Memory Consistency and Cache Coherence.

Cyrus Rashtchian

February 11, 2013

Consistency and Coherence

- **Consistency** is the specification of correctness for memory accesses, defining guarantees on when loads and stores will happen and when they will be seen by the different cores.
- **Coherence** is the guarantee that caches will never affect the observable functionality of a program... they may only affect the timing of memory accesses. (Coherence is implemented using a communication protocol between the caches.)

Memory Model - Sequential Consistency (SC)

- **Sequential Consistency:** a multiprocessor is sequentially consistent if *"the result of any execution is the same as if the operations of all processors (cores) were executed in some sequential order, and the operations of each individual processor (core) appear in this sequence in the order specified by its program."*

SC Relaxation - Total Store Order (TSO)

- Often, there is a FIFO write buffer, so that after you perform a store, the data might go into a buffer for a bit before going into the main memory.
- With the SC guarantee, there needs to be a way to ensure that you don't read from an address while the write data to this address are still in the buffer.
- TSO allows a Store then a Load to be executed as a Load then a Store (this is the buffer issue), but then offers suitable solutions if they are to the same address.

Questions

Q1: If the potential consistency/coherence issues are architecture dependent (e.g. instruction reordering), then what's the benefit of talking about them at an abstract level?

Q2: Can you explain "Coherence typically deals with one cache block at a time and is silent on the interaction of accesses to multiple cache blocks. Real programs access variables across numerous cache blocks."

Q3: Why does Intel keep its memory consistency model a secret?