HW6 - James Hall

Monday, December 10, 2012
11:30 PM

1. Necessary and sufficient conditions for deadlock
   What is the difference between *necessary* and *sufficient* in Holt's *Some Deadlock Properties of Computer Systems*?
   In a General Resource Graph (GRG), a cycle is necessary for deadlock. As in, a cycle must exist for a deadlock to exist, but doesn't guarantee deadlock
   Also in GRGs, a knot is a sufficient condition for deadlock, if-and-only-if the graph is expedient. So all processes having requests are blocked, and a knot exists in the graph, then the system is deadlocked.
   Necessary means that the condition must be true for the system to be deadlocked, sufficient means that if the condition is true the system *is* deadlocked.
2. General resource graphs and single unit requests
   Why do Holt's algorithms require single unit requests?
   Without single unit requests, a single process can request all the resources, and essentially block all other processes in the system. This is especially true in consumable resource scenarios, where there may be two producers, and if one producer grabs all resources, the other can't produce anything.
3. Correctness of deadlock detection algorithms
   What are the two conditions for a deadlock detection algorithm to be consistent?
   No undected deadlocks, and no false reports of deadlocks.
4. Bracha and Toueg's distributed deadlock detection algorithm
   What type of request model is Bracha and Toueg's algorithm? What does this mean?
   It is an OR type request model. A process requests a set of resources, and once it is granted any of the set, it continues on.
5. Ho and Romamoorthy's one-phase deadlock detection algorithm
   How does the one-phase algorithm correctly prevent false deadlocks?
   Every process maintains two tables, one for the resource status and one for the process status. The algorithm only uses the information common to both tables to detect deadlock, that way, if a message is in transit, it is not counted .
6. Ho and Romamoorthy's hierarchical deadlock detection algorithm
   How does the hierarchical version of Ho and Ramamoorthy's algorithm work?
   It designates a central site, which picks control sites for each cluster. Each control site runs the one-phase Ho and Ramamoorthy algorithm. The control site detects intracluster deadlocks, and passes the resulting information to the central site. The central site uses the collected information to detect intercluster deadlocks.
7. Chandy's distributed deadlock detection in resource models
   Why use a triplet when sending deadlock PROBE messages?
   The triplet consists of: the initiating process, the sending process, and the destination process. If the initiating and destination process are the same, then the system is deadlocks. The sending process is included, so the algorithm can work backwards to determine the cycle that is creating the deadlock.
8. Chandy's distributed deadlock detection in communication models
   What are the two major differences between the communication model and resource model of deadlock?
   In the communication model, a process typically know which other processes it is waiting on to continue, whereas in the resource model, some form of detection must be done for this information to be known. The other major difference is the request model. In the resource model, the requests are *AND* requests, where a process must receive ALL the resources it requests, but the communication model is an *OR* model, where a process must receive ONE of the

resources (messages) it requests.

