Analytic Models

Assignment 2

Mina Mousavifar - 11279515 - sem311

Question 1.

a)

Based on Little's law so far we have: $N = X(R + Z)$
Where N = number of users in system, X = access rate to webpage, R = mean residence time, Z = think time.
In addition access rate to disk is 16 $\frac{access}{second}$ and each webpage access needs 2 accesses to disk. So the access rate for a webpage is: $X_{webpage} = 16/2 = 8\ \frac{access}{second}$
So:

$$36 = 8 * (1.5 + Z) \rightarrow 4.5 = 1.5 + Z \rightarrow Z = 3seconds$$

b)

Based on Little's law and previous part: $N = X * Z$
Where N = average number of users thinking, X = access rate to webpage, Z = think time.

$$N = 3 * 8 = 24$$

Question 2.

a)

In order to maximize throughput of the system, we need to maximize utilization by keeping slower server busy without overwhelming it. So at time of T, the slower server serve 1 request and the faster server would serve $\frac{S_{slow}}{S_{fast}}$. Then the total of requests would be $1 + \frac{S_{slow}}{S_{fast}}$. For the faster server we have:

$$f = \frac{\frac{S_{slow}}{S_{fast}}}{1 + \frac{S_{slow}}{S_{fast}}} = \frac{S_{slow}}{S_{slow} + S_{fast}}$$

Moreover, we should check that $U_{slow} \leq 1$ which is satisfied.

b)

So as to minimize request delay, we need to minimize request time, which means users waiting for the faster server. So f should be 1 in this condition.

Question 3.

a)

Based on Forced flow law we have: $D_k = \frac{B_k}{C}$ where D is service demand, k is each service part, B is resource busy time and C is number of requests completed.

$$D_{processor} = \frac{B_{processor}}{C} = \frac{400}{5000} = 0.08seconds$$

$$D_{disk1} = \frac{B_{disk1}}{C} = \frac{500}{5000} = 0.1 seconds$$
$$D_{disk2} = \frac{B_{disk2}}{C} = \frac{600}{5000} = 0.12 seconds$$

So the system service demand would be:

$$D = \sum D_k = 0.08 + 0.1 + 0.12 = 0.3 seconds$$

Therefor we find that disk2 is our bottleneck with the most service demand.

b)

Processor is made 10x faster, so it's service demand has been changed: $D_{processor} = \frac{D_{processor}}{10} = \frac{0.08}{10} = 0.008 seconds$
and the system service demand has been changed: $D_{proc10x} = \sum D_k = 0.008 + 0.1 + 0.12 = 0.228 seconds$
Then we calculate N^* which is the intersection of heavy load and light load.

$$N^* = \frac{D + Z}{D_{max}} = \frac{0.3 + 2}{0.12} = \frac{2.3}{0.12} = 19.1\overline{6}$$

$$N^*_{proc10x} = \frac{D_{proc10x} + Z}{D_{max}} = \frac{0.228 + 2}{0.12} = \frac{2.228}{0.12} = 18.5\overline{6}$$

