Thread-level parallelism

How architectures are built to exploit TLP

Before the multi-core era

- Lots of growth in single-core performance
  - Especially between the mid 1980s and the mid 2000s

- But the growth couldn't last forever…
  - Limits to instruction-level parallelism
  - Power consumption

- Meanwhile, multiprocessors were improving
  - Servers, supercomputers
  - Cloud computer, software-as-a-service

What is a chip multiprocessor (CMP)?

Definition

- A collection of tightly couped processors
  - Called cores
- Shared memory system
  - And, for programs, a shared address space
- Managed by a single operating system

How do CMPs exploit parallelism?

- Multi-threaded programs
  - 1 program, n threads
- Multi-programming
  - n independent programs, or
  - 1 program duplicated n times

How are CMPs organized?

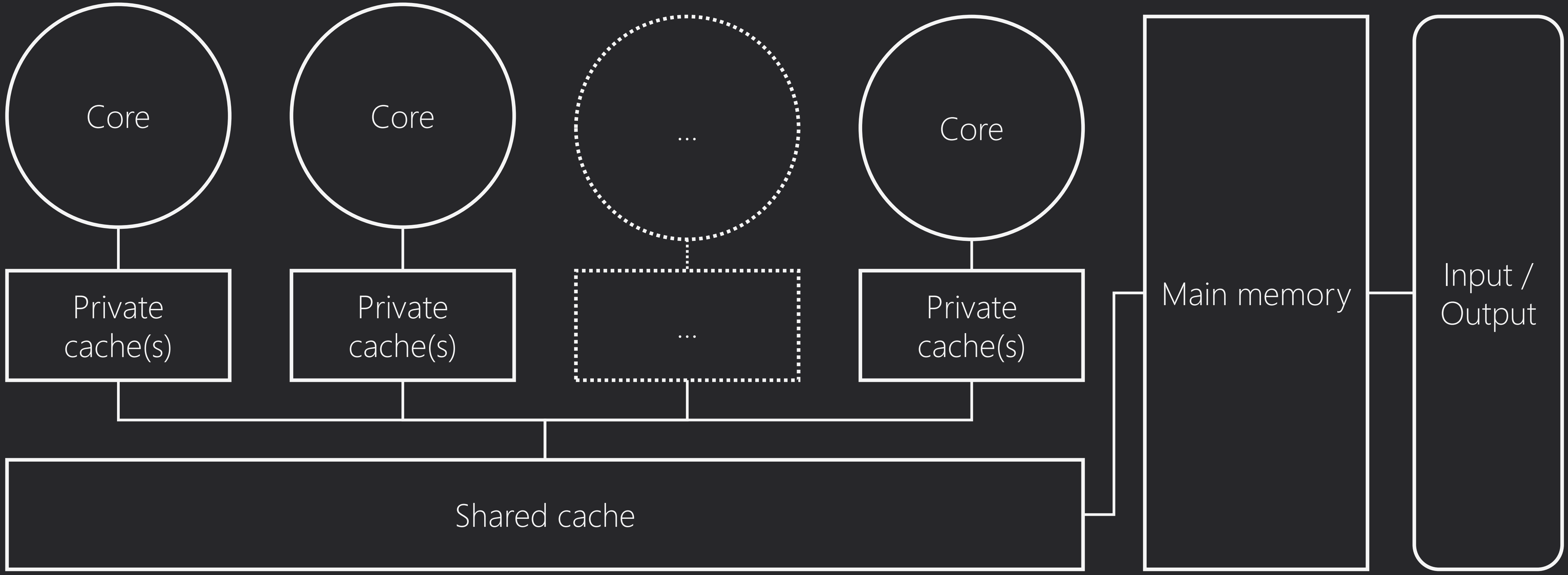
- Symmetric multiprocessors (SMP)

- Distributed shared memory (DSM)
  - Non-uniform cache access (NUCA)
  - Non-uniform memory access (NUMA)

