Parallel and Distributed Computing
Lecture # 2

By,

Dr. Ali Akbar Siddique

Introduction to Hardware Architectures for Parallel Computing

- Parallel computing relies on specialized hardware to achieve high performance.
- Different architectures are designed to support parallelism at various levels.
- This lecture covers:
  - Multi-core and many-core architectures
  - GPU architectures
  - Shared vs. distributed memory models

Multi-core vs. Many-core Architectures

Multi-core Architectures

- A multi-core processor is a single computing component with two or more independent processing units (cores). Each core can execute instructions independently, allowing for parallel execution.
- **Characteristics**:

  •Typically **2 to 16 cores** in modern CPUs.

  •Each core has its own **L1 cache** but may share L2/L3 caches.

  •Designed for **general-purpose computing** and optimized for single-threaded performance.

  •Common in **desktops, laptops, and servers**.
- **Example**:

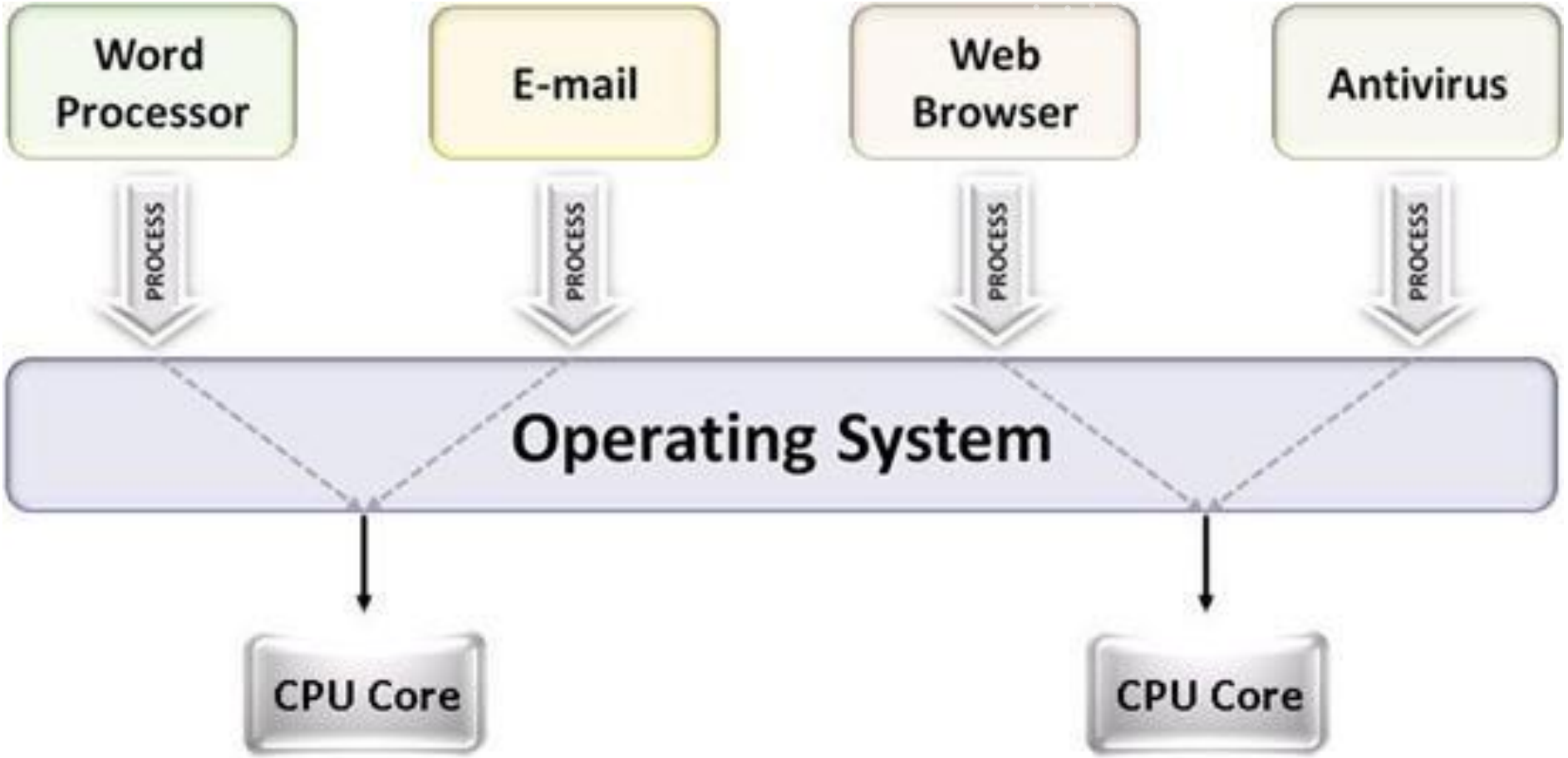
  •Intel Core i7-12700K (12 cores, 20 threads)

  •AMD Ryzen 9 5900X (12 cores, 24 threads)
- **Use Case**:

  •Running multiple applications (e.g., web browsing, video streaming, gaming).

  •Multitasking (e.g., running a code compiler while editing a document).