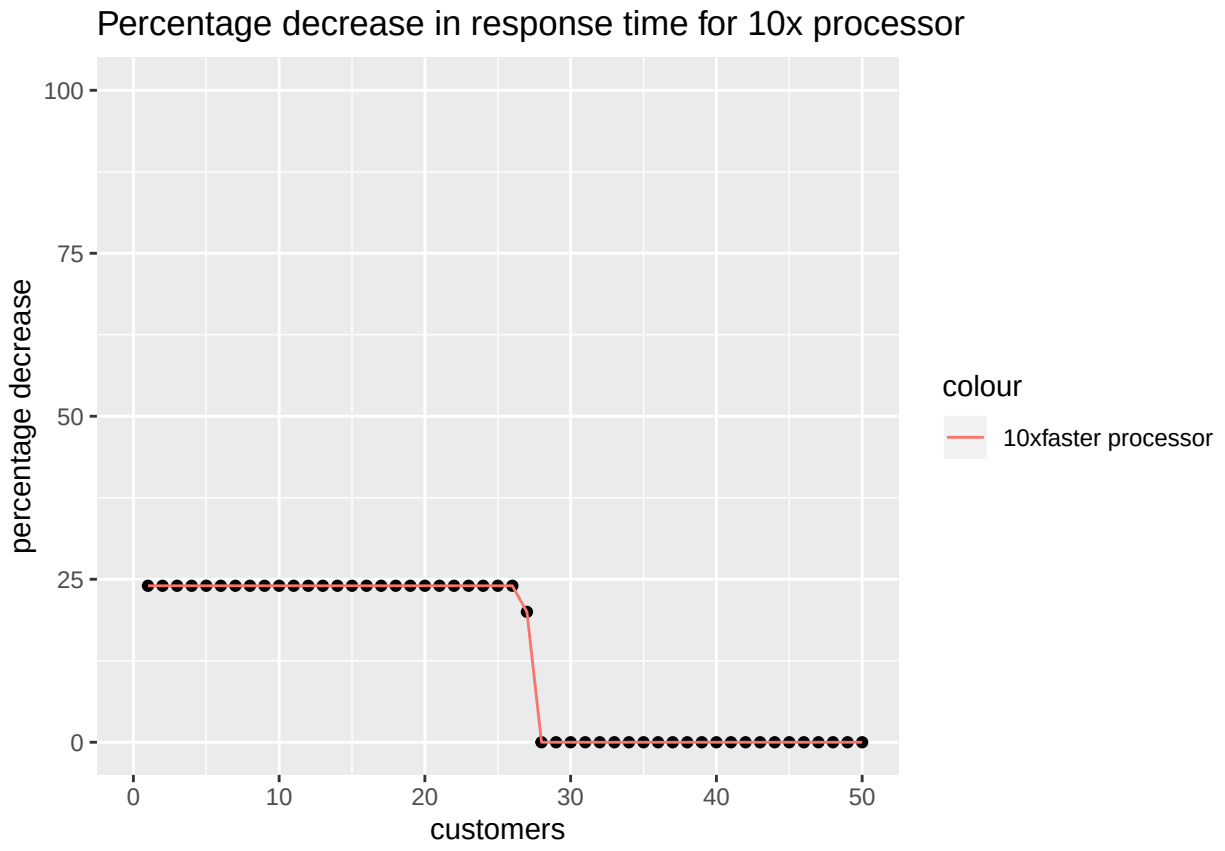
For bounds on throughput we have:

$$max(D, ND_{max} - Z) \leq R(N) \leq ND \rightarrow max(0.3, 0.12N - 2) \leq R(N) \leq 0.3N$$

$$max(D_{proc10x}, ND_{max} - Z) \leq R_{proc10x}(N) \leq ND_{proc10x} \rightarrow max(0.228, 0.12N - 2) \leq R(N) \leq 0.228N$$

```
library(dplyr)
# calculate lower bounds for original system
Rs <- data.frame("N"=1:50, "R"=rep(0,50), stringsAsFactors = FALSE)

for(i in 1:50){
  Rs[i,2] = max(0.3, (0.12*Rs[i,1]) - 3)
}

# calculate lower bounds for 10x faster processor
Rs10x <- data.frame("N"=1:50, "R"=rep(0,50), stringsAsFactors = FALSE)

for(i in 1:50){
  Rs10x[i,2] = max(0.228, (0.12*Rs10x[i,1]) - 3)
}

# calculate percentage decrease
des <- data.frame("N"=1:50, "change"=rep(0,50), stringsAsFactors = FALSE)

for(i in 1:50){
  des[i,2] = ((abs(Rs10x[i,2]-Rs[i,2]))*100)/Rs[i,2]
}
despre <- des

# plotting
library(ggplot2)
p <- ggplot(data = des, mapping = aes(x = N, y = change)) +
  xlim(0,50) + ylim(0, 100) +
```

```
  ggtitle("Percentage decrease in response time for 10x processor") +
  xlab("customers") + ylab("percentage decrease") +
  geom_point() +
  geom_line(data = des, aes(x = N, y = change, color = "10xfaster processor"))
p
```



c)

