George Suarez

CSE 460

Homework 3

1. The aging algorithm with a = 1/2 is being used to predict run times. The previous four runs, from oldest to most recent are 40, 20, 40, and 15 msec. What is the next run time?

(((40 + 20) / 2 + 40) / 2 + 15) / 2

= ((30 + 40) / 2 + 15) /2

= (35 + 15) / 2 = **25**

1. Measurement of a certain system have shown that the average process runs for a time T before blocking on I/O. A process switch requires a time S, which is effectively wasted ( overhead ). For round robin scheduling with quantum Q, give a formula for the CPU efficiency for each of the following.
2. Q = infinity
3. Q > T
4. S < Q < T
5. Q = S
6. Q nearly 0

Evaluate the efficiency when S = 1, Q = 5, and T = 20.

1. T / (T + S) = 20 / (20 + 1) = 20 / 21 = **.95 => 95%**
2. T / (T + S)) = 20 / (20 + 1) = 20 / 21 = **.95 => 95%**
3. T / (T + (ST / Q)) = 20 / (20 + (1 \* 20 / 25)) = 20 / (20 + .8) = **.96 => 96%**
4. Q / (Q + Q) = 5 / (5 + 5) = 5 / 10 = **.50 => 50%**
5. Efficiency goes to zero as Q goes to 0.
6. Write a multithreaded program using SDL threads or POSIX threads. The program uses a number of threads to multiply two matrices. The multiplication of an M X L matrix A and an L X N matrix B gives an M X N matrix C, and is given by the formula,

http://cse.csusb.edu/tongyu/courses/cs460/images/matmul.png

Basically, each element Cij is the dot product of the i-th row vector of A with the j-th column vector of B. The program uses one thread to calculate a dot product. Therefore, it totally needs M x N threads to calculate all the elements of matrix C.

*sdl\_matrix.cpp:*

#include <SDL2/SDL.h>

#include <SDL2/SDL\_thread.h>

#include <vector>

#include <ctime>

#include <iostream>

using namespace std;

int matrixA[3][2] = {{5, 4}, {2,6}, {9, 2}};

int matrixB[2][3] = {{5,6,2}, {4, 2, 8}};

int matrixC[3][3] = {{0, 0, 0},{0, 0, 0},{0, 0, 0}};

int dotProduct(void \*data)

{

char \*threadname = (char \*)data;

for (int row = 0; row < 3; row++)

{

for (int col = 0; col < 3; col++)

{

for (int product = 0; product < 2; product++)

{

matrixC[row][col] += matrixA[row][product] \* matrixB[product][col];

}

}

}

return 0;

}

void printMatrixA(int matrix[][2])

{

cout << "Matrix A: " << endl;

for (int row = 0; row < 3; row++)

{

for (int col = 0; col < 2; col++)

{

cout << matrixA[row][col] << " ";

}

cout << endl;

}

cout << endl;

}

void printMatrixB(int m[][3])

{

cout << "Matrix B: " << endl;

for (int row = 0; row < 2; row++)

{

for (int col = 0; col < 3; col++)

{

cout << matrixB[row][col] << " ";

}

cout << endl;

}

cout << endl;

}

void printMatrixC(int m[][3])

{

cout << "Matrix C: " << endl;

for (int row = 0; row < 3; row++)

{

for (int col = 0; col < 3; col++)

{

cout << matrixC[row][col] << " ";

}

cout << endl;

}

cout << endl;

}

int main()

{

SDL\_Thread \*sumThread = SDL\_CreateThread(dotProduct, "Sum Thread", (void \*)"Dot Product Thread");

if (sumThread == NULL)

{

cout << "SDL\_CreateThread failed: \n" << SDL\_GetError() << endl;

}

else

{

int returnValue;

SDL\_WaitThread(sumThread, &returnValue);

printMatrixA(matrixA);

printMatrixB(matrixB);

cout << "Equals to ";

printMatrixC(matrixC);

cout << endl;

}

return 0;

}

*Output:*

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Homework/Homework 3 on master\***

**$** ./sdl\_matrix

Matrix A:

5 4

2 6

9 2

Matrix B:

5 6 2

4 2 8

Equals to Matrix C:

41 38 42

34 24 52

53 58 34

1. *Sdl\_reader\_writer.cpp:*

#include <SDL2/SDL.h>

#include <SDL2/SDL\_thread.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <signal.h>

#include <unistd.h>

#include <iostream>

#include <fstream>

using namespace std;

SDL\_bool condition = SDL\_FALSE;

SDL\_mutex \*mutex1;

SDL\_cond \*readerQueue; //condition variable

SDL\_cond \*writerQueue; //condition variable

int readerCount = 0;

int writerCount = 0;

bool quit = false;

string fileName = "counter.txt";

int reader(void \*data)

{

while (!quit)

{

SDL\_Delay(rand() % 3000);

SDL\_LockMutex(mutex1);

while (!(writerCount == 0))

SDL\_CondWait(readerQueue, mutex1);

readerCount++;

SDL\_UnlockMutex(mutex1);

//read

int count = -1;

ifstream inFile;

inFile.open(fileName.c\_str());

if (inFile.good())

{

inFile >> count;

inFile.close();

}

SDL\_LockMutex(mutex1);

printf("\nThis is %s thread: %d\n", (char \*)data, count);

printf("Counter value: %d\n", count);

if (--readerCount == 0)

SDL\_CondSignal(writerQueue);

SDL\_UnlockMutex(mutex1);

}

}

int writer(void \*data)

{

while (!quit)

{

SDL\_Delay(rand() % 3000);

SDL\_LockMutex(mutex1);

while (!((readerCount == 0) && (writerCount == 0)))

SDL\_CondWait(writerQueue, mutex1);

writerCount++;

SDL\_UnlockMutex(mutex1);

int count = -1;

ifstream inFile;

inFile.open(fileName.c\_str());

if (inFile.good())

{

inFile >> count;

inFile.close();

}

ofstream outFile;

outFile.open(fileName.c\_str());

if (outFile.good())

{

outFile << count;

outFile.close();

}

SDL\_LockMutex(mutex1);

writerCount--; //only one writer at one time

count++;

printf("\nThis is %s thread: %d\n", (char \*)data, count);

printf("Counter value: %d\n", count);

SDL\_CondSignal(writerQueue);

SDL\_CondBroadcast(readerQueue);

SDL\_UnlockMutex(mutex1);

}

}

int main()

{

SDL\_Thread \*idr[20], \*idw[3]; //thread identifiers

char readerNames[20][10];

char writerNames[3][10];

for (int i = 0; i < 20; i++)

{

cout << readerNames[i] << "Reader: " << i + 1 << endl;

idr[i] = SDL\_CreateThread(reader, "Reader Thread", readerNames[i]);

}

for (int i = 0; i < 3; i++)

{

cout << writerNames[i] << "Writer: " << i + 1 << endl;

idw[i] = SDL\_CreateThread(writer, "Writer Thread", writerNames[i]);

}

readerQueue = SDL\_CreateCond();

writerQueue = SDL\_CreateCond();

for (int i = 0; i < 20; i++)

{

SDL\_WaitThread(idr[i], NULL);

}

for (int i = 0; i < 3; i++)

{

SDL\_WaitThread(idw[i], NULL);

}

SDL\_DestroyCond(readerQueue);

SDL\_DestroyCond(writerQueue);

SDL\_DestroyMutex(mutex1);

return 0;

}

*Output:*

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Homework/Homework 3 on master\***

**$** ./sdl\_readers\_writers

Reader: 1

Reader: 2

Reader: 3

Reader: 4

Reader: 5

Reader: 6

Reader: 7

Reader: 8

Reader: 9

Reader: 10

Reader: 11

Reader: 12

Reader: 13

Reader: 14

Reader: 15

Reader: 16

Reader: 17

Reader: 18

Reader: 19

Reader: 20

Writer: 1

Writer: 2

Writer: 3

This is thread: 0

Counter value: 0

This is thread: -1

Counter value: -1

This is thread: 0

Counter value: 0

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: 0

Counter value: 0

This is thread: -1

Counter value: -1

This is thread: -1

Counter value: -1

This is thread: 0

Counter value: 0