Task Distribution in Multi-Core Processor



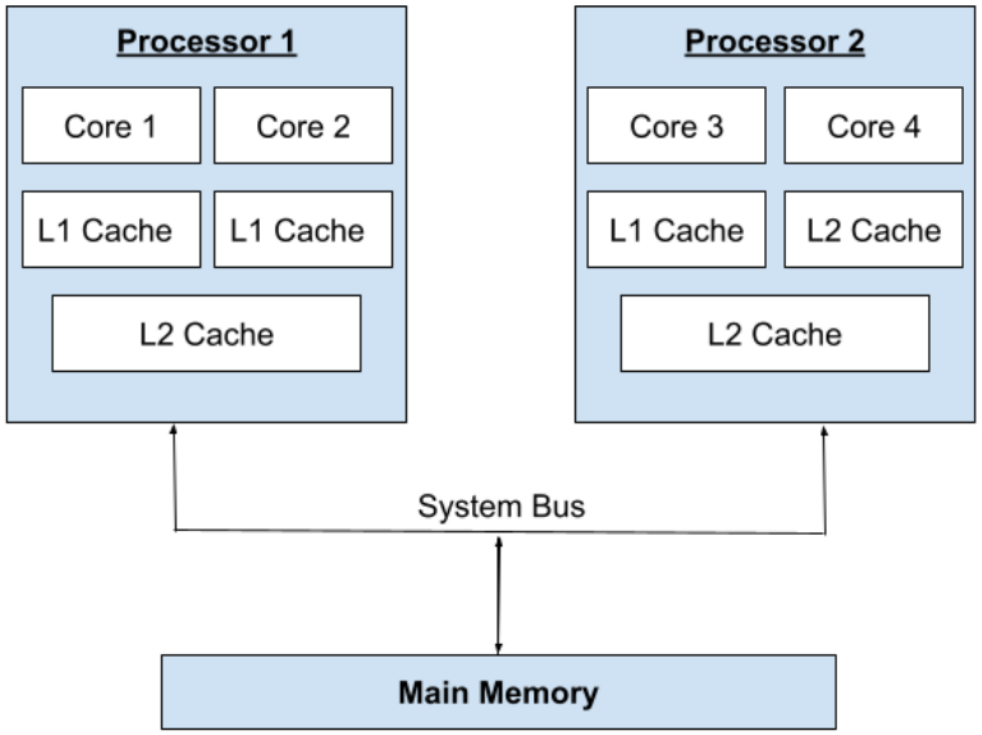
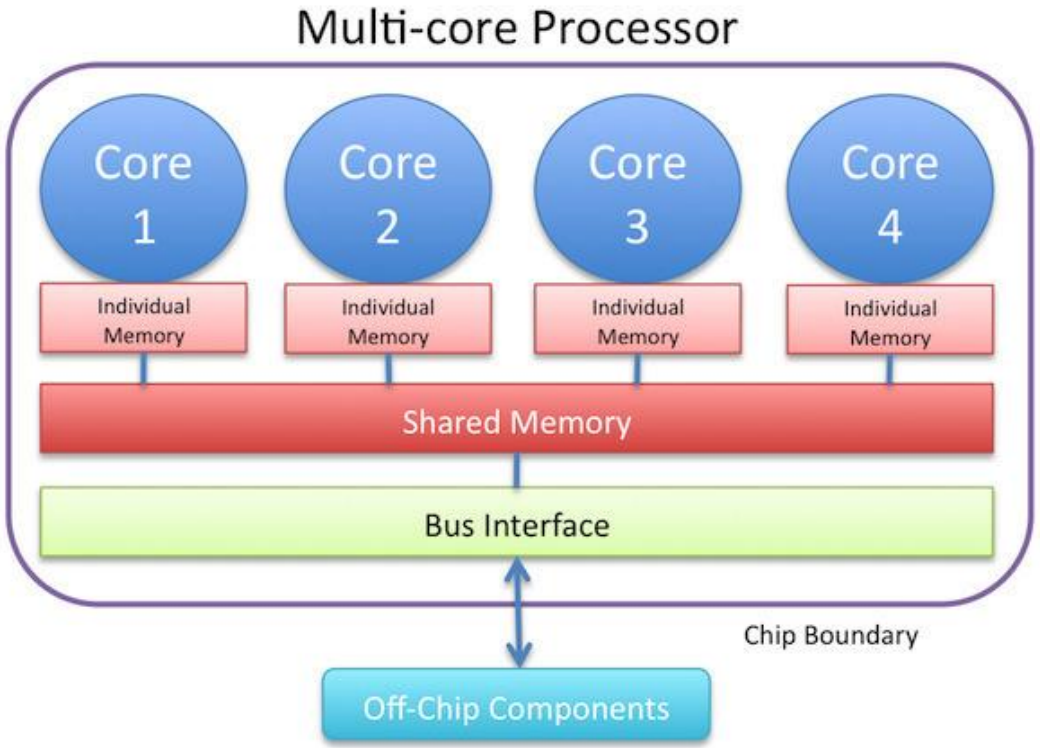


Fig. 2: Multi-core Processor Architecture



Cache Memory – L1, L2, and L3

1. What is Cache Memory?

- Cache memory is a small-sized, high-speed memory located close to the CPU.
- It stores frequently accessed data and instructions to **reduce latency** and improve processing speed.
- Faster than RAM but smaller in size.

