Keyan Vakil – CS350 HW2

1. Overall hit rate = 95% (hit in L1 or miss L1, hit L2)
L1 hit rate = 75%
L2 hit rate = 90%

a. The caching is independent since a hit in L1 doesn't necessarily influence whether the data is in L2. Additionally, a miss in L1 doesn't necessarily influence whether the data is in L2. The same statements could be said when comparing L2's influence on L1. This can be determined since, if they were directly dependent, the overall hit rate would be L1 hit rate + L1 miss rate*L2 hit rate = 0.75, 0.25*0.9 = 0.975 != 0.95. (see assumptions below)

b. Since we are assuming that the L1 cache is independent of the L2 cache, the probability of data being located in both caches can be represented by the hit rate of L1*hit rate of L2.
Pr(L1 hit) = .75; Pr(L2 hit) = 0.90; Pr(L1 hit and L2 hit) = Pr(L1 hit)*Pr(L2 hit)
.75*.9 = 0.675 = 67.5%

c. Assuming the requests are random and the system has been running for long enough to receive a sufficient number of requests (reached a stable state, the probability that 10 requests will result in no overall misses is (.95)^10 = .60 = 60%

d. Using the same assumptions in c, the probability that there will be exactly one miss is:
10C1*((0.05)*(0.95)^9) = 0.32 = 32%
Note: 10C1 represents the number of combinations of 1 miss (0.05^1) and 9 hits (0.95^9)

e. Using the same assumptions in c, the probability that there will be exactly two misses is:
(10C2)*(((0.05)^2)*(0.95)^8) = 0.075 = 7.5%
Note: 10C2 represents the number of combinations of 2 misses (0.05^1) and 8 hits (0.95^9)

f. The average number of requests that will be successfully served in a row from L1 for the 10 requests is (assuming this is a Geometric distribution) before a miss is 1/(1-.75) = 4. Thus, the average number of requests that will be successfully served in a row should actually be 4 – (1 miss) = 3 hits in a row.

g. The average number of requests that will be served by the cache system out of 100 requests is 100*(.95) = 95 requests.

h. Assuming that the number of requests is sufficiently large to be distributed approximately normally, the probability that out of 1000 requests at least 960 of the requests will result in overall hits is the probability that the z-score is greater than the z-score corresponding to the 96th percentile of this distribution:
mean = 1000*.95 = 950 requests
sd = sqrt(mean*probability of success*probability of failure)
sd=sqrt(950*(.95*.05)) = 6.72
z = (score - mean)/sd
To find the probability that at least 960 of the requests will be processed.
z > (960 - 950)/6.72

z > 1.49
Pr(z<1.49) --> .9319
Pr(z>1.49) = 1 - .9319= .0681
6.81%

i. Technically, for problems a and b, the answers may depend on the design of the cache. "Some processors use an inclusive cache design (meaning data stored in the L1 cache is also duplicated in the L2 cache) while others are exclusive (meaning the two caches never share data)." (http://www.extremetech.com/extreme/188776-how-l1-and-l2-cpu-caches-work-and-why-theyre-an-essential-part-of-modern-chips). For the above problems I assumed that the caches are independent since if they were dependent in some simple way, the overall hit rate probability would not match up with the given one. For problems c, d, and e, I assumed that the requests are random and the system is at a stable state (ie not instantiating itself or cleaning up data). For part f, I assumed that the distribution of requests is geometric since the events are assumed to be independent of each other. For part h I assumed that the distribution of the requests can be represented as a normal distribution. For all the problems I assumed that the givens are perfectly representative of the caching system in the future.

2. Arrival is independent. Average inter-arrival time = 50 milliseconds. --> lambda = 1 request/50 ms = 0.02 requests/ms

a. Assuming that all of the requests are correctly serviced immediately, the throughput of the server is 1 request per 50 ms = 0.02 requests/ms

b. Assuming the random variable of requests follows an exponential distribution, f(x) = lambda*e^(-lambda*x), where lambda=0.02 requests/ms

c. Assuming arrivals follow a the distribution in b, the CDF of a the distribution would be f(x) = 1 – e^(-lambda*x) where lambda=.02 requests/ms. We can use this CDF to determine the probability that the requests would be separated from one another by less than 80 ms and then take 1-that probability to find out the probability that they would be separated by more than 80 ms.
1 – f(x) = 1 – (1 – e^(-lambda*x)) = e^(-0.02*80) = 0.2019
There is a 20.129% chance that the requests would be separated by more than 80 ms.

