Memory Models

CDP 236370

Written By: Adi Omari
(Revised and corrected in 2012 by others)
Tutorial Outline

• Coherence
• Sequential consistency
• Causal consistency
• Release consistency (if time allows)
The Happened Before Relation in Shared Memory

• We say that event \( e \) **happened before** event \( e' \) (and denote it by \( e \rightarrow e' \) or \( e < e' \)) if one of the following properties holds:
  1) **Processor Order**: \( e \) precedes \( e' \) in the same process
  2) **Write-Read**: \( e \) is a write and \( e' \) is the corresponding read (reading the same value)
  3) **Transitivity**: exists \( e'' \) s.t. \( e < e'' \) and \( e'' < e' \)

• Two events \( e, e' \) are said to be **independent** or **concurrent** (denoted by \( e \parallel e' \)) if not \( e < e' \) and not \( e' < e \).
The Happened Before Relation in Shared Memory - Example

(1) Processor Order: e precedes e’ in the same process
(2) Write-Read: e is a write and e’ is the corresponding read (reading the same value)
(3) Transitivity: exists e” s.t. e < e” and e” < e’
Coherence – Reminder

• A program $P$ is coherent if for every variable $x$, there exists an order of $\text{read}(x)/\text{write}(x)$ which is consistent with program order in all the processes.

(1) Processor Order: $e$ precedes $e'$ in the same process
(2) Write-Read: $e$ is a write and $e'$ is the corresponding read (reading the same value)
(3) Transitivity: exists $e''$ s.t. $e < e''$ and $e'' < e'$
Coherence – Example 1

• A program $P$ is **coherent** if for **every variable** $x$, there **exists** an order of $\text{read}(x)/\text{write}(x)$ which is consistent with program order in all the processes.

![Diagram](image)

- The execution is not **coherent**. There is no order of $\text{read}(x)/\text{write}(x)$ that is consistent with both process 1 and 2.

(1) Processor Order: e precedes e’ in the same process
(2) Write-Read: e is a write and e’ is the corresponding read (reading the same value)
(3) Transitivity: exists e” s.t. e < e” and e”< e'
Coherence – Example 2

• A program $P$ is **coherent** if for **every variable** $x$, there **exists** an order of $\text{read}(x)/\text{write}(x)$ which is consistent with program order in all the processes.

```
Process 1                Process 2
Read y,1 → Read x,2
Write x,2 ← Write y,1
```

This execution is **coherent**. We found an order of $\text{read}(x)/\text{write}(x)$ and $\text{read}(y)/\text{write}(y)$ that is consistent with both process 1 and 2.

For variable $x$ we have: $W\ x,2 < R\ x,2$

For variable $y$ we have: $W\ y,1 < R\ y,1$
Exam Question 1 (1)

• Given the following execution of a program with 4 threads:
• Is the execution **coherent**?
• If no explain, if yes give an execution order for X and Y writes.

**P1:** R X,7;  R Y,17;  R X,13;  R Y,11

**P2:** W X,7;  W Y,11;  R X,7;  R Y,11

**P3:** W X,13;  W Y,17

**P4:** R Y,11;  R X,13
Exam Question 1 (2)

• For commands relating to \( X \) we suggest the following order:

\[
\begin{align*}
P1: & \quad R \ X, 7; \quad R \ Y, 17; \quad R \ X, 13; \quad R \ Y, 11 \\
P2: & \quad W \ X, 7; \quad W \ Y, 11; \quad R \ X, 7; \quad R \ Y, 11 \\
P3: & \quad W \ X, 13; \quad W \ Y, 17 \\
P4: & \quad R \ Y, 11; \quad R \ X, 13
\end{align*}
\]

Then we can order the writes to \( X \) in the following order:

\( W \ X, 7; \quad W \ X, 13; \)
Exam Question 1 (3)

• For commands relating to $Y$ we suggest the following order:

P1: $R X, 7$; $R Y, 17$; $R X, 13$; $R Y, 11$

P2: $W X, 7$; $W Y, 11$; $R X, 7$; $R Y, 11$

P3: $W X, 13$; $W Y, 17$

P4: $R Y, 11$; $R X, 13$

Then we can order the writes to $Y$ as the following: $W Y, 17$; $W Y, 11$;

For each one of the variables $X$ and $Y$ we found a working commands order; therefore the execution is coherent.
Sequential Consistency - Reminder

• A program P is **sequentially consistent** if there **exists** an order of **reads/writes** (of **all** the variables) which is consistent with program order in all the processes.
Sequential Consistency – Example

- A program P is **sequentially consistent** if there exists an order of reads/writes (of all the variables) which is consistent with program order in all the processes.