2. Levels of Cache (L1, L2, L3)

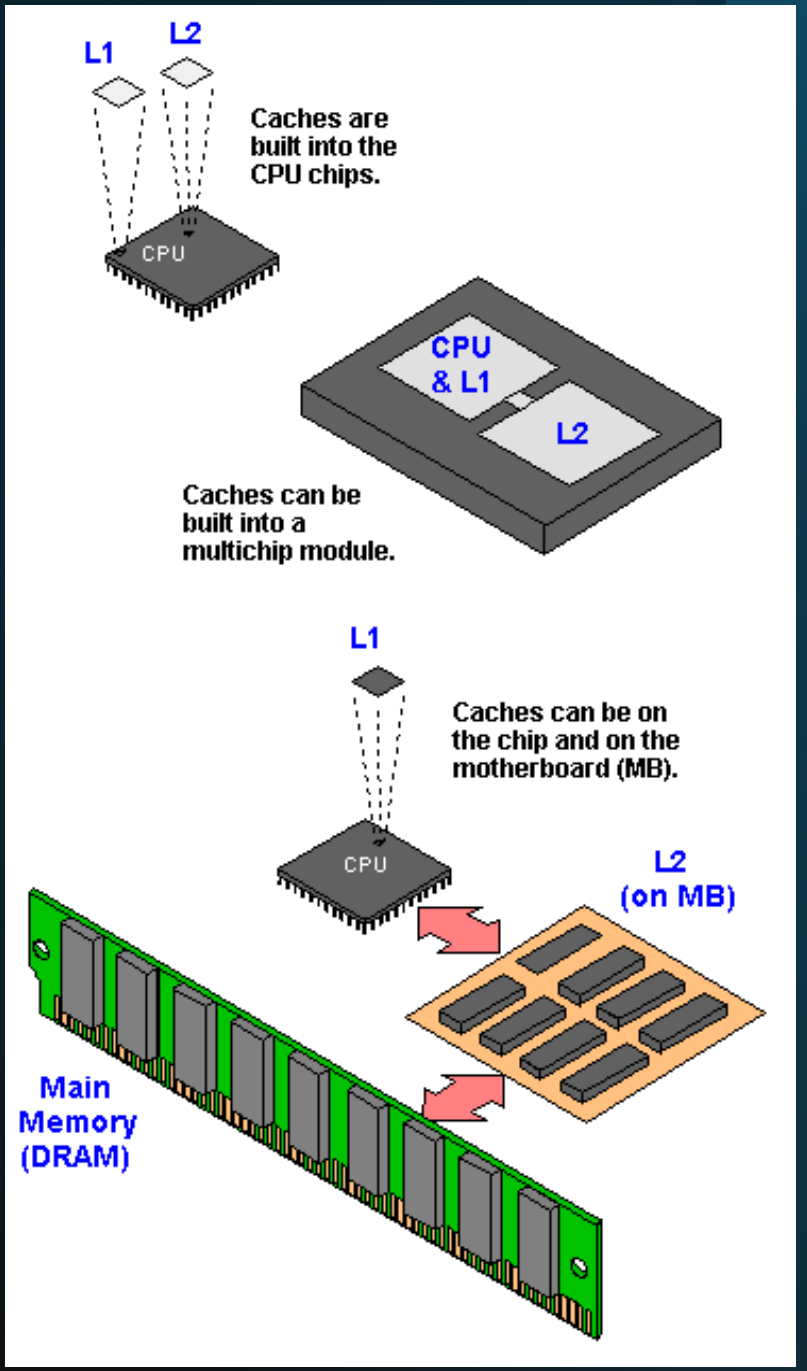
- Modern processors use a multi-level cache hierarchy to balance speed and size.

Cache Hierarchy

Cache Level	Size (Typical)	Speed (Latency)	Location	Function
L1 (Level 1)	32KB - 1MB	Fastest (1-4 cycles)	Inside CPU core	Stores most frequently used instructions & data
L2 (Level 2)	256KB - 8MB	Slower than L1 (5-20 cycles)	Shared per core or per group	Holds data evicted from L1
L3 (Level 3)	4MB - 64MB	Slowest (20-50 cycles)	Shared across all cores	Reduces access time for RAM

Cache Levels – L1

- **L1 Cache (First Level Cache)**
- Smallest but **fastest** cache (closest to the CPU).
- Stores **immediate instructions** and frequently accessed data.
- Typically divided into:
  - **L1 Instruction Cache (L1i)** – Stores CPU instructions.
  - **L1 Data Cache (L1d)** – Stores frequently used data.
- **Example:**
- When a CPU fetches an instruction, it first looks in the **L1 cache** before going to L2 or RAM.



Cache Levels – L2

L2 Cache (Second Level Cache)

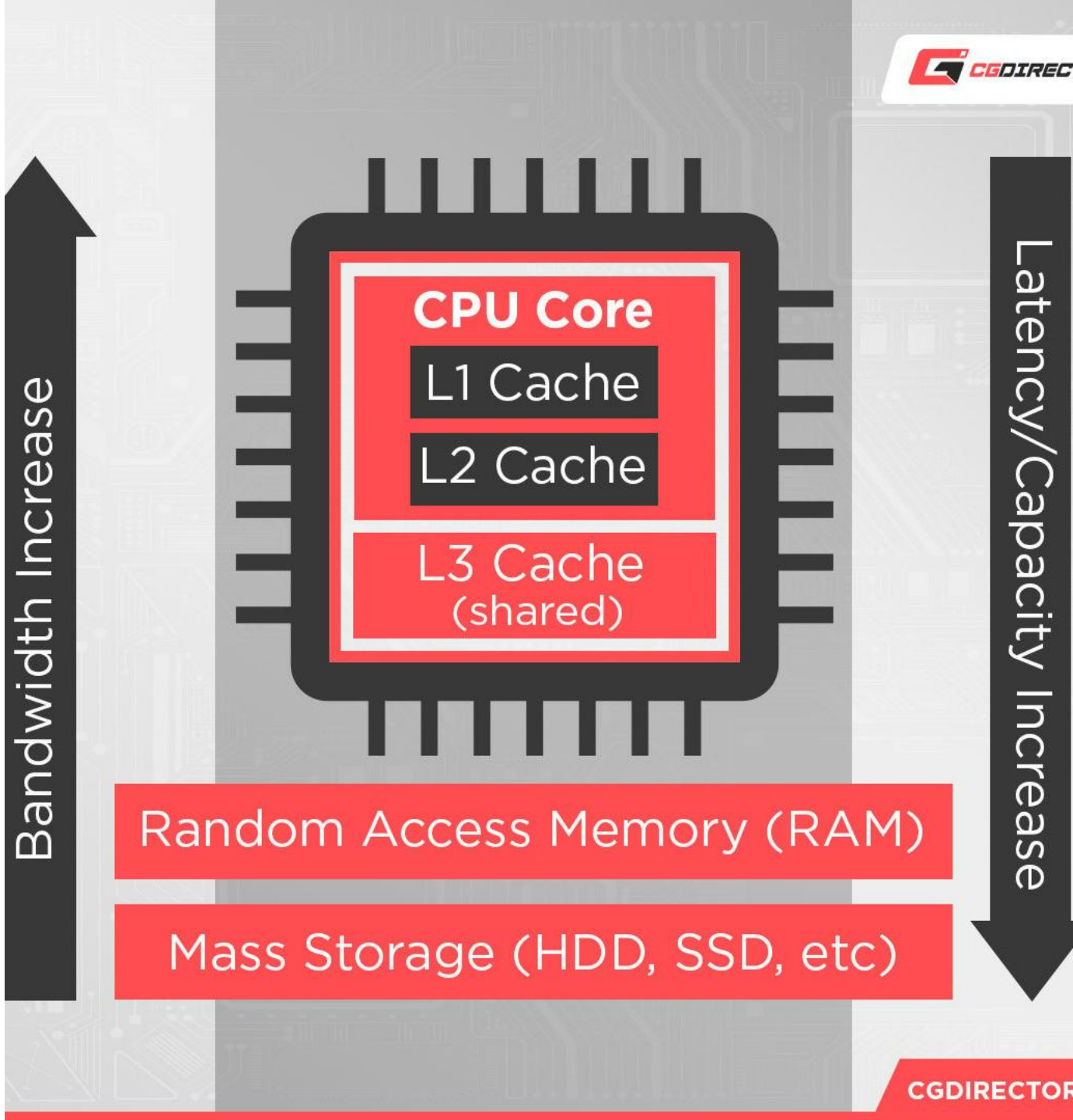
Larger than L1 but slightly slower.

