ECSE 420 Assignment 1 Report
Group 19

Stuart Mashaal
260639962

Oliver Tse Sakkwun
260604362

Due: October 10th, 2018

Question 2

2.1

Deadlock can occur when these four conditions are meet:

1. Mutual exclusion: At least one resource must be held in a non-shareable mode. Otherwise, the processes would not be prevented from using the resource when necessary. Only one process can use the resource at any given instant of time

2. Hold and wait or resource holding: a process is currently holding at least one resource and requesting additional resources which are being held by other processes.

3. No preemption: a resource can be released only voluntarily by the process holding it.

4. Circular wait: each process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource. In general, there is a set of waiting processes, $P = \{P_1, P_2, ..., P_N\}$, such that P_1 is waiting for a resource held by P_2, P_2 is waiting for a resource held by P_3 and so on until P_N is waiting for a resource held by P_1.

2.2

The solution to resolve deadlock is to remove the above conditions by employing the following two strategies:

1. Prevention

   1. Mutual exclusion: In general, this condition cannot be disallowed.
   2. Hold-and-wait: The hold and-wait condition can be prevented by requiring that a process request all its required resources at one time, and blocking the process until all requests can be granted simultaneously.
   3. No preemption: One solution is that if a process holding certain resources is denied a further request, that process must release its unused resources and request them again, together with the additional resource.
   4. Circular Wait: The circular wait condition can be prevented by defining a global ordering of resource types. If a process has been allocated resources of type R, then it may subsequently request only those resources of types following R in the ordering.

2. Detection and reallocation of resources

   - The system constantly monitors processes for deadlocks/unsafe states (for example using the Banker's Algorithm) and when it detects them, it will restart and/or delays all or some of the offending processes.

Question 4

Amdahl's Law:

$$\text{Speed-Up} = \frac{1}{S + \frac{P}{N}}$$

where

- S and P are the sequential and parallel time percentages of the program, respectively. $S + P = 1$.
- N is the number of processors that can be used to paralellize the parallel fraction of the program

4.1

The maximum speed-up of a program occurs when the program is executed on an *infinite* number of processors. So, for a program where the sequential portion is 40% of the program, the maximum speed-up is:

$$\lim_{N\to\infty} \frac{1}{0.4 + \frac{0.6}{N}} = \frac{1}{0.4} = 2.5$$

4.2

Given a fixed number of processors, N, we wish to make a 20% sequential program twice as fast. We wish to do this by decreasing the program's sequential time percentage, S, by a multiplicative factor, k. We can find k by isolating it in the following equation:

$$\begin{aligned}
& 0.2 + \frac{0.8}{N} &&= 2 \cdot \left(0.2k + \frac{1 - 0.2k}{N}\right) \\
\Longrightarrow\ & 0.2N + 0.8 &&= 0.4kN + 2 - 0.4k \\
\Longrightarrow\ & 0.2N - 1.2 &&= 0.4k \cdot (N - 1) \\
\Longrightarrow\ & \frac{0.2 \cdot (N - 6)}{0.4} &&= k \cdot (N - 1) \\
\Longrightarrow\ & \frac{N - 6}{2 \cdot (N - 1)} &&= k
\end{aligned}$$

4.3

Given a fixed number of processors, N, and a program that runs twice as fast when the sequential time percentage, S, is divided by 3, the original sequential time percentage found by isolating S in the following equation:

$$\begin{aligned}
& S + \frac{1-S}{N} &&= 2 \cdot \left(\frac{S}{3} + \frac{1 - \frac{S}{3}}{N} \right) \\
\Longrightarrow\quad & SN - S + 1 &&= \frac{2SN}{3} + 2 - \frac{2S}{3} \\
\Longrightarrow\quad & 3SN - 3S + 3 &&= 2SN + 6 - 2S \\
\Longrightarrow\quad & SN - S &&= 3 \\
\Longrightarrow\quad & S &&= \frac{3}{N-1}
\end{aligned}$$