MCOMD2NOS – Networks & Operating Systems

Nathan Shuttleworth

# Task 1:

### Report on Cluster Monitor proof of concept

The system works by having 3 main scripts: the menu, the cluster status and process analytics, as well as a separate script for functions. The menu is simply used to improve the user experience, it features a loading screen that is handled using a function from the function script. The menu features colour and is designed to be easily used and understood. The three options it has are to open the other 2 main scripts, and the ability to exit.

The cluster status script works primarily through 2 for loops, the first loops through 5 times and goes through each of the nodes, the 2nd loop is inside the fist and it runs 7 times, on each run it gets a different line of data. Head and tail commands are piped together to read from a specific line of the nodes, the line that is read from changes with each run. The data is then added to an array, the values are updated with each run to get the totals. Afterwards, the data is manipulated if it needs to be for instance if it is a % value, and then the status of the whole cluster is clearly displayed to the user.

The process analytics functions slightly differently, but is ultimately built off the same idea of looping and reading data. For this one, all the process data from the nodes is written to a separate text file using a grep command, that matches the pattern of 3 capital letter in a row. Next, variables are initialised to hold the names of most demanding processes for different criteria, as well as the values of them. A for loop that runs for the number of processes there are in the text file is used, this is the total number of processes being carried out across all of the nodes. Inside this are several piped head and tail commands that read the process currently being analysed and compare its data using a series of if statements with the most demanding process, and updates these if it exceeds them. The data is then outputted to the user.

The only one of the requirements for this system that my solution does not meet is displaying the 5 top users of various properties. I was unable to implement this, I made attempts to but it would result in the rest of the process analytics to stop functioning. As a result of this I rolled back to an earlier version of the system, where all of the implemented functions worked.

One ethical/legal issue that arises from my solution relates to how node data is written to a text file. Although this data text file is wiped clean every time the menu is run, this does will still mean that if the system were to crash after running the Process Analytics, the node data will all still be written on the file. If the system was then to be compromised, all this data would be readable. This includes names of users and processes, although thankfully because of the way the system works it would not be known which nodes the processes are running on. Another ethical sided of this is whether the users of the cluster consent to having their process data saved externally from the cluster.

In the node files, the IP address of each node is displayed. However, this is not used in any of the status or analytics scripts, so if this were compromised it would be the fault of the developers of the cluster, rather than me who is just making the analytical scripts.

# Task 2:

## 1a)

Paged Memory Allocation is used to let hard disk space act as memory, for when the main memory is full. It works by dividing a job up into pages, a single application can occupy multiple pages, and the pages do not have to be adjacent. A page table records where the pages are located in the physical memory. Pages that are not being used can be sent to a swap file to open up more space, and limit external fragmentation. When this happens a lot thrashing occurs and this can lower performance.

## b)

Thrashing is when pages are sent to the swap file too much. Swapping uses a lot of the CPUs resources, and so can degrade its performance. Thrashing can be caused by having too few free pages to swap to and having too many different applications loaded in at once. It can also be caused by an SW loop that goes across two pages. This is when the pages refer to each other, and so it is just a loop being passed to and from each other wasting the CPU resources.

### c)

A first in first out (FIFO) algorithm is used when there is a new page request, and there is not enough space in the main memory to allocate it. A queue in the main memory is used to keep track, the first page to enter is the first one to be allocated, when space is freed up. An example is a queue at a theme park, once the car arrives and the last passengers have got off, those at the front of the queue get to go on and have their turn.

## 2)

Dead lock – This is when all of the resources are taken up and no processes are going to release, leading to all of the processes being stuck waiting for resources that will never be freed up. Since nothing is happening, the processes are dead. This is what could be happening when the system hangs, and the only way to fix is to turn the computer off.

Live lock – Refers to when the processes do release resources to help one another be completed, however they are all doing this and so no progress is made, just constantly releasing and gaining resources. This means that the processes are alive as they are doing something, although it is nothing functional. This is what happens when the screen appears to freeze, stuff is still happening behind the scenes but to the user nothing is happening.

Race – When there are 2 processes of the same priority trying to get the same resource, they are both ‘racing’ to get it. Because of the unpredictable nature of which will ‘win the race’ to get the resource, you cannot predict which will be executed.

Starvation – This is what happens when there is a process with a higher priority than the others that will always be allocated the required resources, meaning that all of the other processes will be ‘starved’ of resources and not able to make any progress. It is like race, except one process always wins and none of the other processes get anything.

The four conditions for dead locking are:

Mutual exclusion – This takes place when there is only a single core. It is when a process that is being executed access to a resource exclusively, so nothing else can be executed concurrently.

Resource holding – This can result in memory running out. Some processes that are currently being executed can reserve resource that they will need later on, this however stops others from being able to use it and so will become dead locked.

No pre-emption – When other processes have to wait for another to execute and there is no means to interrupt it. The resources it is using cannot be taken away until the process releases them, this will only happen when it is finished.

Circular wait- When all of the currently executed processes do not have enough resources, and are waiting for each other to release, however because they are all in need this will not happen. It is described as circular as it is a circle of processes waiting for each others resources, it is indefinite and will not end.

[Sahota, (2021)]

References:

Sahota, V. (2021) *Lecture Slides –* Accessed February, March 2021 – Used to answer all exam questions