EVAL Problem 1

a. SLEEP method

1 second	2793.75
2 seconds	2793.53
3 seconds	2793.54
4 seconds	2793.53
5 seconds	2793.50
6 seconds	2793.49
7 seconds	2793.48
8 seconds	2793.47
9 seconds	2793.47
10 seconds	2793.46

MIN: 2793.46 MHz	**MAX:** 2793.74
AVG: 2793.522 MHz	**SD:** 0.0849575057

b. BUSYWAIT method

1 second	2793.42
2 seconds	2793.42
3 seconds	2793.42
4 seconds	2793.42
5 seconds	2793.42
6 seconds	2793.42
7 seconds	2793.42
8 seconds	2793.42
9 seconds	2793.42
10 seconds	2793.42

MIN: 2793.42	**MAX**: 2793.42
AVG: 2793.42	**SD:** 0

c. SCC sessions

Date/Time	**Clock Speed**
Sun Sep 15, 8:53 AM	2793.42
Sun Sep 15, 11:07 AM	2793.42
Sun Sep 15, 1:20 PM	2793.43
Sun Sep 15, 4:37 PM	2793.43
Sun Sep 15, 9:50 PM	2793.43
Sun Sep 15, 11:54 PM	2793.41
Mon Sep 16, 9:54 AM	2793.41
Mon Sep 16, 12:11 PM	2793.42
Mon Sep 16, 2:04 PM	2791.41
Mon Sep 16, 4:53 PM	2791.42

Based on part (a) and (b), the data indicates that the BUSYWAIT method provides a more stable result with no standard deviation, meaning the clock speed measurements are consistent across different runs. The SLEEP method shows some variability, with a noticeable standard deviation, indicating less consistency. The SLEEP method relies on system calls like nanosleep to pause execution. These functions can be affected by the operating system's scheduling policies and other system activities. The BUSYWAIT method continuously checks the elapsed time using a loop, which can provide more accurate and consistent measurements of CPU cycles. Since it doesn't rely on the system's scheduler, it avoids the variability introduced by other system processes. For these reason, for this part, we will be using the BUSYWAIT method to test whether the SCC is comprised of identical machines.

The data suggests that the SCC allocates machines with identical or nearly identical CPUs across sessions. The slight variations in clock speed are within the same range of consistency, and there is no substantial difference to suggest that different types of CPUs are being used across sessions. Therefore, the SCC is comprised of identical machines in terms of CPU architecture.

EVAL Problem 2

a. To compute the average throughput of the server in terms of requests per second:
   i. First identify the first and last timestamps
      1. First <receipt timestamp> : The timestamp for the first request (R0)
         2955320.090126354
      2. Last <completion timestamp> : The timestamp for the last request (R499)
         2955371.378790270
   ii. Then calculate the total active_time
      1. The numerical difference between the <completion timestamp> and <receipt timestamp> would be the active_time.
         2955371.378790270 - 2955320.090126354 = 51.288663916 seconds
   iii. Then count the number of requests
      1. From the output (R0 to R499)
         500 requests
   iv. Calculate the throughput
      1. Throughput = total_requests / active_time
         500 / 51.288663916 = 9.74 requests per second

   Thus, the average throughput of the server is approximately 9.74 requests per second.

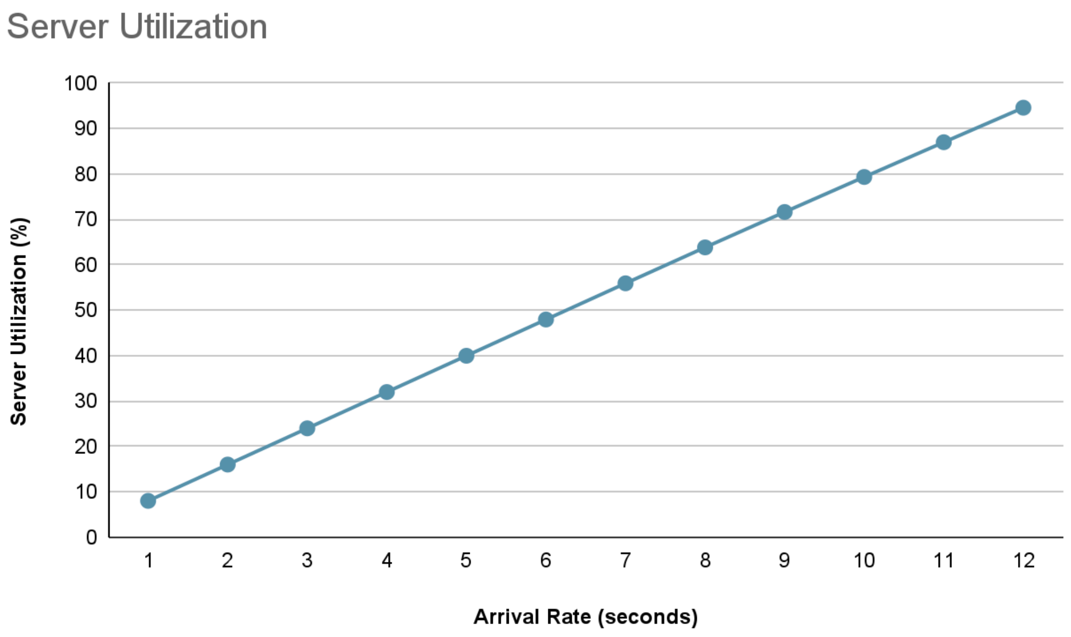
b. To compute the utilization of the server as a percentage:
   i. First identify the first and last timestamps
      1. First <receipt timestamp> : The timestamp for the first request (R0)
         2955670.379898337
      2. Last <completion timestamp> : The timestamp for the last request (R499)
         2955721.668290509
   ii. Then calculate the active_time
      1. The numerical difference between the <completion timestamp> and <receipt timestamp> would be the active_time.
         2955721.668290509 - 2955670.379898337 = 51.288392172 seconds
   iii. Calculate the total_request_time.
      1. Summation of the difference between <completion timestamp> and the <receipt timestamp> for each request
         Σ (<completion_timestamp> - <receipt_timestamp>) for all requests
         = **40.664155**
   iv. Calculate the utilization
      1. utilization = (total_request_time / active_time) * 100
         (40.664155 / 51.288392172) * 100 = 79.285%

