Machine Problem 4: Analysis

Class: 313-200
Name: Daniel Tan

**Analysis:**

In Machine Problem 4, the goal was to create a client program that would load a data server program and interrogate it. The implementation required running both the data server and the client program as concurrent processes. Unlike, machine problem 3, this machine problem utilized a single event handler thread to replace machine problem 3's worker threads. 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 4, the single event handler thread circumvented the use of multiple worker threads by utilizing the select() call. 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 buffer sizes and variable numbers of worker threads; and the results of both implementations from machine problem 3 and 4 are plotted in the following figures:

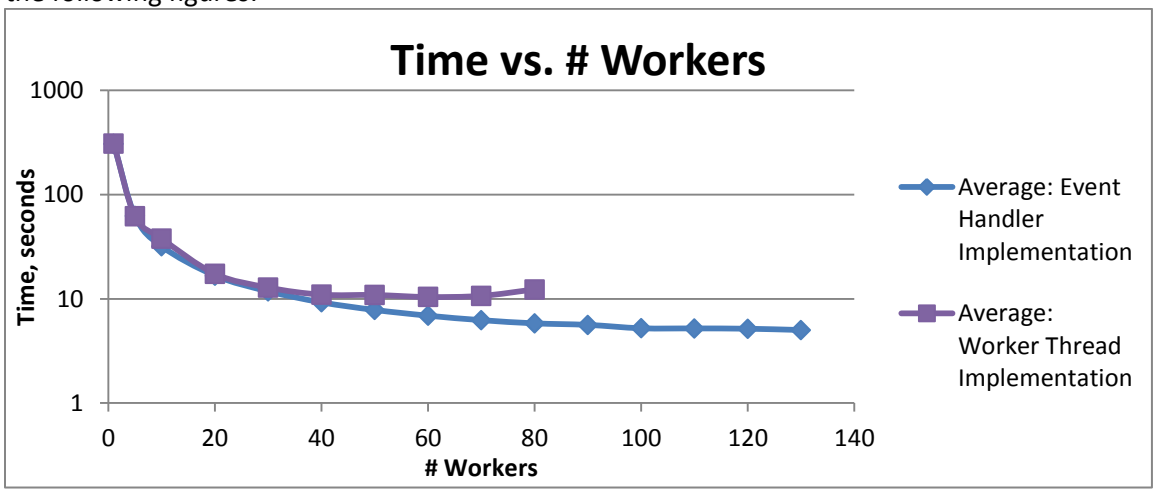


**Figure 1 – This figure plots the program run times vs. the number of workers**

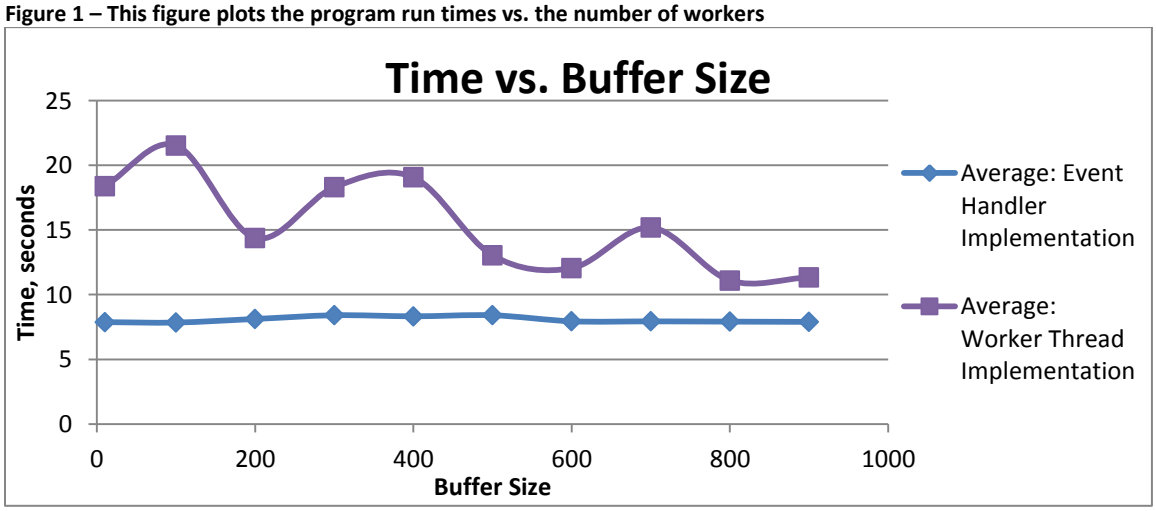


**Figure 2 – This figure plots the program run times vs. the size of the bounded buffer of requests**

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

| # data requests = 10,000 | | |
|---|---|---|
| number of trials = 10 | | |
| bounded buffer size = 1000 | | |
| # channels | Average Time (µsec) CI lower | Average Time (µsec) CI upper |
| 1 | 306.49 | 219.97 |
| 5 | 62.22 | 44.50 |
| 10 | 31.79 | 22.27 |
| 20 | 16.64 | 11.54 |
| 30 | 11.64 | 8.12 |
| 40 | 9.14 | 6.59 |
| 50 | 7.76 | 5.10 |
| 60 | 6.79 | 4.44 |
| 70 | 6.16 | 4.41 |
| 80 | 5.72 | 3.72 |

**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 | | |
| # channels = 50 | | |
| buffer size | Average Time (µsec) CI lower | Average Time (µsec) CI upper |
| 10 | 7.78 | 7.97 |
| 100 | 7.70 | 7.99 |
| 200 | 7.74 | 8.50 |
| 300 | 8.03 | 8.79 |
| 400 | 7.88 | 8.78 |
| 500 | 8.03 | 8.78 |
| 600 | 7.78 | 8.11 |
| 700 | 7.72 | 8.15 |
| 800 | 7.78 | 8.05 |
| 900 | 7.84 | 7.95 |
| 1000 | 7.90 | 8.09 |

**Table 2 – Table of runtimes vs. the buffer 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 event handler implementation averages do not increase unlike the worker thread implementation due to the context switching among worker threads. **Figure 2** demonstrates that the run time of the event handler implementation is almost independent of the buffer size. Finally, the event handler implementation had the advantage of much smaller run time standard deviations with the average std. dev. for all trials being 0.252 compared to the worker thread implementation's average std. dev. of 3.541.