Stores data that doesn't fit in L1.

Can be dedicated per core or shared among multiple cores.

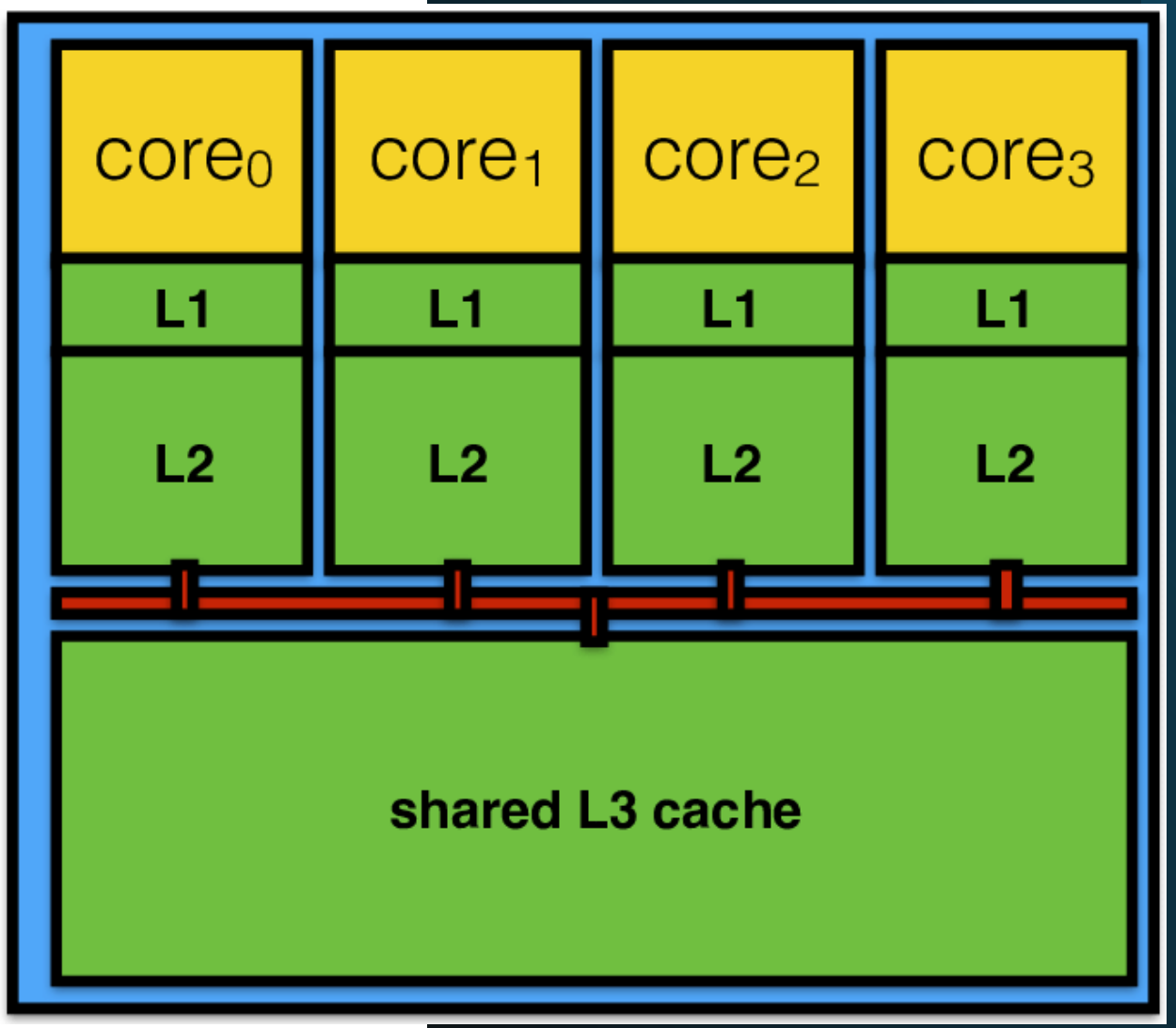
Example:

