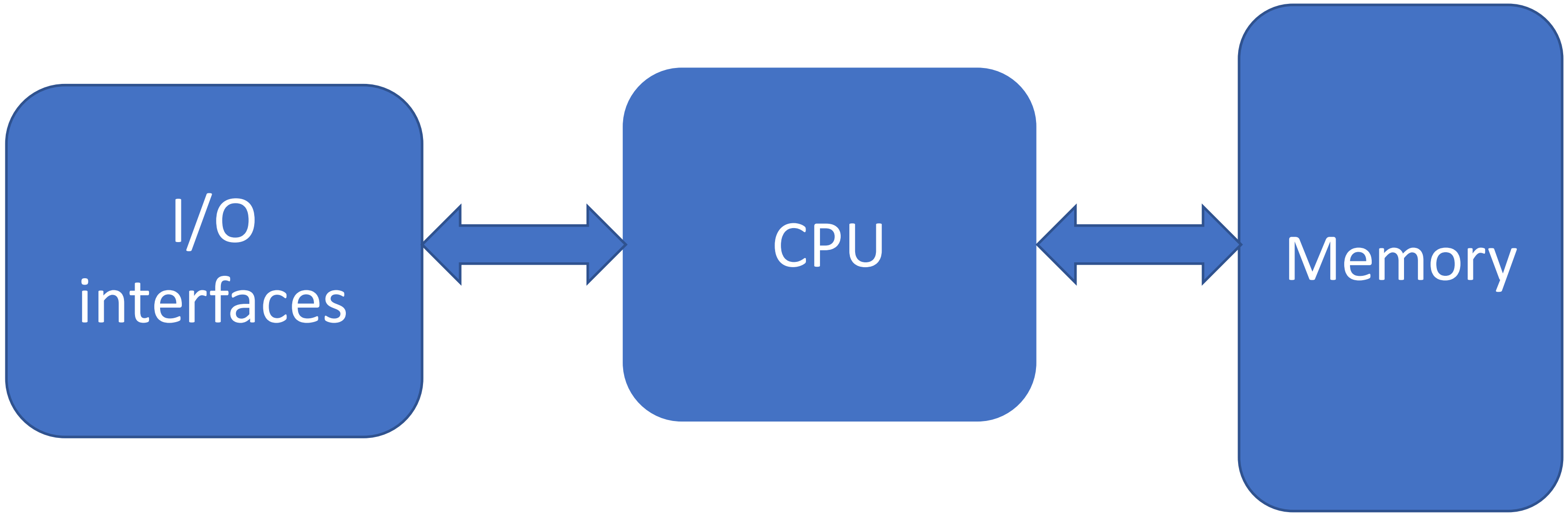
Multicore Systems

Sujoy Sinha Roy

School of Computer Science

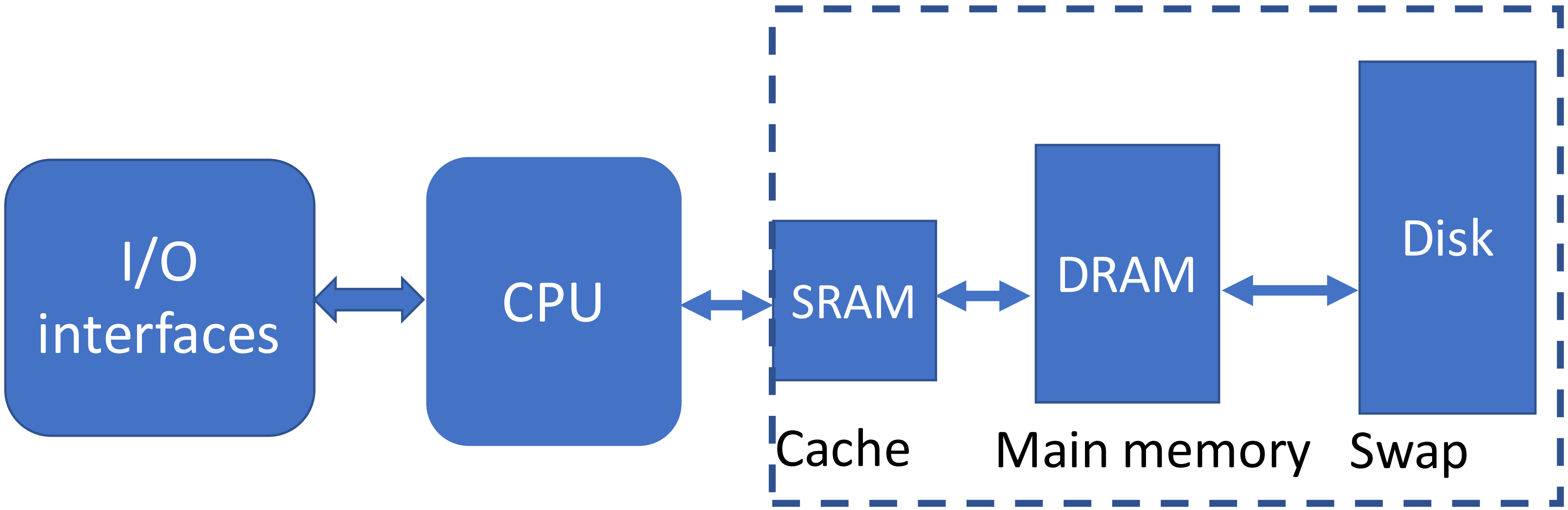
University of Birmingham

Recap: Simple von Neumann Computer



We started C programming assuming a simple von Neumann computation model.
→ We executed program on a single CPU (or processor) step-by-step.
→ We stored data and program instructions in the Memory.

Recap: Memory Hierarchy



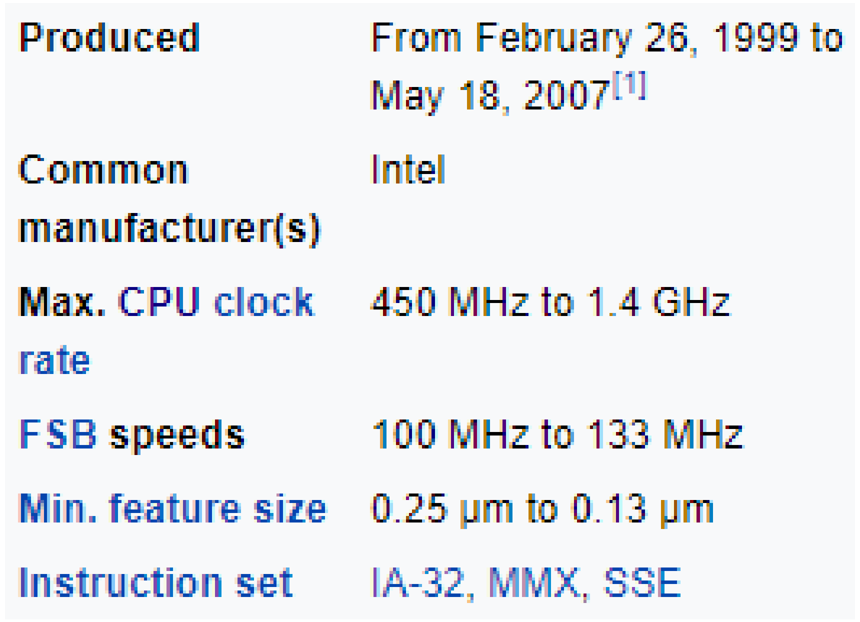
We studied memory hierarchy from a programmer's perspective

→ Learnt how to write efficient C code to exploit memory hierarchy

→ Programs are still executed on a single CPU (or processor) step-by-step but more efficiently intel

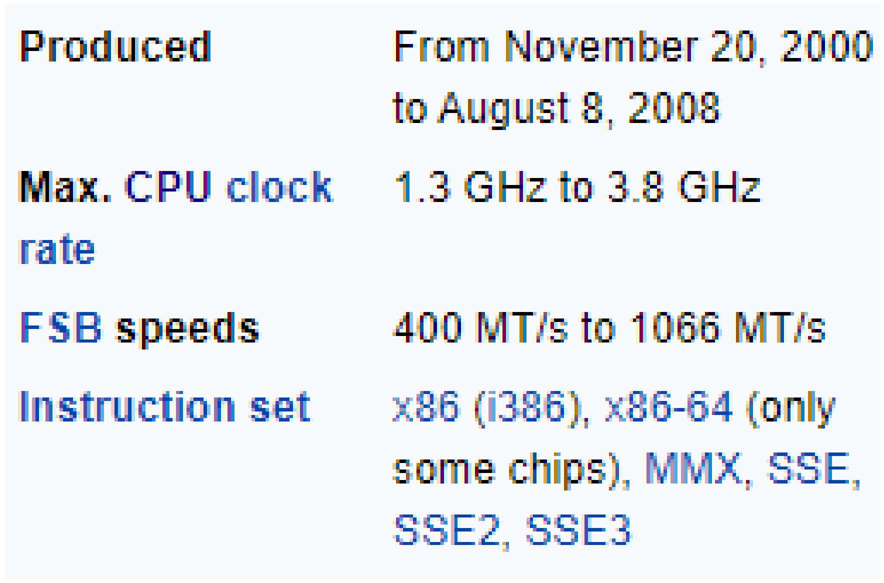
pentium II
PROCESSOR

Produced	From May 7, 1997 to December 26, 2003[1]
Common manufacturer(s)	Intel
Max. CPU clock rate	233 MHz to 450 MHz
FSB speeds	66 MHz to 100 MHz
Min. feature size	0.35 μm to 0.18 μm
Instruction set	IA-32, MMX



Produced	From February 26, 1999 to May 18, 2007[1]
Common manufacturer(s)	Intel
Max. CPU clock rate	450 MHz to 1.4 GHz
FSB speeds	100 MHz to 133 MHz
Min. feature size	0.25 µm to 0.13 µm
Instruction set	IA-32, MMX, SSE

Clock frequency almost doubled w.r.t Pentium II



Produced	From November 20, 2000 to August 8, 2008
Max. CPU clock rate	1.3 GHz to 3.8 GHz
FSB speeds	400 MT/s to 1066 MT/s
Instruction set	x86 (i386), x86-64 (only some chips), MMX, SSE, SSE2, SSE3