9. Sinha's distributed deadlock detection algorithm and undetected deadlocks
   How can this algorithm be used to resolve deadlocks?
   The probe messages being sent identify the lowest-priority transaction that is occurring in the system. The initiating process can then send an ABORT message to the originator of that transaction, which will resolve the deadlock.
10. Choudhary's distributed deadlock detection algorithm and false deadlocks
    What are the two reasons for false deadlock detection Choudhary lists? What do these two mean?
    "False Deadlock Due to External Probe" - a process not in the cycle is requesting a resource and is waiting on a deadlock cycle. When the deadlock cycle is resolved, the external process' probe is now in the system, but it is a request that is not valid, because the cycle no longer exists.
    "False Deadlock Due to Old Information" - when multiple processes are in a deadlock state, and one receives a probe from another process, it stores this probe. If the deadlock is then cleaned up, the first receiving process may still list the external process as owning a resource, and can cause an unnecessary abort or a false deadlock to be reported.
11. Independent recovery and two-site failures
    Extend Skeen and Stonebraker's crash recovery algorithm so it can do independent recovery when two sites fail.
    This cannot be done. Since two sites may fail concurrently, but while in different local states, it is impossible for an algorithm to prevent them from reaching different (namely, *abort* and *commit*) final states.
12. Koo and Toueg's rollback algorithm and message loss
    Is message loss tolerated in Koo and Toueg's rollback algorithm? Is so, how? If not, why not?
    Some message losses are acceptable, and won't deadlock the system. For instance, if a message is lost when waiting for a reply from a downstream process, then the process waiting can assume the node has failed, and return "no." However, if a message is lost when waiting for the decision back from the initiator, the process blocks until it finds out the decision, which could be forever if the initiator has failed.
13. The message and time complexity of Venkatesan's first rollback algorithm
    Ignoring the one-time preprocessing step, the message complexity of Venkatasen's rollback algorithm is $O(md)$ and the time complexity is $O(Dd)$ where m is the number of channels in the system, and D is the diameter of the tree. What is d, what is the worst case for d, and why?
    d is the number of iterations required to settle on a rollback point. The worst case is the max number of application processes that have been sent. Say at every iteration, the variable $UPDTD_i$ for process P_i is set to true. This means that every process would complete another iteration, until the system got to its initial state, where it would settle (there would be no orphaned events). Since not every process may have sent d messages, the process that sent d would be the only process that does processing at every step.
14. The performance of Gifford's voting algorithm
    The performance of reading and writing during a transaction is broken in to 3 pieces, First Read, Subsequent Read, and Subsequent Write. What is point of each phase? Why is there no First Write? And why does a Subsequent Read take 3 messages while a Subsequent write takes 3m?
    The first read is necessary for a node to get the current copy of the file, a subsequent read is to update its copy of the file, and a subsequent write is to update the file. There is no First Write because a file cannot be written to if the file isn't first read, to know what to update and to what version number. A subsequent write takes a majority quorum to update the file, whereas a subsequent read only takes one other (the current) node to approve it.
15. The performance of Jajodia and Mutchler's hybrid voting algorithm
    Compare the performance of Hybrid Voting vs Static Voting.
    Hybrid voting maintains extra state information associated with every file, but uses the same number of messages to determine if a read/write is possible. The performance is near identical.
16. Interactive consistency and the consensus problem

What is the main difference between interactive consistency and consensus?
Consensus is deciding on a single value (even a single bit), where interactive consistency is deciding on a **vector** of values.

17. The Byzantine Generals Problem using oral messages
How many messages are sent if there is 1 general and 6 lieutenants?
The commander runs OM(2), and sends 6 messages.
Each lieutenant runs OM(1), and sends 5 messages (one to each other lieutenant), so 30 messages in total.
Each lieutenant then again runs OM(0), but 5 times, sending 5 messages each time, so 150 messages in total.
By the end, 186 messages have been sent. The complexity is $O(n^3)$.
18. The Byzantine Generals Problem using signed messages
What problems does signing messages solve?
The algorithm is now capable of handling m traitors for any number of generals. The signing of the messages also *greatly* reduces the amount of messages that are passed. It is also possible to determine if a general is a traitor, by comparing what is said to each lieutenant.
19. The performance of the seven different RAID levels
Which RAID levels have the best read performance? The best write performance? What levels are impacted by large reads and writes?
The best read performance are in levels 1, 5, and 6, as they all distribute the read across multiple disks. The best write performance is RAID0, as it doesn't need to do any extra computation to do a write. All levels (except 0) are impacted by large writes, since extra data must be written for redundancy. RAID3 is most impacted by large reads, since only one read request can be serviced at a time.
20. The reliability of the seven different RAID levels
Compare the reliability vs cost of RAID levels 1, 5, and 6.
RAID level 1 is the most costly, but contains the highest reliability against failed disks, since every disk has an exact mirror. RAID1 also affords instantaneous recovery time, as there is a perfect disk on standby.
RAID5 is the least costly, as it doesn't have any specific backup disks, but offers the least reliability, since it can only withstand one disk failure. However, RAID5 has just as good of performance as RAID1, especially comparing the dollar amount.
RAID6 costs between RAID1 and 5 (and is closer to RAID5), but offers much more reliability, since now the array can withstand 2 disk failures, rather than 1. The performance is very near to RAID 5.