Investigating the

differences in CPU schedulers

Operating Systems coursework – 262924 – 19th March 2024

Contents

[I. Running the experiments 3](#_Toc163230565)

[1. Experiment 1: Investigating the throughput and waiting time of the scheduling algorithms with a high number of processes. 4](#_Toc163230566)

[1.1 Introduction 4](#_Toc163230567)

[1.2 Methodology 4](#_Toc163230568)

[1.3 Results 5](#_Toc163230569)

[1.4 Discussion 5](#_Toc163230570)

[1.5 Threats to validity 5](#_Toc163230571)

[1.6 Conclusion 6](#_Toc163230572)

[2. Experiment 2: Investigating the differences in average waiting time for each scheduler. 7](#_Toc163230573)

[2.1 Introduction 7](#_Toc163230574)

[2.2 Methodology 7](#_Toc163230575)

[2.3 Results 8](#_Toc163230576)

[2.4 Discussion 8](#_Toc163230577)

[2.5 Threats to validity 8](#_Toc163230578)

[2.6 Conclusion 8](#_Toc163230579)

[3. Experiment 3: Investigating the turnaround time of the schedulers with a high number of bursts. 9](#_Toc163230580)

[3.1 Introduction 9](#_Toc163230581)

[3.2 Methodology 9](#_Toc163230582)

[3.3 Results 10](#_Toc163230583)

[3.4 Discussion 10](#_Toc163230584)

[3.5 Threats to validity 10](#_Toc163230585)

[3.6 Conclusion 10](#_Toc163230586)

# Running the experiments

A screenshot of a computer code

Description automatically generatedTo run the experiments, you will need to execute the provided bash script. This script, named run.sh, saves logs in run.log. Within this bash script, there are 15 different seeds, with 5 for each experiment. These seeds are used in the input generator to reproduce identical input files when all parameters remain the same. They are randomly generated using a Python script that produces 15 random numbers between 1 and 100,000. Additionally, the bash script defines both the simulator and input parameters for each experiment. These parameters are stored in the input\_parameters and simulator\_parameters folders within each experiment's folder, identified by its number. Also, these parameters are used to execute the two classes: InputGenerator and Simulator. The resulting output files are saved in input\_files and scheduler\_outputs, containing 5 schedulers and 5 outputs for each, totalling 25 outputs per experiment.

# Experiment 1: Investigating the throughput and waiting time of the scheduling algorithms with a high number of processes.

## Introduction

In this experiment, we're looking at the performance of scheduling algorithms with a high workload. Scalability is very important in CPU schedulers and gauges how effectively these algorithms manage resources as the workload increases. The aim is to find out which algorithms excel in resource management with a high number of processes, an important part that all CPU schedulers need to consider.

The focus is on the two key metrics: throughput, which measures the speed at which tasks are completed, and waiting time, which indicates how long tasks are in the ready queue before execution. By analysing these metrics, we gain insights into the effectiveness of scheduling algorithms under different workload scenarios which will help determine which algorithm works best under these intense conditions.

I believe that the algorithm that will be most effective will be the multi-level feedback queue with round robin, this is due to multiple queues being able to adapt to varying process requirements and changing based on the priority of the task also being able to allocate CPU time based on the priorities helps the scheduler have a good throughput even with varying workloads.

## Methodology

In this experiment, I use the same input parameters for each iteration of the experiment, only changing the seed. The value that is most important is the ‘numberOfProcesses’ and this value is set to 500 to allow a high number of processes to test on each scheduler. The parameters used for the input generator are below:

numberOfProcesses=500

staticPriority=0

meanInterArrival=10

meanCpuBurst=30

meanIoBurst=20

meanNumberBursts=5

seed=(x)

The parameters for the simulator are below:

scheduler=(y)

timeLimit=40000

interruptTime=1

timeQuantum=12

initialBurstEstimate=30

alphaBurstEstimate=0.5

periodic=false

The names of the input files are the input-seed\_x where x is the seed used, in this case the x follows these numbers (12382, 68846, 82050, 5112, 97108). The output files are named output-seed\_x following the same number trend. These output files are under a folder with the name of the scheduler the output is from, the y represents the scheduler. All these files can be found under the folder 1.

The metrics I have chosen to look at are the throughput (how many processes are completed in each time) and average waiting time (how long processes spend waiting in the ready queue). This are both plotted on two separate graphs. The reason I have chosen these metrics is that these will a good evaluation about the effectiveness of the scheduler on many processes as the average waiting time will show how well it can swap out processes and not allow processes to stay in the ready queue for a large amount of time. The throughput is good in determining the effectiveness in selecting the correct process to execute.

I have a chosen a high timelimit to have a high enough time for these processes to complete and to get good valid results. The time quantum of 12 means it has a good balance between not switching between processes too fast but also to force the round robin schedule to context switch. I set the initial burst estimate like the mean CPU burst time.

## Results

Blah

## Discussion

blah

## Threats to validity

blah

## Conclusion

blah

# Experiment 2: Investigating the differences in average waiting time for each scheduler.

## Introduction

In this experiment, I investigate how the average wating time changes for each algorithm and what one performs best when trying to minimise average wating time. The waiting time is a measure of how long processes stay in the ready queue and for a scheduler to have a small average waiting time they need to have an effective way to switch between different processes to allow processes not to ‘idle’ in the ready queue for a long amount of time.