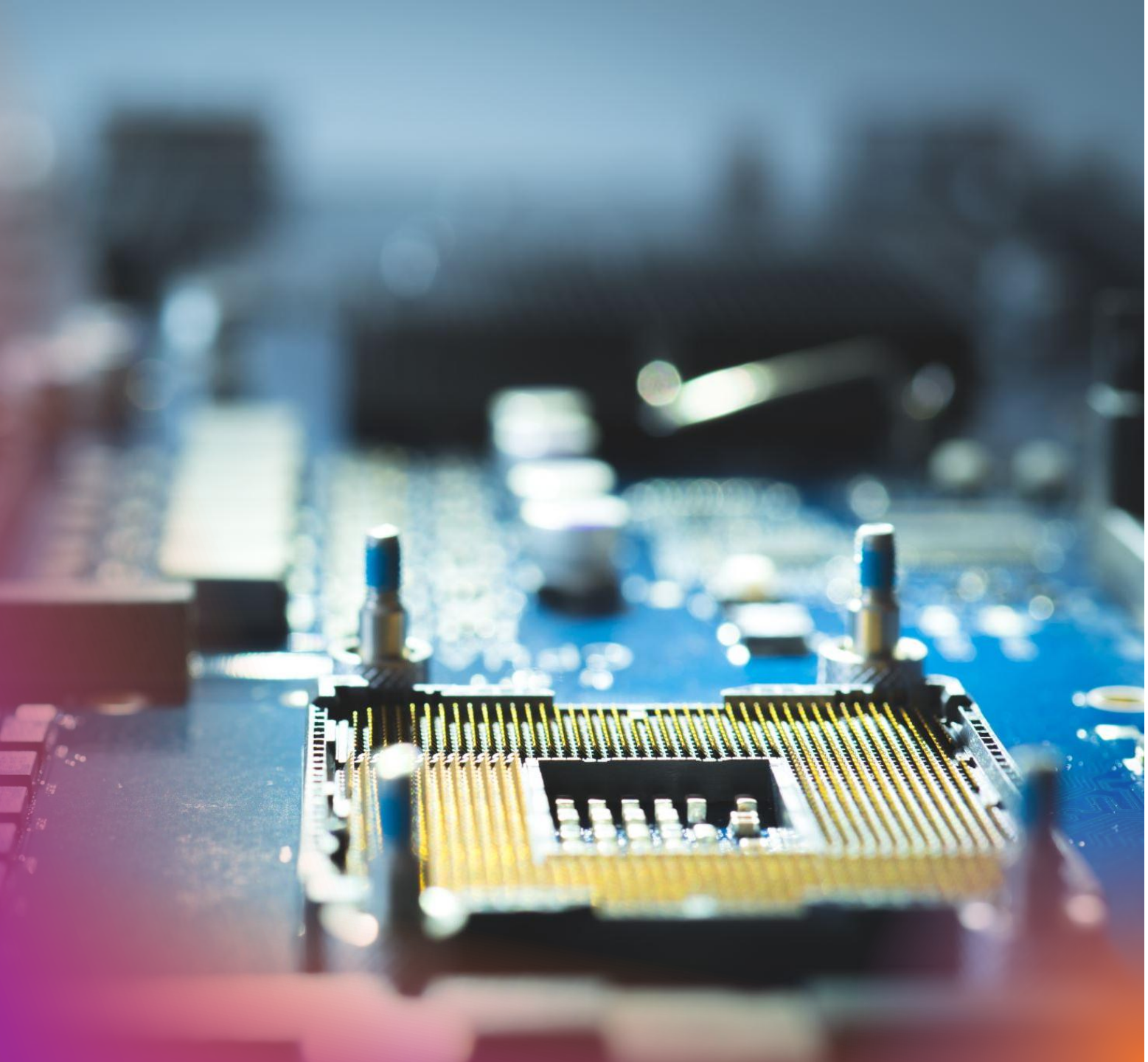
- If L1 cache misses a piece of data, it checks L2 before accessing RAM.



Cache Levels – L3

- **L3 Cache (Third Level Cache)**
  - **Largest and slowest** cache level but **still much faster than RAM**.
  - Typically **shared across all cores** in multi-core CPUs.
  - Reduces **bottlenecks** when multiple cores request the same data.
- **Example:**
  - In gaming or video rendering, **L3 cache helps multiple cores** access frequently used textures or frame data.





Cache Performance Impact

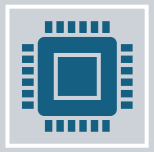
- A well-optimized cache system reduces CPU-RAM interaction, improving processing speed.
- More L1 and L2 cache means faster response times for critical operations.
- L3 cache helps in multi-threaded performance (e.g., AI training, video editing).

Real-World Example (CPU Cache Sizes)

Processor	L1 Cache	L2 Cache	L3 Cache
Intel Core i9-13900K	80KB per core	2MB per core	36MB shared
AMD Ryzen 9 7950X	64KB per core	1MB per core	64MB shared
Apple M2	192KB per core	2MB per core	16MB shared