The execution is not **sequentially consistent**. There is no order of read(x)/write(x) that is consistent with both process 1 and 2. (the same as the first coherent example)

We already verified that this execution is **coherent**
Exam Question 2 (1)

- Given the following execution of a program with 4 threads:
- Is the execution **sequentially consistent**?
- If no explain, if yes give an execution order for X and Y writes.

\[
\begin{align*}
P1: & \quad \text{R X,7}; \quad \text{R Y,17}; \quad \text{R X,13}; \quad \text{R Y,11} \\
P2: & \quad \text{W X,7}; \quad \text{W Y,11}; \quad \text{R X,7}; \quad \text{R Y,11} \\
P3: & \quad \text{W X,13}; \quad \text{W Y,17} \\
P4: & \quad \text{R Y,11}; \quad \text{R X,13}
\end{align*}
\]
Exam Question 2 (2)

(P3) (1) (P2) (2)
W X,13 < W Y,17 < W Y,11 < R X,7 < W X,13

(1) According to P1: R Y,17 < R Y,11 (P1 Order) => W Y,17 < W Y,11

(2) According to P1: R X,7 < R X,13 (P1 Order) => W X,7 < W X,13

=> W X,7 < R X,7 < W X,13 (no other writes)
Causal Consistency

- If event B is caused or influenced by an earlier event A (A happened before B), **Causal Consistency** requires that all the processes first see A, then see B

- **Causally related events**: Events A and B are causally related if A causes or influences B (A happened before B)

- **Concurrent events**: Events A and B are concurrent/independent if they are not causally related

- Formally:

  - **Casual Consistency**: An execution is said to be legal under *causal consistency* if writes that are casually related are seen by all processes in the same order. Concurrent writes may be seen/read in a different order on different machines.
Example – Causally Consistent

$W_1(x), 1 \rightarrow W_2(x), 2$ and $W_1(x), 1 \rightarrow W_1(x), 3$ are the only writes that are causally related.

$W_2(x), 2$ and $W_1(x), 3$ are concurrent events (so it is not required that all processes see them in the same order).

The above sequence of events is Causally Consistent but not Sequentially Consistent (conflicting readings of $x$ in $P_3$ and $P_4$).
Example – Causally Inconsistent

W_2(x),2 is causally related to W_1(x),1 (because the writing of 2 may result from the value read by R_2(x),1).

Since the two writes are causally related all processes must see them in the same order as they relate each-other, however, P3 sees them in a different order.
Example – Causally Consistent

\[ W_1(x), 1 \]
\[ W_2(x), 2 \]
\[ R_3(x), 2 \]
\[ R_4(x), 1 \]
\[ R_3(x), 1 \]
\[ R_4(x), 2 \]

\( W_2(x), 2 \) and \( W_1(x), 1 \) are **concurrent** events (the read \( R_2(x), 1 \) from previous example was removed).

Since the two writes are **concurrent** they may be seen in a different order by different processes.
Exam Question 3 (1)

- Given the following execution of a program with 4 threads:
- Is the execution *causally consistent*?
- If no explain, if yes give an execution order for X and Y writes.

**P1:** R X,7; R Y,17; R X,13; R Y,11

**P2:** W X,7; W Y,11; R X,7; R Y,11

**P3:** W X,13; W Y,17

**P4:** R Y,11; R X,13
Exam Question 3 (2)

- We can order the causally related command such that reads come after the write they related to
  - There is no other causal relations in this program
- This defines a partial order, but that is what causally consistent requires in our case
  - For every group of related commands, the order must be seen the same for all the processes
Exam Question 4 (1)

• Determine the relationship for every pair
• \(\Leftrightarrow, \Rightarrow, \Leftarrow, \text{ or } \Leftrightarrow\) if there is no logical relation

• Coherence ____ Sequential Consistency
• Causal Consistency ____ Sequential Consistency
• Coherence ____ Causal Consistency
Exam Question 4 (2)

• Coherence ⇐ Sequential Consistency
  • Explanation: if there is a “good” order between all the reads/writes (of all the variables together), so particularly there is a “good” order for the commands of every variable separately

• Causal Consistency ⇐ Sequential Consistency
  • For a similar reason as before

• Coherence ⇔ Causal Consistency
  • We will show that using two examples
Exam Question 4 (3)

• Coherence ≠ Causal Consistency
  
P1: R Y,1; W Y,2; R Y,2; W X,1;  
P2: R X,1; W Y,1; R Y,2;

• It is easy to show that it is coherent (do that by your self)
• Let us show that it is not causal consistent:

• (1) from P1 we learn that \( W Y,1 < W Y,2 \) (because \( W Y,2 \) comes after \( R Y,1 \) and every write is related to (may be affected by) all the previous reads.

