CS350 Fundamentals of Computing Systems

Fall 2024

15th Oct 2023, Discussion 6

A government service has been initially deployed a single-queue, single-server system. Call the server S0. The service receives on average, at steady state, about 8280 requests per hour. And it turns out that the number of requests issued by the citizens using the service is Poisson-distributed. You have benchmarked that the service time at steady state is exponentially distributed with an average of 0.4 seconds. Your job is to cost-efficiently optimize the system.

A government service has been initially deployed a single-queue, single-server system. Call the server S0. The service receives on average, at steady state, about 8280 requests per hour. And it turns out that the number of requests issued by the citizens using the service is Poisson-distributed. You have benchmarked that the service time at steady state is exponentially distributed with an average of 0.4 seconds. Your job is to cost-efficiently optimize the system.

(a) **[6 points]** First off, compute the average queue length that the typical user will experience at steady state upon submitting a request?

A government service has been initially deployed a single-queue, single-server system. Call the server S0. The service receives on average, at steady state, about 8280 requests per hour. And it turns out that the number of requests issued by the citizens using the service is Poisson-distributed. You have benchmarked that the service time at steady state is exponentially distributed with an average of 0.4 seconds. Your job is to cost-efficiently optimize the system.

(a) **[6 points]** First off, compute the average queue length that the typical user will experience at steady state upon submitting a request?

Unify unit first!

Solution:

First, we compute $\lambda = 8280/(60 * 60) = 2.3$ req./second.

Next, we recognize that $T_s = 0.4$ seconds.

For this part, we first compute $\rho = \lambda \cdot T_s = 0.92$

Finally, we can compute $q = \frac{\rho}{1-\rho} = \frac{0.92}{1-0.92} = 11.5$.

NOTE: Solutions that computed w instead of q are also valid!

(b) **[6 points]** In an attempt to optimize the performance of the system, you propose the purchase of one more server, call it S1, with identical specifications as S0. The idea is to have all the user requests enqueued in a single, central queue. The request at the head of the queue is then served by S0 or S1, as soon as one of them becomes available. Because the two servers are identical, each will still process a generic request in 0.4 seconds, on average. What queue length will a generic user experience upon submitting a request?

(b) **[6 points]** In an attempt to optimize the performance of the system, you propose the purchase of one more server, call it S1, with identical specifications as S0. The idea is to have all the user requests enqueued in a single, central queue. The request at the head of the queue is then served by S0 or S1, as soon as one of them becomes available. Because the two servers are identical, each will still process a generic request in 0.4 seconds, on average. What queue length will a generic user experience upon submitting a request?

- M/M/N System: $q = C\frac{\rho}{1-\rho} + N\rho$, $w = C\frac{\rho}{1-\rho}$,
  where $\rho = \frac{\lambda T_s}{N} = \frac{\lambda}{N\mu}$, $C = \frac{1-K}{1-\rho K}$, and $K = \frac{\sum_{i=0}^{N-1} \frac{(N\rho)^i}{i!}}{\sum_{i=0}^{N} \frac{(N\rho)^i}{i!}}$.

(b) **[6 points]** In an attempt to optimize the performance of the system, you propose the purchase of one more server, call it S1, with identical specifications as S0. The idea is to have all the user requests enqueued in a single, central queue. The request at the head of the queue is then served by S0 or S1, as soon as one of them becomes available. Because the two servers are identical, each will still process a generic request in 0.4 seconds, on average. What queue length will a generic user experience upon submitting a request?

- M/M/N System: $q = C\frac{\rho}{1-\rho} + N\rho$, $w = C\frac{\rho}{1-\rho}$,

  where $\rho = \frac{\lambda T_s}{N} = \frac{\lambda}{N\mu}$, $C = \frac{1-K}{1-\rho K}$, and $K = \frac{\sum_{i=0}^{N-1} \frac{(N\rho)^i}{i!}}{\sum_{i=0}^{N} \frac{(N\rho)^i}{i!}}$.

Solution:

In this case, we are dealing with an M/M/2 system.

In this case, the utilization of each server is $\rho' = \frac{\lambda T_s}{2} = 0.46$.

We can compute $K = \frac{1+2\rho'}{1+2\rho'+\frac{(2\rho')^2}{2}} = \frac{1.92}{2.3432} = 0.819$

Next, we compute $C = \frac{1-K}{1-\rho' K} = \frac{0.181}{0.623} = 0.29$

And finally $q' = C\frac{\rho'}{1-\rho'} + N\rho' = 0.29 \cdot 0.85 + 0.92 = 1.167$

NOTE: Solutions that computed $w' = q' - N\rho'$ instead of q' are also valid!

(c) **[6 points]** An alternative is to decommission the already-deployed server S0 and to purchase 4 new servers. The 4 new servers (all together!) cost half than S1 because they are much slower. Indeed, each will process a request in 0.9 seconds, on average. In this case, you propose that requests are randomly pre-split to the 4 servers, thus each server now has a per-server queue. What queue length will a generic user experience upon submitting a request?