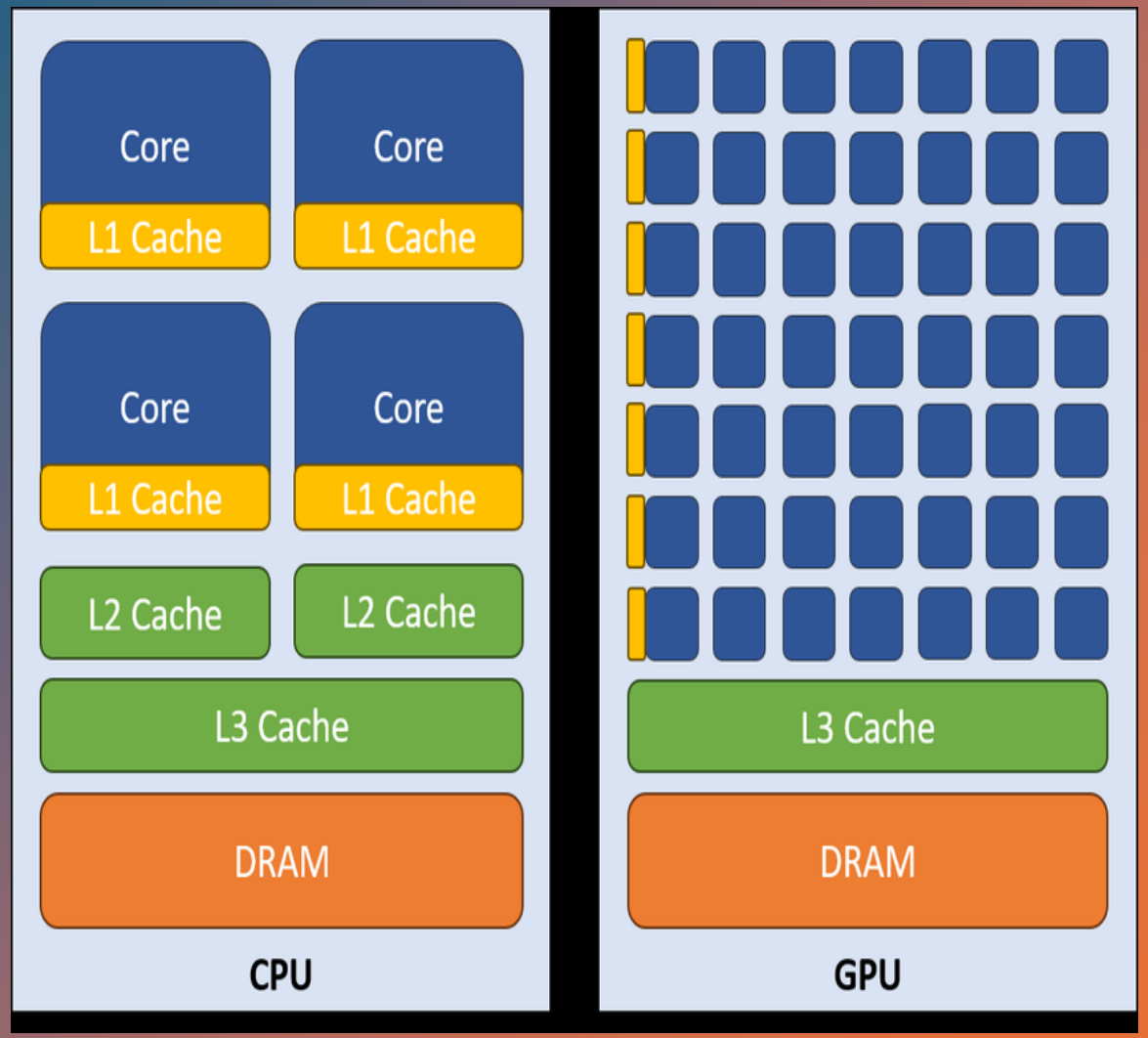
Analogy for Better Understanding

L1 Cache → Like a notepad on your desk (very fast, very small).



L2 Cache → Like a bookshelf next to your desk (a bit slower but larger).

L3 Cache → Like a library in your office building (much larger but slower to access).

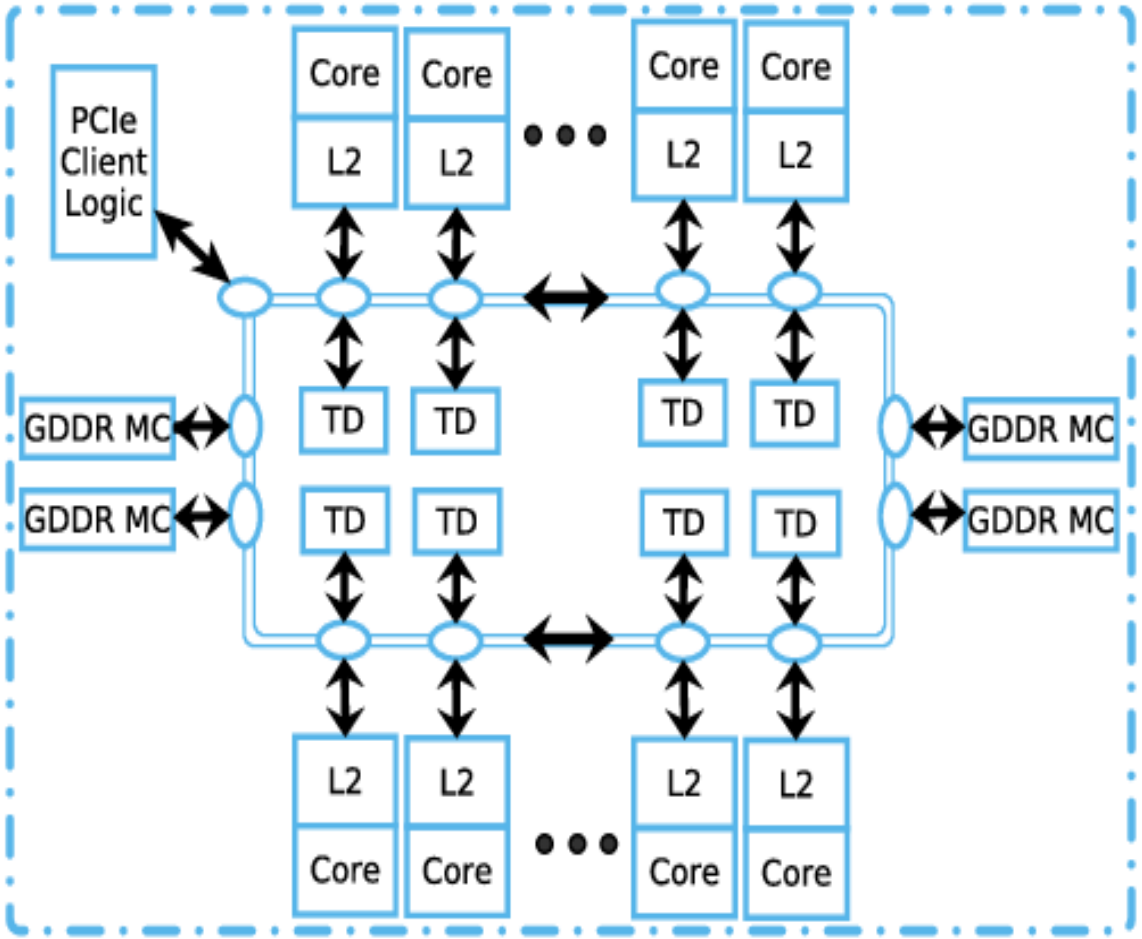


Many-core Architecture

- A many-core processor contains a large number of cores (typically dozens to hundreds) optimized for highly parallel workloads. These architectures prioritize parallelism over single-threaded performance.
- **Characteristics**:
  - Designed for **massive parallelism**.
  - Typically found in **GPUs and specialized accelerators**.
  - Cores may be **simpler and lower power** compared to multi-core CPUs.
  - Used in **scientific computing, AI, and simulations**.
- **Example:**
  - NVIDIA A100 GPU (6912 CUDA cores, 432 Tensor cores)
  - AMD Instinct MI250X (13,312 Stream processors)
- **Use Case:**
  - Training deep learning models using TensorFlow or PyTorch.
  - High-performance computing (HPC) workloads like weather modeling.
  - Rendering 3D graphics in video games or animations.