• (2) from P2 we learn that \( R X,1 < W Y,1 \) (for the same reason as before) and \( W X,1 < R X,1 \) (read of a value is related to - comes after- the write that wrote it) \( \Rightarrow W X,1 < W Y,1 \) (transitivity)

• (3) from P1 we learn that \( W Y,2 < W X,1 \Rightarrow W Y,2 < W Y,1 \) (transitivity)

• From (1) and (3) we get that \( W Y,1 < W Y,2 \) and \( W Y,1 > W Y,2 \) two conflicting requirements that can't be fulfilled together.

• Therefore the execution is not causally consistent.
Exam Question 4 (4)

- Coherence $\neq$ Causal Consistency

- Not coherent: P1 and P2 see different read order of Y.

- Causally consistent: according to the dependencies above we can order the commands in this order
  - For example (there may be more orders):
    - W Y,1; R Y,1 (by p1); R Y,1 (by p2);
    - W Y,2; R Y,2 (by p1); R Y,2 (by p2);
Exam Question 5 (1)

- Given that Java's lock-free memory model obeys Coherence, what values of a,b are possible after the following execution:
  - Initialization: int a = 1, b = 2;
  - After the initializations, both threads are running in parallel

<table>
<thead>
<tr>
<th>Thread A:</th>
<th>Thread B:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a = b;</td>
<td>b = a;</td>
</tr>
</tbody>
</table>

- Note: in this question there can be more than one right answer (mark them all).

A. a = 1; b = 2;
B. a = 2; b = 1;
C. a = 1; b = 1;
Exam Question 5 (2)

• Option A: \(a = 1; b = 2;\)

• Initialization: \(\text{int } a = 1, b = 2;\)

<table>
<thead>
<tr>
<th>Thread A: (a = b;)</th>
<th>Thread B: (b = a;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R b, 1)</td>
<td>(R a, 2)</td>
</tr>
<tr>
<td>(W a, 1)</td>
<td>(W b, 2)</td>
</tr>
</tbody>
</table>

\(W b, 1\) does not exist!
Exam Question 5 (3)

Option B: a = 2; b = 1;

- Initialization: int a = 1, b = 2;

<table>
<thead>
<tr>
<th>Thread A: a = b;</th>
<th>Thread B: b = a;</th>
</tr>
</thead>
<tbody>
<tr>
<td>R b, 2</td>
<td>R a, 1</td>
</tr>
<tr>
<td>W a, 2</td>
<td>W b, 1</td>
</tr>
</tbody>
</table>

Full order of reads and writes $\Rightarrow$ Sequential Consistency
Exam Question 5 (4)

Option C: \( a = 1; b = 1; \)

- Initialization: \( \text{int } a = 1, b = 2; \)

<table>
<thead>
<tr>
<th>Thread A: ( a = b; )</th>
<th>Thread B: ( b = a; )</th>
</tr>
</thead>
<tbody>
<tr>
<td>R ( b, 1 )</td>
<td>R ( a, 1 )</td>
</tr>
<tr>
<td>W ( a, 1 )</td>
<td>W ( b, 1 )</td>
</tr>
</tbody>
</table>

Full order of reads and writes => Sequential Consistency
Release Consistency - Reminder

• Before issuing a write to a memory object or a read from it, the process must acquire the needed lock, and later release it.
• Before a read or write access is allowed to perform, all previous acquire accesses must be performed, and
• Before a release access is allowed to perform, all previous read or write accesses must be performed, and
• Acquire and release accesses are sequentially consistent.
Exam Question 6

• Event e considered happened before event e’ according to Release consistency. Write the 3 cases that defines this formally:

Answer:

1) **Processor Order**: e precedes e’ in the same process (thread).

2) **Release-Acquire**: e is a release and e’ is the following acquire of the same lock.

3) **Transitivity**: exists e'' s.t. e < e'' and e'' < e’