Model Checking Cache Coherence in System-Level Code

Nicholas Allen and Yang Zhao
Oracle Labs Brisbane
June 2016

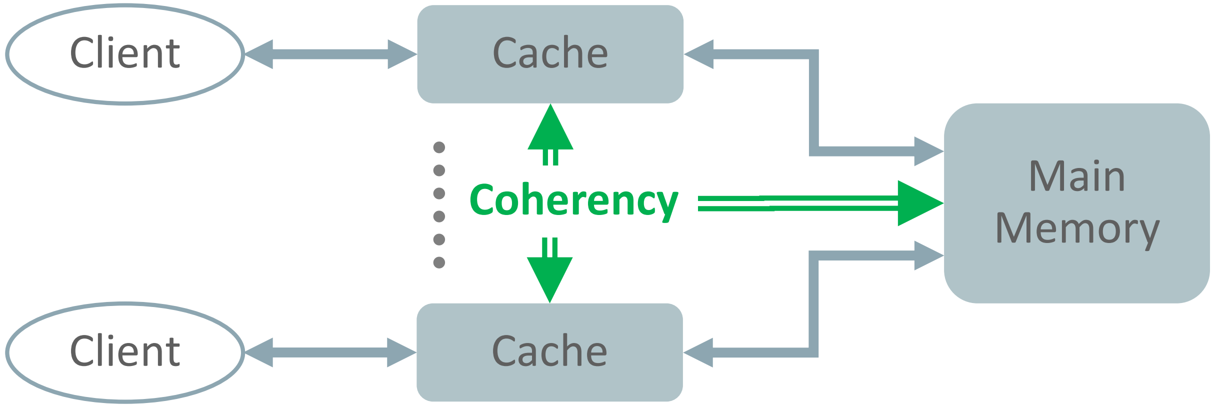
ORACLE

Program Agenda

ORACLE

Cache Coherence Issue

- A shared memory multiprocessor system with a separate cache for each processor.
- Multiple copies for the same data in both main memory and caches.
- Coherence defines the proper access behaviors to the same memory location.



Cache Coherence Issue

- No cache coherence guarantee in hardware level.
- Must be dealt with explicitly in software by programmers.
  - Writing back data from the cache into the memory if the cacheline is “dirty”.

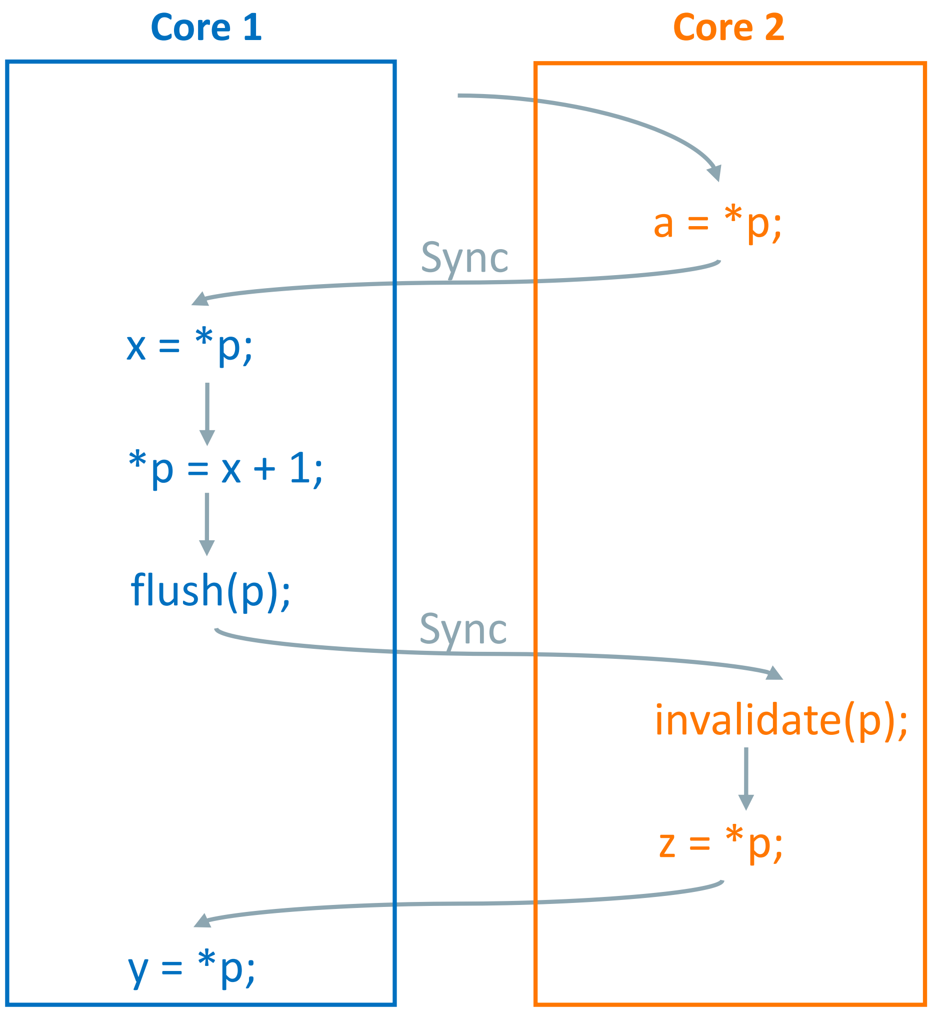
    flush(void addr, size_t size)*

  - Invalidating a cacheline in a cache such that the next load has to fetch data from memory.

