Job Scheduler

1. INTRODUCTION

In this report we will investigate the performance metrics of the current workstation which has a number processing units. As part of our investigation we will report on the processing unit utilization, expected number of idle processors per second, expected number of jobs in the queue per second, and operational cost of the system and then find the maximum number of jobs in queue enough to get a correct answer. We will first describe the model and then summarise our results. Our conclusion is to calculate optimal number of cores to minimize the operational cost.

2. MODEL EXPLANATION

We have been told that there are n=7 processing units. Each processor will either be IDLE or WORKING. While a processor is working, the operating cost is 1.5 pence per second. If the processor is left idle, the operating cost is 1 pence per second. The length of each job is exponentially distributed with mean length 6 seconds. A new job arrives following a Poisson process with rate 0.5 per second. Job arrivals are independent from their processing times. When a new job arrives, the scheduler will assign the job to an IDLE processing unit, picked randomly. Otherwise, if there is no IDLE processing unit, the job is added to the WAITING queue. A delayed job costs 1.25 pence per second in the WAITING queue. When a processor is done processing a job, it will later be in the IDLE state and will be assigned another job from the queue, if one exists. We can assume that the scheduler has a maximum number of jobs m after which it simply rejects jobs, which we first assume m=20 is sufficiently enough to get a correct answer.

Since we know that job arrivals follow a Poisson process, then we know that the process is a time homogeneous Markov process. With that, we know that the jobs in the workstation consists of processing jobs (WORKING processors) and WAITING jobs. Next, if number of jobs does NOT exceed number of IDLE processing units and we know how many processors are currently WORKING, we can deduce how many are IDLE. Furthermore, if we know how many jobs are currently in the system, we can deduce how many jobs are in the WAITING queue. Using this, we can model the number of processors working at any given time as a continuous-time Markov process. Using the Q matrix we have created in APPENDIX A.2., we can get the stationary distribution that will give us the long-term distribution of WORKING processors at any given time.

The stationary distribution we get also shows us the proportion of time the number of processors are WORKING. With this and the information above, we can get the total cost at each state, hence the total expected cost. Other than that, we can also find the stationary distribution and proportion of time the number of processors are working for any number of processors in the workstation. Then we can find the optimal number of cores in the workstation, which is when the operational cost is minimum. Therefore, we can determine the long-term expected cost and the other metrics for different numbers of processors. We then find the optimal number for maximum number of jobs in queue to get the most accurate answer.

3. RESULTS

The assumption of m=20 was made. First, we are required to calculate the interested metrics when n=7. Utilization of 7 processing units means at any given moment of time, for 1 processor, we will have 0.42857 processors working.

Here are the calculated metrics for n=7:

Number of Processors, n	**7**
Utilization of processing units	0.42857
Average number of IDLE processors in a second	4.00000
Average number of WAITING jobs in a second	0.02823391
Operational cost of the system (pence)	8.54

Table 1

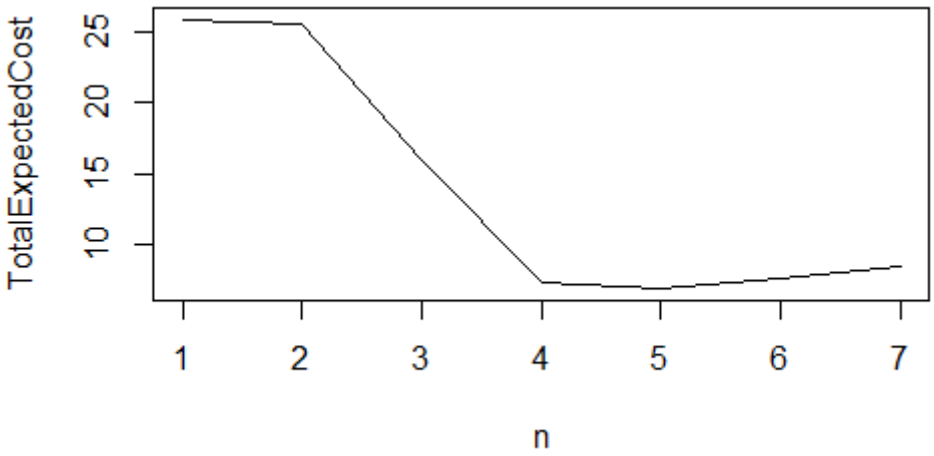
Here are the calculated metrics for n from 1 to 7 (rounded to 5 decimal points):