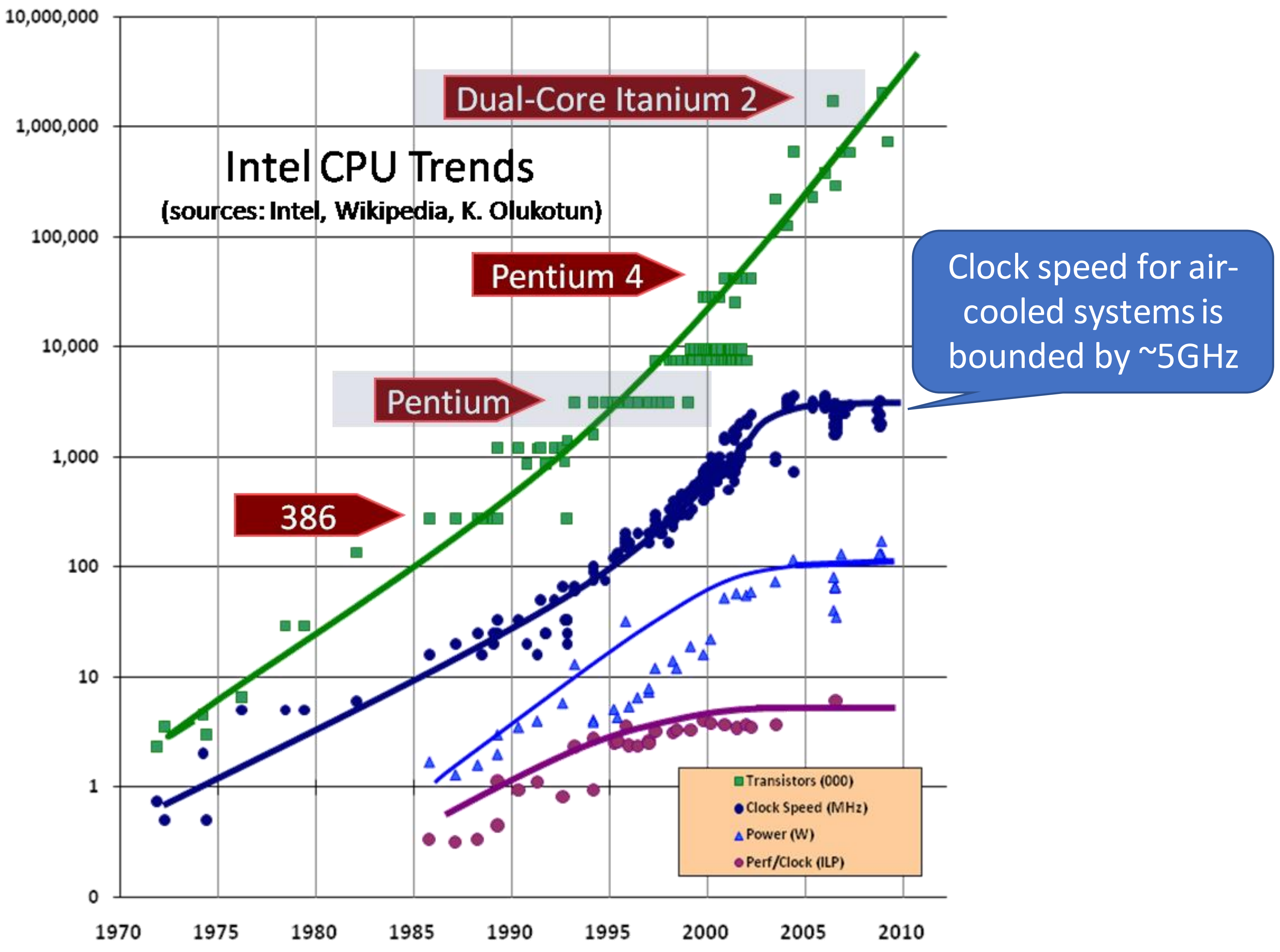
Clock frequency almost doubled w.r.t Pentium III

What is the clock frequency in 2020?

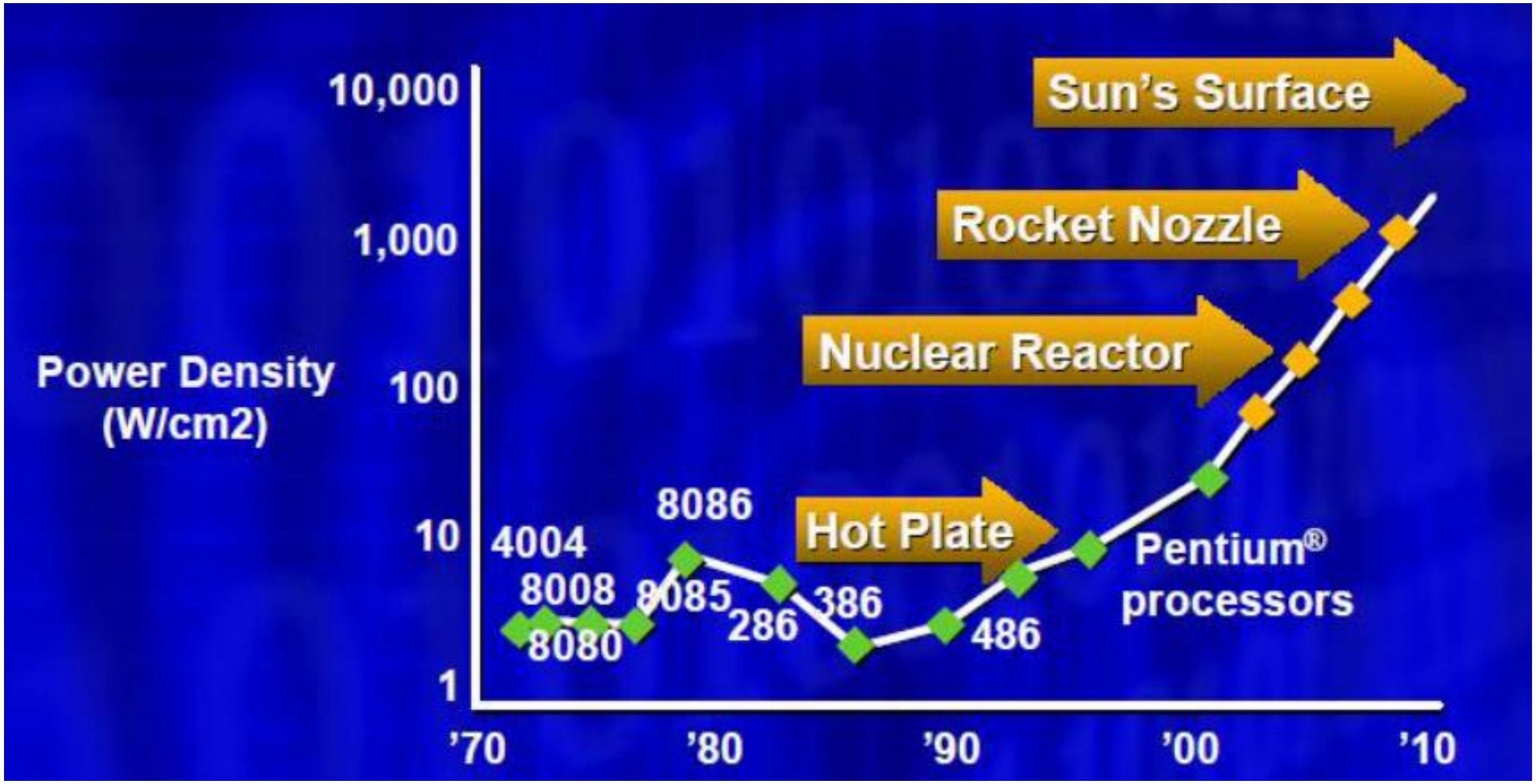
Intel® Core™ i7-10510U Processor (8M Cache, Up to 4.80 GHz)

- 8 MB Intel® Smart Cache Cache
- 4 Cores
- 8 Threads
- 4.90 GHz Max Turbo Frequency
- U - Ultra-low power
- 10th Generation

Little increase in clock frequency after 10 years since Pentium 4



Source: Herb Sutter "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software" available at http://www.gotw.ca/publications/concurrency-ddj.htm



Source: Patrick Gelsinger, Intel Developer's Forum, Intel Corporation, 2004.

Applications are demanding more resources!

Alternative solution:

- Put many processing cores on the microprocessor chip.
- The number of cores doubles with each generation.

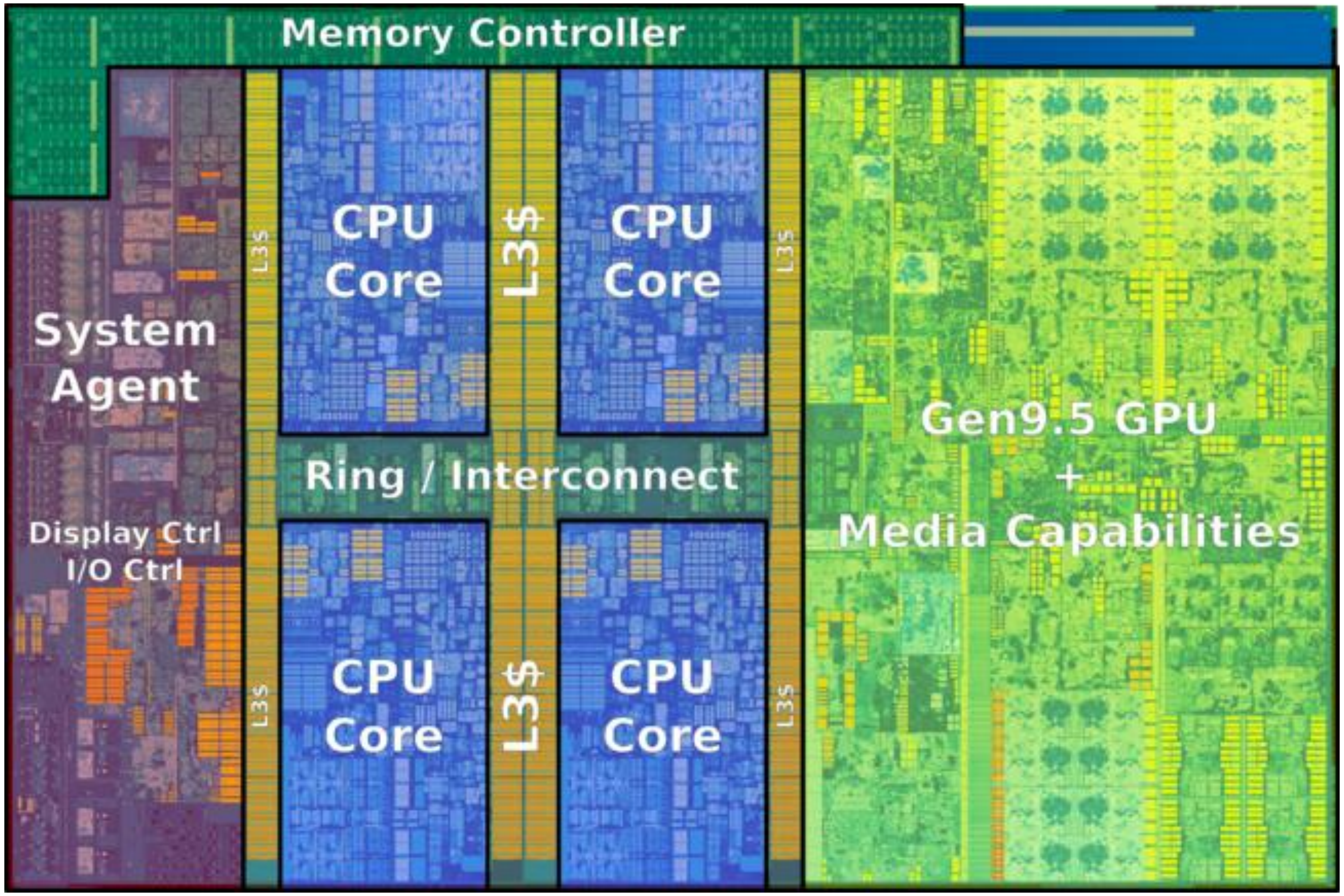


Photo of the quad-core GT2 Kaby Lake processor chip. Used in mainstream desktop computers.

It has 4 CPU cores to compute in parallel.