First we should calculate the new service demand time. Number of visits still stays the same and we have the same S_k because of our identical disks. Also, because of new load balancing new D_k would be the same too. So the demand would be:

$$V_1 + V_2 = \frac{D_1}{S} + \frac{D_2}{S} \rightarrow \frac{V_1 S}{S} + \frac{V_2 S}{S} + \frac{V_3 S}{S} = \frac{1}{S}(0.1 + 0.12) \rightarrow \frac{D_1 S}{S} + \frac{D_2 S}{S} + \frac{D_3 S}{S} = \frac{1}{S}(0.22)$$

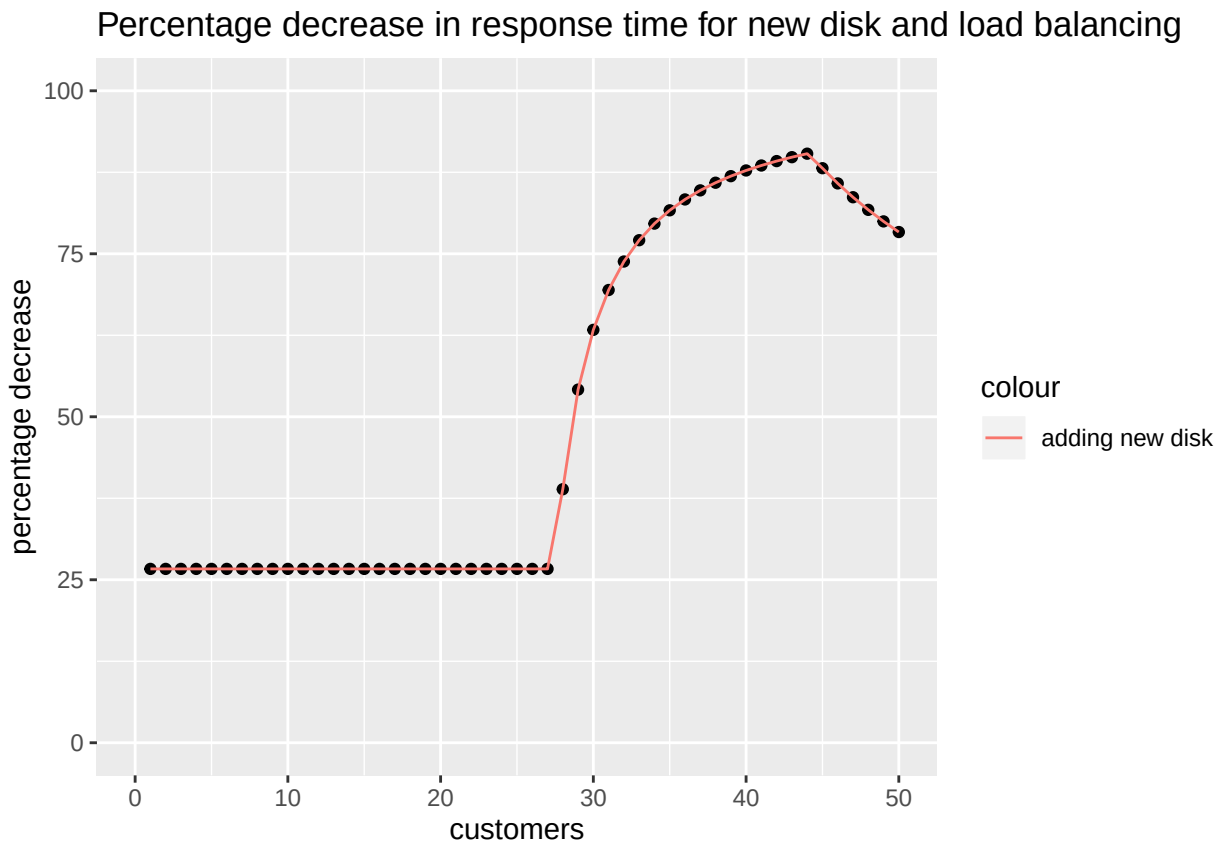
$$D_1 = D_2 = D_3 \rightarrow D_1(\frac{1}{S} + \frac{1}{S} + \frac{1}{S}) = \frac{1}{S}(0.22) \rightarrow D_1 * 3 = 0.22 \rightarrow D_1 = 0.07\overline{3}$$