The metric can be very important as a high waiting time causes a lot of delay for processes and can drastically decrease the performance of the algorithm. Being able to select the processes with shortest execute time is effective in this as it means less processes must wait, as you get the shorter processes out of the way. If you select a process that has a high execute time it means that all processes in the ready queue must wait for this one to finish. This is usually a lot easier said than done.

The scheduler that I think will be most effective in reducing the average waiting time is the multi-level feedback queue using round robin. As this scheduler is pre-emptive it means that it can be interrupted if a higher priority process enters the queue. In addition, it is round robin, so it allows for switching out of the processes when it has used the time quantum, this means that a longer process will not stay running for a high amount of time as it will get switched out.

## Methodology

In this experiment, I will use the same input parameters for each output but will be changing the seed each time. In this experiment it is important to have a lot of processes but with high burst and I/O times to investigate who can select the correct processes to reduce waiting time. The ‘numberOfProcesses’ is set to 100, ‘meanCpuBurst’ of 40, ‘meanIoBurst’ of 20 and ‘meanNumberBursts’ of 5. This allows for a good range of processes with a different CPU and I/O bursts. All the parameters I have used are below:

numberOfProcesses=100

staticPriority=0

meanInterArrival=12

meanCpuBurst=40

meanIoBurst=20

meanNumberBursts=5

seed=(x)

The parameters for the simulator are below:

scheduler=(y)

timeLimit=10000

interruptTime=5

timeQuantum=15

initialBurstEstimate=40

alphaBurstEstimate=0.5

periodic=false

The names of the input files are the input-seed\_x where x is the seed used, in this case the x follows these numbers (11654, 93333, 24870, 8621, 10291). The output files are named output-seed\_x following the same number trend. These output files are under a folder with the name of the scheduler the output is from, the y represents the scheduler. All these files can be found under the folder 2.

I put the initial burst estimate as the same as the CPU mean burst and the time quantum just less than half in order to allow switching of processes.

An amount of processes of 100 allows for a range of processes to be produced using the parameters and to give a good amount of results to allow for a fair test.

The metric average waiting time is plotted together for each seed and then the average waiting time is calculated using all these seeds to get a good evaluation of the overall best algorithm when it comes to minimising the average waiting time.

To validate the results, I have used 5 different seeds in order to compare the average waiting time in order to ensure a fair test and anomalies are accounted for.

## Results

Blah

## Discussion

blah

## Threats to validity

blah

## Conclusion

blah

# Experiment 3: Investigating the turnaround time of the schedulers with a high number of bursts.

## Introduction

In this experiment, I will investigate how a high number of bursts of a process can influence which scheduler will be most effective. Having a high number of bursts means having a lot of CPUS and I/O bursts and each scheduler will have a different way of dealing with them.

In order to measure how well a scheduler can perform with a high number of bursts I will be measuring the average turnaround time as this is a measure of how long it takes from the submission of the process until the time of completion, which includes all of the CPU and I/O bursts.

A scheduler with a lower average turnaround time is more desirable as this shows that the scheduler is highly efficient in dealing with the processes as opposed to something with a higher average turnaround time.

I predict that the scheduler that will perform best with these experiments is the shortest job first (using exponential averaging). This scheduler offers efficient CPU utilization and being able to prioritize the shorter burst times first means that it may lead to lower turnaround times. Also, having a scheduler that can estimate the nest burst duration can become crucial and with the exponential averaging this can have a massive help on how well this scheduler can perform.

## Methodology

In this experiment I will have a high meanNumberBursts, for each input file, of 12 which will help with my investigation. I will be keeping the parameters the same but will be generating different input files using a different seed each time. As discussed, I will be investigating the average turnaround time of each scheduler on a different seed of the input parameters, this will be plotted in order to make decisions based on the best performing and to compare the results for all of the schedulers. The input parameters that I have decided to use are:

numberOfProcesses=50

staticPriority=0

meanInterArrival=12

meanCpuBurst=10

meanIoBurst=10

meanNumberBursts=12

seed=(x)

The parameters for the simulator are below:

scheduler=(y)

timeLimit=10000

interruptTime=1

timeQuantum=5

initialBurstEstimate=10

alphaBurstEstimate=0.5

periodic=false

The names of the input files are the input-seed\_x where x is the seed used, in this case the x follows these numbers (80253, 41649, 72592, 7885, 36334). The output files are named output-seed\_x following the same number trend. These output files are under a folder with the name of the scheduler the output is from, the y represents the scheduler. All these files can be found under the folder 3.

I choose the same low number for the CPU and I/O bursts due to the amount of them being 12, I want the main focus of the experiment to be on the number of bursts. A number of processes of 50 gives a good amount of output data to test on.

In the scheduler I choose an initial burst estimate the same as the CPU mean burst and a time quantum of half of it, so that the schedulers have a chance to swap out processes.

I will run this experiment 5 times with the 5 different input parameters and using this will calculate the average turnaround time for each scheduler on those input files, these will be used to compare the results and have a solid conclusion on the best scheduler. I can also compute an overall average turnaround time using all 5 experiments.

Doing this experiment 5 times allows for any anomalies of the experiment to be hashed out and therefore is one of the input parameters does produce vastly different results than the others I can discard it from the results.

## Results

Blah

## Discussion

blah

## Threats to validity

blah

## Conclusion

blah