(c) **[6 points]** An alternative is to decommission the already-deployed server S0 and to purchase 4 new servers. The 4 new servers (all together!) cost half than S1 because they are much slower. Indeed, each will process a request in 0.9 seconds, on average. In this case, you propose that requests are randomly pre-split to the 4 servers, thus each server now has a per-server queue. What queue length will a generic user experience upon submitting a request?

Solution:

The new organization can be modeled using a 4*M/M/1 system model.

The new $T_s' = 0.9$ sec.

In this case, we recompute the per-server utilization as: $\rho'' = \frac{\lambda T_s'}{4} = 0.5175$

A generic user will be enqueued in one of these M/M/1 systems, thus the queue length they will experience is: $q'' = \frac{\rho''}{1-\rho''} = 1.073$.

NOTE: Solutions that computed w'' instead of q'' are also valid!

(d) **[5 points]** Which solution between what you have considered in Part **(b)** and Part **(c)** is more desirable? Motivate your answer.

(d) [5 points] Which solution between what you have considered in Part **(b)** and Part **(c)** is more desirable? Motivate your answer.

Solution:

The solution proposed in Part **(c)** is more desirable because it's ~~both the~~ cheaper option ~~and provides better quality of service to the citizens, if the length of the queue as perceived by a new customer is the metric chosen.~~

On the other hand, if we consider the averege response time as the chosen metric to reason abot the system's quality of service we have:

1. For Part **(b)**: $T'_q = q'/\lambda = 1.167/2.3 = 0.5$ sec.
2. For Part **(c)**: $T''_q = \frac{q''\cdot 4}{\lambda} = 1.866$ sec.

If we look at the response time, the 4 servers are half the cost of S1, but the response time seen by the users would be $3.7\times$ worse compared to the solution proposed in Part **(b)**. Thus, one could conclude that the solution in part Part **(b)** is a better cost/performance trade-off.

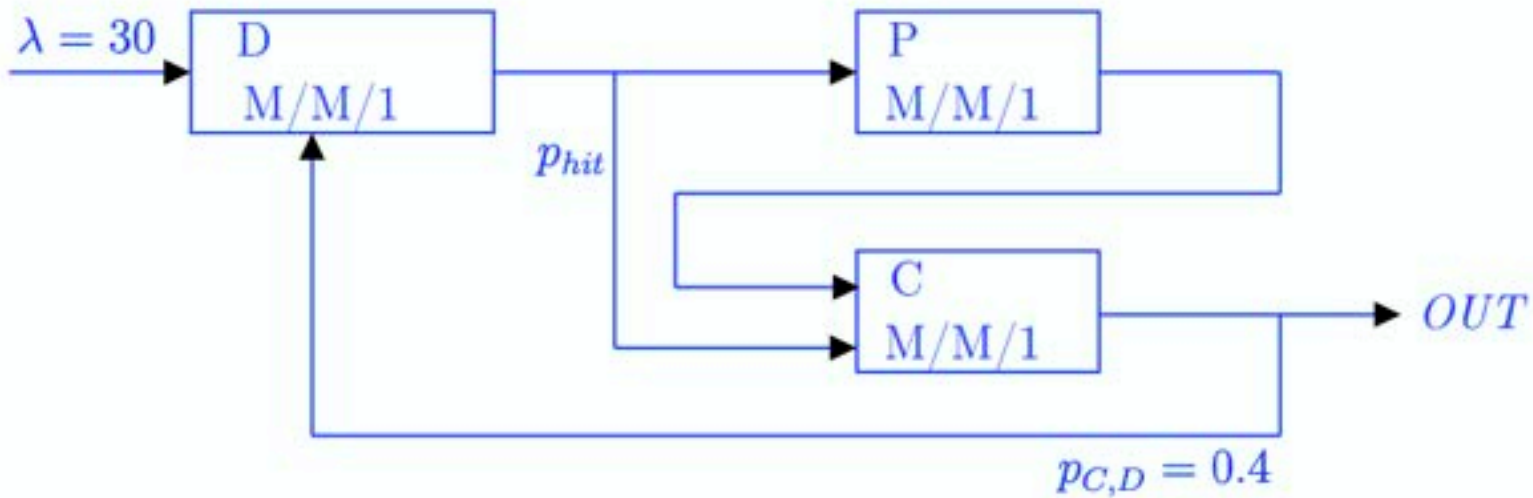
NOTE: Solutions that reasoned on w' vs. w'' and/or T'_w vs. T''_w are also valid. The conclusions are the same.