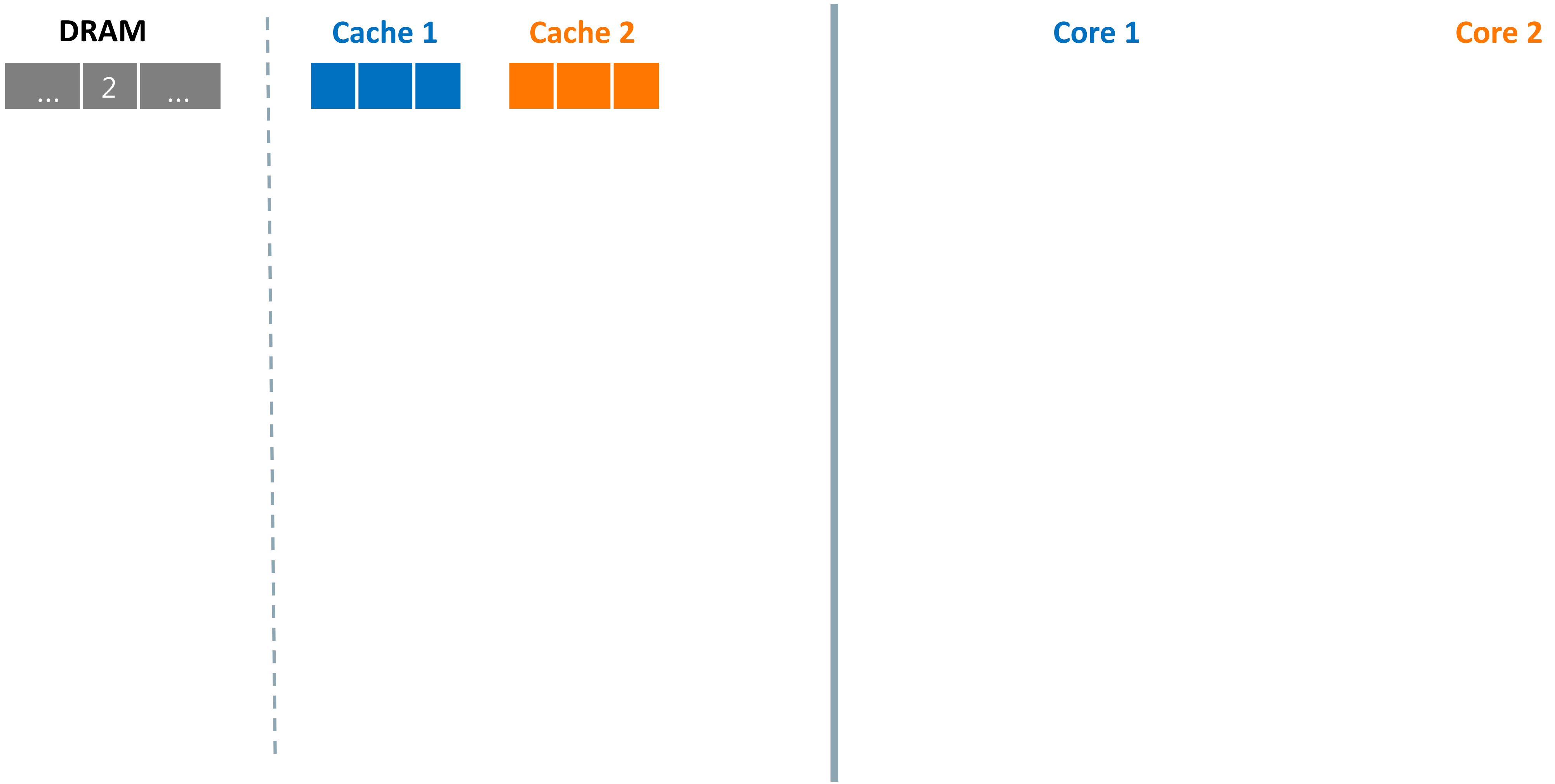
    Invalidate(void addr, size_t size)*

ORACLE

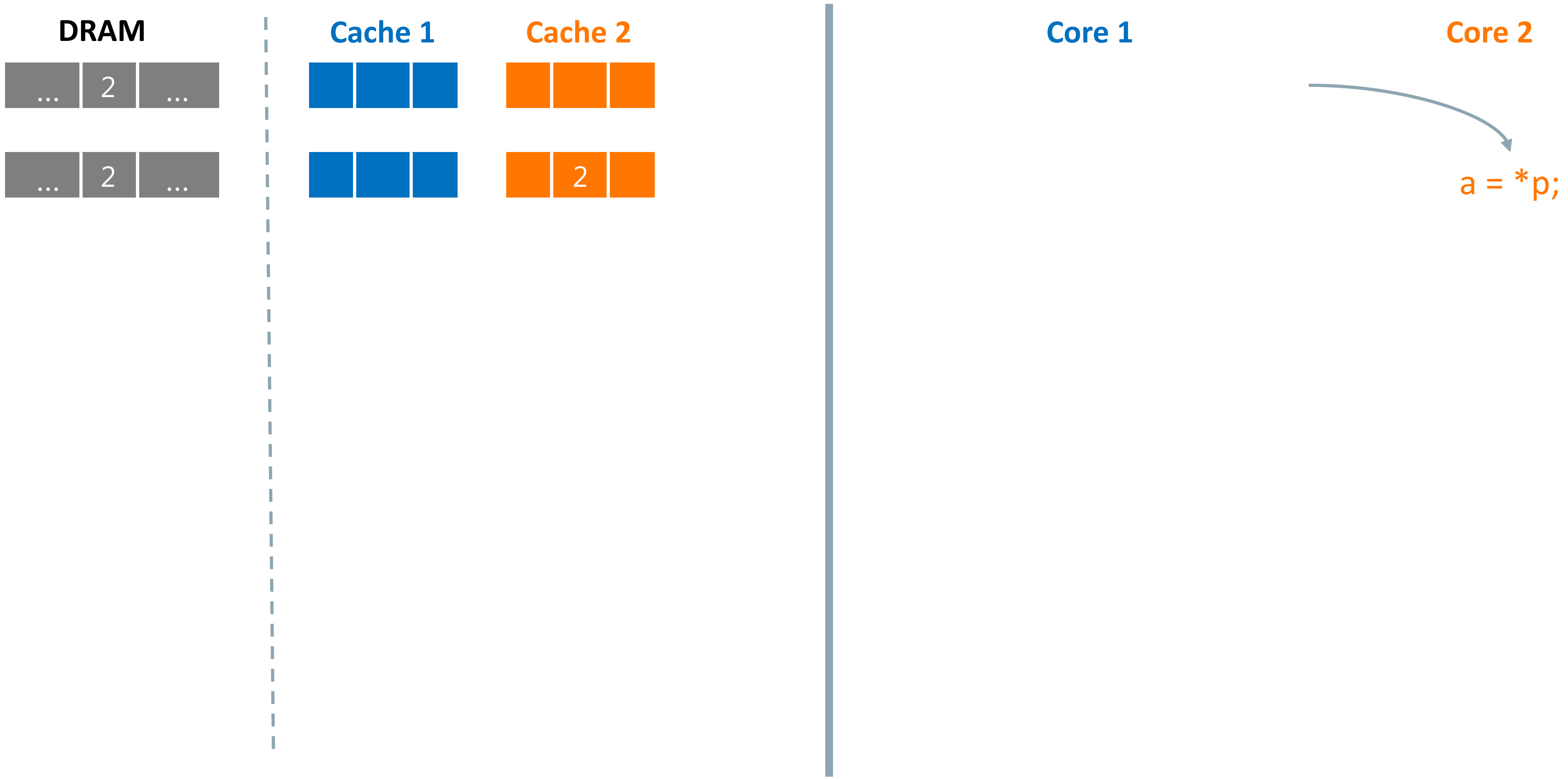
Cache Coherence Issue



Cache Coherence Issue

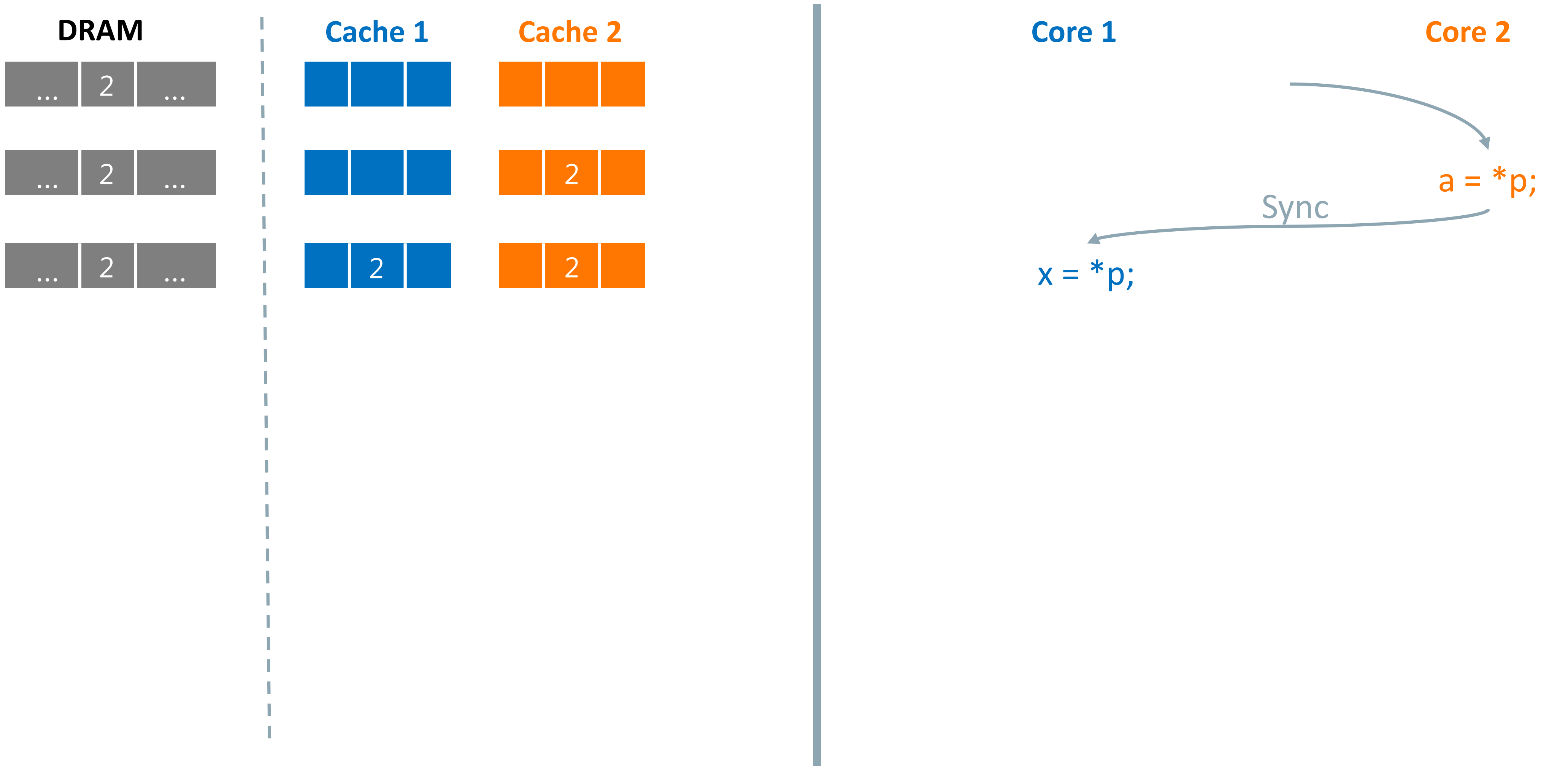


Cache Coherence Issue



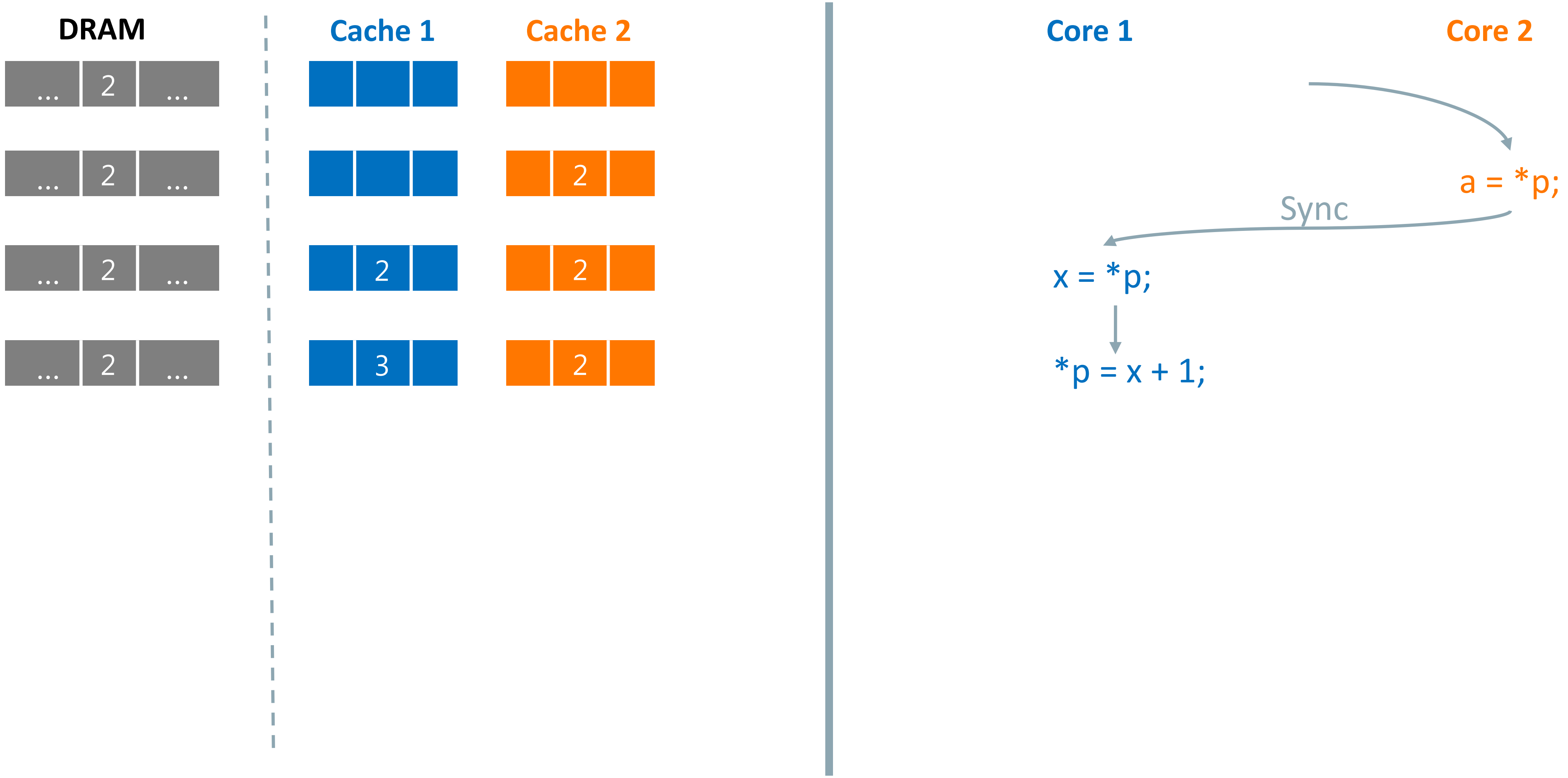
ORACLE

Cache Coherence Issue

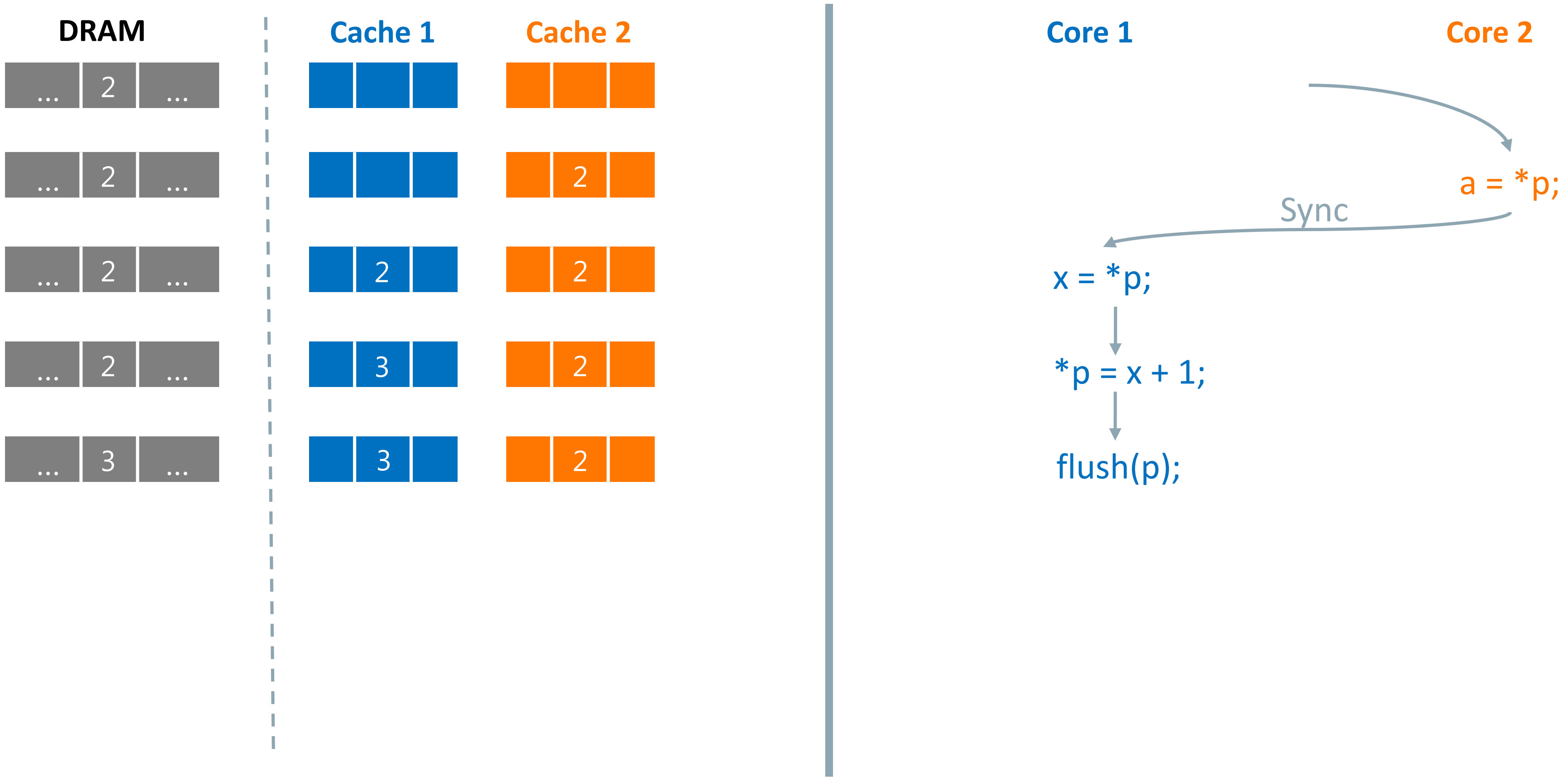


ORACLE

Cache Coherence Issue

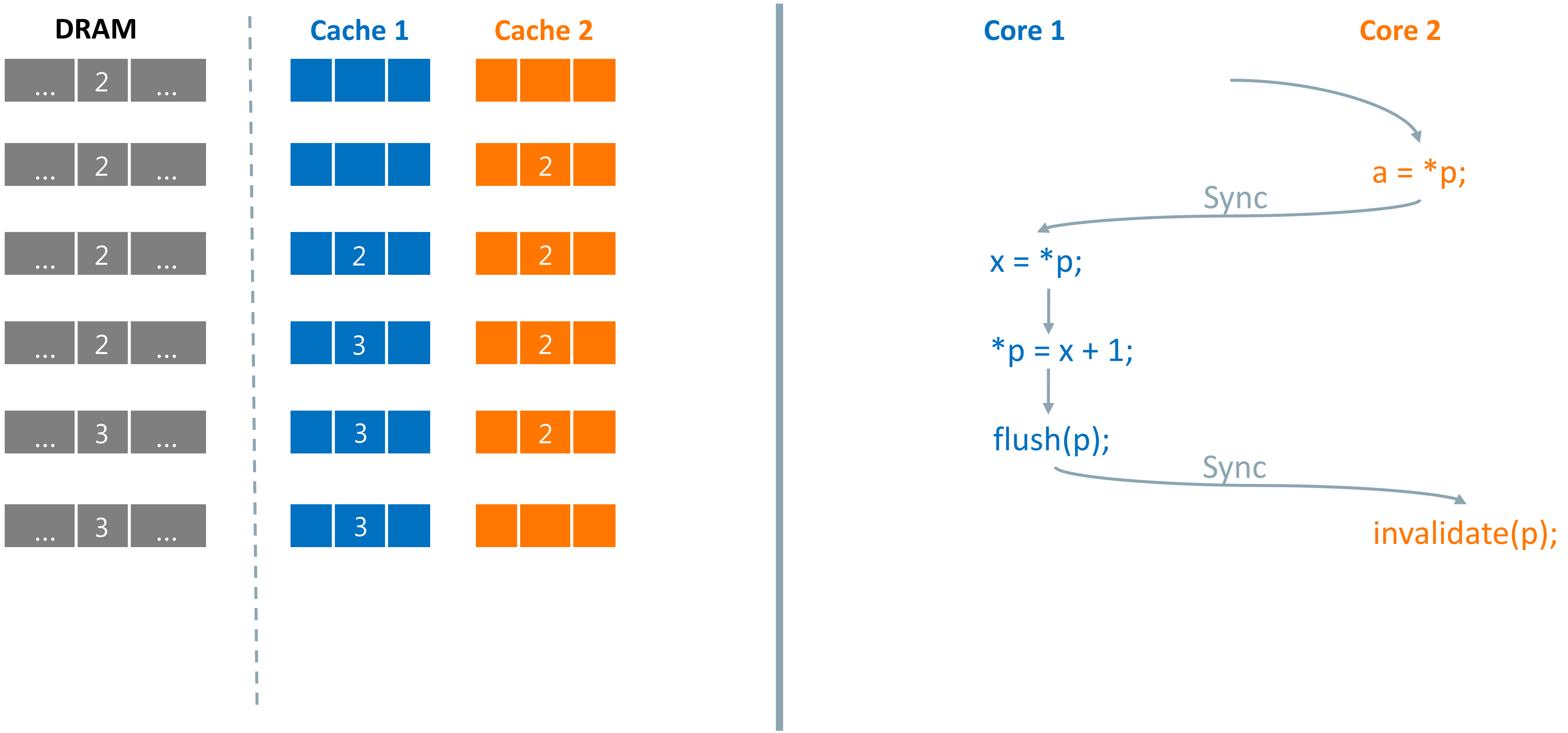


