Machine Problem 5: Analysis

Class: 313-200
Name: Daniel Tan

Analysis:

In Machine Problem 5, the goal was to create two distinct programs: a data server program and a client program that would interrogate the data server. The implementation required running both the data server and the client program as concurrent processes on separate machines. Unlike, machine problem 3, this machine problem utilized a different request channel implementation for network communication. The data server allowed communication with the client through more than one channel, allowing multiple requests to be simultaneously processed. The client operation remained the same as machine problem 3's implementation as it was divided into three sets of threads that handled the entire data server interrogation. However, in machine problem 5, both the client and the data server were rewritten to utilize the network request channel implementation for Interprocess communication. As in machine problem 3, the client maintained a variable sized buffer to temporarily store the requests deposited by the request threads. The run time of the client program was again measured for variable numbers of worker threads and server backlog sizes; and the results of both implementations from machine problem 3 and 5 are plotted in the following figures:

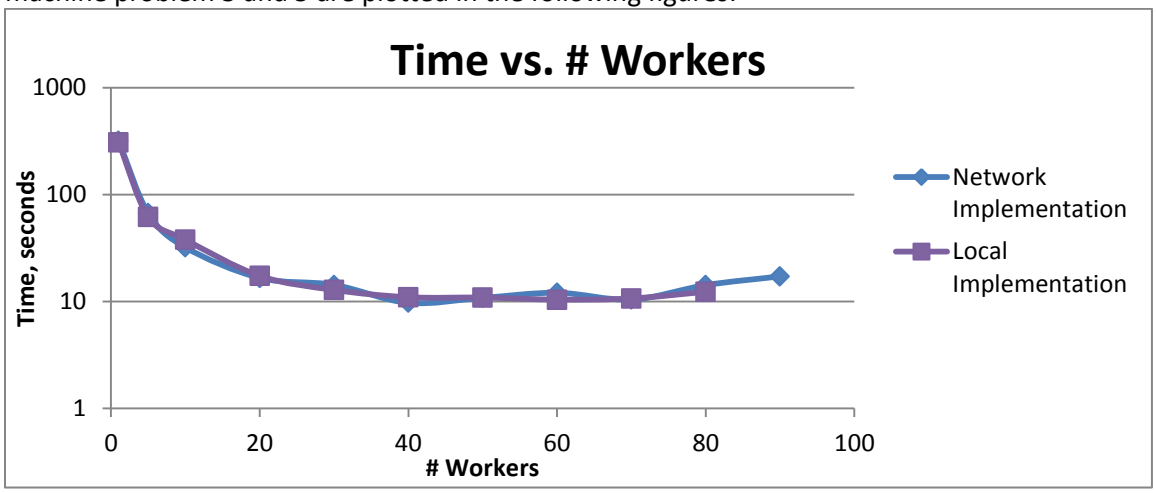

Figure 1 – This figure plots the program run times vs. the number of workers

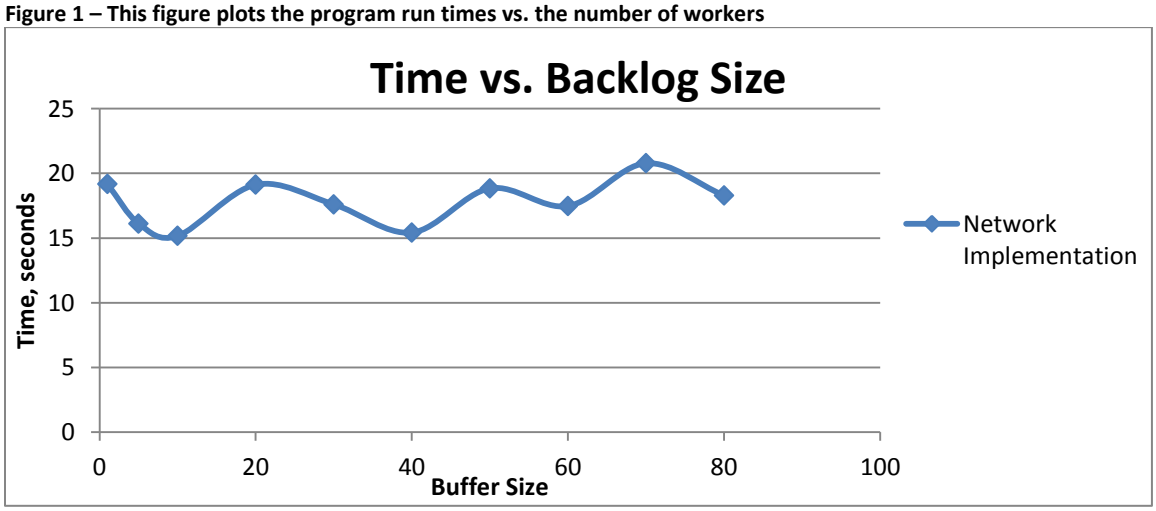

Figure 2 – This figure plots the program run times vs. the size of the backlog of the server

The statistics of the data for the network implementation are contained in the following tables:

# data requests = 10,000		
number of trials = 10		
bounded buffer size = 1000		
backlog size = 100		
# workers	Average Time (µsec) CI lower	Average Time (µsec) CI upper
1	311.90	233.92
5	64.82	49.66
10	31.67	22.72
20	16.49	11.63
30	12.64	11.62
40	8.85	7.67
50	8.81	10.68
60	8.85	8.80
70	8.25	9.49
80	11.05	20.17
90	13.70	15.61
100	12.08	21.84

Table 1 – Table of runtimes vs. the # of worker threads; CI = confidence interval, LB = lower bound, UB = upper bound

# data requests = 10,000		
number of trials = 10		
bounded buffer size = 1000		
# workers = 100		
backlog size	Average Time (µsec) CI lower	Average Time (µsec) CI upper
1	15.40	22.95
5	12.24	20.00
10	12.24	18.08
20	14.97	23.26
30	14.06	21.14
40	11.59	19.26
50	15.55	22.12
60	12.74	22.21
70	15.74	25.81
80	15.65	20.93
90	15.78	23.77
100	12.08	20.81

Table 2 – Table of runtimes vs. the backlog size; CI = confidence interval, LB = lower bound, UB = upper bound

The number of samples run was 10; given that sample size, normal distribution cannot be assumed, thus T-statistics were used. The confidence intervals for all trials for both the varying number of workers and varying buffer sizes were calculated at a T alpha level of 2.26215716 (i.e. 95% Confidence Interval).

Figure 1 demonstrates that the network implementation averages differ very slightly compared to the local machine implementation, indicating that the run times were mostly unaffected by the fact that the communication was over a network. If the network speeds were much slower then the disparity between the local and network implementations would be greater. The network implementation still had the same issue of slowing down when the large number of threads led to too many context switches. **Figure 2** demonstrates that the run time of the network implementation seems to be independent upon the backlog size. In this particular implementation where the clients connecting are few and all connection requests are issued in a single for loop in the client, the likelihood of many connection requests remaining on the backlog is very minimal; thus the size of the backlog is not what is primarily responsible for the run time variations Finally, the network implementation had on average a slightly smaller run time standard deviations with the average std. dev. for all trials being 2.931 compared to the local implementation's average std. dev. of 3.541.