CS-342

Operating Systems

Project #2

Kerem Ayöz
21501569 / Section:2

Cansu Yıldırım

21502891 / Section:2

PART A

syn_phistogram.c

```c
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <sys/syscall.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/shm.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/ipc.h>
#include <semaphore.h>


double *shared_data;
sem_t *childwait;

void create_intermediate(const char *file_name, double min_val, double max_val, int
     bin_count) {

  double w = (max_val-min_val) / bin_count;
  int bins[bin_count];
  double num;

  for (int i = 0; i < bin_count; i++)
    bins[i] = 0;

  FILE *file;
  file = fopen(file_name, "r");

  if (file== NULL){
    printf("Error! file cannot be opened");
    exit(1);
  }
  while(fscanf(file,"%lf",&num)!=EOF){
    if(num==max_val)
      ++bins[bin_count-1];

    int i,j;
    for(i=0,j=1; i<=bin_count-1 && j<=bin_count; i++,j++)
      if((num>=min_val+i*w) && (num<min_val+j*w))
        ++bins[i];

  }

  fclose(file);
  sem_wait(childwait);

  for (int i = 0; i < bin_count; i++)
    shared_data[i] = bins[i];

  exit(0);
}
```

```
57
58 int main(int argc, char const *argv[]) {
59
60   double min_val = atof(argv[1]);
61   double max_val = atof(argv[2]);
62
63   int bin_count = atoi(argv[3]);
64
65   int n = atoi(argv[4]);
66
67   pid_t ids[n];
68
69   int combined_bins[bin_count];
70   for (int i = 0; i < bin_count; i++)
71     combined_bins[i] = 0;
72
73   const int SIZE = bin_count * sizeof(double);
74
75   double shm_fd;
76   void* ptr;
77   shm_fd = shm_open("shm", O_CREAT | O_RDWR, 0666);
78   ftruncate(shm_fd, SIZE);
79   ptr = (double *)mmap(0, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);
80   shared_data = (double*)ptr;
81
82   childwait = sem_open("childwait", O_RDWR | O_CREAT, 0660, 1);
83
84   for(int i=0;i<n;i++)  {
85     ids[i] = fork();
86
87     if (ids[i] == 0)
88       create_intermediate(argv[5+i], min_val, max_val, bin_count);
89
90   }
91
92   for(int i=0;i<n;i++)  {
93     wait(NULL);
94     for (int j = 0; j < bin_count; j++)
95       combined_bins[j] += shared_data[j];
96     sem_post(childwait);
97
98   }
99
100
101  FILE *f = fopen(argv[n+5], "w");
102  if (f == NULL) {
103    printf("Error opening file!\n");
104    exit(1);
105  }
106
107  for (int i = 0; i < bin_count; ++i)
108    fprintf(f, "%d:%d\n",i+1,combined_bins[i]);
109
110  fclose(f);
111
112  shm_unlink("shm");
113  sem_destroy(childwait);
114  return 0;
115
116 }
```

PART B

syn_thistogram.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <math.h>
#include <sys/syscall.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/time.h>

#define MIN(x, y) (((x) < (y)) ? (x) : (y))

typedef struct node{
  double data;
  struct node * next;
}Node;

void add(Node** head, double new_item) {
  Node* new_node = (Node *)(malloc(sizeof(Node)));
  new_node->data = new_item;
  new_node->next = NULL;

  if ((*head) == NULL) {
    *head = new_node;
  }

  else {
    Node * cur = *head;
    while (cur->next != NULL)
      cur = cur->next;
    cur->next = new_node;
  }
}

double retrieve(Node** head) {
  Node * retrieved = *head;
  *head = (*head)->next;
  double num = retrieved->data;
  free(retrieved);
  return num;
}

void print_all(Node** head) {
  Node * cur = *head;
  while(cur != NULL) {
    printf("%f\n", cur->data);
    cur = cur->next;
  }
}

int size(Node **head) {
  int size = 0;
  Node *cur = *head;
  while (cur != NULL) {
    cur = cur->next;
    size++;
```

```
  }
  return size;
}

Node * head = NULL;
pthread_mutex_t mutex;
pthread_cond_t cond;

double min_val;
double max_val;
int bin_count;
int n;
double w;
int b;


int finished_count;
pthread_mutex_t mutex2;

void *worker(void *arg) {
  char *file_name = ((char *)arg);
  double send[b];
  for (int i = 0; i < b; i++)
    send[i] = 0;

