Causal-Broadcast Memory

Amir Karniel Ori Lahav
Tel Aviv University

DistMem: Shared memory based on causal broadcast

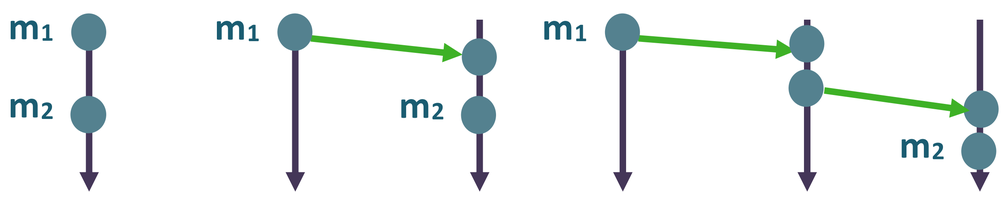
Every process *p* has a local mapping M_p from locations to values. The code for a process *p* is:

```
method write(x,v)
  M_p := M_p[x ↦ v]
  causal_broadcast(Wxv)
  return()

method read(x)
  return(M_p(x))

when Wxv is received
from another process:
  M_p := M_p[x ↦ v]
```

- A message m_1 is a cause of a message m_2 when the send of m_1 comes before the send of m_2 in Lamport's happens-before order:



- Causal-Broadcast requires that if some message m_1 is a cause of a later message m_2, then all processes will receive m_1 before m_2.

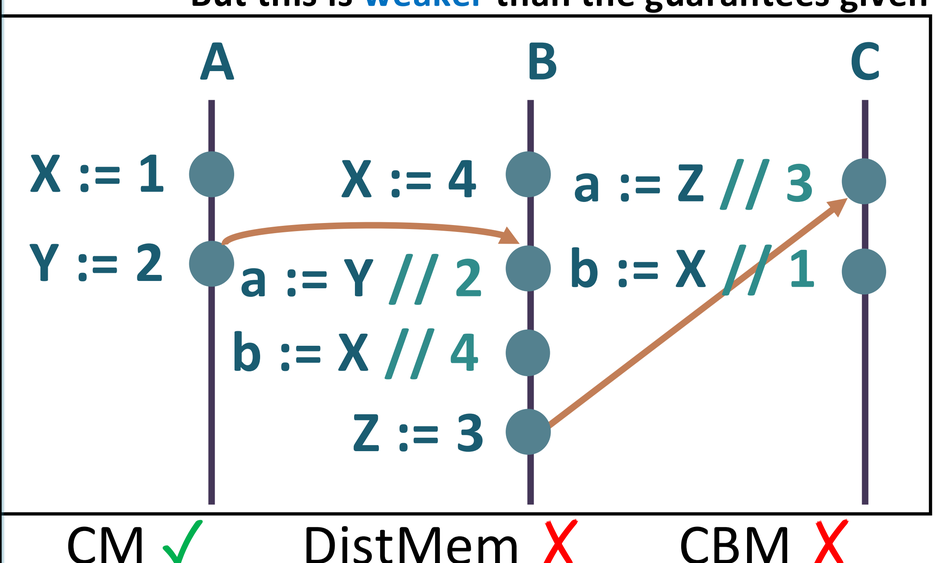
- The above is a classical algorithm given in "Causal memory: Definitions, implementation, and programming".

But what *is* causal memory?

We ask: given an abstract execution (*no* propagation edges, only sequences of memory operations), what makes it "causal-memory-consistent"?

- The above paper suggests the following model, termed CM:
  - There exists a (partial) causal order *co* extending the program order
  - Every process *p* has a serialization S_p of all events, justifying its own read events by their *co*-past
  - Each serialization agrees with *co*

But this is weaker than the guarantees given by the actual implementation!



- CM misses the necessary *co* edge from **X := 1** to **X := 4**.
- We fix this gap by adding the following to the definition of CM, resulting in our novel model **CBM**:
  - If S_p places an event before an event of *p*, there is *co* between them
- **CBM captures precisely the behaviors allowed by DistMem.**

Additional results

- **Testing**: Checking CBM-consistency of an execution graph can be done in polynomial time.
- **Verifying**: Checking state reachability of client programs operating on top of DistMem is undecidable.
- Placing CBM among other memory models (see below) and proving a write-write-race freedom result.
- Generalizing CBM and DistMem to shared objects beyond memory.