- Connected via an interconnection network

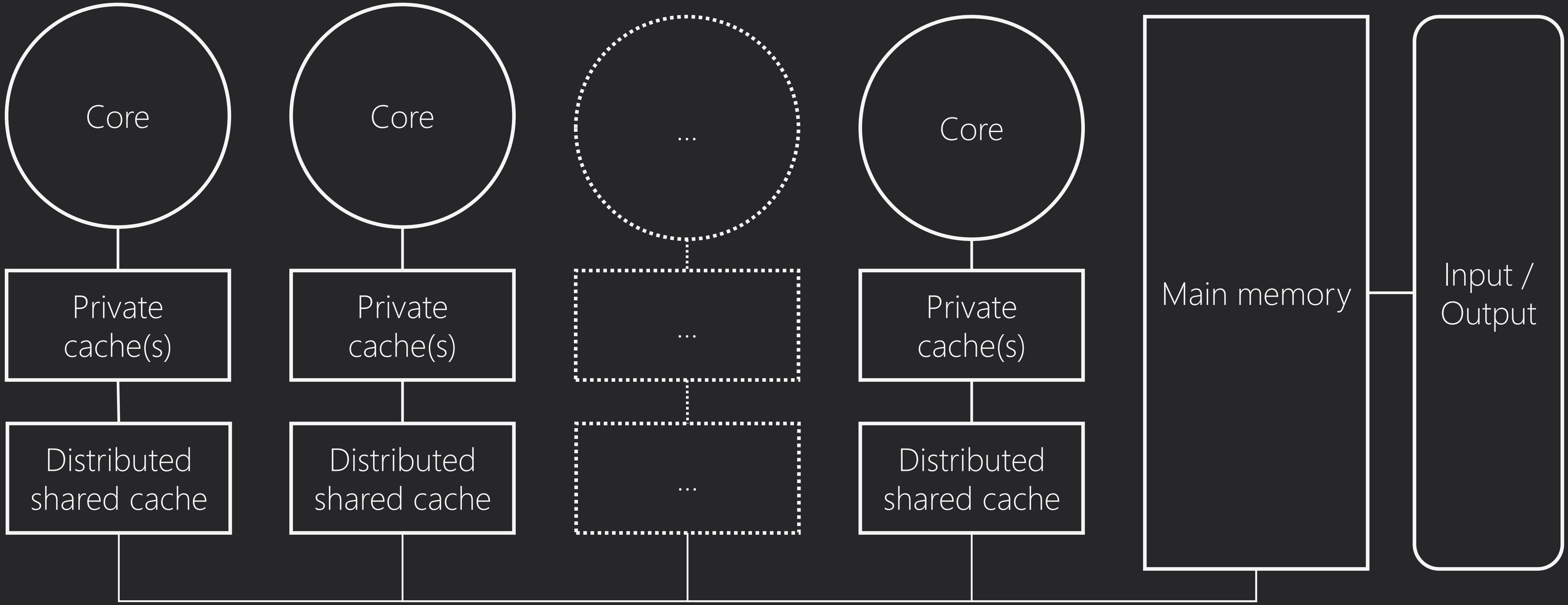
What is a SMP?

- All cores share a centralized memory
- Access to memory is symmetric
  - Uniform memory access time for every core