Therefore the new bounderies would be: $D = \sum D_k = 0.22$ and $D_{max} = 0.07\overline{3}$

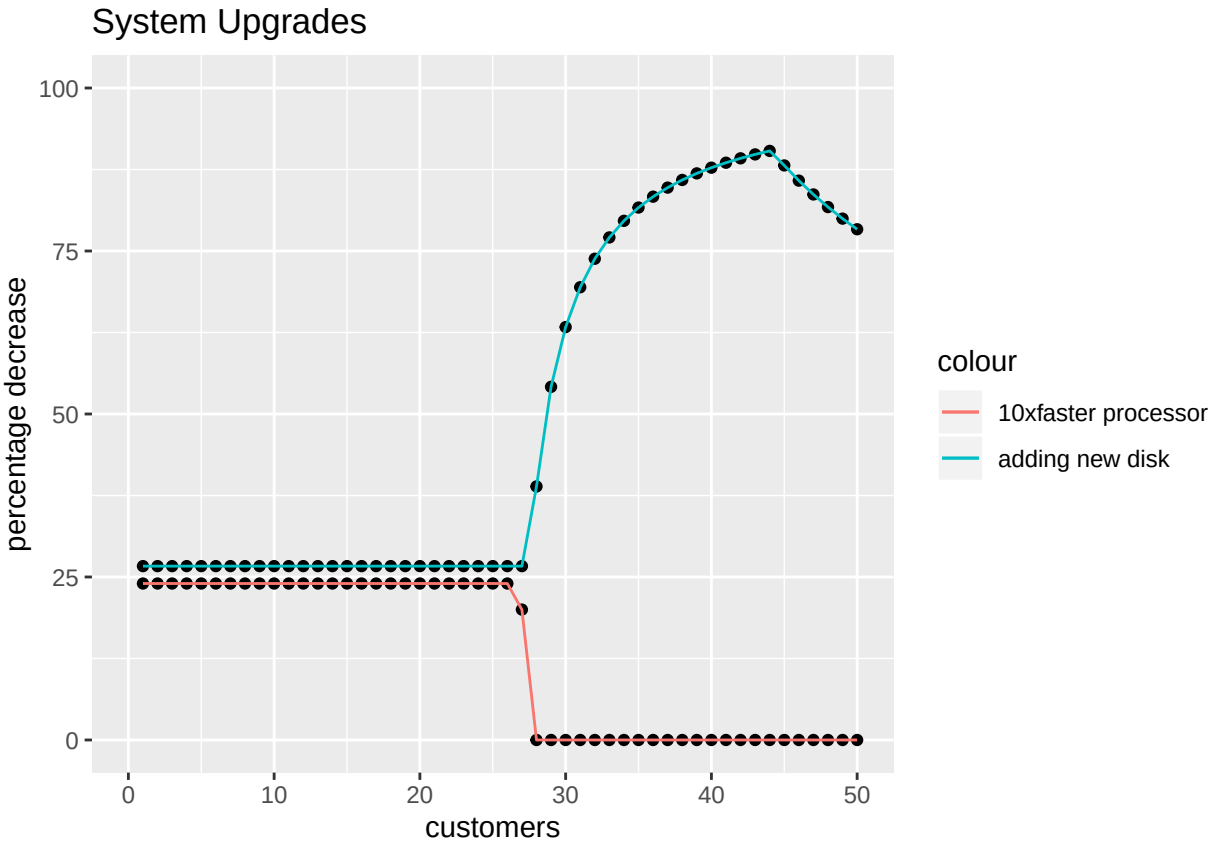
$$max(D_{newdisk}, ND_{max} - Z) \leq R_{newdisk}(N) \leq ND_{newdisk} \rightarrow max(0.22, 0.07\overline{3}N - 2) \leq R(N) \leq 0.07\overline{3}N$$

```
# calculate lower bounds for new disk
Rsd3 <- data.frame("N"=1:50, "R"=rep(0,50), stringsAsFactors = FALSE)

for(i in 1:50){
  Rsd3[i,2] = max(0.22, (0.073*Rsd3[i,1]) - 3)
}
```

```r
# calculate percentage decrease
des <- data.frame("N"=1:50, "change"=rep(0,50), stringsAsFactors = FALSE)

for(i in 1:50){
  des[i,2] = ((abs(Rsd3[i,2]-Rs[i,2]))*100)/Rs[i,2]
}

# plotting
library(ggplot2)
p <- ggplot(data = des, mapping = aes(x = N, y = change)) +
  xlim(0,50) + ylim(0, 100) +
  ggtitle("Percentage decrease in response time for new disk and load balancing") +
  xlab("customers") + ylab("percentage decrease") +
  geom_point() +
  geom_line(data = des, aes(x = N, y = change, color = "adding new disk"))
p
```




```r
p <- ggplot(data = des, mapping = aes(x = N, y = change)) +
  xlim(0,50) + ylim(0, 100) +
  ggtitle("System Upgrades") +
  xlab("customers") + ylab("percentage decrease") +
  geom_point() +
  geom_line(data = des, aes(x = N, y = change, color = "adding new disk")) +
  geom_point(data = despre, mapping = aes(x = N, y = change)) +
  geom_line(data = despre, aes(x = N, y = change, color = "10xfaster processor"))
p
```