Cache Coherence Issue



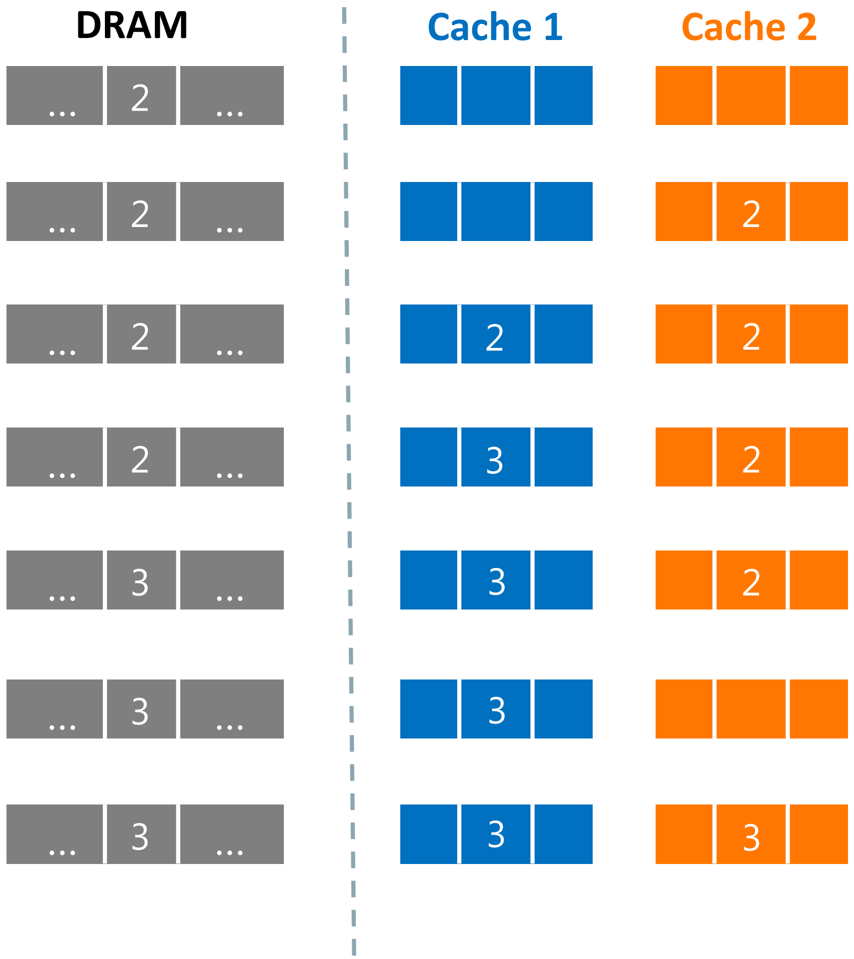
ORACLE

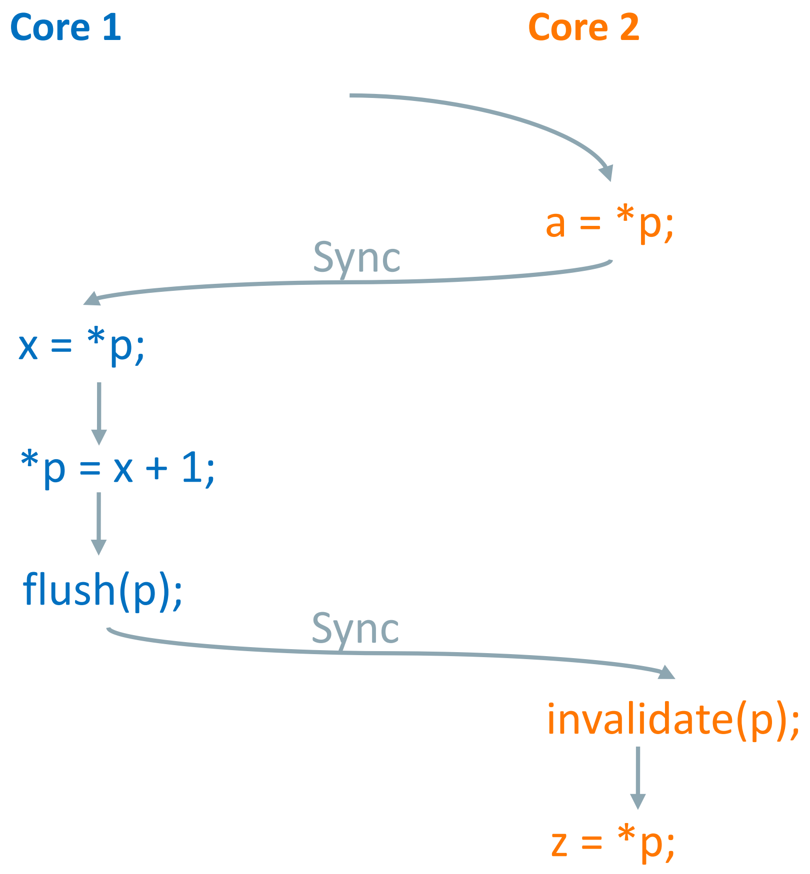
Cache Coherence Issue



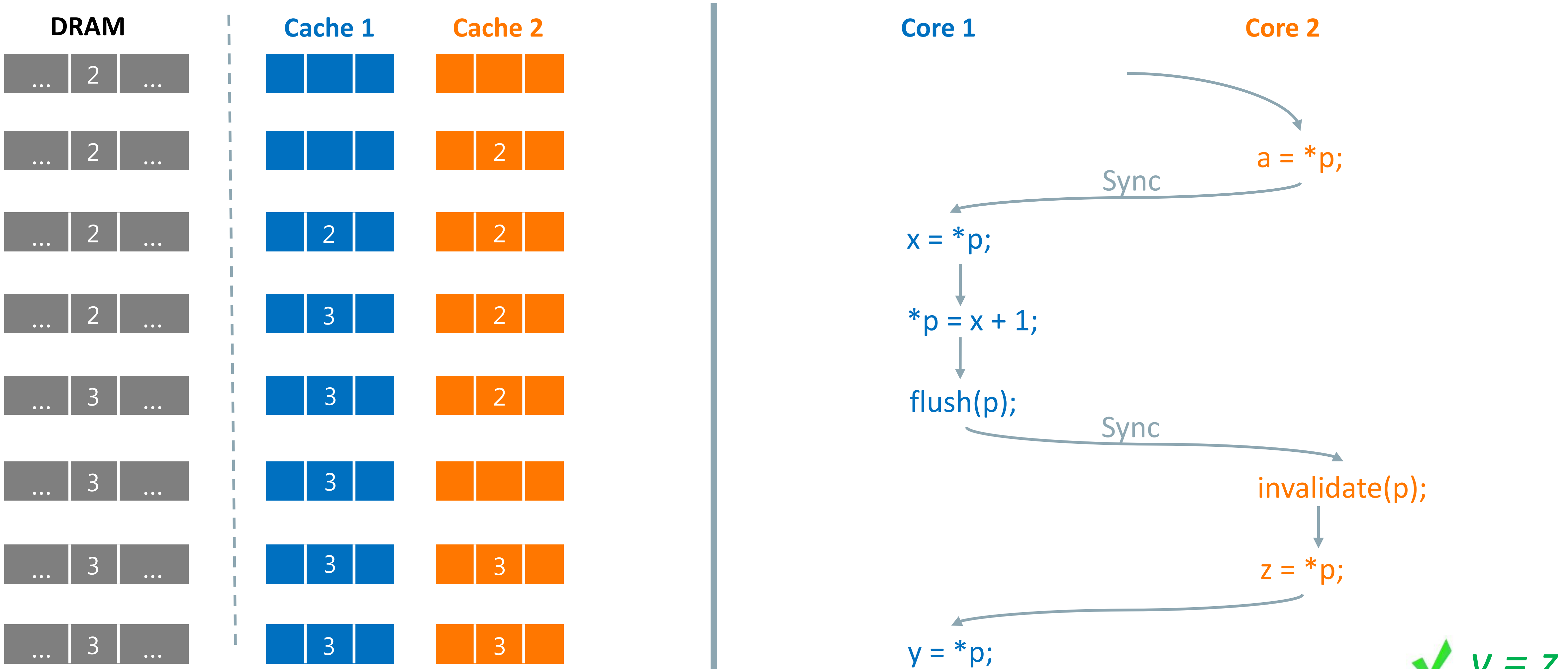
ORACLE

Cache Coherence Issue



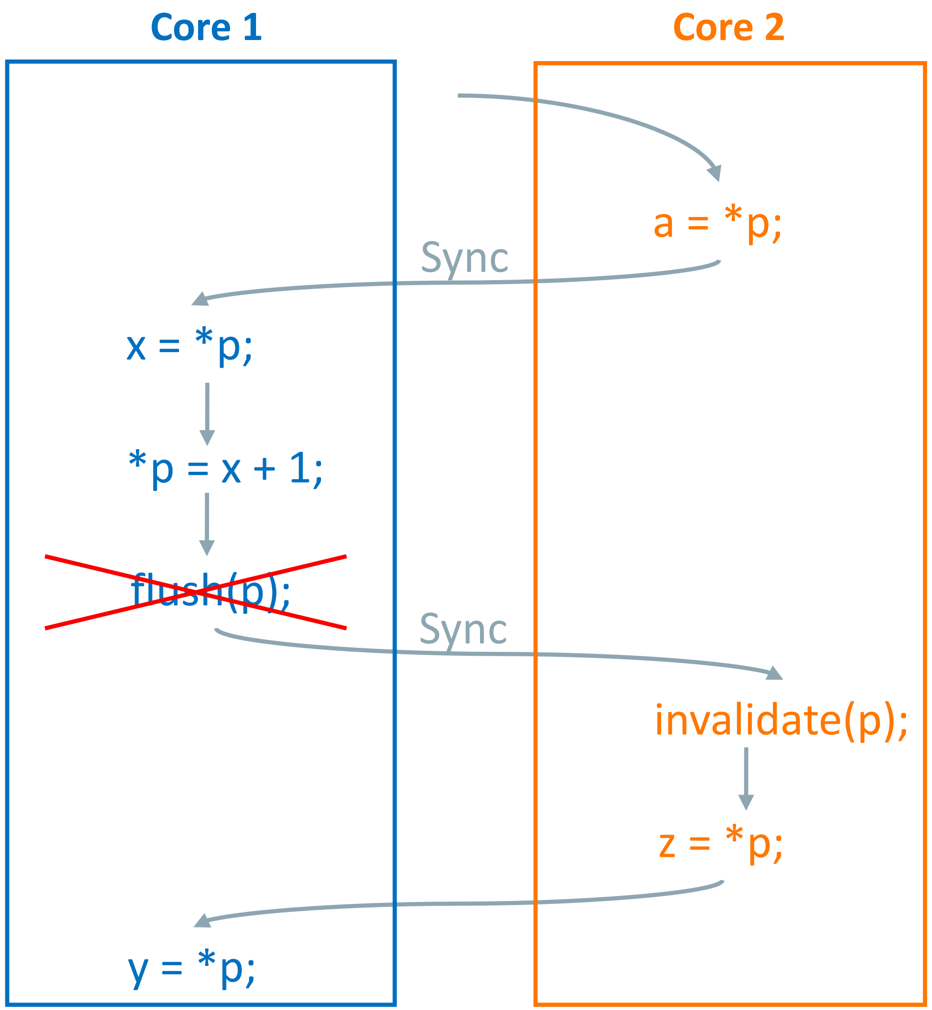


Cache Coherence Issue

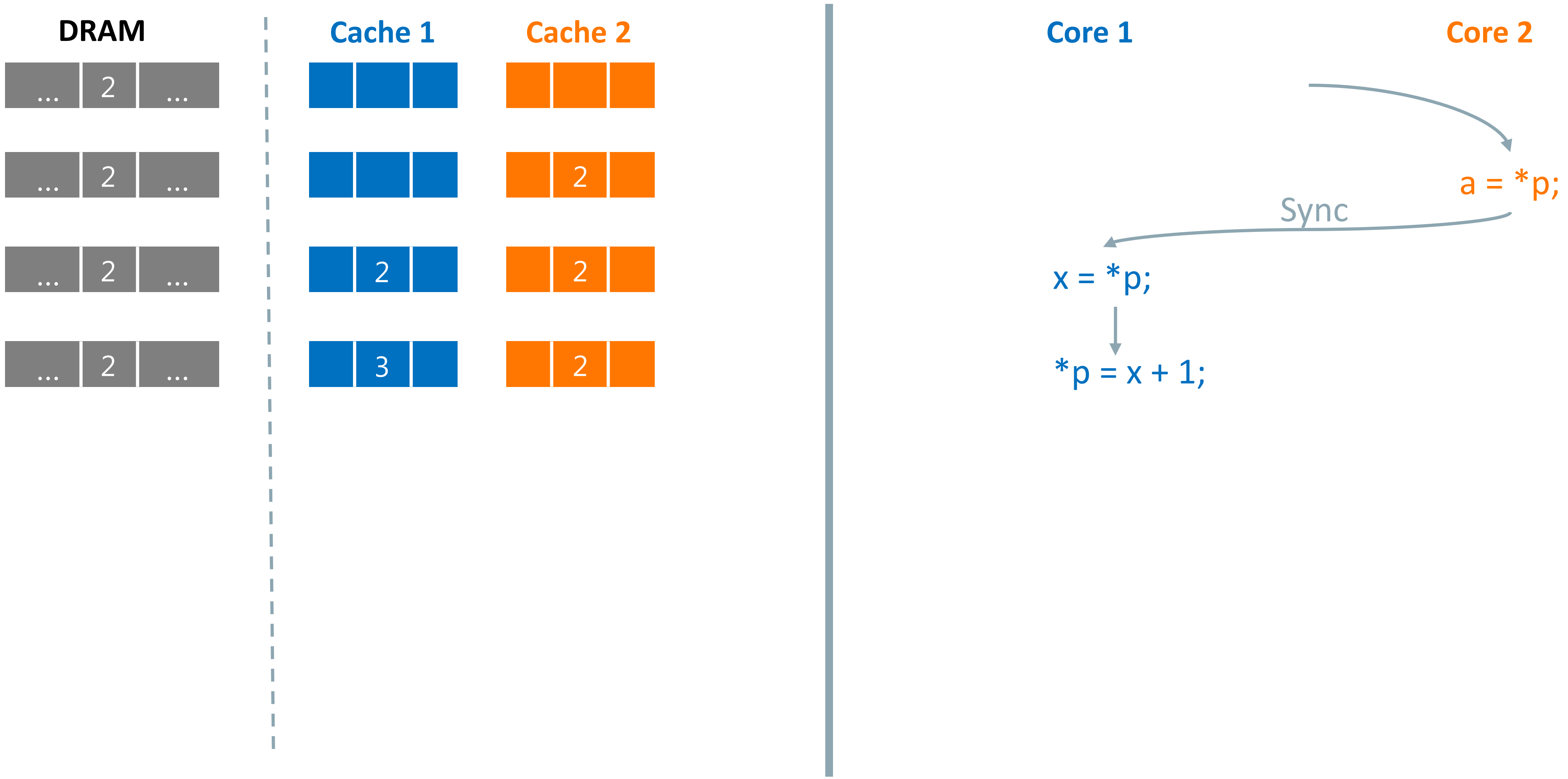


ORACLE

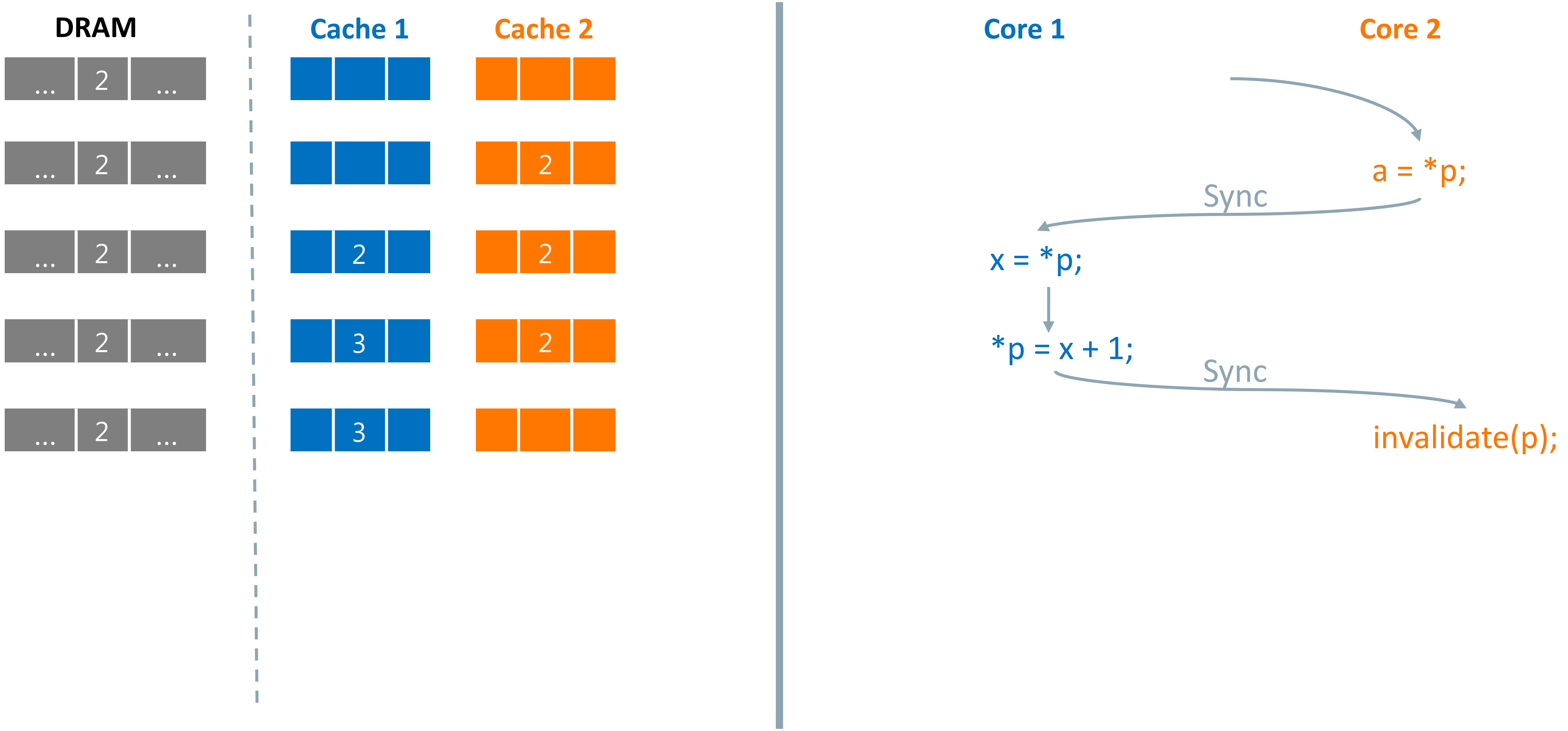
Cache Coherence Issue