Image source https://en.wikichip.org/wiki/intel/microarchitectures/kaby_lake#Die

Samsung Galaxy S10+

Exynos 9 Series (9820)

6.4" Quad HD+ Curved Dynamic AMOLED (3040x1440)

Android 9 (Pie)

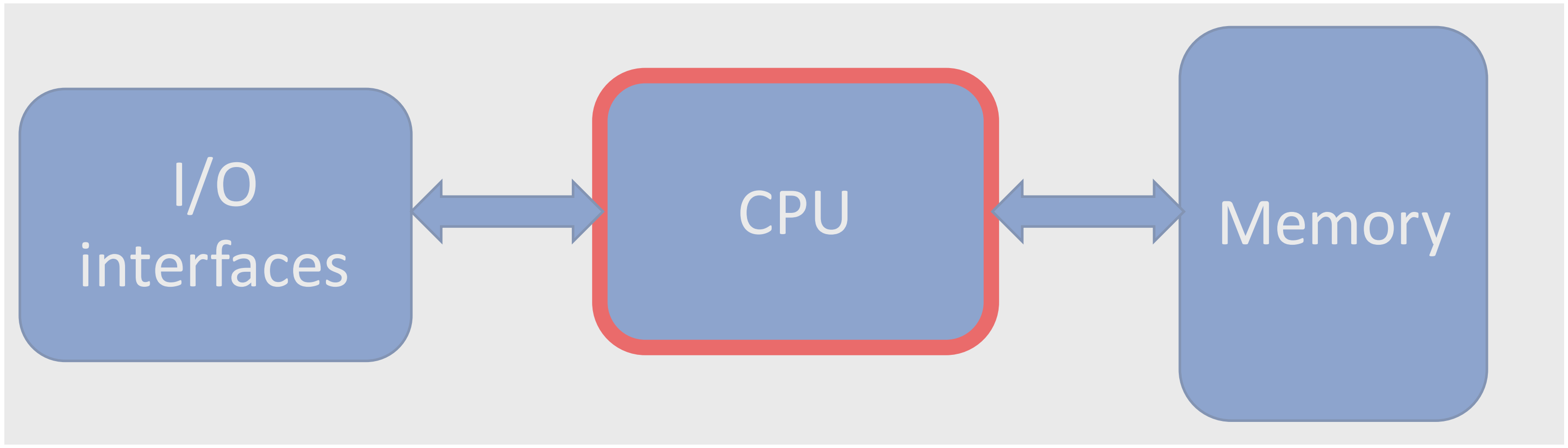
8/12GB RAM, 128/256GB/1TB Storage, microSD(up to 512GB)

Rear: 12MP AF(Wide)+12MP AF(Tele)+16MP FF(Ultra Wide),
Front: 10MP AF+8MP FF(RGB Depth)

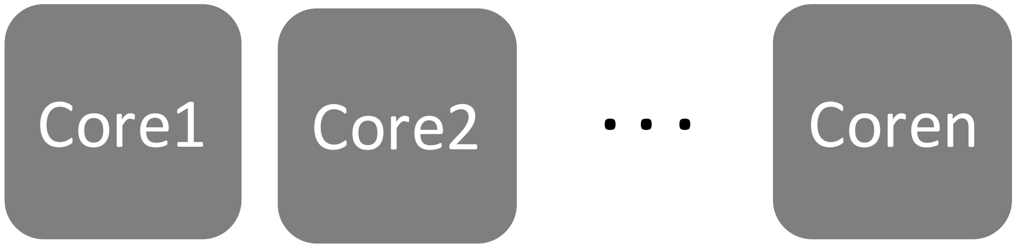
Superior Performance for Seamless Multi-tasking

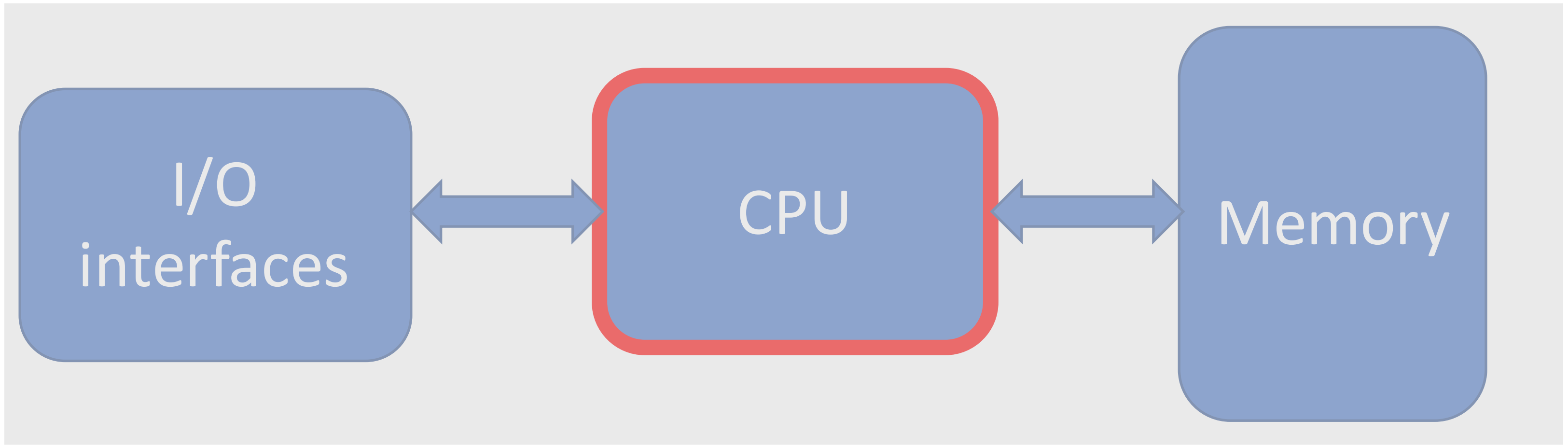
Featuring a 4th generation custom CPU, the Exynos 9820's innovative tri-cluster architecture delivers premium processing power. The CPU consists of two custom cores for ultimate processing power, two Cortex-A75 cores for optimal performance, and four Cortex-A55 cores for greater efficiency, resulting in superior performance that lasts. Tri-cluster with intelligent task scheduler boosts multi-core performance by 15 percent when compared to the Exynos 9810, while the 4th generation custom CPU with enhanced memory access capability and cutting-edge architecture design improves single core performance by up to 20 percent or boosts power efficiency by up to 40 percent.[1]

https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-9-series-9820/



From Single Core
To
Multi Core



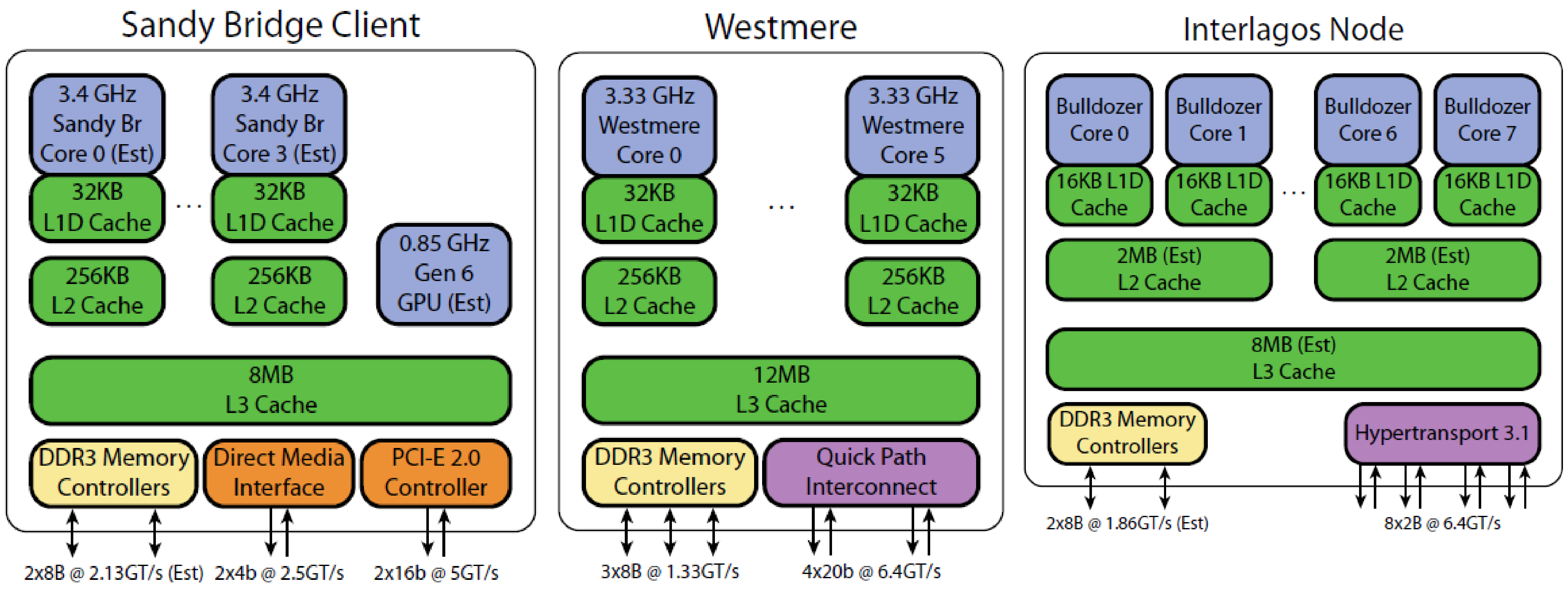


From Single Core
To
Multi Core

What happens to the Memory?
One memory? Multiple memory?