A request processing system is comprised of three machines. Requests always arrive at the first machine called the decoder (D). The decoder understands a request's payload and also checks if the response is already available in the system cache. If the response is not in the cache (cache miss), the request is sent from D to the processing engine (P) for full processing. After full processing, the request is sent from P to the cache machine (C) so that the response produced by P can be added to the cache. Conversely, in case of a cache hit at D, the request is directly sent from D to C for fast response retrieval—in this case, the request essentially bypasses P. Cache hits at D occur with probability p_{hit}. Regardless of how the request reached the C machine, after being served at C, the request might request additional processing from the system. This happens with probability 0.4, in which case it will go back to the decoder (D) as if it was a new request. Otherwise, it leaves the system and it is considered completed. On average, the system receives 30 requests per second. The average service time at D is 11 milliseconds; 28 milliseconds at P; and 9 milliseconds at C.

(a) [2 points] Provide a diagram of the system which shows the inter-connections between the D, P, and C machines. Describe the queuing models used for each machine, the known parameters, and the assumptions you will employ to solve the system.

(a) **[2 points]** Provide a diagram of the system which shows the inter-connections between the D, P, and C machines. Describe the queuing models used for each machine, the known parameters, and the assumptions you will employ to solve the system.

Solution:



Assumptions:

(1) System is at steady state;

(2) All queues infinite and with FIFO discipline;

(3) Arrivals from the outside are Poisson;

(4) All service times are exponentially distributed;

(b) [5 points] Assume that the probability p_{hit} of a cache hit (a.k.a. hit rate) is 30%. What resource constitutes the bottleneck in the system? Motivate your answer.

(b) **[5 points]** Assume that the probability p_{hit} of a cache hit (a.k.a. hit rate) is 30%. What resource constitutes the bottleneck in the system? Motivate your answer.

Solution:

The arrival rate from the outside is $\lambda = 30$ req./sec.

After completing at C, requests can go back to D with probability $p_{C,D} = 0.4$.

We can find $\lambda_D = \lambda + p_{C,D}\lambda_D$. It follows that $\lambda_D = \frac{\lambda}{1-p_{C,D}} = \frac{30}{0.6} = 50$ req./sec.

Since any request going through D also goes through C, we have $\lambda_C = \lambda_D$.

Conversely, if we call the cache hit rate p_{hit}, $\lambda_P = (1 - p_{hit})\lambda_D = 50 \cdot 0.7 = 35$ req/sec.

We can compute all the utilizations.

$\rho_D = \lambda_D \cdot T_{s,D} = 50 \cdot 0.011 = 0.55$;

$\rho_C = \lambda_C \cdot T_{s,C} = 50 \cdot 0.009 = 0.45$;

$\rho_P = \lambda_P \cdot T_{s,P} = 35 \cdot 0.028 = 0.98$;

Thus, P is definitely the bottleneck since it has the highest utilization.

(c) **[5 points]** With the same hit rate, what is the average response time of the entire system?

(c) **[5 points]** With the same hit rate, what is the average response time of the entire system?

Solution:

For this part, we first compute the various q.

$q_D = \rho_D/(1-\rho_D) = 1.22$ requests;

$q_C = \rho_C/(1-\rho_C) = 0.82$ requests;

$q_P = \rho_P/(1-\rho_P) = 49$ requests;

$q_{tot} = q_D + q_C + q_P = 51.04$ requests; and

$T_{q,tot} = q_{tot}/\lambda = 21.04/30 = 1.7$ seconds.

(d) [5 points] With the same hit rate, what is the capacity of the system?

(d) [5 points] With the same hit rate, what is the capacity of the system?

Solution:

The capacity is the value of throughput λ^* at which the bottleneck reaches 100% utilization.

We can setup the following equation: $\rho_P = \lambda_P \cdot T_{s,P} = 1$

We know that $\lambda_P = 0.7\lambda_D = \frac{(1-p_{hit})\lambda^*}{0.6} = \frac{0.7\lambda^*}{0.6}$.

Thus, we can write $\frac{0.7\lambda^*}{0.6} T_{s,P} = 1$ and solve for λ^*.

It follows that $\lambda^* = \frac{0.6}{0.7T_{s,P}} = 30.6$ req./sec.

(e) **[5 points]** Suppose you were able to introduce a new caching heuristic that increases the hit rate p_{hit} from the previous 30% to 50%. What is the resulting increase in system capacity?

(e) **[5 points]** Suppose you were able to introduce a new caching heuristic that increases the hit rate p_{hit} from the previous 30% to 50%. What is the resulting increase in system capacity?

Solution:

The previous capacity was $\lambda^* = 30.6$, which was computed as $\lambda^* = \frac{0.6}{(1-p_{hit}^{old})T_{s,P}} = 30.6$ req./sec.

With the new heuristic, the new capacity is $\lambda'^* = \frac{0.6}{(1-p_{hit}^{new})T_{s,P}} = \frac{0.6}{0.5 \cdot 0.028} = 42.86$ req/sec. req./sec.

This corresponds to an increase in overall system capacity of about 1.40×, a.k.a. about 40% better.