Cache Coherence Issue

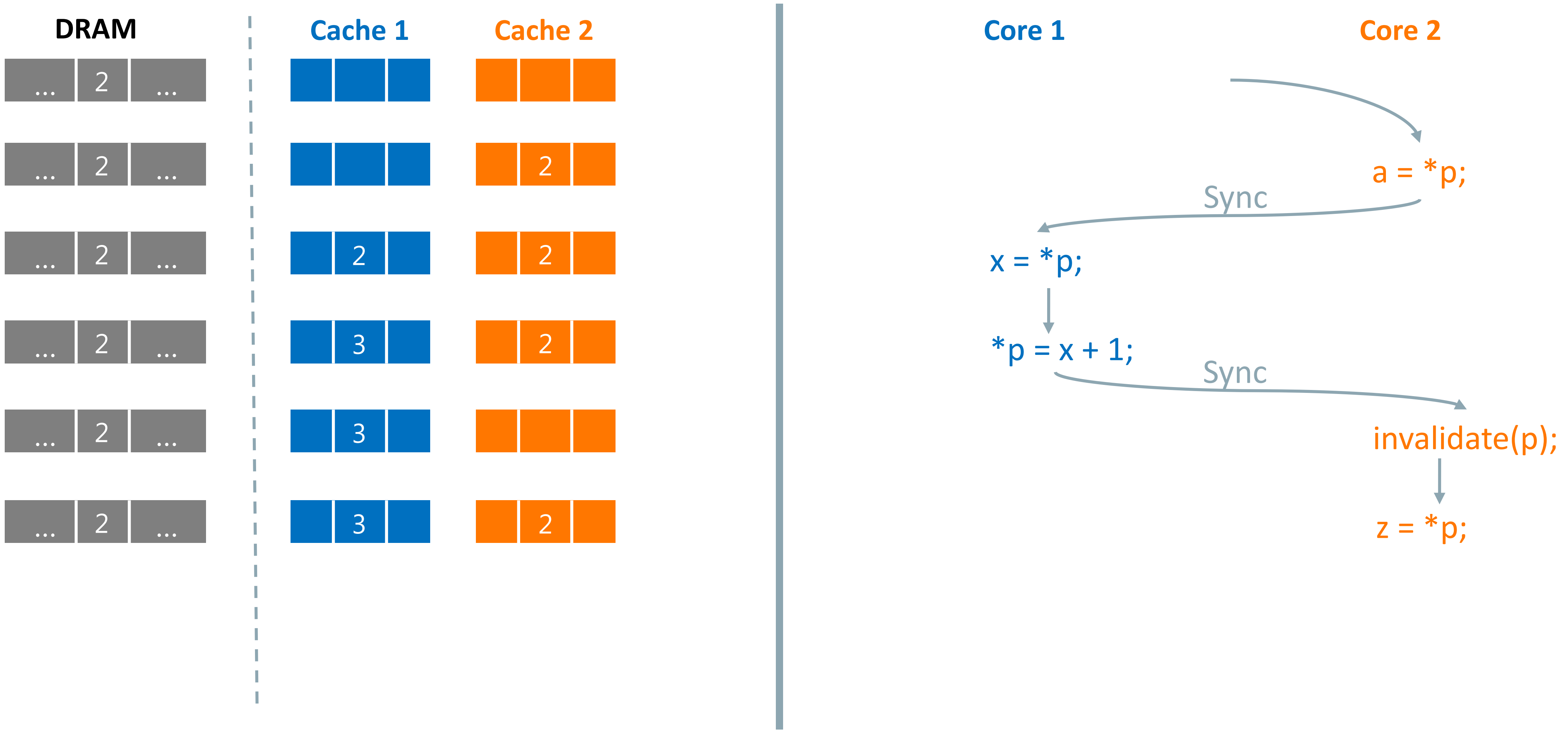


Cache Coherence Issue

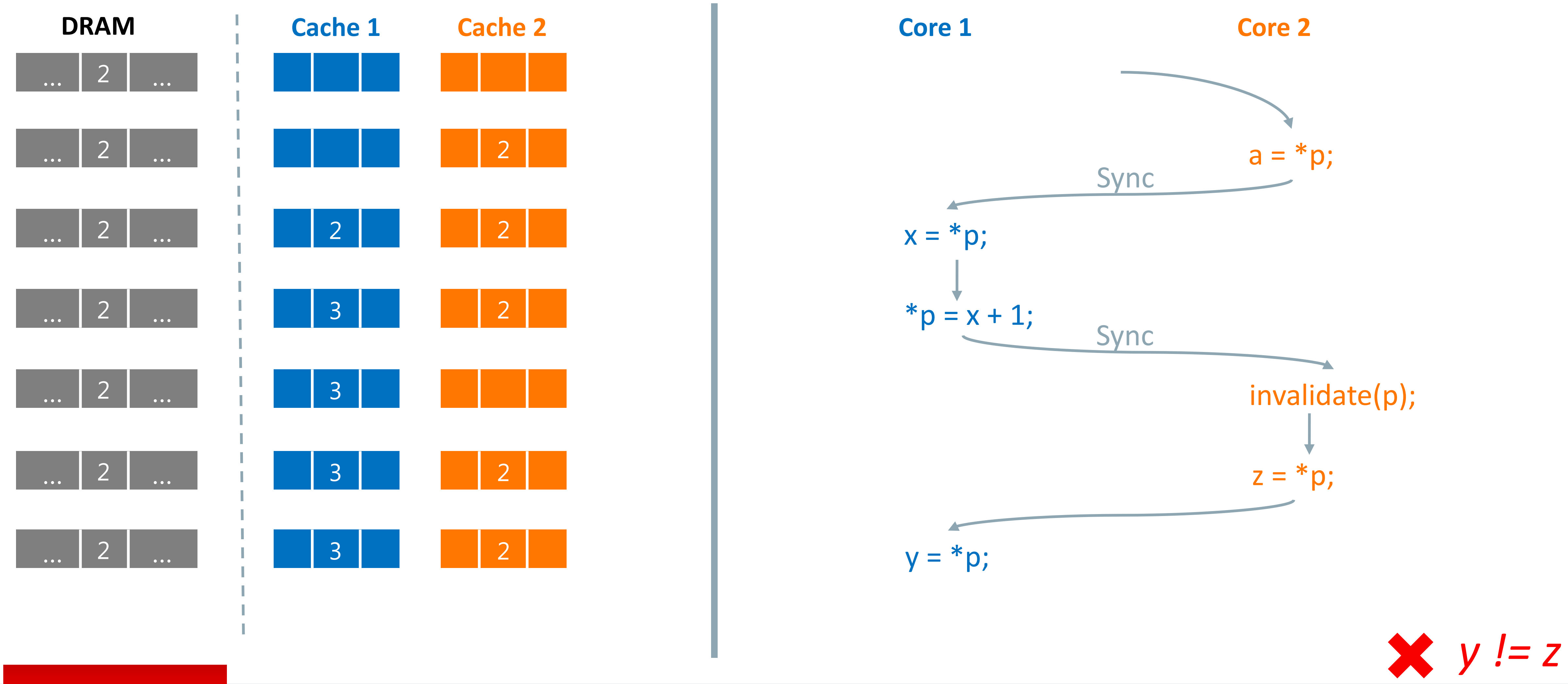


ORACLE

Cache Coherence Issue



Cache Coherence Issue



Cache Coherence Issue

- Hardware
  - 32 Cores
  - 8 GB DRAM
- Software
  - ~50,000 lines of C code
- Approach
  - Investigate software model checking to verify cache coherence

ORACLE

Model Checking

- A formal technique which automatically verifies the desired behavioral properties ***p*** of a given system , on the basis of a user-defined model ***M*** and initial state ***s***.

$$M, s \models p$$

- Verification procedure is an exhaustive search of the state space of the design.
  - No proofs
  - Counterexamples
  - State-space explosion

ORACLE

Model Checking Tools

- Initial investigation suggests SPIN is the best choice.