  FILE *file;
  file = fopen(file_name, "r");

  int count = 0;
  double num;

  while(fscanf(file,"%lf",&num)!=EOF){
    if (count == b) {
      pthread_mutex_lock(&mutex);
      for (int i = 0; i < count; i++)
        add(&head, send[i]);
      pthread_cond_signal (&cond);
      pthread_mutex_unlock(&mutex);
      count = 0;
    }
    send[count++] = num;
  }

  pthread_mutex_lock(&mutex);
  for (int i = 0; i < count; i++)
    add(&head, send[i]);
  pthread_cond_signal (&cond);

  finished_count++;

  pthread_mutex_unlock(&mutex);
  fclose(file);
  pthread_exit(0);
}

int main(int argc, char *argv[]) {

  min_val = atof(argv[1]);
  max_val = atof(argv[2]);
  bin_count = atoi(argv[3]);
```

```
  n = atoi(argv[4]);
  w = (max_val-min_val) / bin_count;
  b = atoi(argv[6+n]);

  finished_count = 0;
  pthread_cond_init(&cond,NULL);

  pthread_t workers[n];
  int indexes[n];
  pthread_mutex_init(&mutex, NULL);
  int combined_bins[bin_count];
  for (int i = 0; i < bin_count; i++)
  combined_bins[i] = 0;

  for(int i=0;i<n;i++)  {
    indexes[i] = i;
    char *file_name = argv[5+indexes[i]];
    (void) pthread_create(&workers[i], NULL, worker, file_name);
  }

  while (1) {

    pthread_mutex_lock(&mutex);
    if (finished_count == n && size(&head) == 0)
      break;

    while (size(&head) == 0)
      pthread_cond_wait (&cond, &mutex);

    int retrieve_count = MIN(size(&head),b);
    for (int i = 0; i < retrieve_count;i++) {
      double num = retrieve(&head);
      if(num==max_val)
        ++combined_bins[bin_count-1];

      int i,j;
      for(i=0,j=1; i<=bin_count-1 && j<=bin_count; i++,j++)
        if((num>=min_val+i*w) && (num<min_val+j*w))
          ++combined_bins[i];
    }
    pthread_mutex_unlock(&mutex);
  }

  for (int i = 0; i < n; i++)
    (void) pthread_join(workers[i], NULL);

  FILE *f = fopen(argv[n+5], "w");
  if (f == NULL) {
    printf("Error opening file!\n");
    exit(1);
  }

  for (int i = 0; i < bin_count; ++i)
    fprintf(f, "%d:%d\n",i+1,combined_bins[i]);

  fclose(f);

  pthread_mutex_destroy(&mutex);
  return 0;
}
```

PART C

In the syn_thistogram program, value size B is given as a parameter in the command line. A worker thread reads B numbers from the input file and put them into linked list. Main thread gets the numbers from the linked list in batches of B numbers.

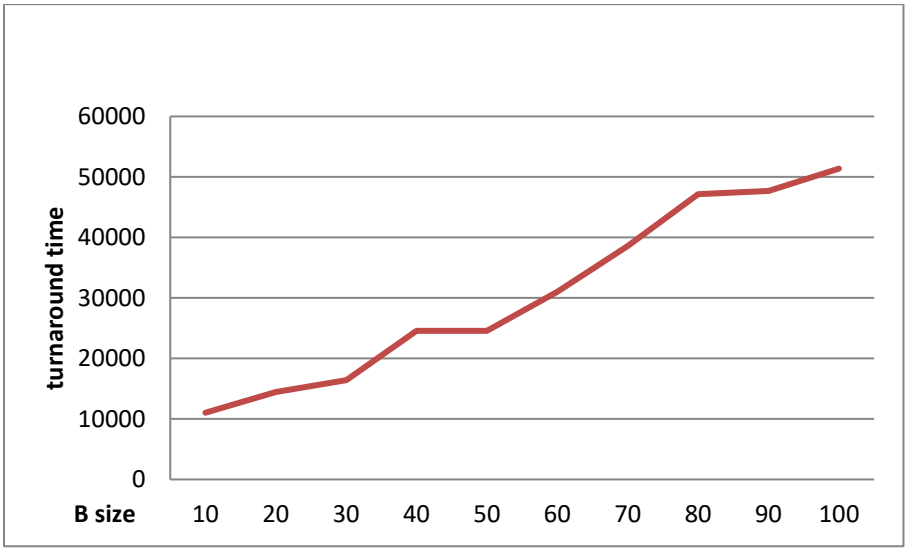
To find out whether this value B affects the turnaround time of the program, we did some experiments.

4 input files are given with sizes 500, 1000, 1500 and 2000 lines. The min value is 0, maxvalue is 10000 and bincount is 9 for all of these input files.

As B, we tried the values 10,20,30,40,50,60,70,80,90 and 100.

The results are as follows:

B size	10	20	30	40	50	60	70	80	90	100
Turnaround time	11027	14439	16432	24534	24565	31011	38579	47129	47661	51373



Worker threads lock the critical section when they start putting the values into the linked list. Also, main thread locks while it receives the values from the linked list and puts them into the histogram array. Thus, if the B size increases, waiting time between the locks for both worker threads and main thread will be increased. Our observation proved this.