- Memory is partitioned into parts which are private to the different computational units
- ‘Remote’ parts of memory are accessed via an interconnect network

PARALLELISATION PARADIGMS



QUIZ

What is ‘program state’?

1. The memory address of the CPU instruction that is currently being executed
2. Whether a program executed successfully or not and which error it encountered (e.g. segmentation fault)
3. For a specific execution of a program the values of all variables used by the program at a single point in time

DISTRIBUTED STATE & MESSAGE PASSING

Distributed State

Program state is partitioned into parts which are private to the different processes.

Message Passing

- Parts of program state are transferred from one process to another for coordination
- Primitive operations are active send and active receive

MPI

- Implements a form of Distributed State and Message Passing
- (But also Shared State and Synchronization)

SHARED STATE & SYNCHRONIZATION

Shared State

The whole program state is directly accessible by the parallel threads.

Synchronization

- Threads can manipulate shared state using common loads and stores
- Establish agreement about progress of execution using synchronization primitives, e.g. barriers, critical sections, …

OpenMP

- Implements Shared State and Synchronization
- (But also higher level constructs)