# Memory Barriers: A Hardware view for Software Hackers

Paul E. McKenney

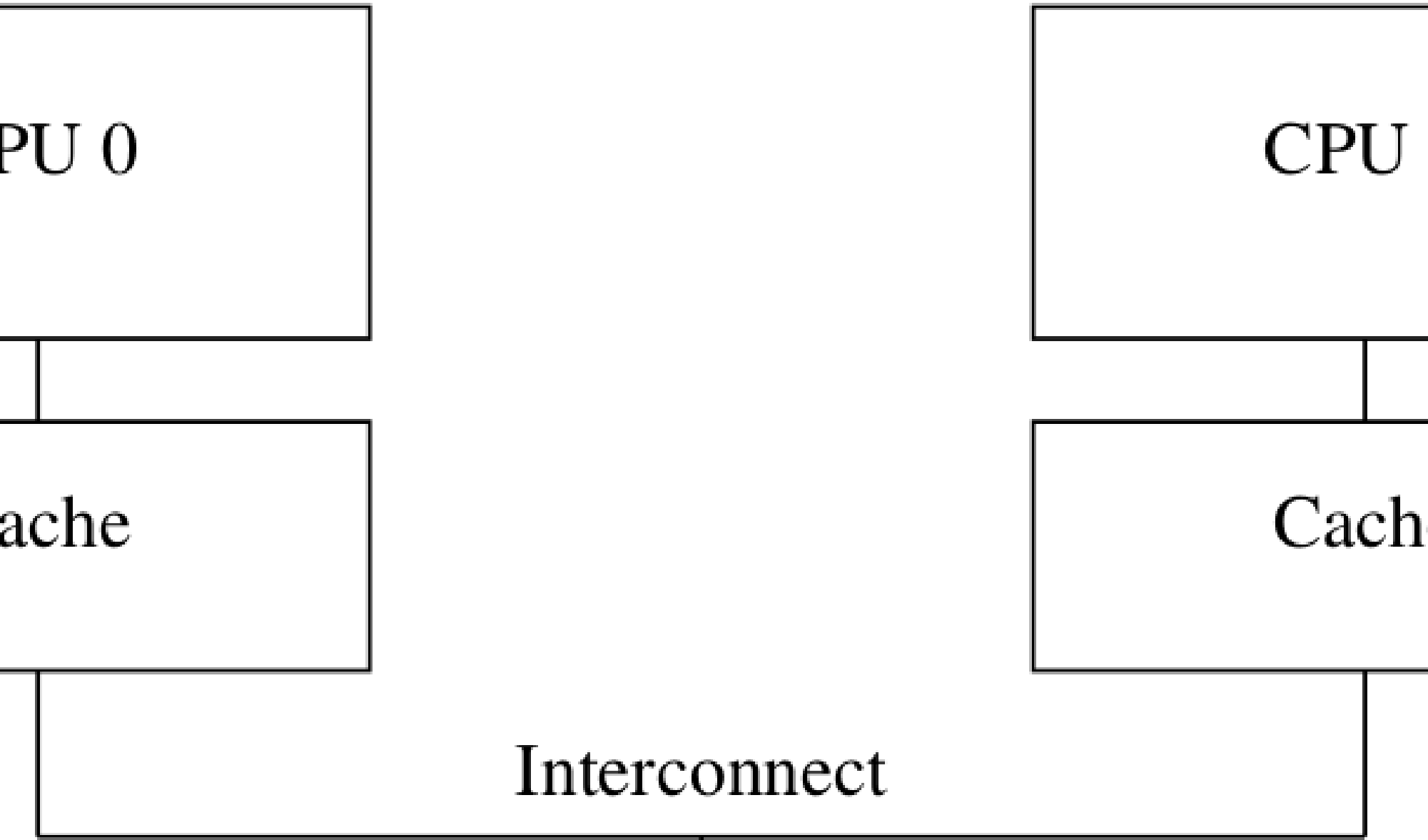
Presented by
Akshay Gopalakrishnan

November 23, 2021

# Introduction

- This paper represents the inner working of hardware that results in several non-sequential behaviors of our concurrent programs
- The paper is rife with examples as well as showcasing the reasons for having such hardware features which in turn help in our programs performing better.
- Along with the positives the author also carefully cautions why such rampant changes for performance might result in highly non-trivial behaviors being showcased by the hardware running our programs.
- The paper concludes by discussing the then versions of several concurrent hardware that exhibit different non-sequential behaviors.

# Cache structures



# The usefullness of Caches

- Used for faster access of memory.
- Useful when a single shared memory location is being accessed several times but not changed.
- Need to read from main memory which is at least 10x times slower than accessing caches.
- Overall performance of program is significantly improved.

# Memory Barriers: A Hardware view for Software Hackers

## The usefullness of Caches

The usefullness of Caches

- Used for faster access of memory.
- Useful when a single shared memory location is being accessed several times but not changed.
- Need to read from main memory which is at least 10x times slower than accessing caches.
- Overall performance of program is significantly improved.

In modern systems there are multiple cache levels that exist, each of which is based on the scope. For instance, L1 cache is local to just one core. Whereas L2 is to multiple cores in the same processor (could also be others). All this layering is done for performance, part of the reason why we have such highly non-trivial behaviors of our concurrent programs.

# THe MESI protocol

- Multiple caches need to be in synchronization to ensure no stale data in caches exist.
- Caches can communicate with each other via the interconnect network.
- We require a protocol to ensure that caches lines are updated accordingly.

An example of such a protocol is MESI (Modified Exclusive Shared and Invalid).

- Modified - cache line has the upto date data which resides in memory and the data has been stored by the corresponding CPU.
- Exclusive - cache line is not updated with the recent memory store done by the corresponding CPU.
- Shared - cache line is in read-only state (CPU needs to consult with other CPU caches before being able to write to it)
- Invalid - "empty" cache line and new data can be put here.

# Memory Barriers: A Hardware view for Software Hackers

## └THe MESI protocol

THe MESI protocol

- Multiple caches need to be in synchronization to ensure no stale data in caches exist.
- Caches can communicate with each other via the interconnect network.
- We require a protocol to ensure that caches lines are updated accordingly.

An example of such a protocol is MESI (Modified Exclusive Shared and Invalid).

- Modified - cache line has the upto date data which resides in memory and the data has been stored by the corresponding CPU.
- Exclusive - cache line is not updated with the recent memory store done by the corresponding CPU.
- Shared - cache line is in read-only state (CPU needs to consult with other CPU caches before being able to write to it)
- Invalid - "empty" cache line and new data can be put here.

Modified state is when my CPU is writiing to memory some data as well as duly updated the cache line. Now this line has the only copy of the latest data in memory. Exlcusive is one step behind modified state, wherein the CPU has updated memory, but not just its cache line. Shared state is when more than one cache line has the same data of memory. This only means, if I want to change one cache line, I must inform the others too. Invalid state can contain any stale data that is never going to be read by the CPU (as it is stale).

# Example of Cache Communication methods

# Example

# Need for Write Buffers

# Added Complications

# Example

# Here comes Write Memory Barriers

# Need for Invalidate Queues

# Added Complications

# Example

# Here comes Read Memory Barriers

# Barrier instructions offered by Linux

# Hardware Example: Alfa

# Hardware Example: x86

# Conclusion