Core1
Core2
. . .
Coren

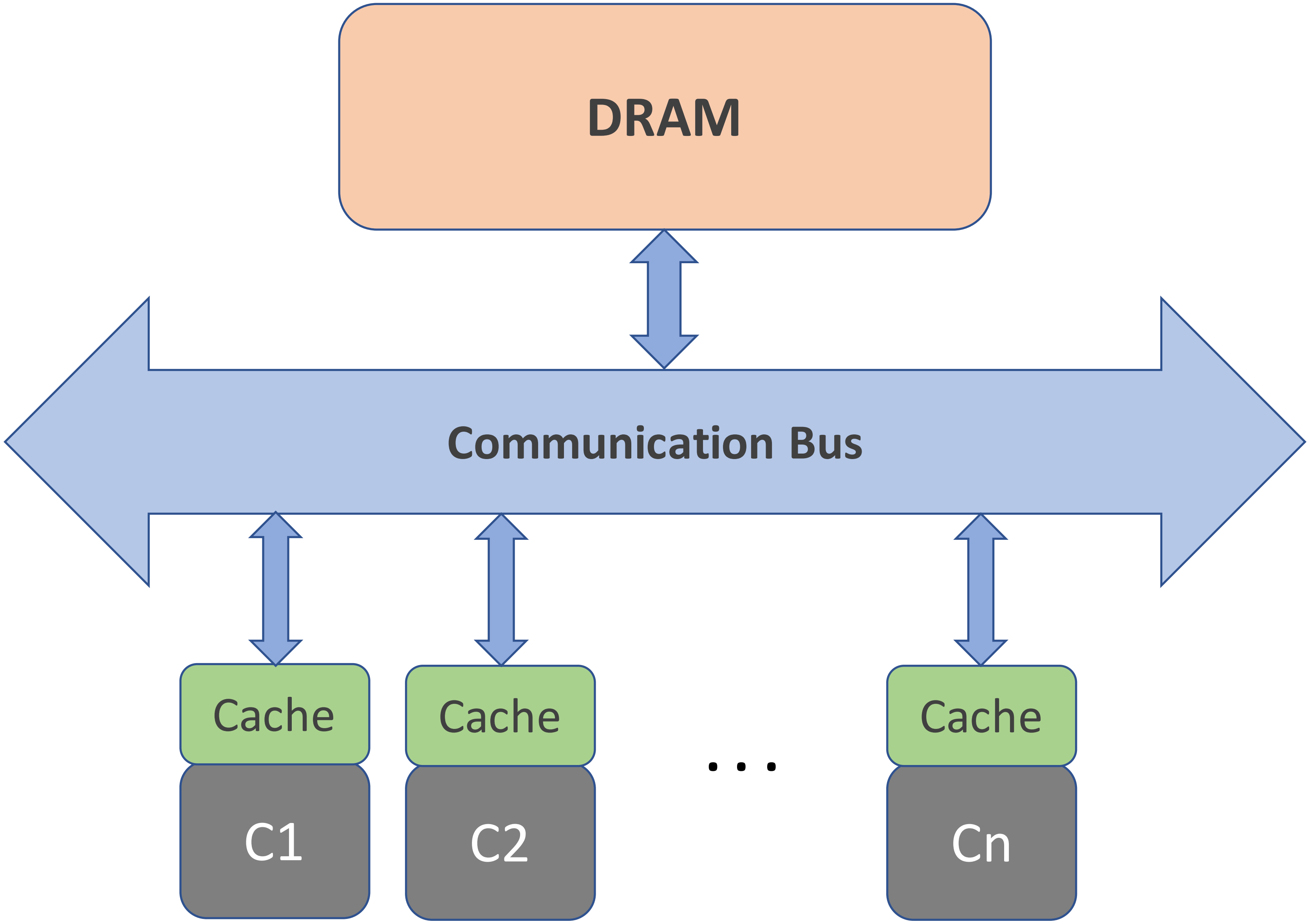
Examples of Intel Multicore Processors



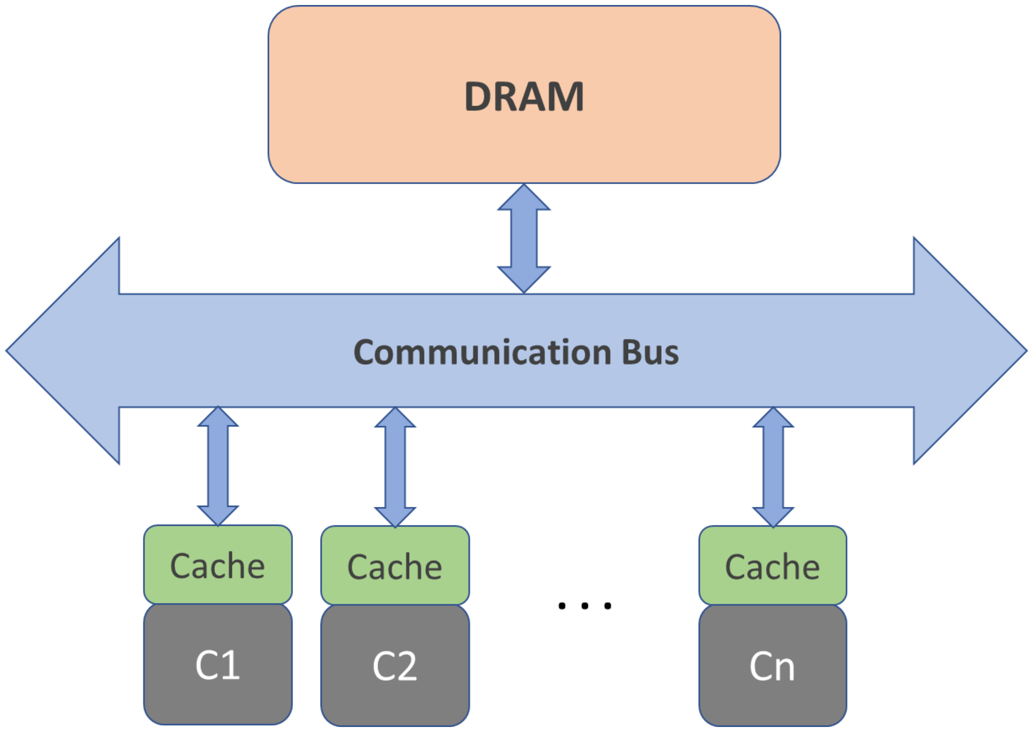
- Each Core has its own L1 Cache
- In some processors, each core has its own L2 Cache
- All cores share one L3 cache
- All cores share one DDR3 DRAM (main memory)

Image source https://www.realworldtech.com/sandy-bridge/2/

Simplified Multicore Architecture



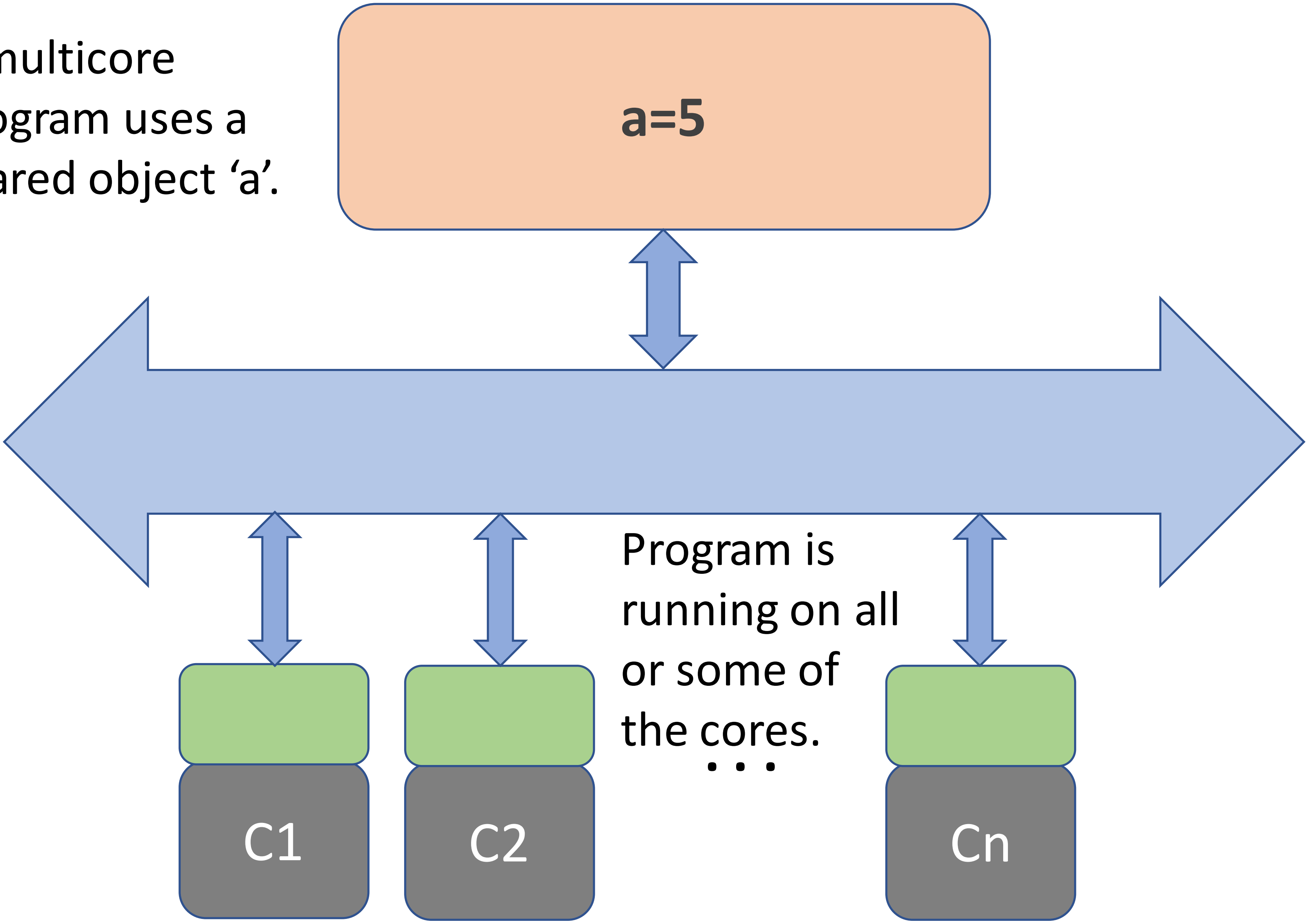
Coherence of data



- ‘Coherence’ is the quality of being logical and consistent. During program execution, data must remain coherent.
- On a multicore system, a data variable may reside in multiple caches and might get updated ‘locally’ by its local core → this causes coherence problem

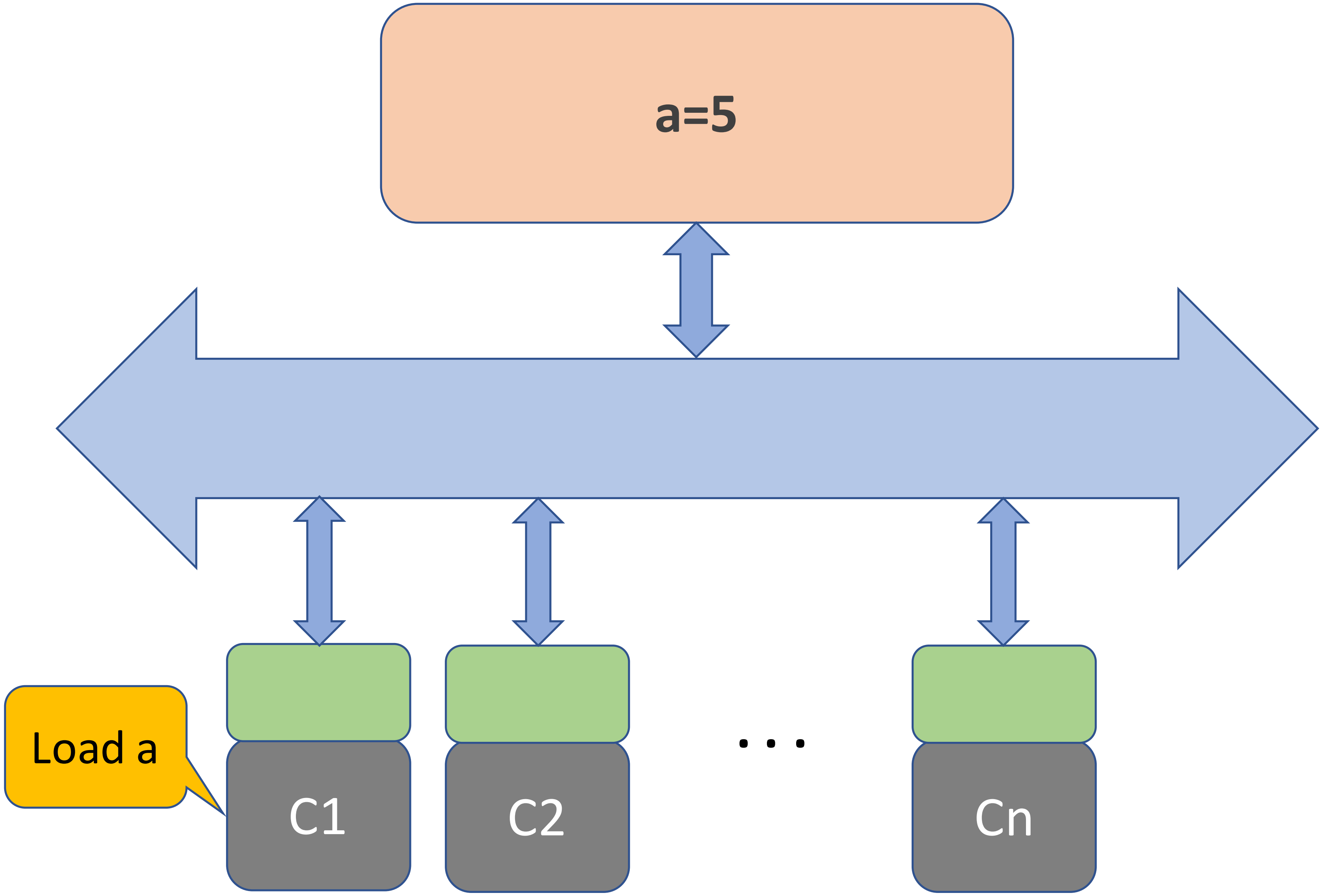
Example: Cache coherence problem

A multicore program uses a shared object ‘a’.