Key Components in Many Core Architecture

- **Cores & L2 Cache:**
  - Each **core** (processing unit) has access to **L2 cache**, which stores frequently used data to reduce memory access latency.
  - The **cores are organized in groups**, likely representing Streaming Multiprocessors (SMs) in a GPU.
- **TD (Thread Dispatching) Units:**
  - These units help distribute workloads efficiently among the processing cores.
  - They manage threads and optimize parallel execution.
- **GDDR Memory Controllers (MC):**
  - These handle data transfers between the **GPU and the global memory (GDDR memory).**
  - GDDR (Graphics Double Data Rate) memory is optimized for high bandwidth.
- **PCIe Client Logic:**
  - Responsible for communication between the **GPU and the CPU/system memory** via the **PCIe (Peripheral Component Interconnect Express) bus**.



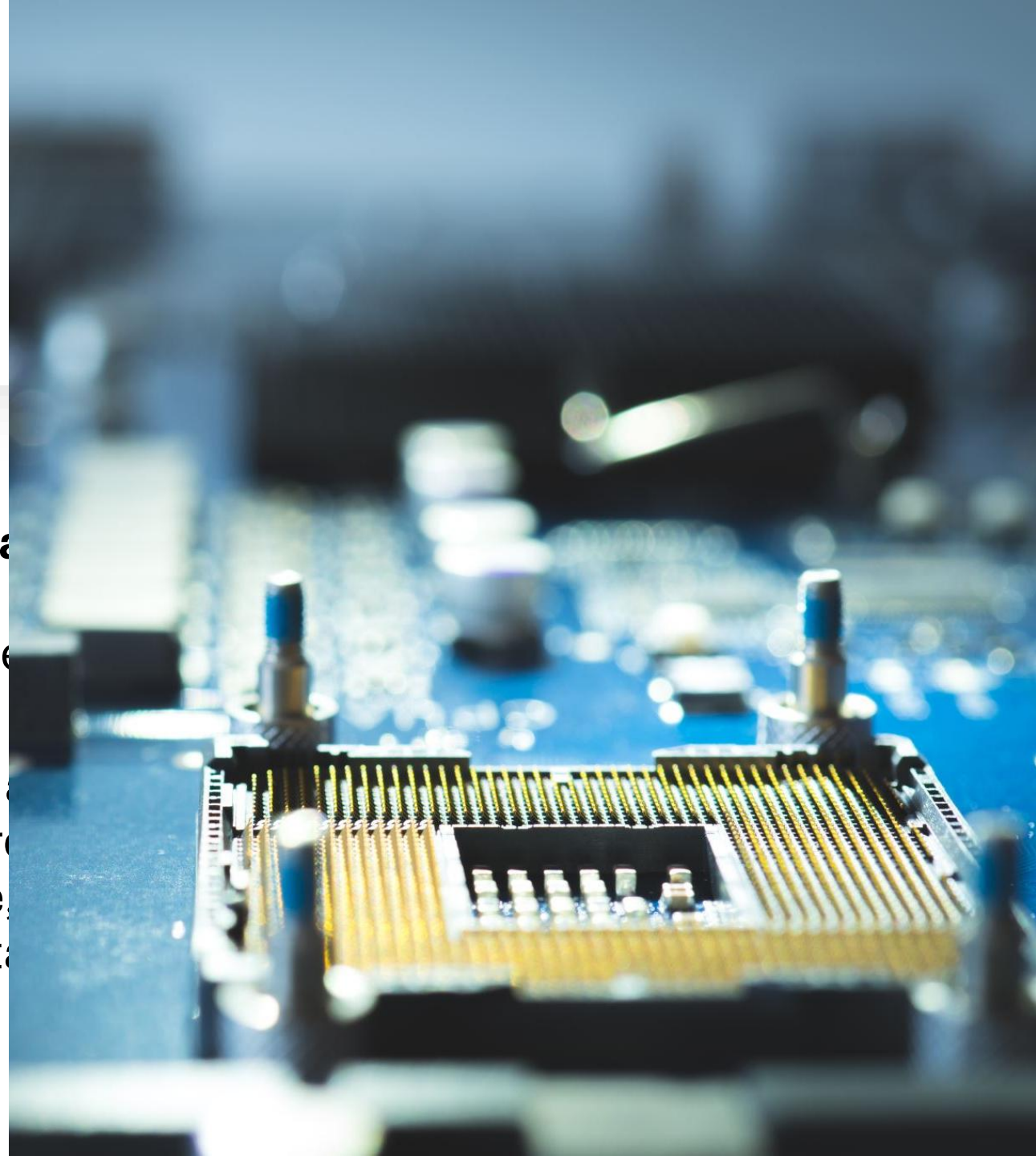
- **Interconnects (Arrows & Rings):**
  - The black arrows and circular connections represent **data flow and interconnects** that enable efficient communication between processing cores, caches, and memory controllers.

Key Differences

Feature	Multi-core Architecture	Many-core Architecture
Number of Cores	2 - 16 cores	Dozens to thousands of cores
Optimization	General-purpose computing	Highly parallel tasks
Cache Structure	L1 per core, shared L2/L3	Smaller caches, often per group of cores
Thread Handling	Efficient for multitasking	Optimized for high-thread parallelism
Use Cases	General computing, gaming, office apps	AI, scientific computing, simulations

Analogy for Better Understanding

- **Multi-core CPUs** are like **a sma**
  of highly skilled workers, each
  of doing a variety of tasks indepe
  but with limited numbers.
- **Many-core GPUs** are like **an**
  workers, each handling small, r
  tasks but at an enormous scale
  them perfect for highly parallel t

