CSL862 Major

Answer all 24 questions 20/11/2012 Max Marks: 45

Hybrid Transactional Memory

1. Explain the following code generated by the hybrid transactional memory compiler and also explain how it preserves transactional semantics [3]

```
txn_begin handler-addr
if (!canHardwareRead(&X)) {
   txn_abort;
}
tmp= X;
if (!canHardwareWrite(&Y)) {
   txn_abort;
}
Y = tmp + 5;
txn_end;
```

2. Consider the “sequence locks “ described in the TxLinux paper. Why do you expect sequence locks to perform better than simple reader-writer locks? [2]

3. From the TxLinux paper, look at the cx_exclusive(lock) function below:

```
void cx_exclusive(lock) {
   if (xgettxid)  xrestart(NEED_EXCLUSIVE);          //line 1
   while (1) {
      while (*lock != 1);    //spin
      disable_interrupts();
      if (xcas(lock, 1, 0))  break;                  //line 5
      enable_interrupts();
   }
}
```

   a. Explain the logic of “line 1”. What is xgettxid and why is xrestart needed? [1]

b. What are the semantics of xcas? How are they different from compare-and-swap? [1]

4. In two of the papers/material we read, we found that it is not safe to write to a page table within a transaction. Why? [1]

5. On TxLinux, explain using pseudo-code/diagram, how cxspinlocks can cause deadlocks in code that would otherwise not have had any deadlock (if using a simple spinlock). [2]

6. Explain “eager version management” vs. “lazy version management” and their tradeoffs in the context of “Operating Systems Transactions” paper. [2]

7. In “OS Transactions”, both transactional and non-transactional threads co-exist. If the two threads (transactional and non-transactional) conflict, what are some contention management possibilities? For example, can the non-transactional thread be rolled back? Explain how starvation of transactional threads is prevented? [3]

8. In Table 4 of “Operating System Transactions” paper, why does “NoTx” perform worse than “Static” column? Also, the mean overhead of the “Tx” column is 6.61x – is this acceptable performance? Why or why not? [2]

9. In Table 6 of “Operating System Transactions” paper, the caption states that “LDIF-TxOS provides the same crash consistency guarantees as BDB with more than double the write throughput”. Explain what this statement means, and why LDIF-TxOS could be performing better? [2]

10. In Figure 5 of “Operating Systems Transactions” paper, explain why TxOS performs better than “Linux-rename” at 8 processors? [1]

11. Consider the experiment in Figure 2 and Figure 3 of the FlexSC paper. The direct cost dominates at high interrupt frequencies, and the indirect cost dominates at low interrupt frequencies. Explain why. [1]

12. Explain the following statement:
    “The FlexSC system relies on the relatively fast cache-to-cache communication (in the order of 10s of cycles) available in modern multicore processors for performance of their architecture” [2]

13. FlexSC uses a M-on-N threading package (M user threads executing on N kernel-visible threads) to harvest independent system calls by switching threads, in user-mode, whenever a thread invokes a system call. The “Scheduler Activations” paper suggested that this is a bad idea in general. For example, if one of the user thread busy waits for another thread that is currently switched out, it could degrade performance (or worse result in a deadlock on a strict priority system). Does it make sense to use scheduler activations with FlexSC? Why or why not? [4]

14. In FlexSC paper Figure 9, why is “flexsc” performing worse than “sync” when number of batched requests is 1 but shows improvement at 2 or more batched calls? [1]

15. Why is Figure 10 different from Figure 9? Why does FlexSC show improvements over sync even in this case at batching factor greater than 32? [1]

16. How do the FlexSC authors explain the disparity between the throughput improvement (94%) and the IPC improvement (71%) in the 4-core Apache throughput experiment? [2]

17. In Barrelfish paper Figure 3, why is “SHM4” performing worse than “MSG8” but “SHM2” is performing better than “MSG8”? [2]

18. Explain the following statement:
    "Message passing allows operations that might require communication to be *split-phase*".
    [1]

19. Briefly explain why a "multikernel" is distinct from a "microkernel"? [1]

20. Briefly provide three reasons why a multikernel model is expected to be the fit OS architecture for future processors. [2]

21. In Barrelfish paper Figure 6 (TLB Shootdown experiment), explain why NUMA-aware multicast scales much better with the number of cores than either unicast or broadcast. [2]

22. Why does Corey use a separate network stack for each core that requires network access? What are some implications of this to the external networked hosts and how are they handled? [2]

23. Consider Corey paper Figure 8a. Why is “Dedicated” achieving better throughput than “Polling at less than 10 cores? Why do they perform similarly at 10 cores and higher? [2]

24. Consider Corey paper Figure 12. The experiment involves checksumming a file on a webserver. The “locality mode” offers better throughput when filesize is between 256KB and 2048KB; otherwise both “locality” and “random” modes perform similarly. Explain why. [2]