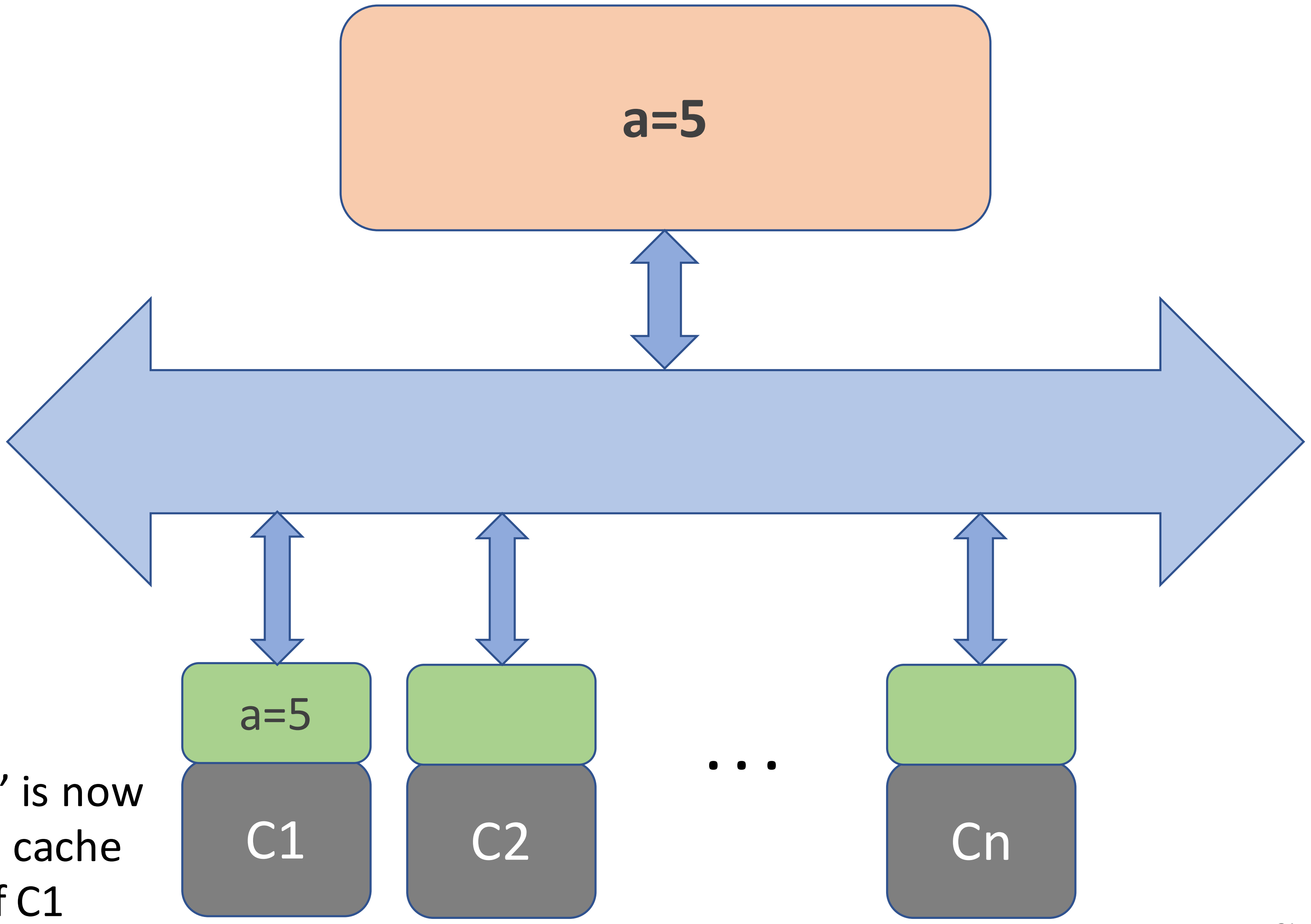


Program is running on all or some of the cores.