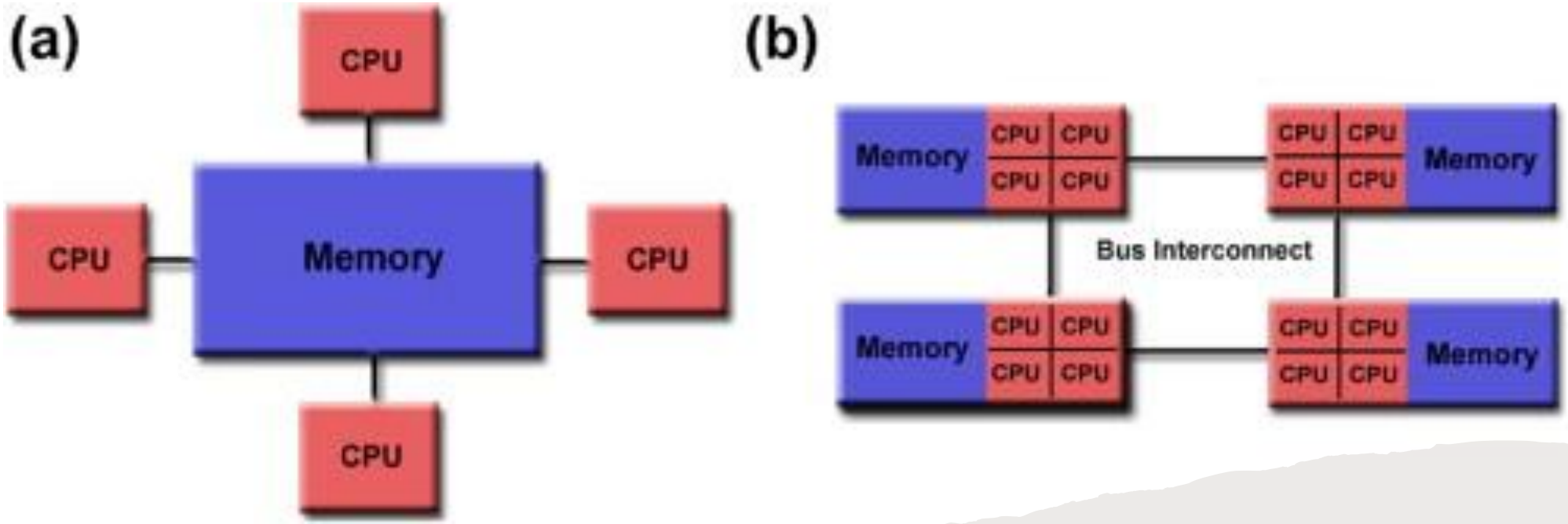


Shared Memory Model

- Multiple processors access the same memory space.
- Key Characteristics:
  - ✓ Fast data sharing between processors.
  - ✓ Requires synchronization to avoid conflicts (e.g., using mutexes and semaphores).

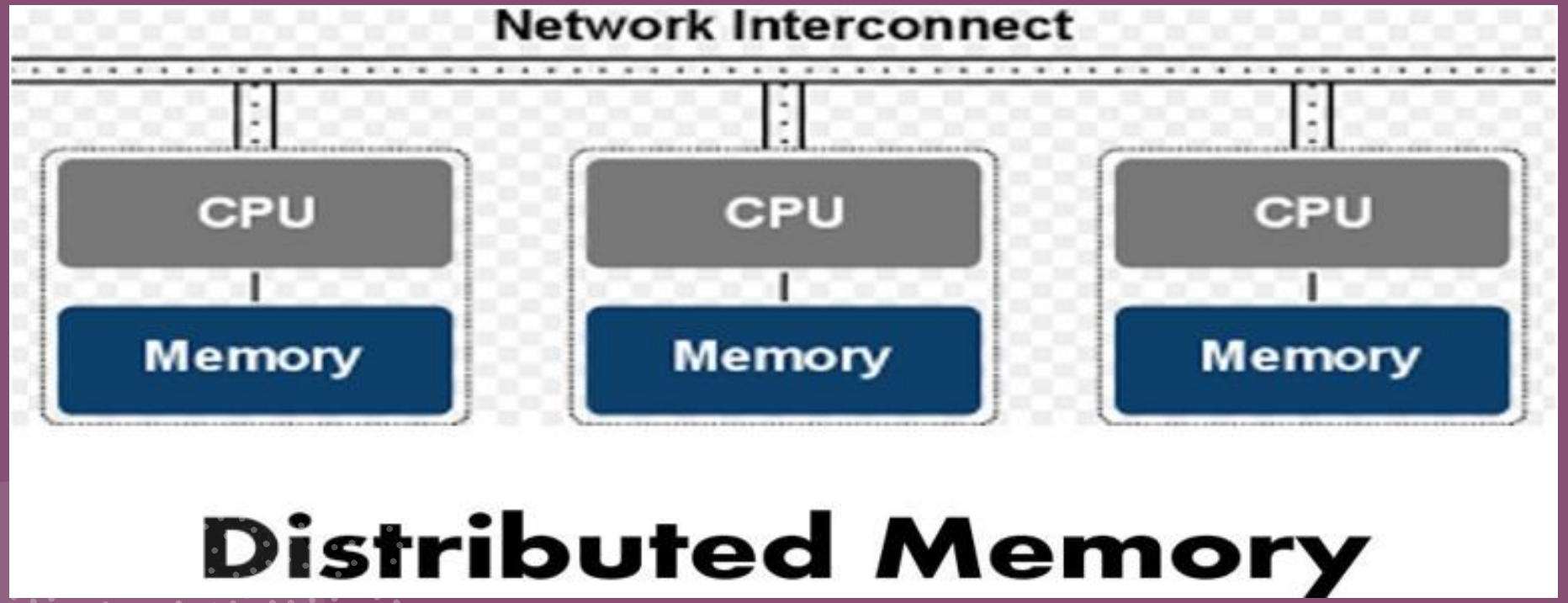
Example:

✓ Multi-threaded programming in a multi-core processor.



Distributed Memory Model

- Each processor has its own local memory and communicates via a network.
- Key Characteristics:
  - ✓ No direct memory access between processors.
  - ✓ Requires message passing (e.g., MPI - Message Passing Interface).
- Example:
  - ✓ Supercomputers using distributed clusters to solve large-scale problems



Shared vs. Distributed Memory

FEATURE	SHARED MEMORY	DISTRIBUTED MEMORY
Memory Access	Single memory space shared by processors	Each processor has its own memory
Communication	Direct memory access	Message passing (e.g., MPI)
Scalability	Limited to memory bandwidth	Highly scalable for large computations
Example	Multi-core CPUs	Cluster computing, cloud systems