d. Assuming the distribution is the same as in part b, the mean would be equal to the standard deviation, since the means of all exponential distributions by definition have the same value as the standard deviation. Since the mean of this data is 50 ms, the standard deviation must also be 50ms.

e. The probability distribution function can be represented as a Poisson distribution, following the function, f(x) = (((lambda*t)^x)/(x!))*e^(-lambda*t) where x represents the number of packets received within 100ms, lambda is the arrival rate = 0.02 requests/ms, and t is the timeframe = 100ms. Thus, f(x) = (((0.02*100)^x)/(x!))*e^(-0.02*100)

f. Assuming that the distribution in e is correct, Pr(>=3 packets) = 1- Pr(0 packets) – Pr(1-packet) – Pr(2 Packets)
f(0) = (((0.02*100)^0)/(0!))*e^(-0.02*100) = 0.135
f(1) = (((0.02*100)^1)/(1!))*e^(-0.02*100) = 0.271
f(2) = (((0.02*100)^2)/(2!))*e^(-0.02*100) = 0.271
Pr(>= 3) = 1- 0.135 – 0.271 – 0.271 = 0.323
There is a 32.3% of receiving 3 or more packets in 100 ms.

3. Single threaded buffer, service time per request is exponential, u = 1 request/48 ms = 0.02 requests/ms, uses server in previous problem

rho = utilization of system = rate of arrivals/rate of service

rho = (1 request/50 ms)/(1 request/48 ms) = 0.96 ---> The server is at a steady state

a. The capacity of the HTTP daemon is 1/48 requests/ms = 0.02 requests/ms

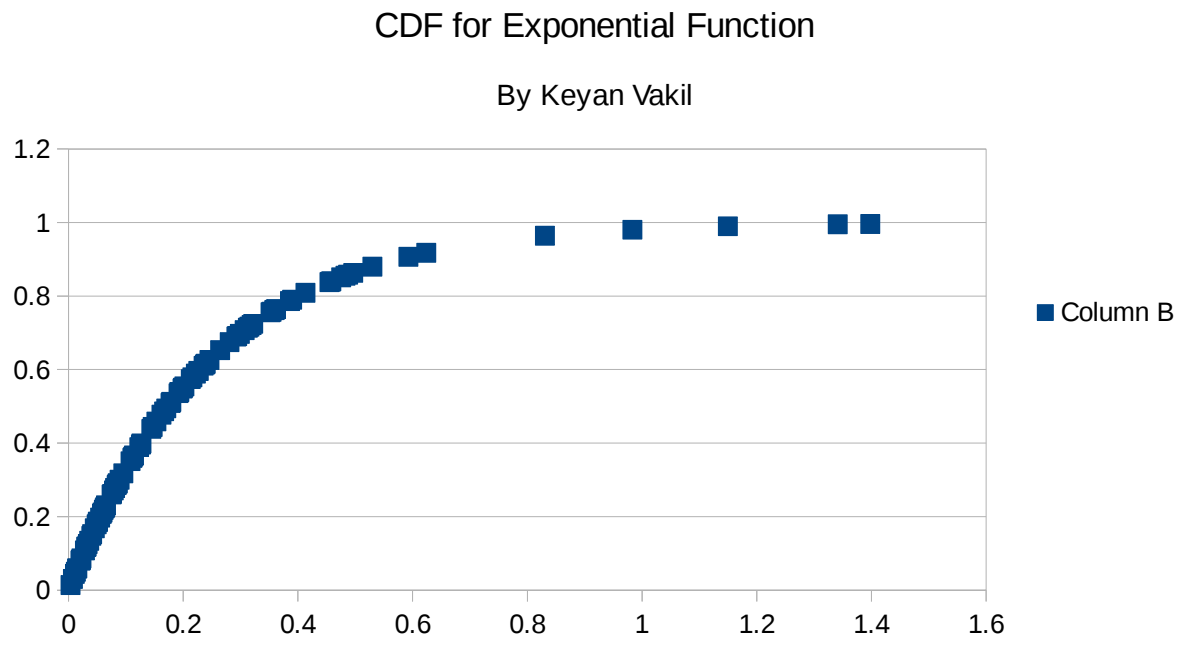
b. The probability distribution of the number of HTTP requests queued up for service or being serviced would be f(x) = (1-p)*(p)^x where p = rho = 0.96 and x = the number of requests waiting to be serviced. (1-p) represents the probability of the current request being serviced, and we would multiply that by the probabilities of each of the other requests in the queue to the power of the number of iterms in the queue.