Processors	1	2	3	4	5	6	7
Total Expected Cost (pence per second)	25.88	25.50	15.90	7.38	6.94	7.62	8.54
Utilization of processing units	1.00000	0.99994	0.95631	0.74970	0.60000	0.50000	0.42857
Average number of IDLE processors in a second	6.37327E-11	1.11393E-4	1.31068E-1	1.00121	2.00001	3.00000	4.00000
Average number of WAITING jobs in a second	19.50000	18.00261	9.17476	1.50104	0.35411	0.09914	0.02823

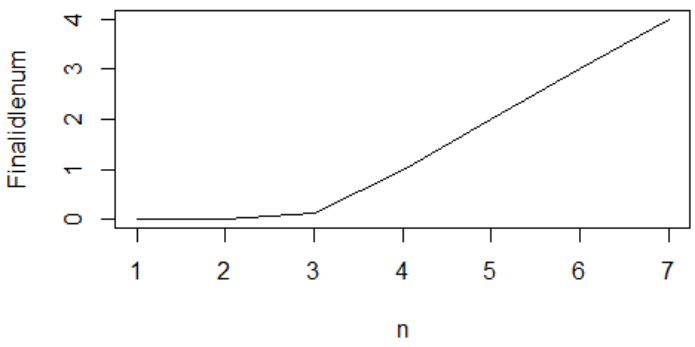
Table 2

In Table 1 and Table 2, the first row gives the number of processors. The second row gives the total expected cost per second. The third row gives the utilization of processing units. The fourth row gives the average number of IDLE processors per second. The fifth row gives the average number of WAITING jobs per second.

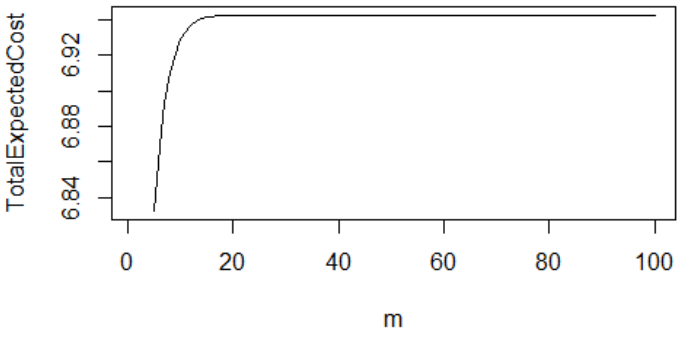
Multiplying the average number of processors at each state with the costs of one unit per second for each state, we get the expected cost of each state. Summing that up, we get the total expected costs per second. For n=7 in Table 1, we follow the said instructions, then we repeat for other values of n to find the optimal n and we get Table 2.



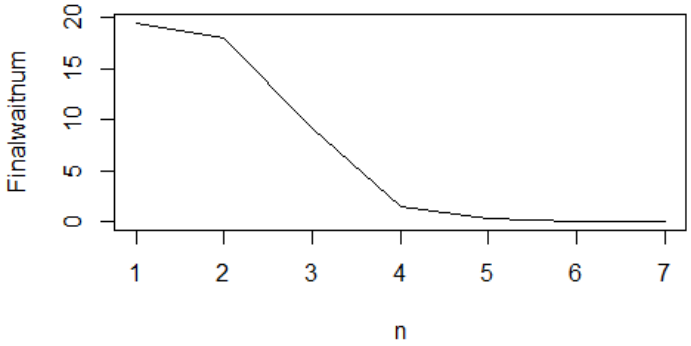
Graph 1 Total Expected Cost against number or processors