So we can see that the change in the processor has minimal effect in light load which disappears in the heavy load. However, the change in the disks lead to a significant improvement, especially in heavy load condition. Because our bottleneck was the disk performance.

Question 4.

a)

$$U = \lambda * S = 8000\frac{packet}{second} * 0.1millisecond * \frac{1second}{1000millisecond} = 0.8$$

b)

Because of poisson arrival and exponential distributed service time we have M/M/1 queue. So we have:

$$R = \frac{S}{1-\lambda * S} = \frac{S}{1-U} = \frac{0.1millisecond * \frac{1second}{1000millisecond}}{1-0.8} = 0.0005 = 0.5millisecond$$

$$L_Q = \frac{U^2}{1-U} = \frac{0.8^2}{1-0.8} = 3.2$$

c)

Because of poisson arrival and deterministic service time we have M/D/1 queue. So we have:

$$R = S + \frac{S}{2}(\frac{\lambda S}{1-\lambda S}) = 0.0001 + 0.00005(\frac{0.8}{0.2}) = 0.0003 = 0.3millisecond$$

$$L_Q = \frac{1}{2}\frac{U^2}{1-U} = 0.5 * \frac{0.8^2}{1-0.8} = 1.6$$

Question 5.

a)

Because of poisson arrival and deterministic service time we have M/D/1 queue, so the residence time is:

$$R = \frac{S_{fast}}{2}\frac{2-U_{fast}}{1-U_{fast}} + \frac{S_{slow}}{2}\frac{2-U_{slow}}{1-U_{slow}} = 3\frac{2-U_{fast}}{1-U_{fast}} + 6\frac{2-U_{slow}}{1-U_{slow}}$$

The mean delay in the system is:

$$Delay = \frac{1}{2S_{fast}}\frac{2-U_{fast}}{1-U_{fast}} + \frac{1}{2S_{slow}}\frac{2-U_{slow}}{1-U_{slow}} = \frac{1}{3}\frac{2-U_{fast}}{1-U_{fast}} + \frac{1}{6}\frac{2-U_{slow}}{1-U_{slow}}$$