	Model Language	Properties Language	Concurrency Support	Scalability
SPIN	Promela (C-like)	LTL User assertions	Yes	Multi-core, State compression, Partial order reduction
CBMC / LLBMC	C/C++	Built-in options User assertions	Partial	SAT-solver/ SMT-solver Bounded checking
NuSMV / NuSMV2	SMV	LTL and CTL	Yes	SAT-solver
BLAST / CPAChecker	C	Program instrumentation	No	Counterexample-driven, Refinement
CADP / FDR2	LOTOS / CSP	Automata / User assertions	Yes	Compressing states

ORACLE

Cache Coherence Model in Promela

- Model the main memory
  - Accurate
  - Flat memory: an array of bytes

```
byte    DRAM[MAX_ELEMENT];
```

    - Each C language pointer in Promela is an integer, which is used to index the `DRAM` array.
- Model the cache
  - For each particular memory location, it has 4 possible states (U/I/S/M) at all `N` cores.

```
typedef CACHE {
            State state[N];
}
CACHE   cache[MAX_ELEMENT];
```

U: Unknown
I: Invalid
S: Shared
M: Modified

ORACLE

Cache Coherence Model in Promela

- State Transition Diagram
  - Operations: LOAD, STORE, MALLOC, FREE, FLUSH and INVALIDATE

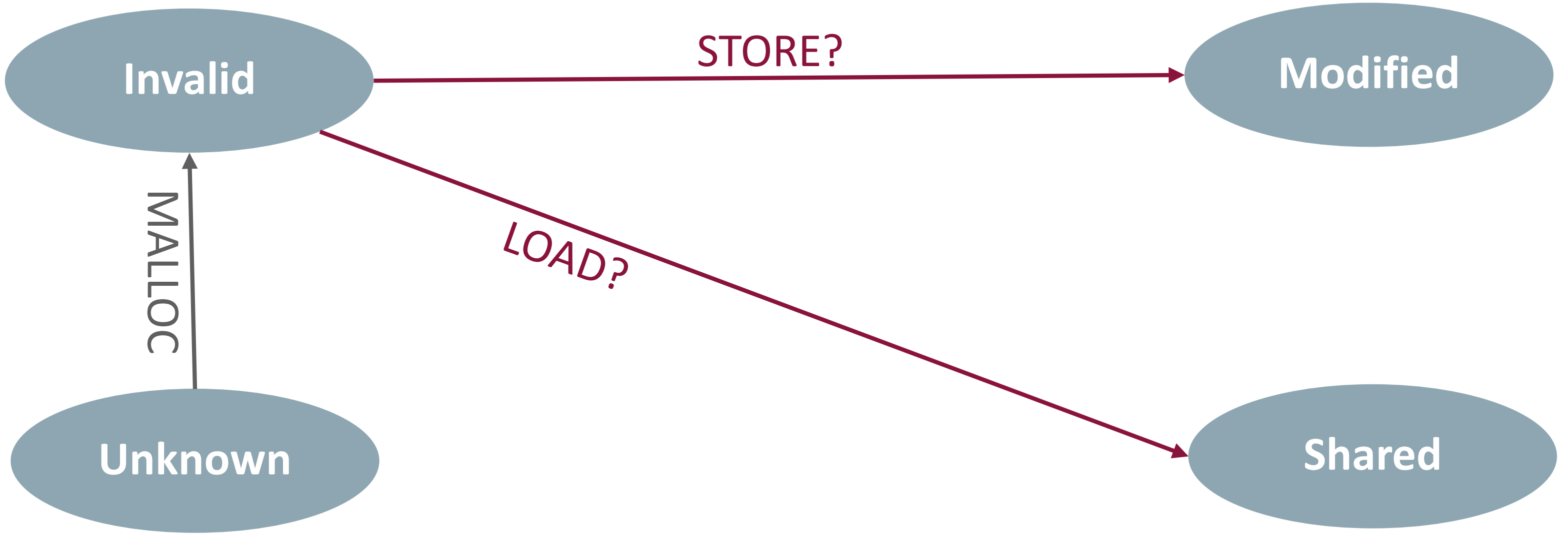
ORACLE

Cache Coherence Model in Promela

- State Transition Diagram
  - Operations: LOAD, STORE, MALLOC, FREE, FLUSH and INVALIDATE

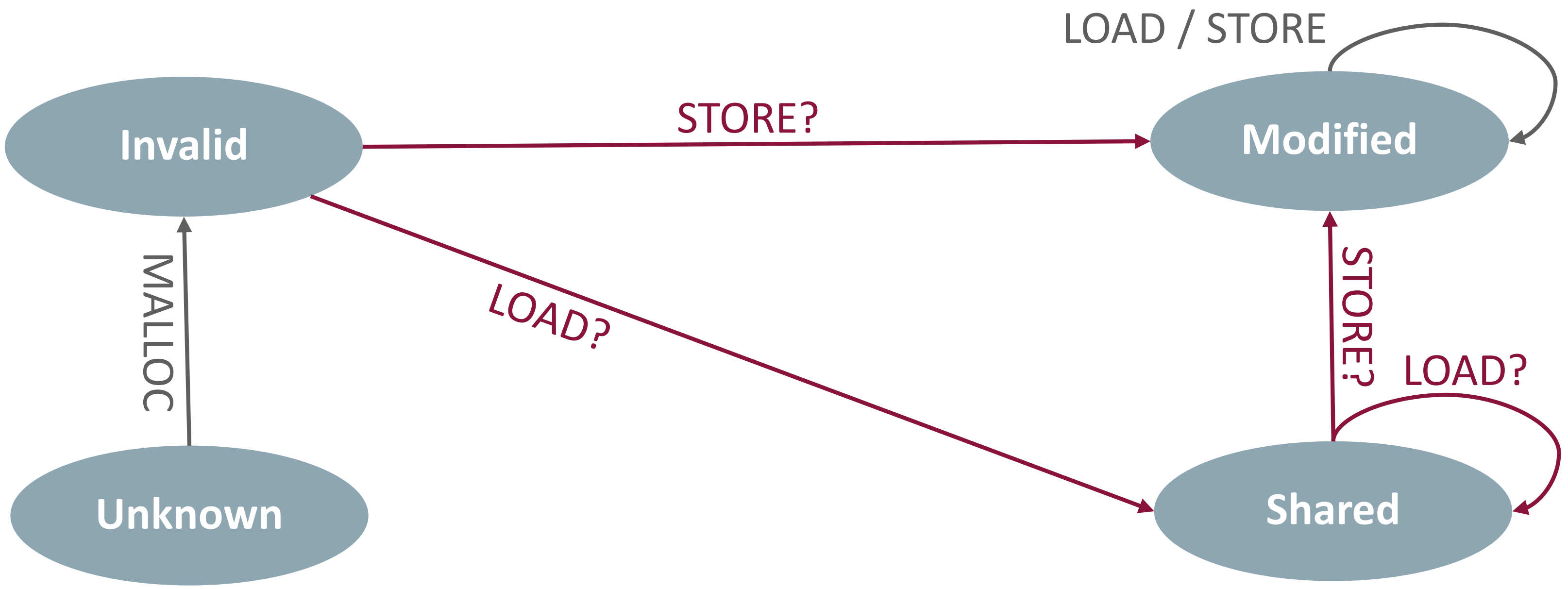
Cache Coherence Model in Promela

- State Transition Diagram
  - Operations: LOAD, STORE, MALLOC, FREE, FLUSH and INVALIDATE



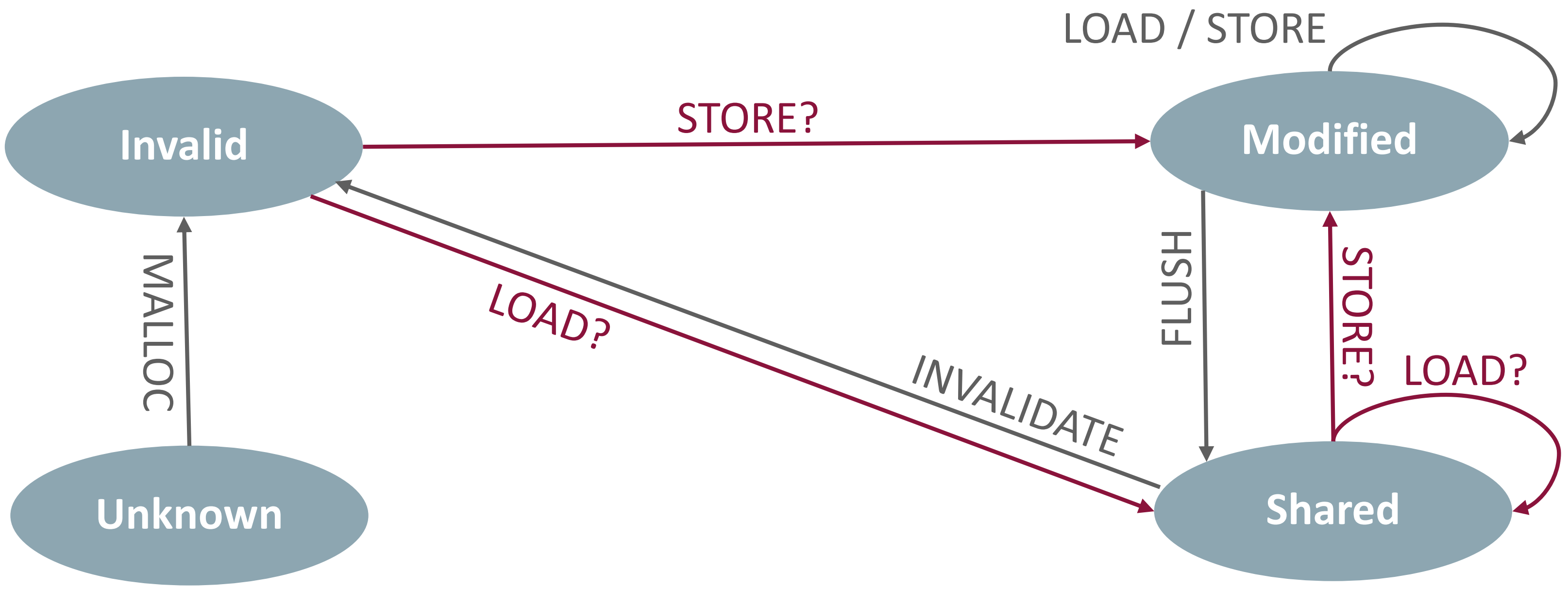
Cache Coherence Model in Promela

- State Transition Diagram
  - Operations: LOAD, STORE, MALLOC, FREE, FLUSH and INVALIDATE



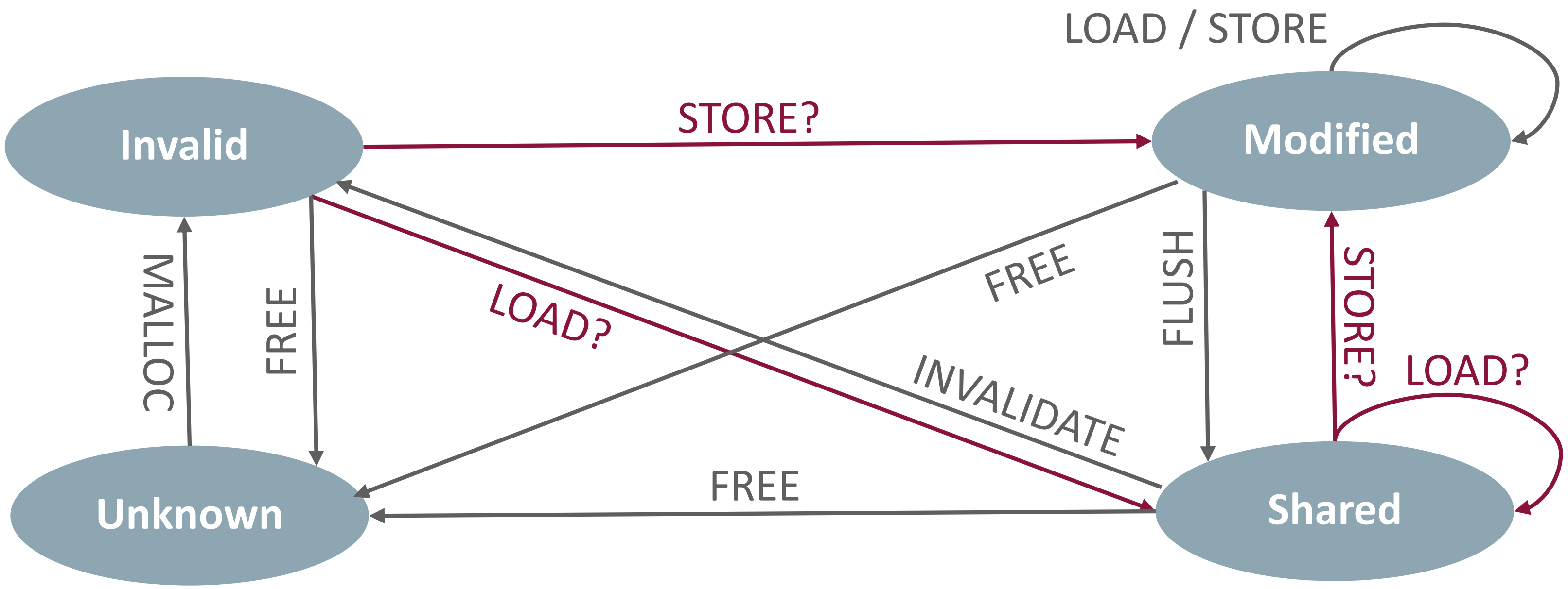
Cache Coherence Model in Promela

- State Transition Diagram
  - Operations: LOAD, STORE, MALLOC, FREE, FLUSH and INVALIDATE