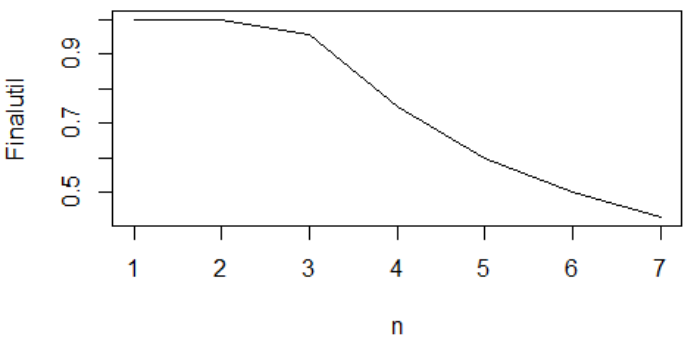
Graph 2 Average number of IDLE processors per second against number of processors



Graph 3 Total Expected Cost against maximum jobs in queue



Graph 4 Average number of WAITING jobs per second against number of processors



Graph 5 Utilization of processing units against number of processors

Graph 1 shows the total expected cost per second for different number of processors. From the graph, we can get the minimum total expected cost when n=5. We can see that Graph 2 slope upwards as n increases because there will the more processors to be left IDLE. From Graph 3, we know that the total expected cost will reach a stagnant value starting from m=20, so we can say that 20 is a sufficiently large number to use to get a correct answer, we can also get this inference from the graph of stationary distribution as well. Graph 4 slope downwards and approaches 0 because as n increases, the jobs will be assigned to processors instead of to the queue. Graph 5 slope downwards because that is when n is more than 5, the number of jobs done with still be the same.

4. ANALYSIS

According to the graphs and the table we know that the optimum number of cores is 5, which gives a minimum operational cost of 6.94 pence per second. On average at any given moment of time, 60% of the processing units will be WORKING.

With n=5 processors at any moment of time, we know that there will be approximately 2 processors will beat IDLE, therefore 3 processors will be WORKING. Otherwise, starting from 1 processor, average number of IDLE processors in a second is close to 0, since a new job arrives or comes from queue faster than a job can be done, so the only processor will always be working. And for 7 or more processors, the average number of IDLE processors in a second is more than half of the total number of processors, which shows that the workstation has more than enough processors. And because IDLE processors increase cost too, hence total expected cost increases.

Moreover, we can see that as the number of processors increases, the average number of WAITING jobs in a second decreases because there is more processors for job to be assigned to. Furthermore, we say that m can take a value of 20 because that is when the states are in stationary state and will not have any changes in proportion.

5. CONCLUSION

We would take the optimum number of cores at 5 in the workstation. This will give us a utilization of processing units at 0.60000, approximately 2 IDLE processors in a second, 0.35 jobs WAITING in queue in a second, and an operational cost of 6.94 pence per second. And we can assume m=20 as it is a sufficiently large number to get a correct answer.

APPENDIX A. OUTLINE OF CALCULATIONS

A.1. **Markov Model.** For any time t, let X(t) be the number of WORKING processors, Y(t) the number of IDLE processors, n the number of total processors. Since all processors will be in one of the states, so

$$X(t) + Y(t) = n$$

If there are new job arrivals or jobs WAITING and IDLE processors, the job will immediately be assigned to the said processor. So the processors will only be IDLE if there is no job.