Example: Cache coherence problem

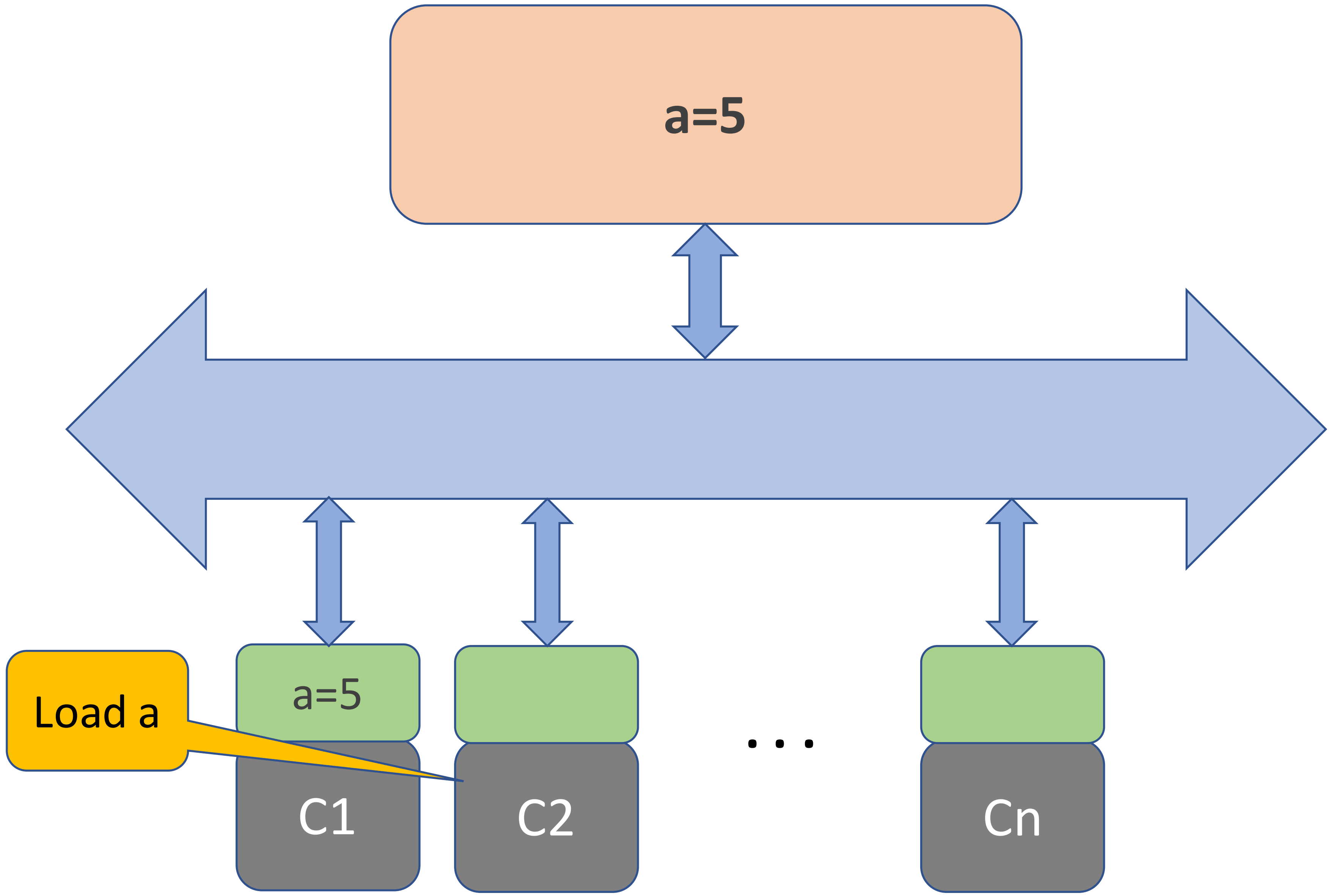


Example: Cache coherence problem

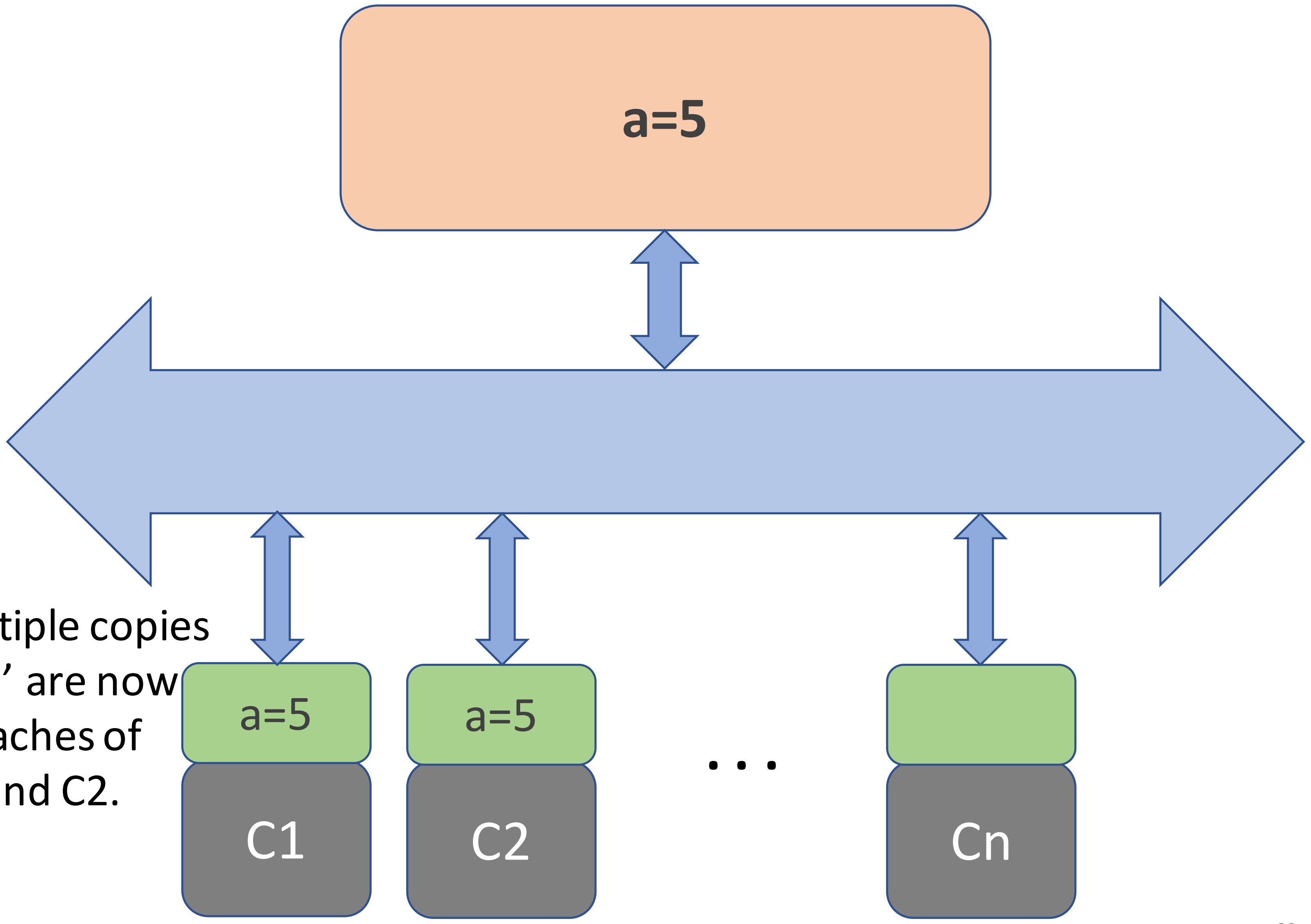


'a' is now in cache of C1

Example: Cache coherence problem

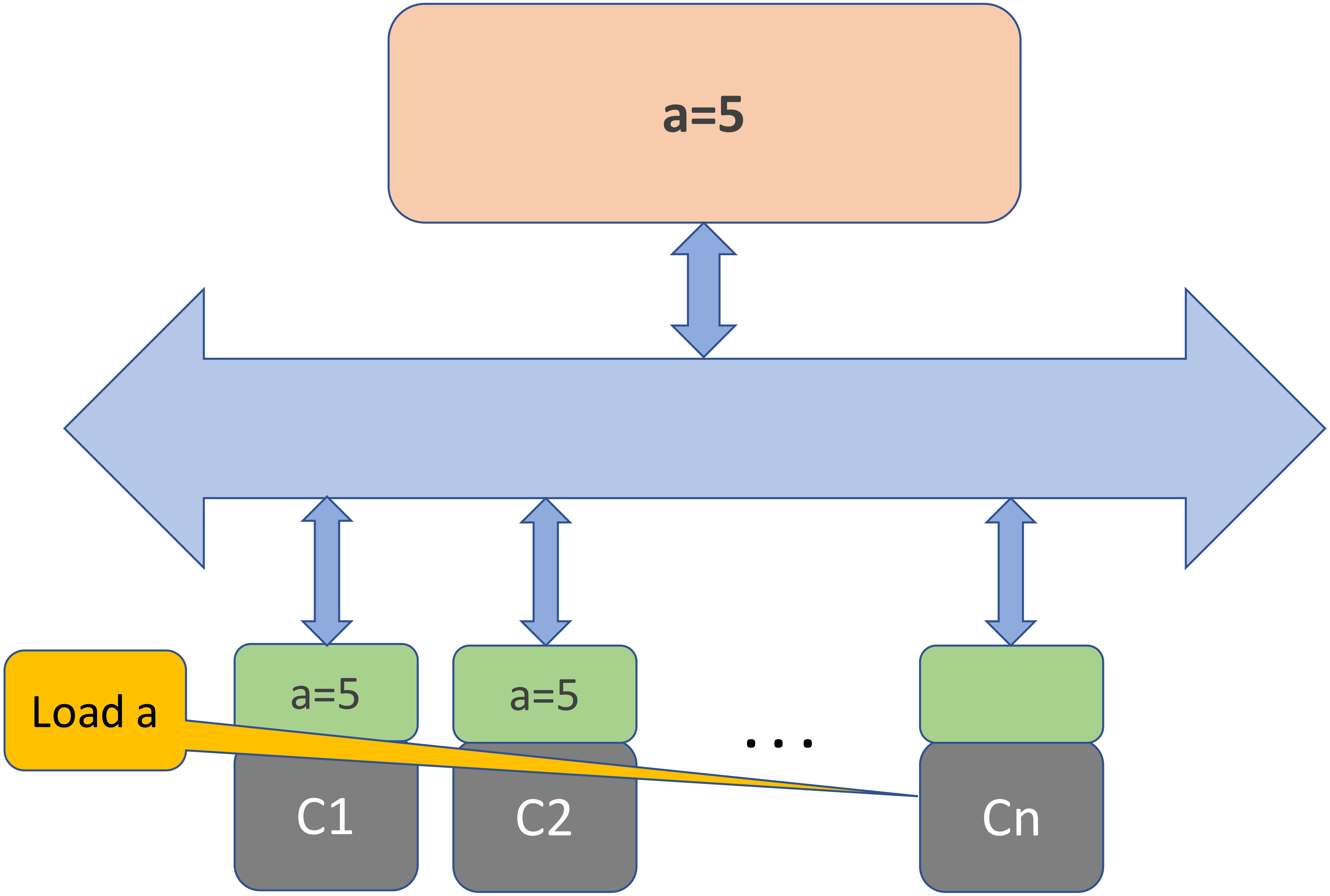


Example: Cache coherence problem

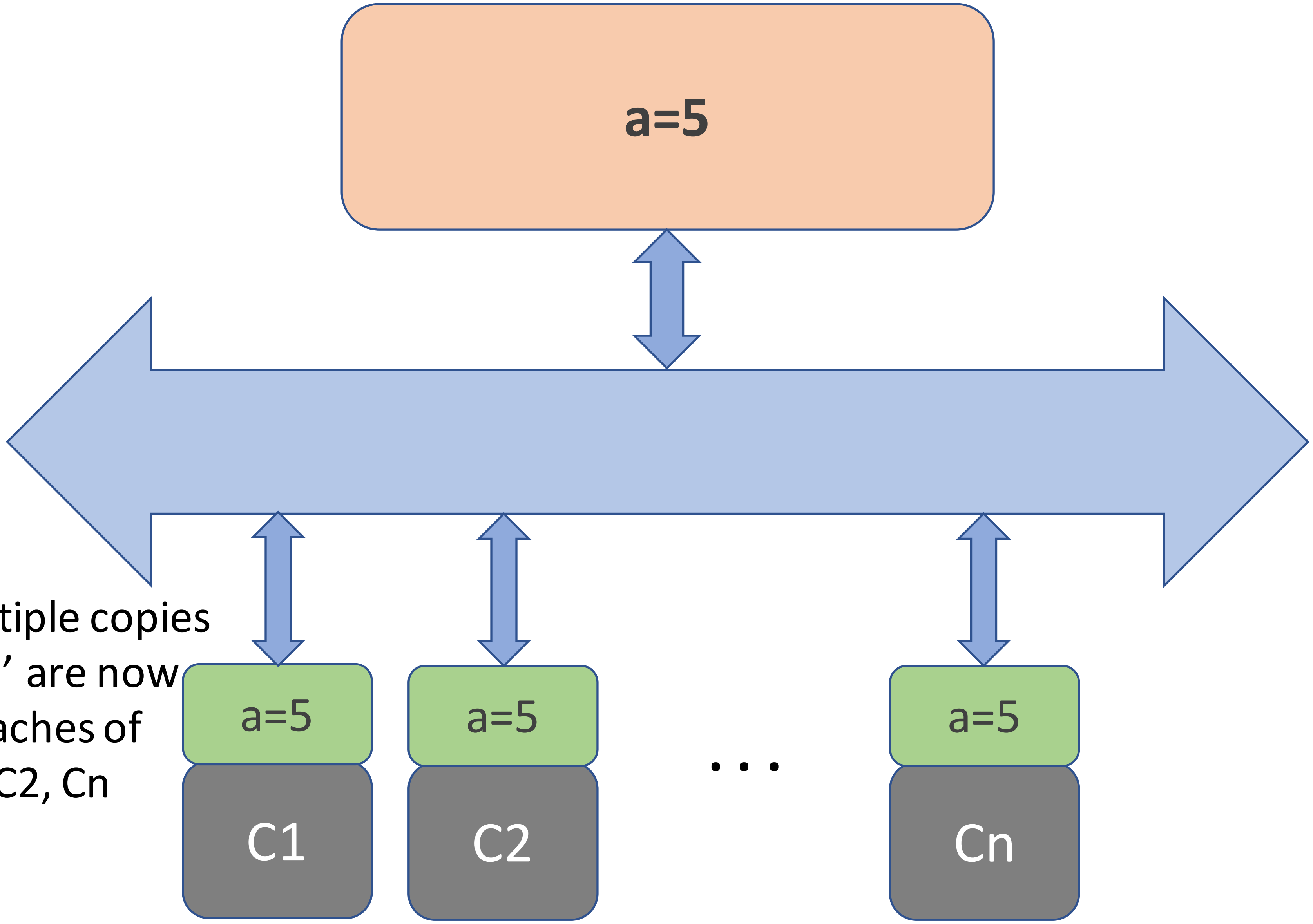


Multiple copies of ‘a’ are now in caches of C1 and C2.

