CSE 461 Homework 2

Ken Lin

006198682

Total points: 50 points assumed

1. (10 points assumed)

The following figure shows a resource graph for a system with consumable resources only.

A resource is represented by a rectangle with thick lines and labeled as Ri. A process is

represented by a circle, labeled Pi.

(a) Is the graph a claim-limited graph? Why?

Yes. Assuming a request edge represents consumption, an assignment edge represents production, and that each resource would have been otherwise labeled had there been resources available (that is, there would have been markers inside a resource if there were units of that resource available), then the graph:

Has 0 available units of R2 and R2.

And has a request edge if and only if a process consumes a resource. (This is an assumption.)

Which fulfils the definition of a claim-limited graph.

(b) Is the graph reducible? Why?

Yes. There's a producer, P3, which isn't a consumer, therefore allowing the graph to be completely reduced.

2. (10 points)

Assume a system has P processes and R identical units of a reusable resource. If each process

can claim at most N units of the resource, determine whether each of the following is true or

false and prove your claim:

(a) If the system is deadlock free then $R \geq P(N - 1) + 1$.

False. A system could be deadlock free even if $R = N$. (However, the if $R < N$ then the system is guaranteed not to complete.) In cases where P is greater than 1 and N is greater than or equal to 1, N is less than $P(N - 1) + 1$, therefore R is not greater than $P(N - 1) + 1$.

As an example of a situation where a system has only N resources yet P >1 without ever being in deadlock:

Each process could be started one at a time, with the rest inactive, and only the active process being able to be assigned resources. In such a case, the first process would initialize, be allowed the maximum amount of resources N, and, after it completes, the next process would be allowed to initialize, so on and so forth until every process completes.

(b) If $R \geq P(N - 1) + 1$ then the system is deadlock free.

True.

Proof:

A system is not in a deadlock state if and only if the resource graph is not completely reducible.

Therefore, let's assume a system with $R \geq P(N - 1) + 1$ is in a deadlock and prove a contradiction. First, for a system to be in a deadlock, all resources must be able assigned to processes without any single process having enough to complete even if all available resources were assigned to that process.

Therefore, no process has N resources, otherwise they would immediately complete. Therefore, the maximum number of assigned resources is P(N-1), because otherwise at least one process would have N resources, allowing it to complete.

Because $R \geq P(N - 1) + 1$, this means there is at least 1 resource available to be assigned. However, this is a contradiction. Because each process is assigned N – 1 resources, the assignment of even a single resource, which we have available, would allow a process to complete, freeing up its resources to allow other processes to complete.

Because assuming a system with $R \geq P(N - 1) + 1$ is in a deadlock proves a contradiction, it means that all systems with $R \geq P(N - 1) + 1$ is deadlock free.

3. (10 points assumed)

The following figure shows a resource graph for a system with reusable resources only. A

resource is represented by a rectangle, in which a small square indicates a unit of the resource.

(a) Is the graph expedient? Why?

The graph is not expedient. A graph is expedient if all processes having outstanding requests are blocked, but P3, which has a request for R2, is not blocked, because R2 has 2 resources but only assigned 1. (P1 also isn't blocked, but 1 unblocked process is sufficient.)

(b) Is there any knot in the graph? Why?

There is a knot, {P1, R2, P2, R1}. A knot is a non-empty set of nodes such that all nodes in the knot can reach every node in the knot, and can only reach nodes in the knot. P1 can reach R2, P2, R1, and back to P1, but cannot reach P3.

(c) Is there any deadlock in the system? Why?

There is no deadlock. The graph is completely reducible.

As an example, P3 can be assigned a unit of R2, allowing it to complete, afterwards, P1 can be assigned the unit of R2 freed up after P3 completed, which allows it to complete. At that point, P2 can be assigned a unit of R1, allowing it to complete.

All processes are complete at that point, and the graph is completely reduced.

4. (10 points assumed)

In this problem you are to compare reading a file using a single-threaded file server and a multithreaded server. It takes 15 msec to get a request for work, dispatch it, and do the rest of the necessary processing, assuming that the data needed are in a cache in main memory. If a disk operation is needed, as is the case one-third of the time, an additional 75 msec is required, during which time the thread sleeps. How many requests/sec can the server handle if it is single threaded? If it is multithreaded?

Single threaded:

15ms + 75ms/3 = 40ms.

1000ms/s / 40ms/request = 25 requests per second, on average.

Multi-threaded:

Assuming thread switching takes no time (unrealistic), the time to perform a disk operation can be entirely discounted as the CPU can switch to a different thread.

Therefore the time it takes to complete a single thread is 15ms.

1000ms/s / 15ms/request = 66.67 requests/second.

The multi-threaded server is almost three times faster, assuming that switching between threads costs negligible time.

5. (10 points assumed)

Consider the state of a system with processes P1, P2, and P3, defined by the following matrices:

Max-Avail A = (5 2 4)

Max-Claim B = (2 2 2)

(1 2 2)

(3 1 3)

Allocation C = (1 1 0)

(1 0 1)

(1 1 1)

(a) Find the available matrix D and the need matrix E in this state.

Available D = (2 0 2)

Need E = (1 1 2)

(0 2 1)

(2 0 2)

(b) Suppose now process P1 makes a request with F1 = (0 0 1).

If the request were granted, what would be D, C, and E in the resulted state?

Allocation C = (1 1 1)

(1 0 1)

(1 1 1)

Available D = (2 0 1)

Need E = (1 1 1)

(0 2 1)

(2 0 2)

(c) To ensure the system be safe, should the request be granted? Why? Give your reasons

in detail.

No, the request should not be granted.

The original system was in a safe state, with the safe path allocating F3 = (2 0 2) to process 3, allowing it to terminate, thereby making available D = (3 1 3). Afterwards, F1 = (1 1 1) is allocated to process 1, making available D = (4 2 3). At that point, F2 = (0 2 1) can be allocated to process 2, allowing all processes to finish.

However, once F1 = (0 0 1) is granted, the available resources become D = (2 0 1) while the need E becomes:

Need E = (1 1 0)

(0 2 1)

(2 0 2)

At that point, available D = (2 0 1) can no longer satisfy the maximum need of any of the three processes available, meaning the system is no longer safe from deadlock. (Although deadlock would not be guaranteed, as it's possible one of the processes could release enough resources for another process to complete.)