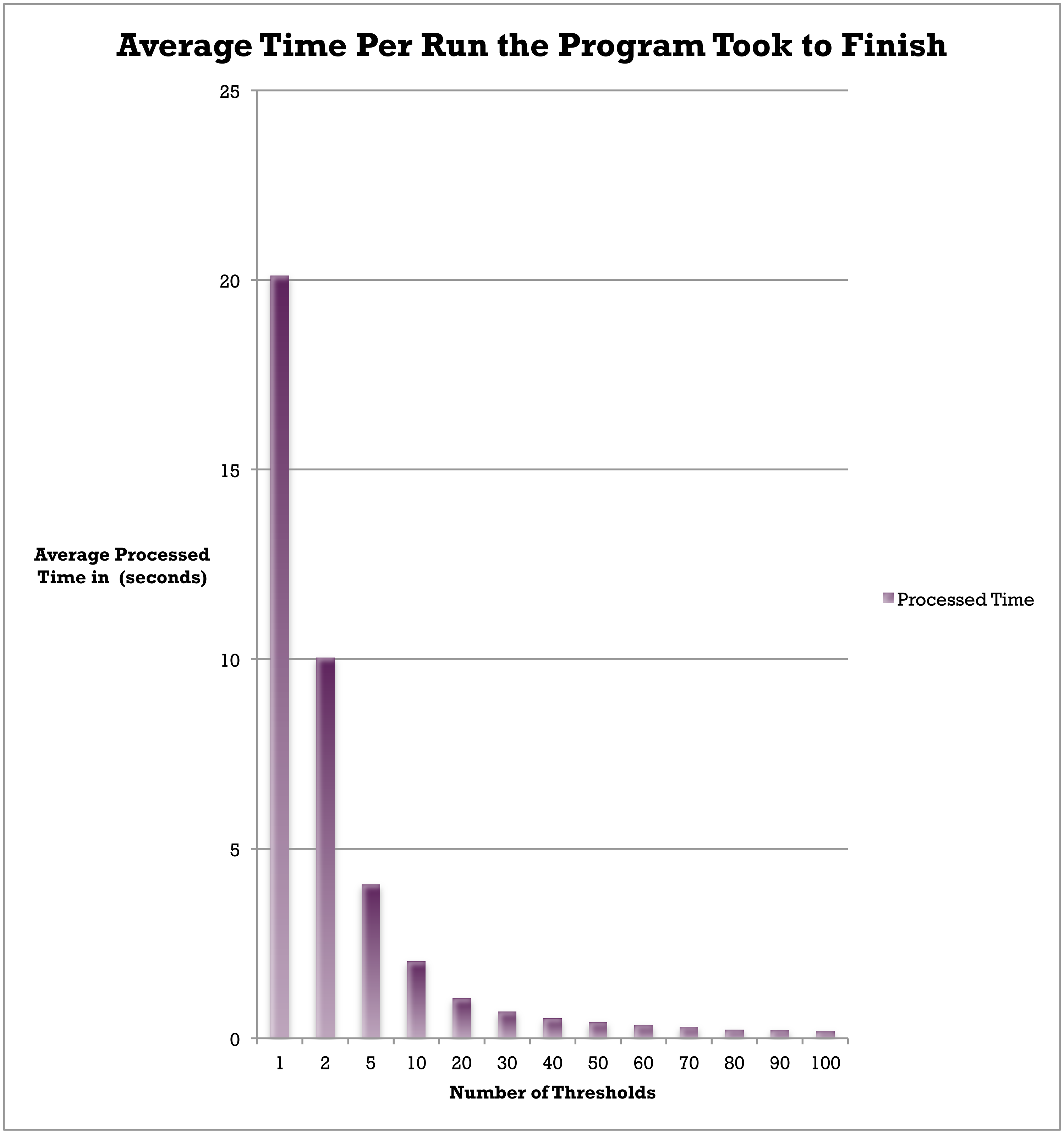
Weber State University

Lili Peng

CS3100 Homework 6 – Pthreads and Sloppy Counter Report

December 3rd, 2017

Average Times Per Run The Program Took to Finish



The Meaning of The Chart on Page 2:

The chart shows that when the number of thresholds is low, the processed time is longer which means the performance is poor, and when the number of thresholds is high, the processed time is shorter which means the performance is excellent.

The bash script I used to run my program 65 times:

```
#!/bin/sh

#  HW6.sh
#
#
#  Created by Lili Peng on 12/3/17.
#

for input in 1 2 5 10 20 30 40 50 60 70 80 90 100;
do
    for ((counter = 0; counter < 5; counter++));
    do
        ./HW6 $input;
    done
done
```

C Program Source:

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <pthread.h>
#include <time.h>

static volatile int counter = 0;
int threshold;
pthread_mutex_t lock;

void *mythread(void *arg) {

    int i;
    int localCounter = 0;

    for (i = 0; i < 1000000; i++)
    {
        localCounter++;
        if (localCounter == threshold) {
            pthread_mutex_lock(&lock);
            counter = counter + threshold;
            pthread_mutex_unlock(&lock);
            localCounter = 0;
        }
    }
    return NULL;
}

int
main (int argc, char *argv[])
{
    if (argc < 2)
    {
        printf("usage: name number\n");
        return 0;
    }
    threshold = atoi(argv[1]);
    pthread_t p1,p2,p3,p4,p5;
    int pres = pthread_mutex_init(&lock, NULL);

    int rc;

    printf("threshold: %d\n", threshold);

    // Start recording time
    clock_t t;
    t = clock();

    rc = pthread_create(&p1, NULL, mythread, "A");
    rc = pthread_create(&p2, NULL, mythread, "B");
    rc = pthread_create(&p3, NULL, mythread, "C");
    rc = pthread_create(&p4, NULL, mythread, "D");
    rc = pthread_create(&p5, NULL, mythread, "E");

    rc = pthread_join(p1, NULL);
    rc = pthread_join(p2, NULL);
    rc = pthread_join(p3, NULL);
    rc = pthread_join(p4, NULL);
    rc = pthread_join(p5, NULL);

    // Calculate the time it takes for counting this threshold
    t = clock() - t;
    double time_taken = ((double)t) / CLOCKS_PER_SEC;

    printf("processed time: %f seconds\n\n", time_taken);
    return 0;
}
```