Algorithms Review for Job Interview

Jenny Huang

December 20, 2014

Contents

1 dfkdjf

2

```
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>

#define SIZE 8   // Size by SIZE matrices

using namespace std;

int num_thrd;   // number of threads

int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE];
int idx[] = {0, 1, 2, 3};

void print_matrix(int m[SIZE][SIZE]) {
    int i, j;
    for (i = 0; i < SIZE; i++) {
        printf("\n\t| ");
        for (j = 0; j < SIZE; j++)
            printf("%2d ", m[i][j]);
        printf("|");
    }
}

void init_matrix(int m[SIZE][SIZE]) {
    int i, j, val = 0;
    for (i = 0; i < SIZE; i++)
        for (j = 0; j < SIZE; j++)
            m[i][j] = val++;
}

// thread function: taking "slice" as its argument
void* multiply(void* slice) {
    int s = *((int*)slice);   // retrive the slice info
    //printf("s value: %d\n", s);
    int from = (s * SIZE)/num_thrd; // note that this 'slicing' works fine
    int to = ((s+1) * SIZE)/num_thrd; // even if SIZE is not divisible by num_thrd
    int i,j,k;

    printf("computing slice %d (from row %d to %d)\n", s, from, to-1);
    for (i = from; i < to; i++) {
```

```
        for (j = 0; j < SIZE; j++) {
            C[i][j] = 0;
            for ( k = 0; k < SIZE; k++)
                C[i][j] += A[i][k]*B[k][j];
        }
    }
    printf("finished slice %d\n\n", s);
    return 0;
}

int main(int argc, char* argv[]) {
    pthread_t* thread;  // pointer to a group of threads
    int i;
    if (argc!=2) {
        printf("Usage: %s number_of_threads\n",argv[0]);
        exit(-1);
    }
    num_thrd = atoi(argv[1]);
    printf("num_thrd: %d\n", num_thrd);
    init_matrix(A);
    printf("\n");
    init_matrix(B);
    thread = (pthread_t*) malloc(num_thrd*sizeof(pthread_t));

    for (i = 1; i < num_thrd; i++) {
        //printf("address i: %d\n", i);
        int rc = pthread_create(&thread[i], NULL, multiply, &idx[i]);
        if (rc != 0) {
            perror("Can't create thread");
            free(thread);
            exit(-1);
        }
    }

    // main thread works on slice 0
    // so everybody is busy
    // main thread does everything if threadd number is specified as 1
    //int tmp = 0;
    multiply((void*)(&(idx[0])));

    // main thead waiting for other thread to complete
    for (i = 2; i <= num_thrd; i++)
        pthread_join(thread[i-1], NULL);

    printf("\n\n");
    print_matrix(A);
    printf("\n\n\t       * \n");
    print_matrix(B);
    printf("\n\n\t       = \n");
    print_matrix(C);
    printf("\n\n");

    free(thread);

    return 0;
}
```