Example: Cache coherence problem

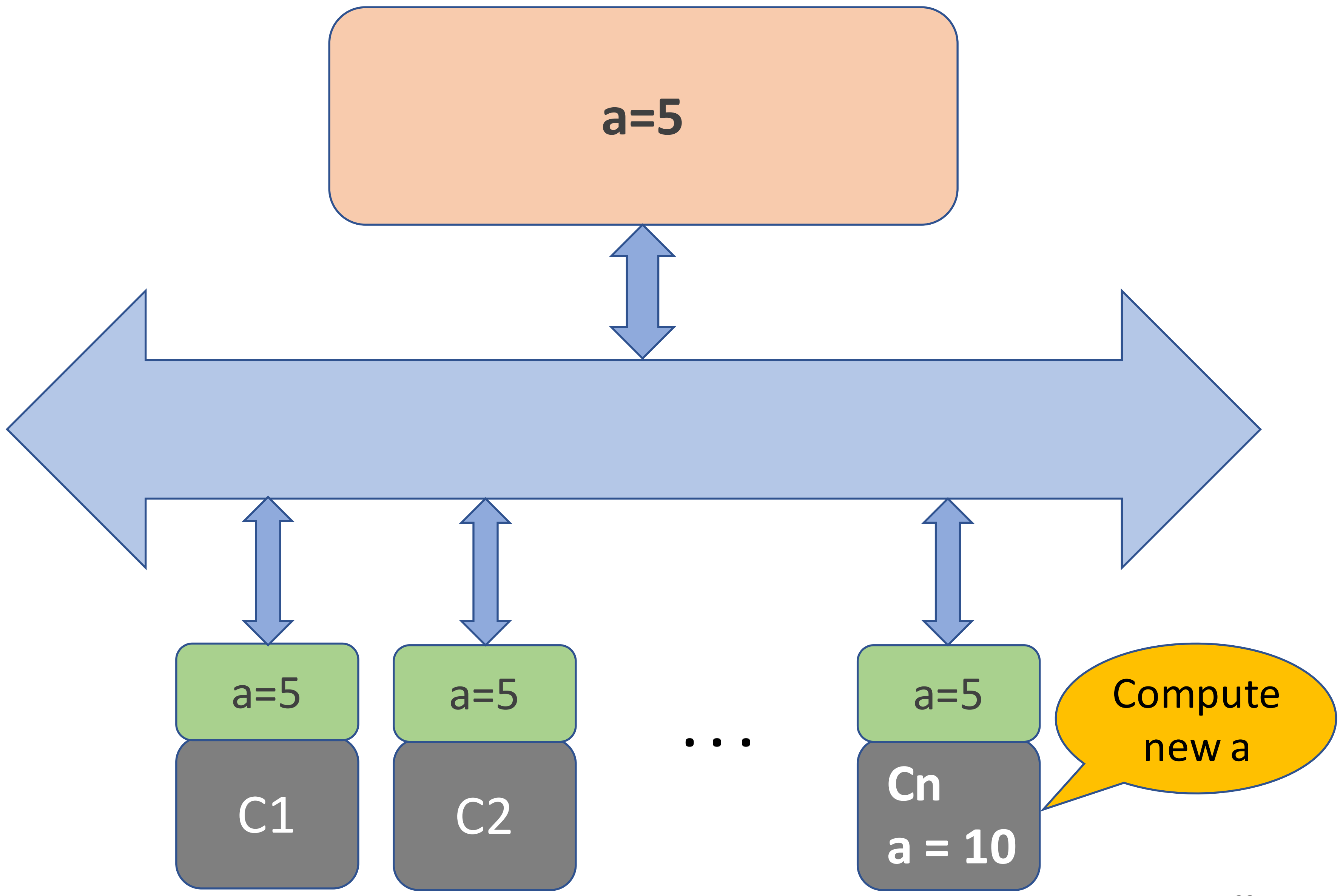


Example: Cache coherence problem

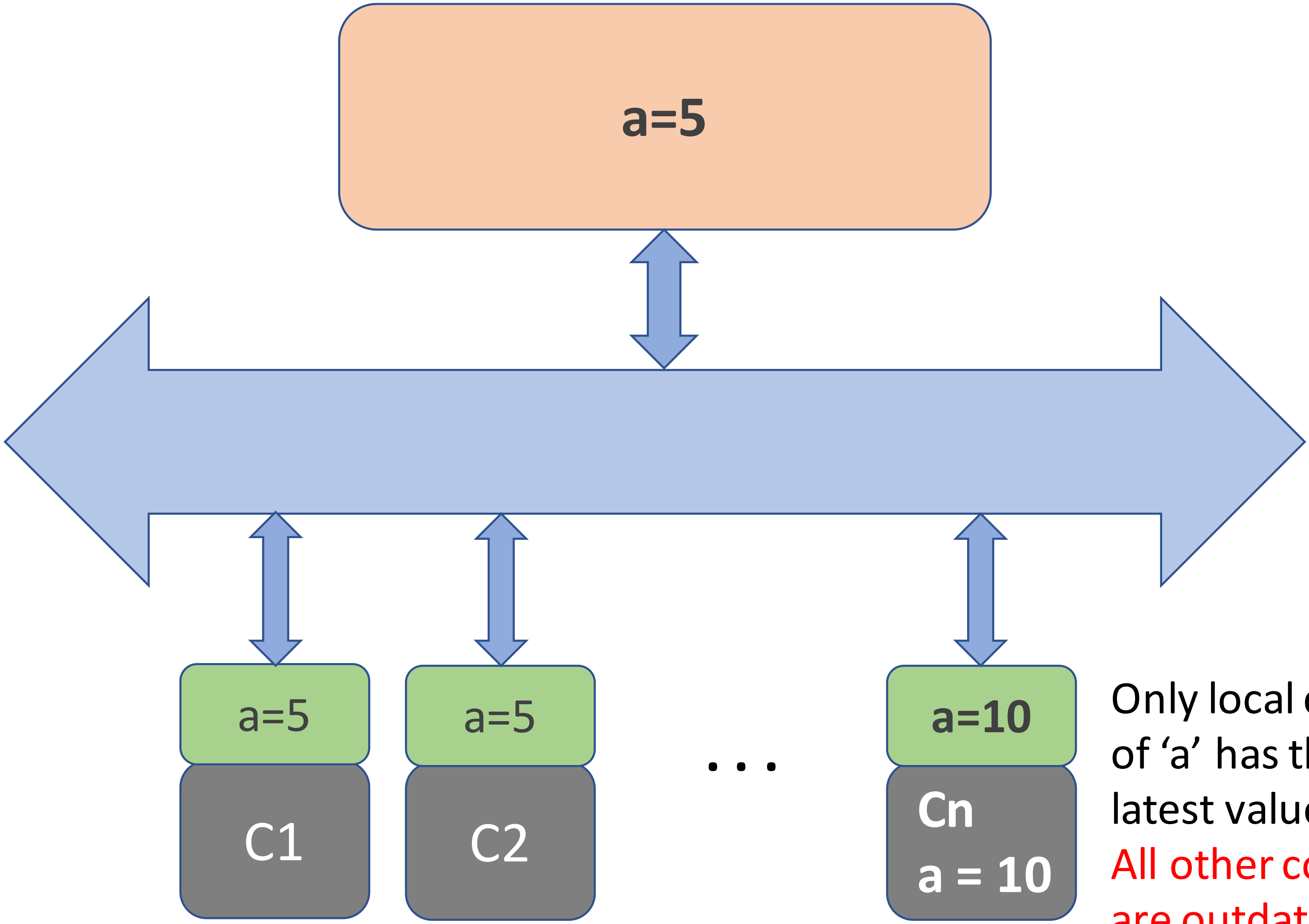


Multiple copies of ‘a’ are now in caches of C1, C2, Cn

Example: Cache coherence problem

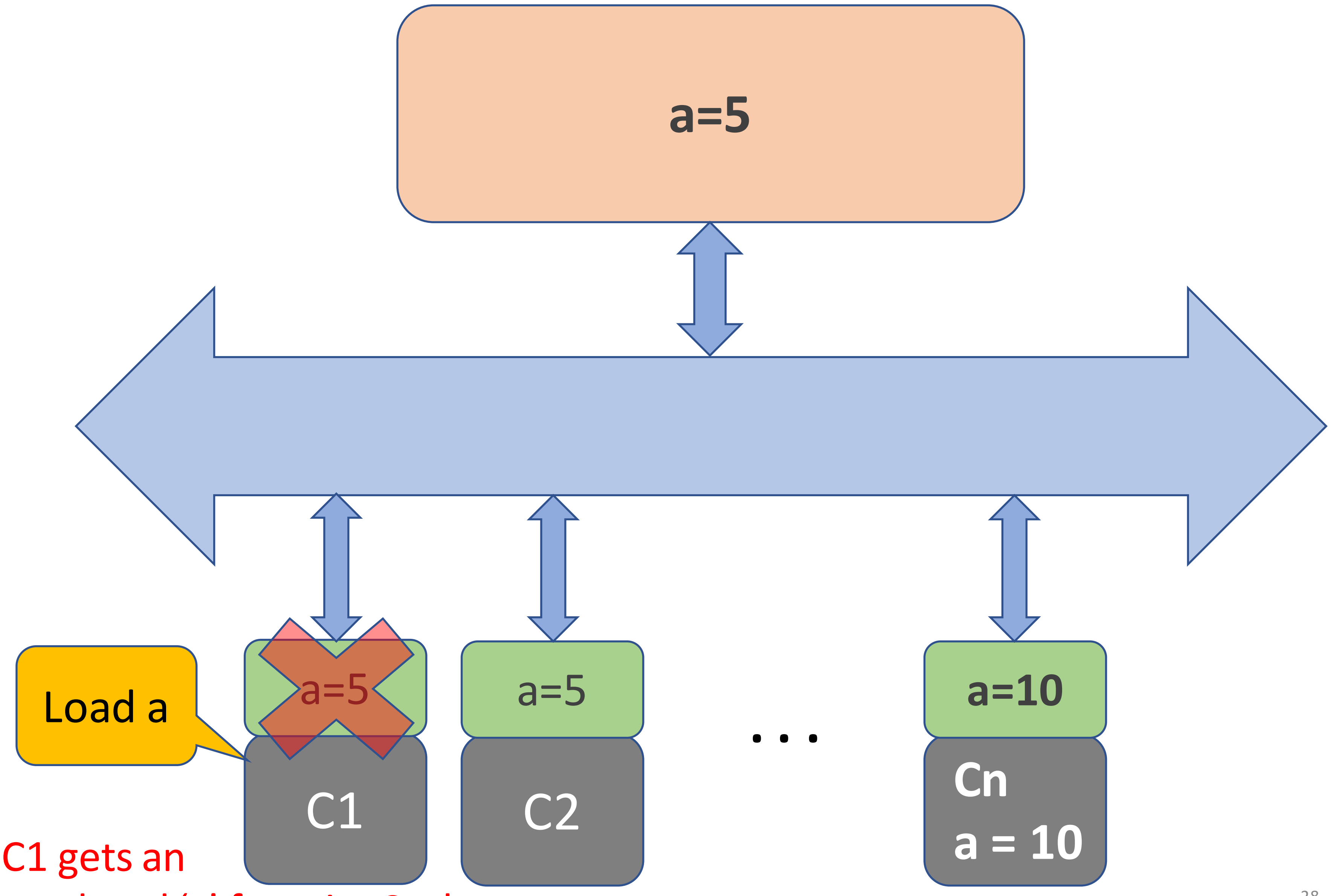


Example: Cache coherence problem



Only local copy of 'a' has the latest value. All other copies are outdated.

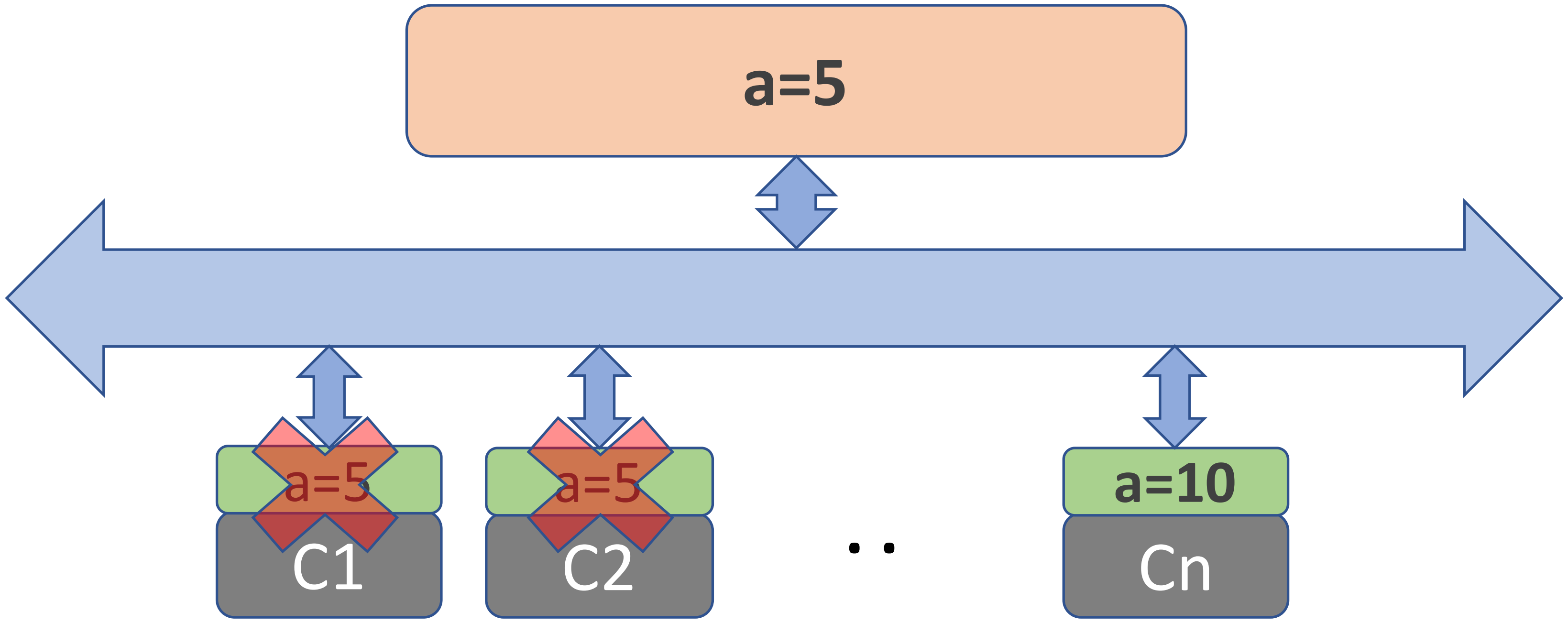
Example: Cache coherence problem



C1 gets an outdated ‘a’ from its Cache.

Coherence mechanisms

To achieve coherence:
a **write to a memory location** must cause all other copies of this location to be **removed from the caches** they are in.



Do: As soon as ‘a’ gets updated by a core, invalidate or delete all other copies of it.

Coherence protocols

- Coherence protocols apply cache coherence in multicore systems.
- Goal is that two cores **must never see different values** for the same shared data.

How this is achieved?

Additional information is stored and passed

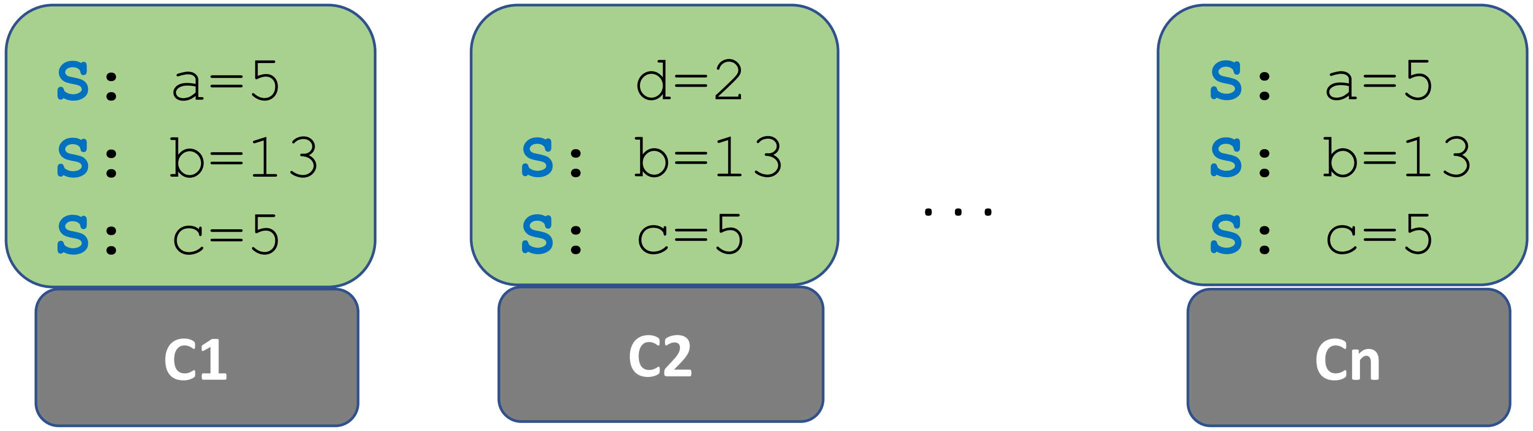
- Is this data stored in multiple caches?
- Has this cache line been modified?