   Thus, the utilization of the server is approximately 79.285%.

c. My calculated server utilization of 79.285% does closely match the system-reported CPU usage of 79%.

d. Data:

1	7.99%
2	15.98%
3	23.96%
4	31.95%
5	39.93%
6	47.91%
7	55.88%
8	63.79%
9	71.58%
10	79.29%
11	86.94%
12	94.54%

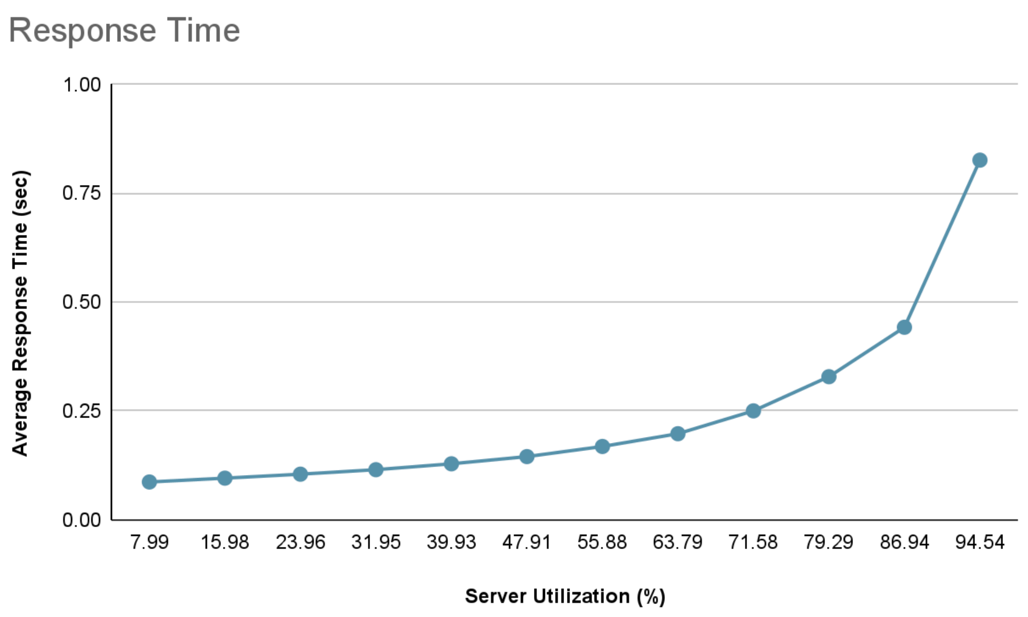


There seems to be a linear correlation between the arrival rate and the server utilization percentage. The server utilization percentage increases proportionally with the arrival rate.

e. Average: 0.328690 seconds
   Max: 1.445948 seconds
   Min: 0.000397 seconds
   Standard deviation: 0.314118 seconds

f.

-a	Max	Min	Avg	SD
1	0.516737	0.000364	0.086924	0.082453
2	0.533848	0.000369	0.095720	0.093710
3	0.568310	0.000398	0.104829	0.103963
4	0.598332	0.000398	0.115155	0.114637
5	0.638501	0.000395	0.128674	0.126948
6	0.684164	0.000399	0.145218	0.142456
7	0.741369	0.000396	0.168461	0.163076
8	0.784278	0.000396	0.197764	0.187568
9	0.914556	0.000394	0.250192	0.228978
10	1.445948	0.000397	0.328690	0.314118
11	1.920404	0.000385	0.442153	0.400139
12	2.335854	0.000550	0.825858	0.507142



The graph shows a clear relationship between server utilization and average response time. As the server utilization increases, the average response time also increases, indicating a direct proportional relationship between the two quantities. This relationship is not linear, in fact it seems to be more of an exponential one. As the utilization climbs, the response time climbs at a faster rate. This might be due to the server struggling more at higher utilization since it is given much more work at higher arrival times leading to higher average response times.