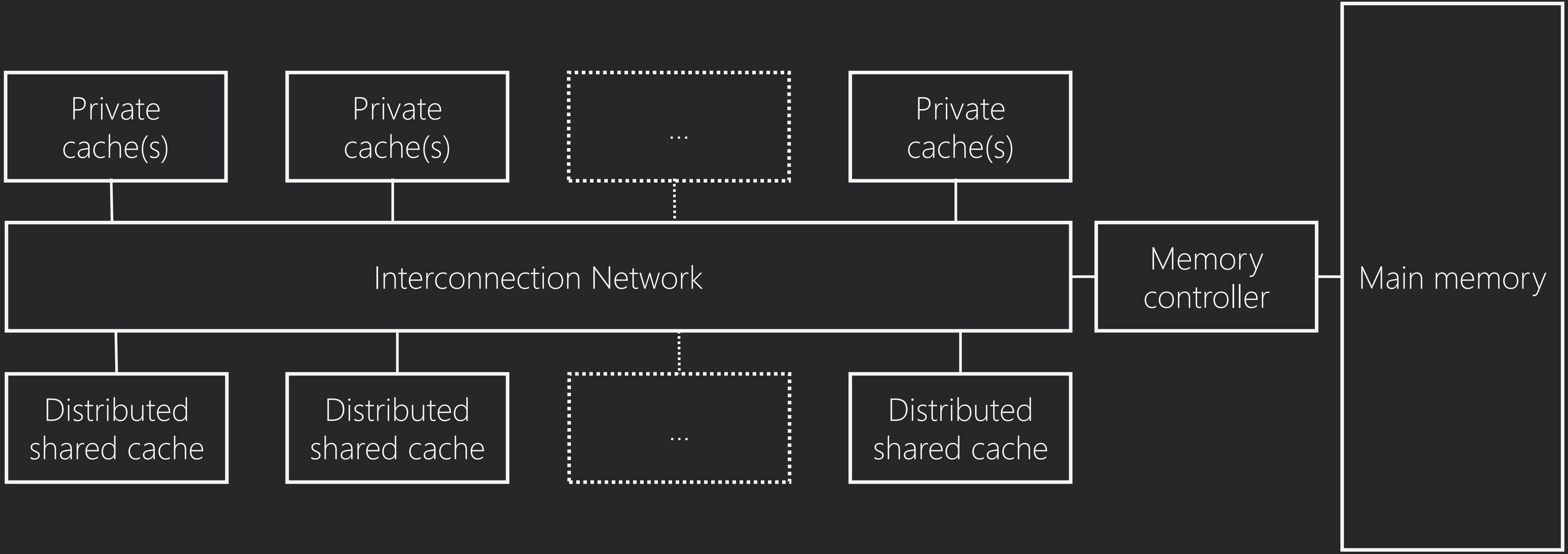
What is NUCA/NUMA?

- Not a true SMP
- Last-level cache is distributed across the chip
- Time to access last-level cache varies



What is an interconnection network?

- An infrastructure to support sending messages across the chip
- Lots of wires and logic dedicated to communication



What are the challenges with parallel processing?

1. Limited parallelism
   - The programmer’s “free lunch” is over

2. The cost of communication (is high)
   - Cache coherence
   - Memory consistency
   - Synchronization

What is cache coherence?

Defines the behaviour of reads and writes to the same memory location ("What values can be returned by a read.")

Time	Event	A's private cache	B's private cache	Main memory
0				42
1	A reads foo	42		
2	B reads foo	42	42	42
3	A writes to foo	16	42	16

What is a coherent memory system?

Given cores P1 and P2 and memory location foo,

1. A write to, followed by a read from, foo by P1 always returns the value written by P (provided P2 does not write to foo between the write and read)
   - Preserves program order (same for single-cores)
2. A write by P2 to foo, followed by a read by P1 from foo, returns the value written by P2 (*provided the read/write are sufficiently spaced apart*)
3. Writes to foo are seen in the same order by all processors

What is memory consistency?

Defines the behaviour of reads and writes of access to other memory location ("When a written value will be returned by a read")

- Last slide: A write by P2 to foo followed by a read by P1 from foo returns the value written by P2 (*provided the read/write are sufficiently spaced apart*)

- So *when* will a written value from P2 be seen by P1? For now,
  - A write is not "complete" until all processors have seen the write
  - A processor cannot change the order of any write with respect to any other memory access

- There are alternatives to this (strict) definition

A summary of CMPs

- There are many ways to organize and connect CMPs
  - SMP, NUMA, NUCA
  - Interconnection networks

- Communication in CMPs is challenging
  - Varying latencies
  - Cache coherence
  - Memory consistency