And based on Little's law the utilization is: $U_{fast} = f * \lambda * S$ and $U_{slow} = (1-f) * \lambda * S$.

```
r_rate = (1:9)*0.1
f_val = (0:100)*0.01

for(i in r_rate){
  optimum_point = -1 # finding best f for minimum delay
  optimum_r = -1  # finding best f for minimum residence time
  d_delay = 1000000
  r_delay = 1000000
  for(j in f_val){
     # check utilization conditions
    Uf = j*i*1.5
    Us = (1-j)*i*3
    if (Uf <= 1 && Us <= 1){
      d = ((2-Uf)/(3*(1-Uf))) + ((2-Us)/(6*(1-Us)))
      r = ((3*(2-Uf))/(2*(1-Uf))) + ((6*(2-Us))/(2*(1-Us)))
      if (d < d_delay){
        d_delay = d
        optimum_point = j
      }
      if(r < r_delay){
        r_delay = r
        optimum_r = j
      }
    }
  }
  cat("Request rate=", i, "   optimum f=", optimum_point,"   optimum r=", optimum_r, "\n")
}
```

```
## Request rate= 0.1    optimum f= 0.67    optimum r= 1
## Request rate= 0.2    optimum f= 0.67    optimum r= 1
## Request rate= 0.3    optimum f= 0.67    optimum r= 1
## Request rate= 0.4    optimum f= 0.67    optimum r= 0.92
## Request rate= 0.5    optimum f= 0.67    optimum r= 0.83
## Request rate= 0.6    optimum f= 0.67    optimum r= 0.78
## Request rate= 0.7    optimum f= 0.67    optimum r= 0.74
## Request rate= 0.8    optimum f= 0.67    optimum r= 0.71
## Request rate= 0.9    optimum f= 0.67    optimum r= 0.69
```

This shows that if we are in light load f should be 1 and the heavier the load gets, our output converges to $f = 0.67$ which is the formula provided in question 2.

$$f = \frac{S_{slow}}{S_{slow} + S_{fast}} = \frac{3}{3 + 1.5} \simeq 0.67$$

b)

First we find the mean service time of these two servers: $S = \frac{k\alpha}{\alpha - 1}$

$$S_{fast} = \frac{\frac{5}{6} * 2.25}{2.25 - 1} = \frac{1.875}{1.25} = 1.5$$

$$S_{slow} = \frac{\frac{5}{3} * 2.25}{2.25 - 1} = \frac{1.875}{1.25} = 3$$

Moreover, remaining service time equals to second moment which is: $\frac{k^2 * \alpha}{\alpha - 2}$

$$S_{remfast} = \frac{(\frac{5}{6})^2 * 2.25}{2.25 - 2} = \frac{1.875}{0.25} = 6.25$$

$$S_{remslow} = \frac{(\frac{5}{3})^2 * 2.25}{2.25 - 2} = \frac{1.875}{0.25} = 25$$

And for M/G/1 queue we have:

$$R = \frac{-U_{fast} * S_{fast} + U_{fast} * S_{remfast} + S_{fast}}{1 - U_{fast}} + \frac{-U_{slow} * S_{slow} + U_{slow} * S_{remslow} + S_{slow}}{1 - U_{slow}}$$

```
library(PtProcess)
Sremf = 6.25
Srems = 25
for(i in r_rate){
  optimum_point = -1
  delay = 1000000
  Sf = rpareto(1, 5/6, 2.25)
  Ss = rpareto(1, 5/3, 2.25)
  for(j in f_val){
     # check utilization conditions
    Uf = j*i*Sf
    Us = (1-j)*i*Ss
    if (Uf <= 1 && Us <= 1){
      r = ((Uf*Sremf + Sf - Uf*Sf)/(1-Uf)) + ((Us*Srems + Sf - Us*Ss)/(1-Us))
      if (d < delay){
        delay = d
        optimum_point = j
      }
    }
  }
  cat("Request rate=", i, "   optimum f=", optimum_point,"\n")
}
```

```
## Request rate= 0.1    optimum f= 0
## Request rate= 0.2    optimum f= 0
## Request rate= 0.3    optimum f= 0.19
## Request rate= 0.4    optimum f= -1
```

```
## Request rate= 0.5    optimum f= -1
## Request rate= 0.6    optimum f= -1
## Request rate= 0.7    optimum f= -1
## Request rate= 0.8    optimum f= -1
## Request rate= 0.9    optimum f= -1
```

Example output: Request rate= 0.1 optimum f= 0
Request rate= 0.2 optimum f= 0.35
Request rate= 0.3 optimum f= 0.18
Request rate= 0.4 optimum f= 0.2
Request rate= 0.5 optimum f= 0.2
Request rate= 0.6 optimum f= 0.4
Request rate= 0.7 optimum f= -1
Request rate= 0.8 optimum f= -1
Request rate= 0.9 optimum f= -1

Because of the variance in the service time, the queue might get longer, and the difference between two servers is also smaller. So we need to apply more balanced traffic between servers.