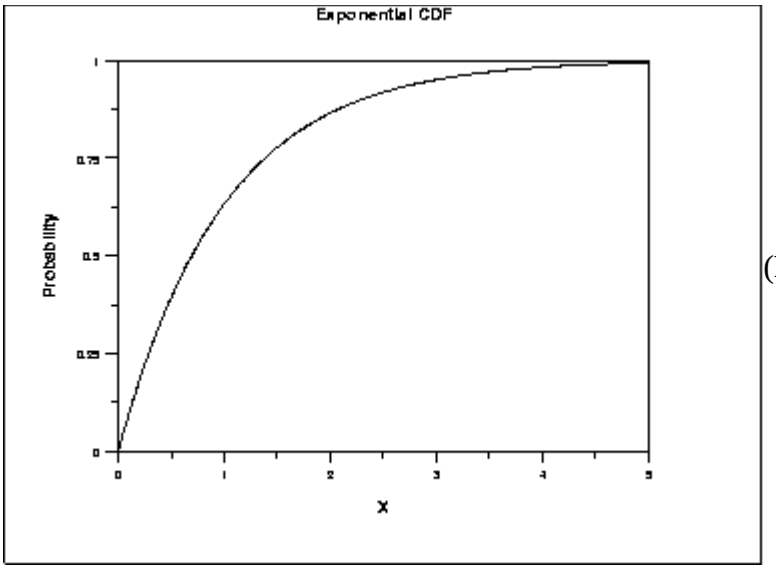
c. w = rho^2/(1-rho) = (0.96)^2/(1-0.96) = 23.04 requests

d. Tq = 1/(u*(1-rho)) = 1/((1/48)*(1-0.96)) = 1200 ms

e. Slowdown = 1/(1-rho) = 1/(1-0.96) = 25

4.

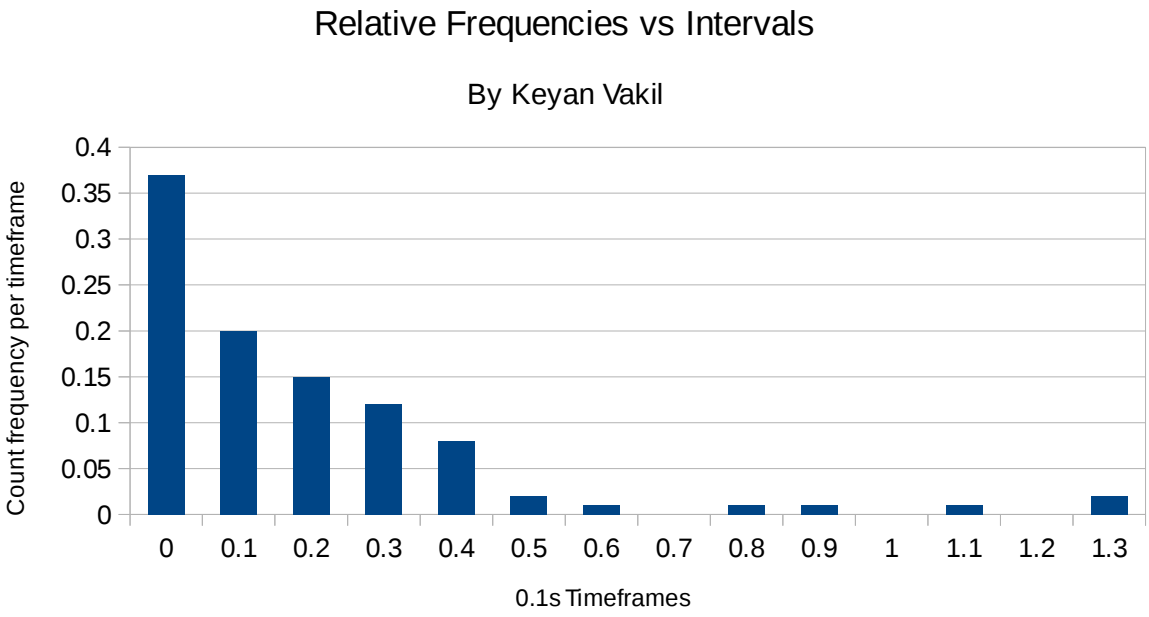




(From http://www.itl.nist.gov/div898/handbook/eda/section3/gif/expcdf.gif)

