Homework 10 solutions

Problem 1

a) The following sequence of instructions will cause deadlock:

Transaction A	Transaction B
A acquires write lock on x read x; if x > 0	
	B acquires write lock on z B acquires write lock on y read z; read y;
Deadlock	

b) To prevent deadlock, the transactions should look like:

Transaction A	Transaction B

{
A acquires write lock on x
A acquires write lock on y
A acquires write lock on z
read x;
if x > 0 {
write y ;
else {
write z ;
}
}
	{
	B acquires write lock on x
	B acquires write lock on y
	B acquires write lock on z
	read z;
	read y;
	write x ;
	}

c) It's not efficient, and cancels the notion of multi tasking .

Problem 2

a) Time = T * N

Assume the node with the smallest ID will initiate the snapshot. The message will be passed from one node to the next, until it reaches that first node, and then the message passing will stop. Creating a total of N rounds each taking time T.

b) Time = T * log N

The message will be passed from the root node to its children, then from each tree node to its children and so on (log (N) -1 times). When the message reaches the leaf nodes, they will send it to their parents, and then the message passing will stop. Creating a total of log N rounds each taking time T.

Problem 3

- The access to the data is blocked to ensure that re-mirrored data is an exact replica of the original data.
- As mentioned in this paragraph : " When the EBS cluster in the affected Availability Zone entered the re-mirroring storm and exhausted its available capacity, the cluster became unable to service ?create volume? API requests. Because the EBS control plane (and the create volume API in particular) was configured with a long time-out period, these slow API calls began to back up and resulted in thread starvation in the EBS control plane. The EBS control plane has a regional pool of available threads it can use to service requests. When these threads were completely filled up by the large number of queued requests, the EBS control plane had no ability to service API requests and began to fail API requests for other Availability Zones in that Region as well." The EPS control plane requests are the processes that faced starvation because they needed to be served by the EPS cluster, and were backed up because of their long time-out period configuration.
- In the Amazon system, a negotiation must take place between the EBS nodes with the volume data to elect one of these nodes as original replica (the leader), and the rest of the nodes are used for re-mirroring. The original replica is the node that the EC2 instance interacts with. It is crucial to have only one agreed-upon leader to prevent any inconsistencies in the data.
- As described in the documents, the time-out period of the EPS API requests is configured by the designers to enforce an upper bound on the level of concurrency. A small time-out will lead to a lower level of concurrency, thus worse performance. However, setting it to a very high value, can lead to starvation issues as shown in this case.
- As mentioned above, if the time-out period was a variable depending on the performance of the system, the system would have benefitted.

Problem 4

To solve this problem, you need two MapReduce jobs:

1)

You need to count how many edges does each node has (the degree of the node)

```
map(key, value)
// key: line number; value: text in a single line
for each word w in value
{
        if(w is number)
                emit(w, 1)
}

reduce(key, values)
// key: a node id; values: an iterator over counts
result = 0
for each count v in values
{
        result += v
}
emit(key,result)
```

2)

You need to count how many nodes have the same degree

```
map(key, value)
// key: node id; value: degree of the node
for each word w in value //There should be only one
{
        emit(w, 1)
}

reduce(key, values)
// key: a degree value; values: an iterator over counts
result = 0
for each count v in values
{
        result += v
}
emit(key,result)
```