Sequential Consistency and Transactional Memory

Arnamoy Bhattacharyya
Scalable Parallel Computing Laboratory

ETH Zurich

DPHPC 2014

Coherence vs Consistency:

Coherence guarantees:

1. Write propagation
2. Write serialization

Consistency the ordering of writes and reads to DIFFERENT memory locations.

Every hardware guarantees a certain consistency model

P1	P2
Instr a	Instr A
Instr b	Instr B
Instr c	Instr C
Instr d	Instr D

We assume:

1. Within a thread, the program order is preserved
2. Each instructions executes atomically
3. Instructions from different threads can be interleaved arbitrarily

Valid Executions:

AbAcBCDd..... or ABCDabcd or aAbBcCdD

Programmers assume SC, makes it easier to reason about program behaviors

Hardware innovations may disrupt the SC model

For example, if we assume write buffers or and out of order execution, or if we drop ACKS in the coherence protocol, the prev. programs may yield unexpected outputs.

Problem in a OOO processor

Intiially A = B =0

P1	P2
A ← 1	B ← 1
….	
If (B == 0) Crit section	if(A == 0) Crit Section

A load may proceed from Load-store queue even not at the top of Reorder buffer

One solution: Speculative load

Second Solution: Fences

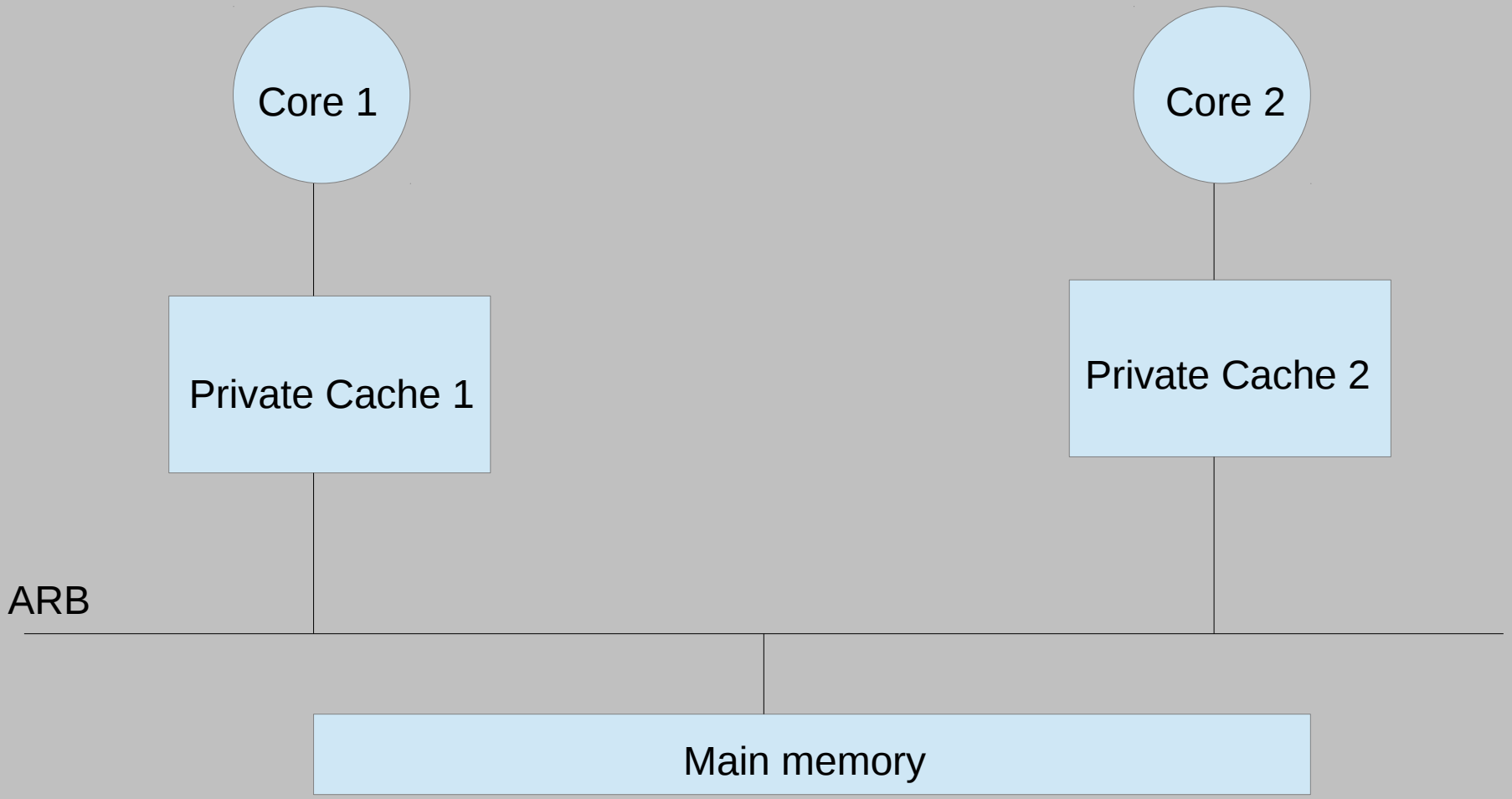
```
P1
{
        No race
}

Fence
        Acuire Lock
Fence

{
        Race
}

Fence
        Release lock
Fence
```

```
    P2
{
        No race
}

Fence
        Acuire Lock
Fence

{
        Race
}

Fence
        Release lock
Fence
```

Alternative of locks: Transactional Memory

Hardware extensions for Transactional Memory

1. Cache checkpointing
2. Read write bits



Cons of Transactional Memory and Solutions:

1. *Cache Overflow*

2. *Starvation*

Use an Arbitration (ARB), which acts as a token.

1. When a cache line is to be evicted, request token

2. When rolled back a threshold number of times, request token

Homework on Sequential Consistency

Check Website