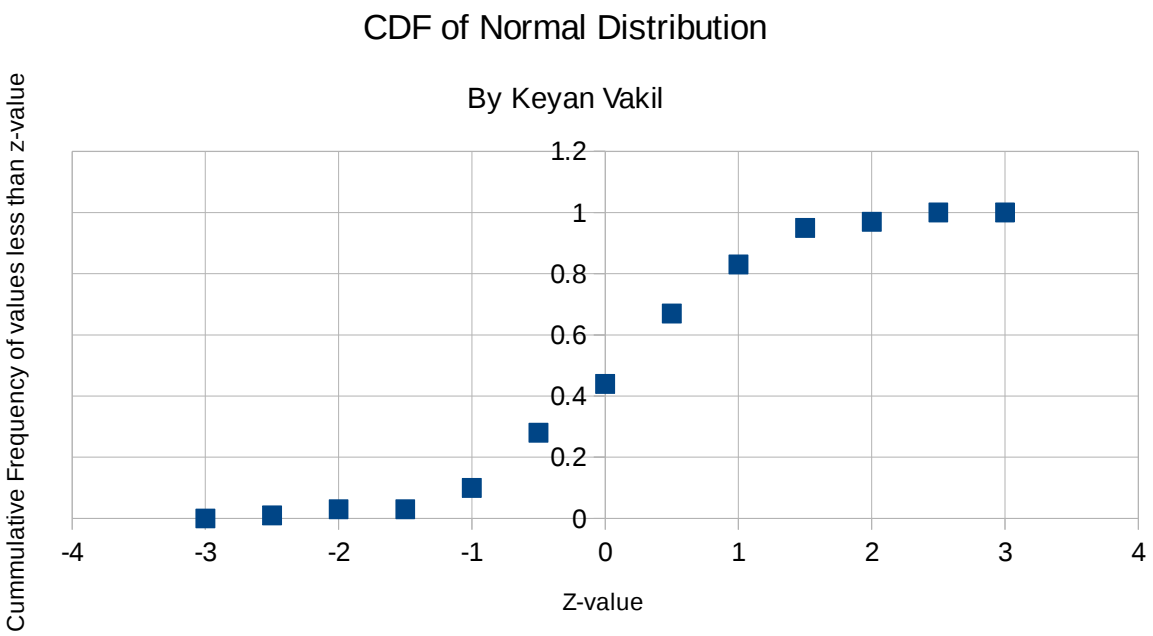
a. My CDF looks logarithmic in nature and, indeed appears very similar, to the analytical CDF of the Exponential Function. Both have x values representing the random variable in question, although my range of x values is smaller due to the nature of my variables. The slope of the probabilities both seem to decrease as the x values get higher, and both approaches 1 asymptotically, which is to be expected of CDFs in general.

b.



This graph represents a Poisson distribution since we applied the Poisson Process to exponential data. The formula that best characterizes this histogram is: f(x) = (((lambda*t)^x)/x!)*e^(-lambda*t), where x = 0, 0.1, 0.2 …, t = 0.1, and lambda = 4.

5.



a.
My Z-values:

Pr(z<0) = 0.44
Pr(z<1) = 0.83
Pr(z<2) = 0.97
Pr(z<3) = 1.00

Actual Z-values:

Pr(z<0) = 0.5
Pr(z<1) = 0.8413
Pr(z<2) = 0.9772
Pr(z<3) = 0.9987

Overall, considering n=100, I would say my z-values are quite accurate, since more variation in the probabilities is to be expected near z=0 and the variation in probabilities will decrease as the absolute value of z increases. The reason my value is similar to the actual z-values is that z-values are actually normalized random variables that represent the CDF of the normal distribution commonly used in statistics. Thus, since my graph was generated through using the CLT, which postulates that any distribution that follows the process in the CLT can be represented as a normal distribution, it should not be surprising that my graph is an accurate representation of the normal distribution.

b.
My program computed the probability to be 0.3500 = 35.00%.

Analytical answer:
Mean = 72; StdDev = 16; LowerScore = 66; UpperScore = 80;
LowerZScore = (66-72)/16 = -0.375
UpperZScore = (80-72)/16 = 0.500
Find: Pr(-0.375 < z < 0.500)

$Pr(-0.375 < z < 0.500) = Pr(z < 0.500) – Pr(z < -0.375)$
$Pr(z < 0.500) = 0.6915$; $Pr(z < -0.375) = 0.3557$
$Pr(-0.375 < z < 0.500) = 0.6915 – 0.3557 = 0.3358$
33.58 %

Considering how n=100, the probability I calculated using my random variable is very close to the actual probability using actual z-scores.