Cache Coherence Model in Promela

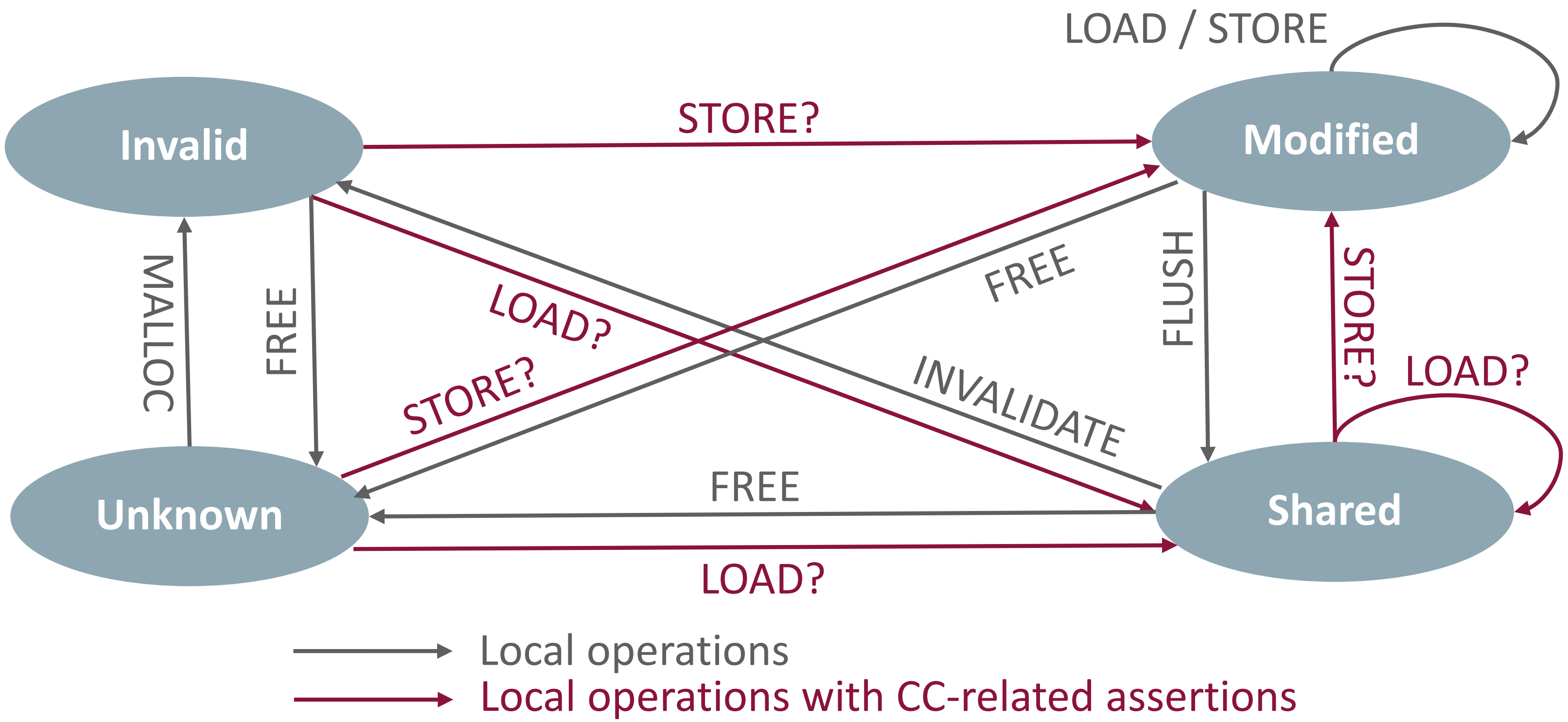
- State Transition Diagram
  - Operations: LOAD, STORE, MALLOC, FREE, FLUSH and INVALIDATE



ORACLE

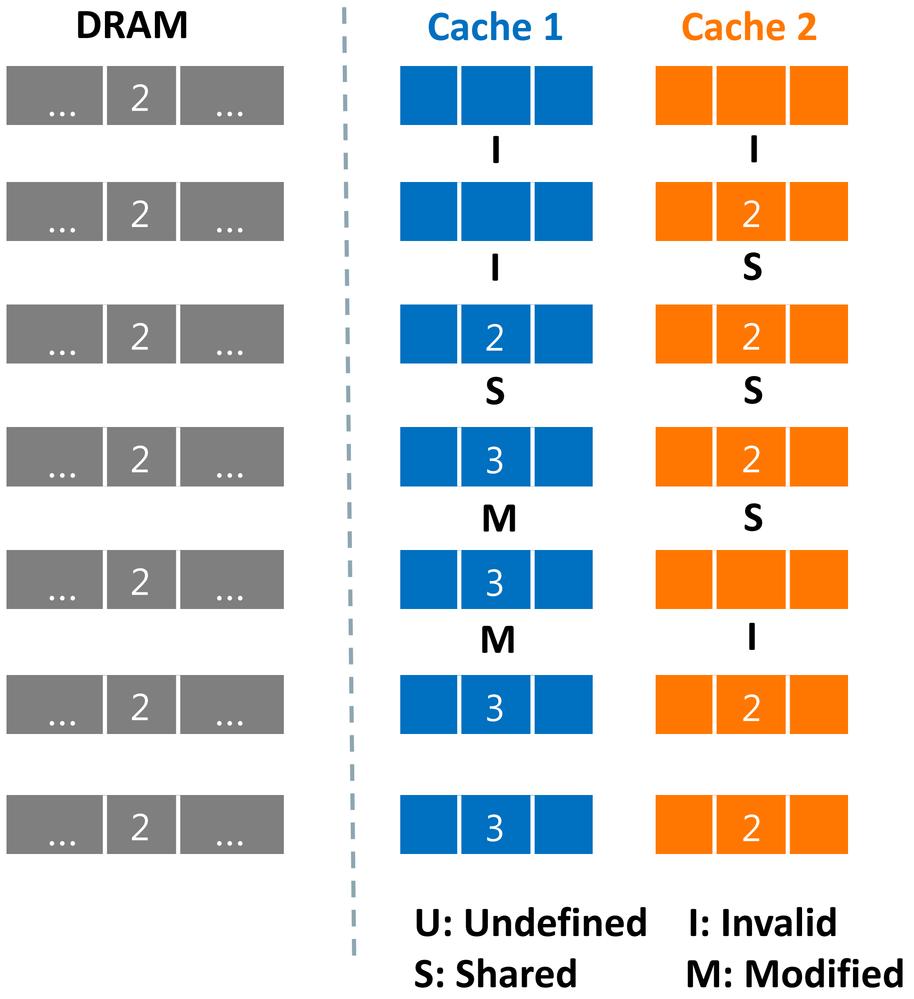
Cache Coherence Model in Promela

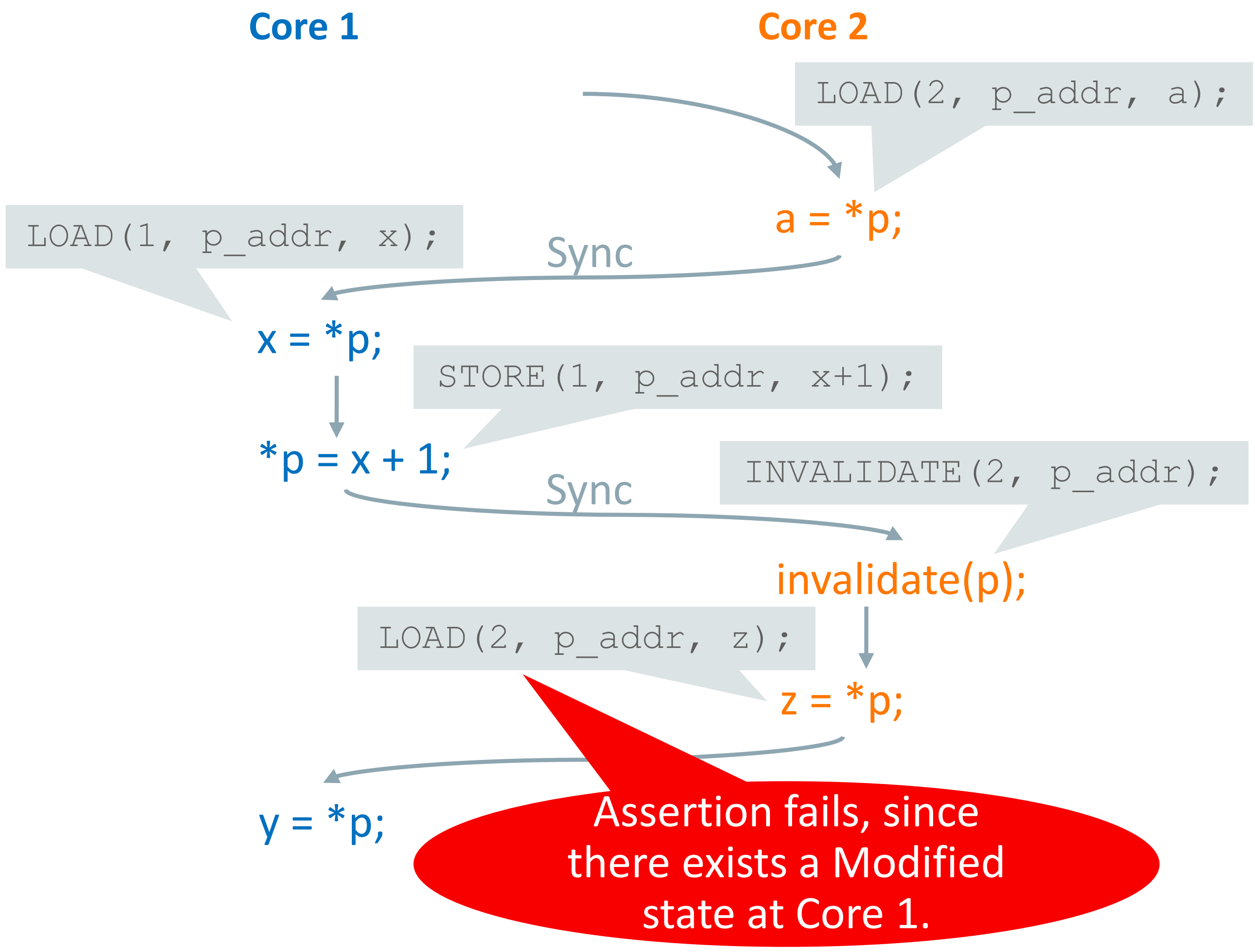
- State Transition Diagram
  - Operations: LOAD, STORE, MALLOC, FREE, FLUSH and INVALIDATE



