CS152 Worksheet 11 (Cache Coherence)

Q1: MOESI Coherence

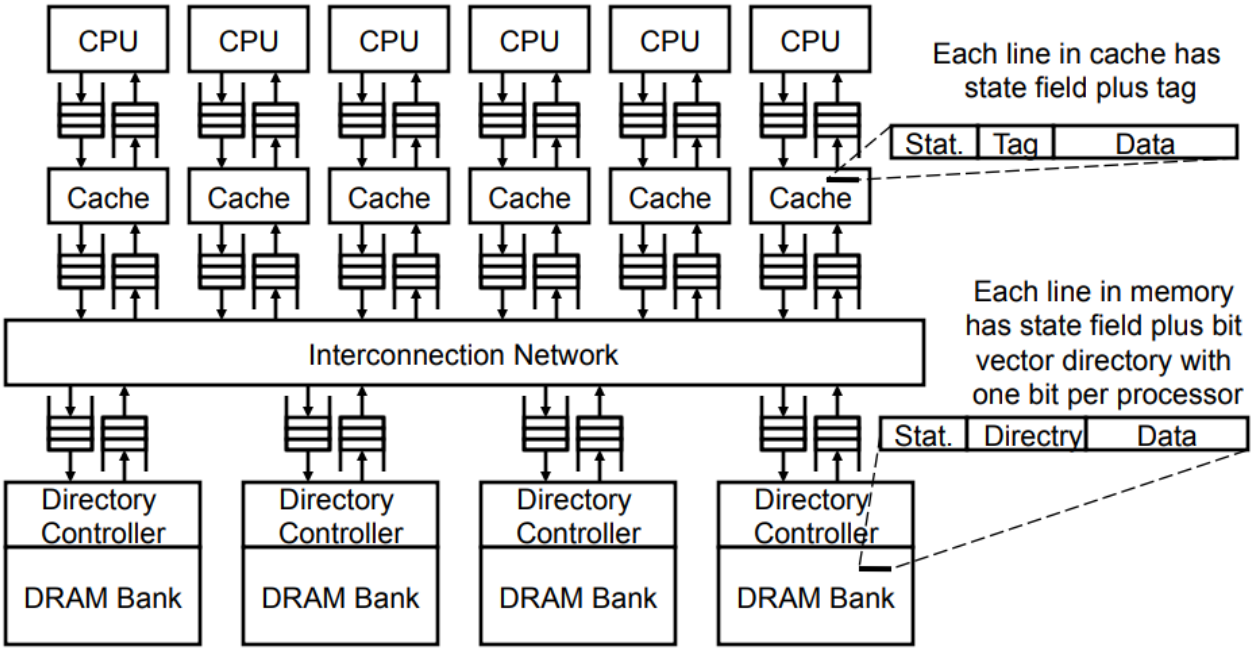
MOESI is an improvement on the MESI cache coherence protocol that adds a fifth “Owned” state which allows caches to hold dirty data without invalidating sharers. Only one cache can be in the “Owned” state while the other caches are in “Shared”. The owner is responsible for sending data to other caches requesting the line and must write back the data when it downgrades.

For each of the following new state transitions, indicate whether it is a valid transition. If it is a valid transition, explain what triggers it, what conditions must be true (i.e. do other sharers exist?), and what actions must be taken during the transition.

	Trigger	**Condition**	**Action**
I → O			
O → I			
S → O			
O → S			
E → O			
O → E			
M → O			
O → M			

Q2: Directory Cache Coherence

In lecture, it was mentioned that directory cache coherence protocols can scale up more easily than snoopy cache coherence protocols. Let's examine how such a system scales as we drastically increase the number of cores in the system.



In the simplest design, each directory entry contains the state of the cache line and a bit vector with one sharer bit per processor. Assume the directory lines have four states.

Q2.1: Full bit vector scheme #1
How many bits does the directory need to store per cache line if there are 128 cores, each with its own private cache?

Q2.2: Full bit vector scheme #2
How many bits does the directory need to store per cache line if there are 1024 cores?

Q2.3: Hierarchical bit vector scheme

It should be obvious from the previous calculation that storing one bit for each sharer per line will not be practical for massively multicore systems. For the 1024-core system, if cache lines are 64 bytes in size, twice as much memory is used for the sharer bits alone than for actual data storage.

We therefore decide to collect cores into groups and represent each group as a single bit in the bit vector. Invalidations must now be sent to all cores in a group if that group's bit is set.

For a 1024-core system with 64-byte cache lines, how many cores must be in each group to reduce the amount of directory state to 10% the amount of physical memory?

Q2.4: Reducing storage overhead

One inefficiency of this system is that you must store directory bits for every line in memory, no matter if it is cached or not. How could you reduce this inefficiency?