Lecture 27 (Memory Consistency Models)

1 Valid Outcomes

1. Different situations can have different outcomes for the same program
2. Thus, we have a set of valid outcomes for a single program
3. Every processor (and memory system) has a set of specifications that specify the allowed outcomes/behaviours

2 Memory Consistency vs Coherence

1. Coherence deals with a single memory location
2. Consistency is about what are the allowed values during execution of program

3 Observer

1. Each observer has a different PoV
2. Completion time might be anytime between start and end time (or even later)

4 Sequential Execution vs Legal Sequential Execution

1. Sequential is simply ordered
2. Legal is when every read reads the latest write
3. Observer at memory location sees a legal sequential execution

5 Atomicity

1. Each operation has a single global completion time
2. Can generate a sequential execution if this property satisfies

$P|T$: all operations issues by thread T in order

$P|T \equiv S|T$: one to one mapping from parallel to sequential execution

$P \equiv S$: for all T, $P|T \equiv S|T$

6 Sequential Consistency

1. Atomicity leads to sequential consistency
2. Additionally we assume that program order (within thread) is preserved
3. SC is gold standard (kinda)

math is to state the obvious by scaring you

6.1 Per Location SC

1. Provides the illusion of a single memory location even if we have a distributed cache
2. SC is difficult to implement but PLSC is needed

6.2 Why not SC?

1. Loads will need to be issued at commit time
2. Benefits of OOO and LSQ will go away
3. For high performance SC needs to be sacrificed

7 Non-Atomic Writes

1. Writes might be seen at different times by different threads
2. This cannot be in SC
3. However, this can be in PLSC and we are happy
4. We allow non-atomic writes but PLSC should be preserved