MSI protocols

MSI protocol is a simple cache coherence protocol.
In this protocol, each cache line is labeled with a state:

- `M`: cache block has been modified.
- `S`: other caches may be sharing this block.
- `I`: cache block is invalid

Initial state of caches before computation

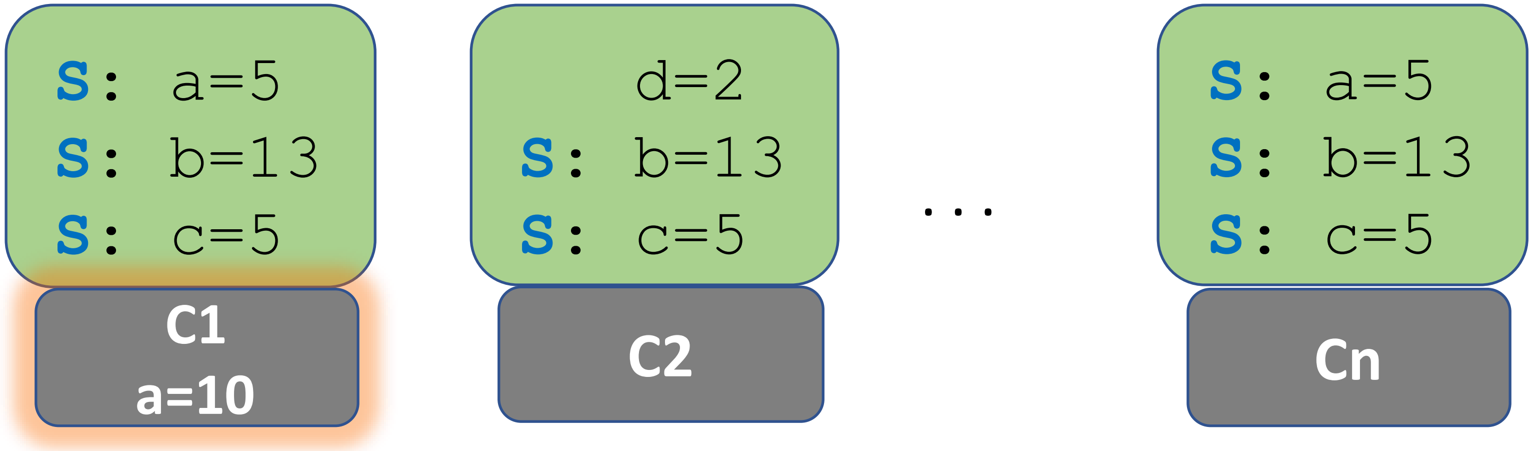


MSI protocols

MSI protocol is a simple cache coherence protocol.
In this protocol, each cache line is labeled with a state:

- **M**: cache block has been modified.
- **S**: other caches may be sharing this block.
- **I**: cache block is invalid

C1 computes new value a=10 in register

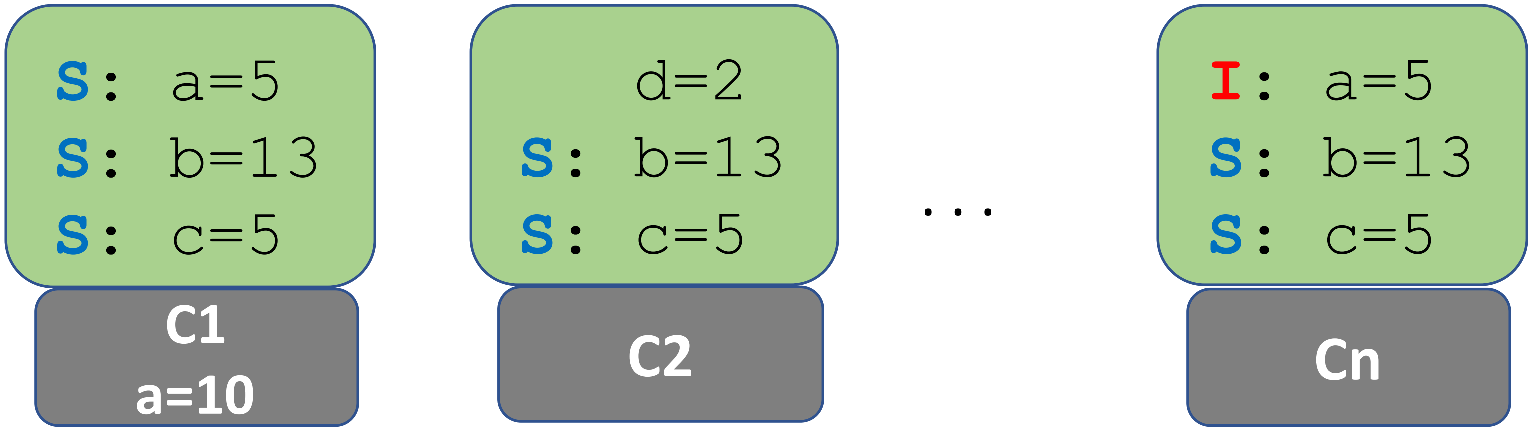


MSI protocols

MSI protocol is a simple cache coherence protocol.
In this protocol, each cache line is labeled with a state:

- `M`: cache block has been modified.
- `S`: other caches may be sharing this block.
- `I`: cache block is invalid

Hardware invalidates all other copies of a

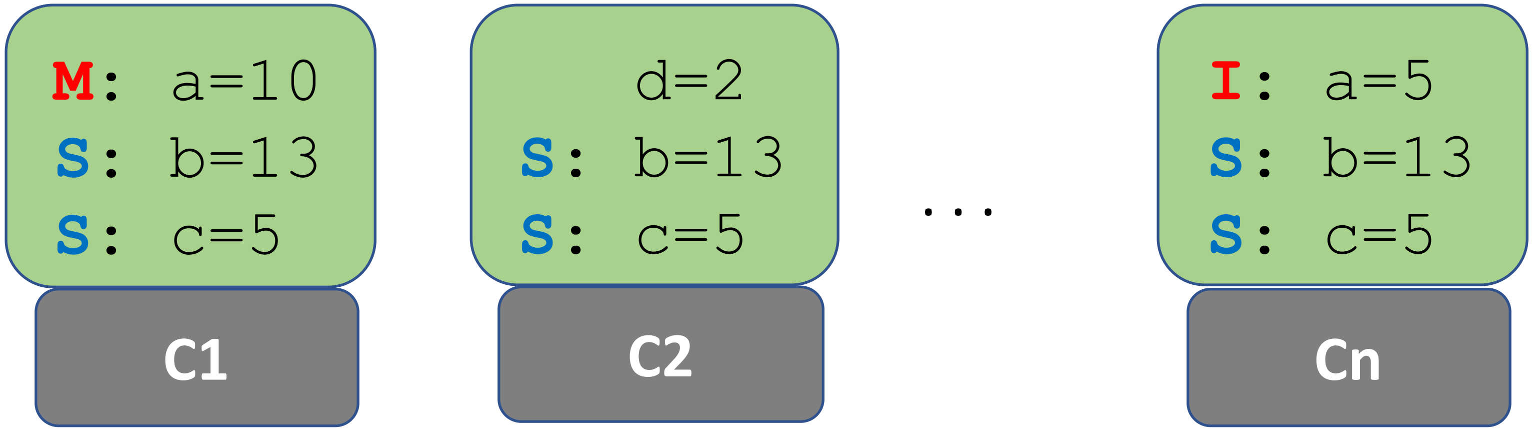


MSI protocols

MSI protocol is a simple cache coherence protocol.
In this protocol, each cache line is labeled with a state:

- **`M`**: cache block has been modified.
- **`S`**: other caches may be sharing this block.
- **`I`**: cache block is invalid

Hardware updates a in cache of C1 and marks it with 'M'

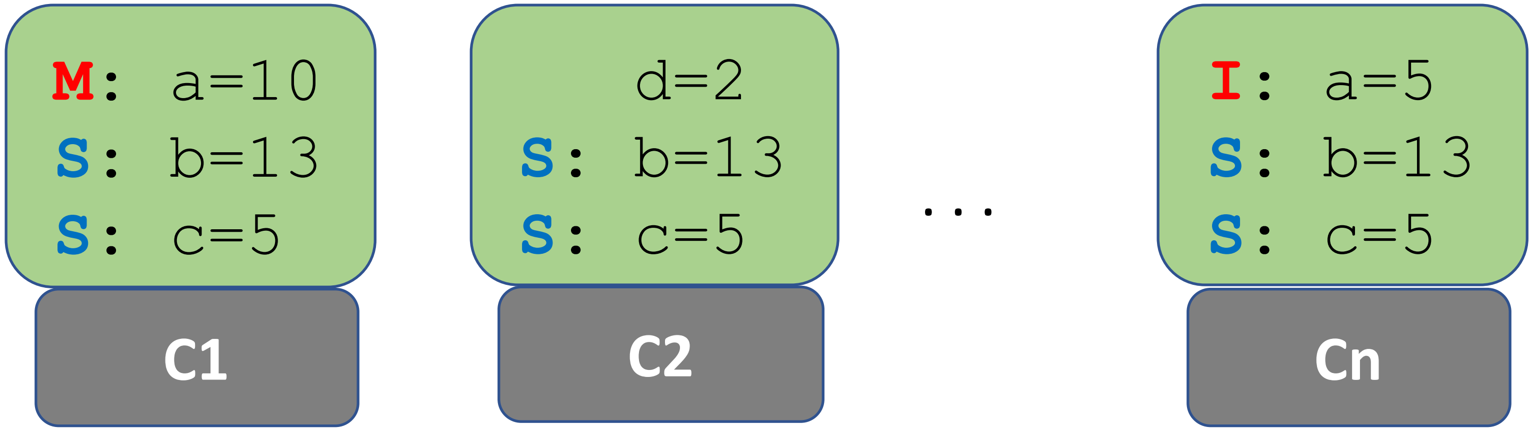


MSI protocols

MSI protocol is a simple cache coherence protocol.
In this protocol, each cache line is labeled with a state:

- **M**: cache block has been modified.
- **S**: other caches may be sharing this block.
- **I**: cache block is invalid

Hardware also updates 'a' in the shared memory

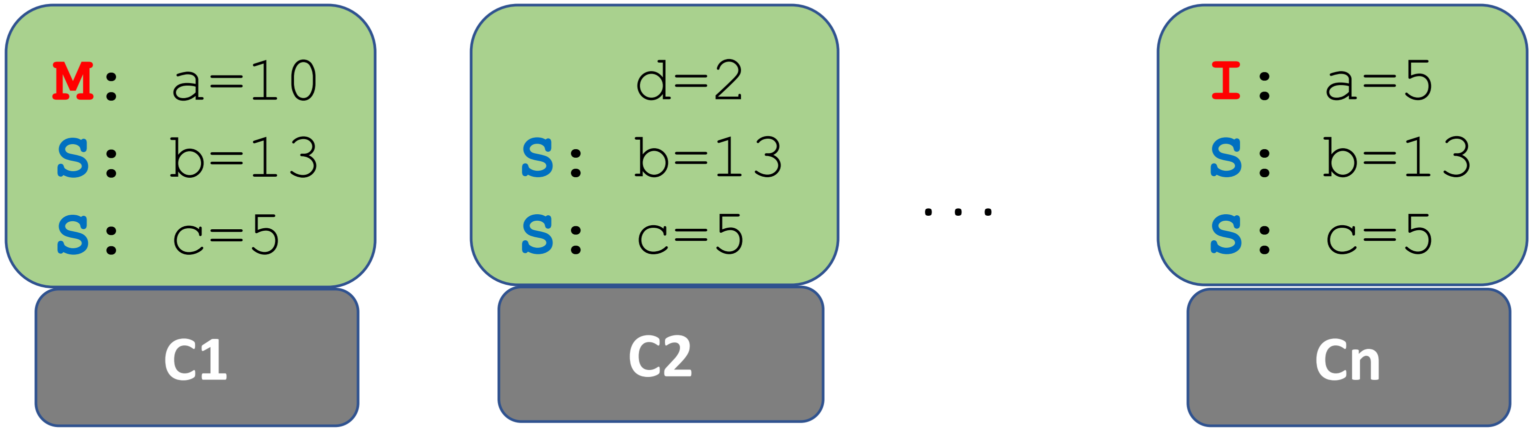


MSI protocols

MSI protocol is a simple cache coherence protocol.
In this protocol, each cache line is labeled with a state:

- **M**: cache block has been modified.
- **S**: other caches may be sharing this block.
- **I**: cache block is invalid

If Cn wants to load 'a' then 'a' will be brought from the shared memory to the local cache of Cn.



Programming multicore platforms

- Option 1: Program directly targeting processor cores
  - ➢ Programmer takes care of synchronization
  - ➢ Painful and error-prone

- Option 2: Use a *concurrency platform*
  - ➢ It abstracts processor cores, handles synchronization and communication protocols, and performs load balancing.
  - ➢ Hence offers much easier multicore programming environment
  - ➢ Examples:
    - **Pthreads** and WinAPI threads
    - OpenMP