Lecture 12: Relaxed Consistency Models

- Topics: sequential consistency recap, relaxing various SC constraints, performance comparison
Relaxed Memory Models

• Recall that sequential consistency has two requirements: program order and write atomicity

• Different consistency models can be defined by relaxing some of the above constraints → this can improve performance, but the programmer must have a good understanding of the program and the hardware
Potential Relaxations

• Program Order: (all refer to different memory locations)
  ➢ Write to Read program order
  ➢ Write to Write program order
  ➢ Read to Read and Read to Write program orders

• Write Atomicity: (refers to same memory location)
  ➢ Read others’ write early

• Write Atomicity and Program Order:
  ➢ Read own write early
Write → Read Program Order

- Consider three example implementations that relax the write to read program order:
  - IBM 370: a read can complete before an earlier write to a different address, but a read cannot return the value of a write unless all processors have seen the write.
  - SPARC V8 Total Store Ordering (TSO): a read can complete before an earlier write to a different address, but a read cannot return the value of a write by another processor unless all processors have seen the write (it returns the value of own write before others see it).
  - Processor Consistency (PC): a read can complete before an earlier write (by any processor to any memory location) has been made visible to all.
## Relaxations

<table>
<thead>
<tr>
<th>Relaxation</th>
<th>W → R Order</th>
<th>W → W Order</th>
<th>R → RW Order</th>
<th>Rd others’ Wr early</th>
<th>Rd own Wr early</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM 370</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSO</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PC</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- **IBM 370**: a read can complete before an earlier write to a different address, but a read cannot return the value of a write unless all processors have seen the write.
- **SPARC V8 Total Store Ordering (TSO)**: a read can complete before an earlier write to a different address, but a read cannot return the value of a write by another processor unless all processors have seen the write (it returns the value of own write before others see it).
- **Processor Consistency (PC)**: a read can complete before an earlier write (by any processor to any memory location) has been made visible to all.
Examples

Initially, A=Flag1=Flag2=0
P1                               P2
Flag1=1                        Flag2=1
A=1                            A=2
register1=A                    register3=A
register2=Flag2                 register4=Flag1

if (A==1)
    register1=A

Result: reg1=1;reg3=2;reg2=reg4=0

Initially, A=B=0
P1                               P2                     P3
A=1                            A=1                            if (A==1)
B=1                            B=1                            if (B==1)

    register1=A

Result: B=1,reg1=0

<table>
<thead>
<tr>
<th>Relaxation</th>
<th>W → R Order</th>
<th>W → W Order</th>
<th>R → RW Order</th>
<th>Rd others’ Wr early</th>
<th>Rd own Wr early</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM 370</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSO</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PC</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Safety Nets

• To explicitly enforce sequential consistency, safety nets or fence instructions can be used

• Note that read-modify-write operations can double up as fence instructions – replacing the read or write with a r-m-w effectively achieves sequential consistency – the read and write of the r-m-w can have no intervening operations and successive reads or successive writes must be ordered in some of the memory models
Optimizations Enabled

- $W \rightarrow R$: takes writes off the critical path
- $W \rightarrow W$: memory parallelism (bandwidth utilization)
- $R \rightarrow WR$: non-blocking caches, overlaps other useful work with a read miss
Weak Ordering

• An example of a model that relaxes all of the above constraints (except reading others’ write early)

• Operations are classified as data and synchronization

• A counter tracks the number of outstanding data operations and does not issue a synchronization until the counter is zero; data ops cannot begin unless the previous synchronization op has completed
Release Consistency

• RCsc relaxes constraints similar to WO, while RCpc also allows reading others’ writes early

• More distinctions among memory operations
  ➢ RCsc maintains SC between special, while RCpc maintains PC between special ops
  ➢ RCsc maintains orders: acquire $\rightarrow$ all, all $\rightarrow$ release, special $\rightarrow$ special
  ➢ RCpc maintains orders: acquire $\rightarrow$ all, all $\rightarrow$ release, special $\rightarrow$ special, except for sp.wr followed by sp.rd
Programmer Viewpoint

• Weak ordering will yield high performance, but the programmer has to identify data and synch operations

• An operation is defined as a synch operation if it forms a race with another operation in any seq. consistent execution

• Given a seq. consistent execution, an operation forms a race with another operation if the two operations access the same location, at least one of them is a write, and there are no other intervening operations between them

```
P1                  P2
Data = 2000         while (Head == 0) { }
Head = 1            ... = Data
```
Performance Comparison

• Taken from Gharachorloo, Gupta, Hennessy, ASPLOS’91

• Studies three benchmark programs and three different architectures:
  - MP3D: 3-D particle simulator
  - LU: LU-decomposition for dense matrices
  - PTHOR: logic simulator

  - LFC: aggressive; lockup-free caches, write buffer with bypassing
  - RDBYP: only write buffer with bypassing
  - BASIC: no write buffer, no lockup-free caches
Performance Comparison

Figure 3: Relative performance of models on LFC

Figure 7: Performance of MP3D under LFC, RDBYP, and BASIC implementations.
Summary

• Sequential Consistency restricts performance (even more when memory and network latencies increase relative to processor speeds)

• Relaxed memory models relax different combinations of the five constraints for SC

• Most commercial systems are not sequentially consistent and rely on the programmer to insert appropriate fence instructions to provide the illusion of SC
Title

• Bullet