Homework 4 Solution

CSE 661 - Parallel and Vector Architectures

5.6 (6 pts)

(a) (*A*, *B*) will have the final value (1, 1). All combinations of (*u*, *v*, *w*) are possible under sequential consistency, except that (*u*, *v*, *w*) cannot be equal to (1, 1, 0). The value (1, 1, 0) violates SC since P3 would have written *w before v*, rather than in program order.

(b) Again, (*A*, *B*) will have the final value (1, 1). All combinations of (*u*, *v*, *w*, *x*) are possible under sequential consistency, except that (*u*, *v*, *w*, *x*) cannot be equal to (1, 0, 1, 0). The value (1, 0, 1, 0) violates SC since then P2 and P4 would see the writes to *A* and *B* happening in opposite order, which violates the atomicity of the writes.

(c) If they are atomic then (*u*, *v*) = (0, 0) and (1, 1) are not possible. If they are not atomic then (0,0) is possible but (1, 1) is not. Note that (*u*, *v*) = (0, 1) or (1, 0) is possible in both cases.

5.9 (4 pts) No, they are not sufficient. Here is a counter-example:

P1	**P2**	**P3**
A = 1	*u* = *A*	*B* = 1
	v = *B*	*w* = *A*

Assuming that all initial values are 0, (*u*, *v*, *w*) cannot be equal to (1, 0, 0) under Sequential consistency. However, it is possible under the conditions of this exercise because these conditions do not guarantee write atomicity.

It is not sufficient that all operations that were performed with respect to P*i* (P1 in the above example) before it issued the store (*A* = 1) to be performed with respect to P*j* (P2 in the above example) that observed the value written by P*i* (P1). All write operations should be performed with respect to ALL processors and observed in the same order. In the above example, P3 did not observe the value written by P1, although it was observed by P2.

8.3 (4 pts) Number of nodes = 512 / 8 = 64 nodes.

(a) Full bit vector scheme:
Directory memory overhead = 64 bits per 64 bytes cache line = 1/8 = 12.5%

(b) Dir$_i$ B with *i* = 3:
Directory memory overhead = $3 \times \log_2 64 = 18$ bits per 64 bytes cache line = 1/8 = 3.5%

8.12(6 pts) **Situation in which write atomicity is violated, assuming an update-based protocol:**
P2 sees the writes to *A* (by P1 and P4) as having happened in a different order than P3 sees them. This can happen in an update-based protocol if P1's write (*A* = 1) reaches and updates P2 before P3, whereas P4's write (*A* = 2) reaches P3 before P2. P2 reads *A* as 1

for *u*, and P3 reads *A* as 2 for *w*. Then, P1's write (*A* = 1) reaches P3 and P2's write (*A* = 2) reaches P2. Then, P2 reads *A* as 2 for *v*, whereas P3 reads *A* as 1 for *x*. The writes to *A* done by P1 and P4 are seen in one order by P2 and in a different order by P4, which is a clear violation to write atomicity.

(a) We can get the result (*u*, *v*, *w*, *x*) = (1, 2, 2, 1), which should not be the case if write atomicity is not violated.

(b) Since the operations are to the same location *A*, write serialization is also violated in the above example. The writes to *A* are seen in a different order with respect to P2 and P3, which is a clear violation to coherence as well.

To solve this problem and ensure serialization (as well as atomicity) of writes done by P1 and P4, all writes should be sent first to the home node and directory. If P1's write to *A* reaches the home node first, then P4's write to *A* has to wait until P1's write is complete and has updated all the shared copies. We can also prevent processors from reading their copy of the shared variable *A*, until all copies have been updated.

(c) Write serialization can also be violated in an invalidate-based protocol if invalidations are sent directly to cached copied without being serialized at the home node and directory. P1's write will invalidate all shared cached copies of *A* (if it knew their places) and P4's write will do the same thing at the same time. Both writes to *A* may end-up taking place at the same time, which is a clear violation to write serialization. *A* might end up having two owners. This is why all writes have to be serialized by going first to the home node to determine the shared copies and to keep track of the last owner. However, if all writes go first to the home node and directory then write serialization and atomicity can be ensured.

(d) On a bus-based machine both P2 and P3 see the write updates done by P1 and P4 in the order that they appear on the serializing bus. Therefore, there is no violation to write atomicity or serialization.