There will also be a WAITING queue, where m is the maximum jobs in the queue, in this case we let m = 20. Also, if there is job WAITING in queue, we know that all the processors are WORKING. Therefore, the number of jobs at any given time in the system is less than or equal to

$$n + m$$

We are told that the job arrival follows a Poisson process with $\lambda = 0.5$ per second and length of each job follows Exp(6). These distributions are independent of the time. It follows that the probability distribution that the future number of WORKING processors only depends on the current number of WORKING processors in each state. So it follows that the process $\{X(t)\}_{t\geq 0}$ is a time homogeneous Markov process.

A.2. **Transition Rates.** The probability of a new job arrives within the short period of h is λ. The probability a job being done within the short period h is the probability of Exp(6) takes value less than or equal to h.

$$Q = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & -(\mu+\lambda) & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2\mu & -(2\mu+\lambda) & \lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6\mu & -(6\mu-\lambda) & \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7\mu & -(7\mu-\lambda) & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7\mu & -(7\mu-\lambda) & \lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7\mu & -(7\mu) \end{pmatrix}$$

Q is a transition matrix with state space S = {0, 1, …, n+m}, let i be the row number and j be the column number. From matrix Q, we can see that when i=1, the current state is 0 processors WORKING, meaning when the system moves from state 0 to state 1, there is a new job arrival with follows a Poisson process with rate λ and this Poisson process happens in each state except for when i=n+m+1, which is when the queue has maximum number of jobs and it rejects jobs. The length of each job follows Exp(6), and we have a mean of μ=1/6. From i=2 to i=7, the rate of job done multiples with the number of processors as the processors are processing jobs at rate following Exp(6). Starting from i=8 is 7 processors WORKING and the rates are the same because the same number of processing units are WORKING and lastly the matrix extends until i=n+m+1. Lastly, the diagonal is filled with the compliments, which are the negative of sum of the two existing elements, to fulfil the condition of the sum of a row must be 0, to achieve a transition rate matrix.

A.3. **Stationary Distribution.** Since all states intercommunicate, and there is a finite number of states (28), there will be a unique stationary distribution, which will be the solution of

$$\vec{\pi} Q = \vec{0}$$

with the constraint

$$\sum_{i=0}^{n+m} \pi_i = 1$$

We then get the stationary distribution by solving the equation. The value π_i is the probability that in the long run there are i number of processors WORKING at any given time. It also tells us the proportion of time the system will be at each state.

A.4. **Metrics of Interest.** We can use the stationary distribution to calculate the metrics of interest. The expected cost per second will be:

The expected proportion of WORKING processors at any given time is:

$$\sum_{i=0}^{n+m} \frac{i \times \pi_i}{n+m}$$

The expected number of processors per second in each state at any given moment of time i:

WORKING: $\sum_{i=0}^{n+m} i \times \pi_i$

IDLE: $\sum_{i=0}^{n-1} (n-i) \times \pi_i$

The expected number of WAITING jobs per second in each state at any given moment of time i:

$$\sum_{i=0}^{m-n} (n+i) \times \pi_{n+i}$$

APPENDIX B. COMPUTER CODE

```
lambda=0.5       #per second
lengthjob=6      #unit: second
mu=1/lengthjob   #parameter for exponential distribution
idlecost=1       #cost per second of an IDLE machine
workcost=1.5     #cost per second of a WORKING machine
waitcost=1.25    #cost per second of a WAITING job

j=100    #number of cores
Finalidlenum=rep(0,j)    #empty vector of average number of IDLE processors per second
Finalworknum=rep(0,j)    #empty vector of average number of WORKING processors per second
Finalwaitnum=rep(0,j)    #empty vector of average number of WAITING jobs per second
TotalExpectedCost=rep(0,j)     #empty vector of TotalExpectedCost
Finalutil=rep(0,j)

for(k in 1:j){   #repeat for different number of cores

m=100            #assumption
n=k              #number of processing units
```

```
#generate empty transition rate matrix Q
Q=matrix(data=0,nrow=n+m+1,ncol=n+m+1,dimnames=list(c(0:(n+m)),c(0:(n+m))),byrow=T)

#fill in Q matrix with transition rates
for(j in 1:(n+m+1)){
 for(i in 1:(n+m+1)){
  if(i+1==j){
   Q[i,j]=lambda
  }
  if(i==j+1 & i<=(n+1)){
   Q[i,j]=(i-1)*mu
  }
  if(i==j+1 & i>(n+1)){
   Q[i,j]=(n)*mu
  }
 }
}

for(j in 1:(n+m+1)){        #fill in diagonal with compliments
 for(i in 1:(n+m+1)){
  if(i==j){
   Q[i,j]=-1*sum(Q[i,])
   }
  }
 }

QT=t(Q)      #transpose of Q matrix

QT[(n+m+1),]=1     #replace the last row of matrix QT with 1

#solve utilization of the processing units
stndist=solve(QT)%*%(matrix(data=(c(rep(0,(n+m)),1)),(n+m+1),1))
stndist=c(stndist)       #get utilization in vector form
```

```
#plot graph of utilization as state increases
plot(stndist,type="l", main="Stationary Distribution")
min(stndist)       #get minimum of stationary distribution

idlenum=rep(0,n+m+1)       #empty vector for number of IDLE machines per second
worknum=rep(0,(n+m+1))     #empty vector for number of WORKING machines per second
waitnum=rep(0,m+1)         #empty vector for number of WAITING jobs per second

for(i in 1:(n+m+1)){       #calculate number of IDLE,WORKING,WAITING per second
 if(i<=(n+1)){
 idlenum[i]=((n+1)-i)*stndist[i]
 worknum[i]=(i-1)*stndist[i]
 }
 else{
 worknum[i]=n*stndist[i]
 waitnum[i]=(i-n-1)*stndist[i]
 }
}

#calculate the average number of IDLE and WORKING processors and WAITING jobs per second of the system
totalidlenum=sum(idlenum)
totalworknum=sum(worknum)
totalwaitnum=sum(waitnum)
util=totalworknum/n

totalcost=(totalidlenum*idlecost)+(totalworknum*workcost)+(totalwaitnum*waitcost)

Finalidlenum[k]=totalidlenum       #average number of IDLE processors per second for each k
Finalworknum[k]=totalworknum       #average number of WORKING processors per second for each k
Finalwaitnum[k]=totalwaitnum       #average number of WAITING jobs per second for each k
Finalutil[k]=util
TotalExpectedCost[k]=totalcost     #total expected cost for each k

}
Finalidlenum           #vector of average number of IDLE processors per second
```

Finalworknum #vector of average number of WORKING processors per second

Finalwaitnum #vector of average number of WAITING jobs per second

Finalutil

TotalExpectedCost #vector of expected costs

#plot graph of Finalidlenum,Finalworknum,Fianlwaitnum,TotalExpectedCost against n

plot(Finalidlenum,xlab="n", type="l")

plot(Finalworknum,xlab="n", type="l")

plot(Finalwaitnum,xlab="n", type="l")

plot(Finalutil,xlab="n", type="l")

plot(TotalExpectedCost,xlab="n",type="l")

min(TotalExpectedCost) #get minimum operational cost

which.min(TotalExpectedCost) #get optimal number of cores

#to find optimal m value, we need to change the below lines

Line 17: m=k #now we change the value of m

Line 18: n=5 #we fix value of n at the optimal value

Line 100: plot(TotalExpectedCost,xlab="m",type="l") #plot graph of total expected cost and m

REFERENCES

Minoccheri A. (2012 November 10). Fast food restaurant simulation: How to “queue” customers? Retrieved from https://stackoverflow.com/questions/13325082/fast-food-restaurant-simulation-how-to-queue-customers?rq=1

Srivastava T. (2016 April 28). How to predict waiting time using Queuing Theory? Retrieved from https://www.analyticsvidhya.com/blog/2016/04/predict-waiting-time-queuing-theory/