ORACLE

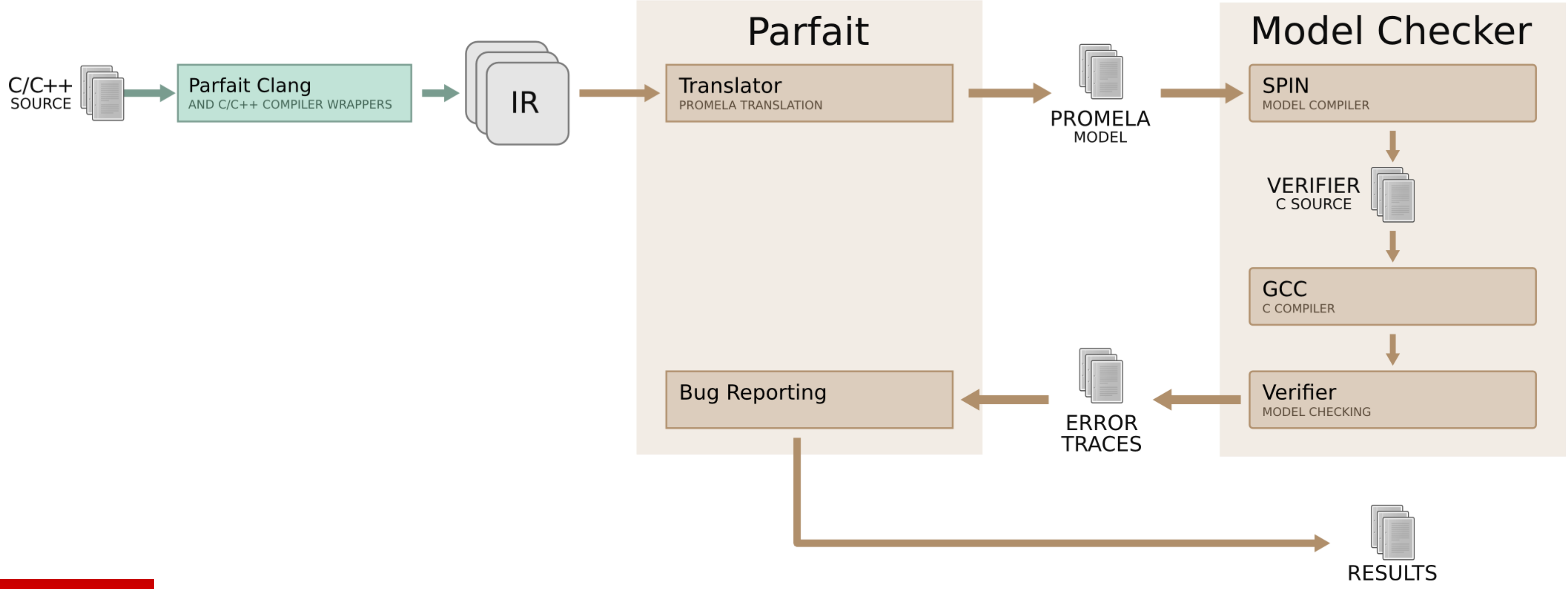
Cache Coherence Model in Promela





Automated Translation of C code to Model

- Translation of program from LLVM IR to Promela
- Generates Promela model with equivalent semantics to original program
  - With instrumentation added for each memory operation to check cache coherence



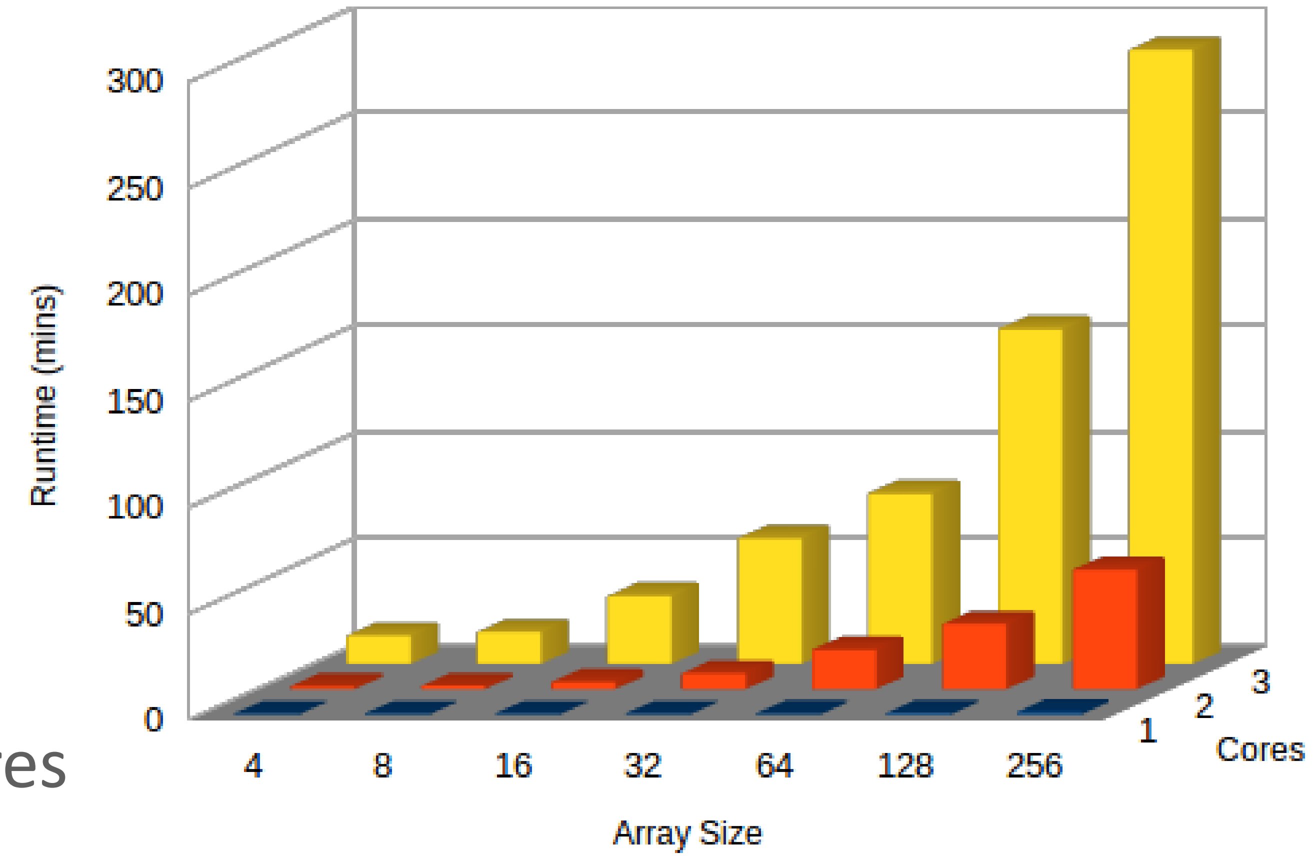
ORACLE

Automated Translation of C code to Model

- Optimisations
  - Group consecutive non-memory operations into atomic blocks
    - Avoids exploration of different interleavings of operations that cannot affect each other
    - 10x speedup in verification
  - Use SPIN's bounded context switch mode
    - *"Relatively low bounds on the number of context switches suffice for a model checker to visit all the reachable states of a model at least once." (Musuvathi, PLDI2007)*
    - Only explore execution paths with the number of preemptions less than a specified bound
    - 8x speedup in verification with a bound of 2

Results - Scalability

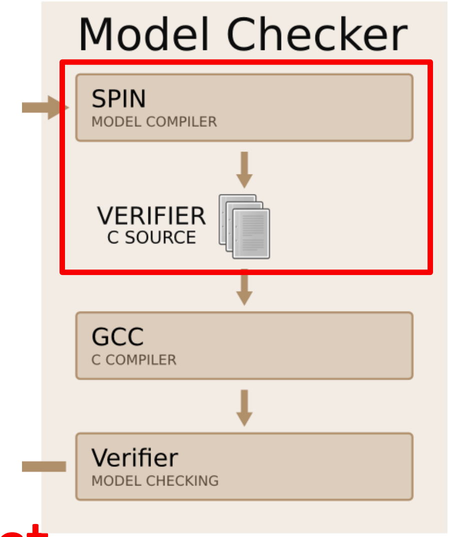
- Larger synthetic example: concurrent quicksort
  - ~200 lines of code
- Tested performance of model checking with different configurations
  - Number of concurrent cores
  - Number of elements to be sorted
- Largest test: 256 elements, 3 cores
  - Verification took ~5 hours



ORACLE

Results - Scalability

- Translated model for a C code base for about 50,000 LoC
  - ~6,000,000 lines of Promela (~1,150,000 with no function inlining)
- SPIN cannot process a model this large
  - Compiling the model did not terminate after 90 hours
- Optimisations implemented in SPIN
  - Compiling the model completes in 4 hours
  - Verifier code generated is 100,000,000 lines of C code
  - Compiling with gcc ran out of memory (64G)
- Current approach using SPIN does not scale for our project.



Further Investigation into Other Model Checkers

- More model checkers have been evaluated
  - Explicit State model checkers: Murphi/PReach, DIVINE
  - Symbolic model checkers: CBMC, LLBMC, ESBMC, SAL, Mocha, Alloy, SATABS, CSeq
  - Hybrid: LTSmin
- Initial experimental result
  - Many of them are designed for state transition systems, and only work well for algorithm verification.
  - Some of them use inlining to handle method calls and then exclude recursive calls.

ORACLE

Further Investigation into Other Model Checkers

Model Checker	Concurrency	Recursion	Quicksort 8 elements, 1 core	Quicksort 32 elements, 1 core	Quicksort 8 elements, 2 cores
SPIN	Yes	Yes	38 s	42 s	112 s
LLBMC	No	Yes (bounded)	40 s	1500 s	N/A
CSeq	Yes	No	N/A	N/A	N/A
LTSMin	Yes	Yes	> 4 h	> 4 h	> 4 h
SATABS	Partial (doesn't support shared dynamic memory)	No (ignores recursion)	> 4 h	> 4 h	N/A
CBMC	Yes	Yes (bounded)	48 s	> 2 h	> 2 h

- SPIN still appears to be the best choice

ORACLE

Conclusion

- Current approach of full software model checking for the verification of cache coherence will not scale for our 50k LoC codebase.
  - Accurate model of program memory and execution produces large models and causes state explosion

ORACLE

Next Steps

- Reduce model size and complexity via abstraction of the program
  - No longer verification (may have false positives / negatives)
  - Use abstracted memory model (e.g. based on points-to analysis)
  - Use slicing to separate the program into multiple smaller parts involving a subset of cores and verify independently
    - May be applied to either accurate or abstracted memory models
  - Manually abstract some auxiliary functions
- Reduce the number of interleavings explored
  - Group successive operations that only have local effects as one atomic operation
    - Use static analysis to determine which memory accesses do not access shared memory
  - Partial Order Reduction

ORACLE

Integrated Cloud

Applications & Platform Services

ORACLE