Relatório do Primeiro Trabalho de Implementação

Problema de Quadratura para Integração Numérica

Gabriel de Souza Ferreira
DRE: 115171168

Larissa Dalimar Lima Viana
DRE: 117213958

Organização

Para visualizar melhor essa parte, o ideal é dar uma olhada no projeto pelo github: https://github.com/gabrielsferre/trab-integral.git

Na pasta do projeto estão localizados os arquivos *.c*, um *makefile*, uma pasta chamada *include*, que contém arquivos *.h*, e uma pasta chamada *ideia_fracassada* (vamos explicar ela melhor mais para a frente neste relatório).

Dentre os *.c's*, estão o *integral_sequencial.c* e o *integral_concorrente.c*, que são os códigos principais do trabalho. Eles implementam o método da integração numérica, um sequencialmente e o outro concorrentemente. Além desses dois, também existem vários *.c's* com os testes, com três deles sendo testes automáticos. O *teste_integral_sequencial.c* testa a corretude e o tempo de execução da função que faz a integral sequencial, o *teste_integral_concorrente.c* faz a mesma coisa para a integral concorrente, e o *teste_comparativo.c* repete os mesmos casos de teste dos anteriores, só que agora calculando o ganho que a função concorrente tem sobre a sequencial. O resto dos testes são aqueles pedidos no enunciado do trabalho, um para cada uma das funções pedidas (essas funções também são testadas nos testes automáticos).

Na pasta *include* estão os *.h's*. Além do *integral_sequencial.h* e do *integral_concorrente.h*, que fazem o que você esperaria de um *.h*, também

existem outros três: *funcoes.h*, que define as funções a-b-c-d-e-f-g e algumas constantes matemáticas; *conc_util.h*, que possui wrappers para algumas funções da *pthread* e para a função *malloc* (esses wrappers servem para não precisarmos ficar testando por erros o tempo todo no código); *timer.h*, que é o mesmo que usamos nas aulas de laboratório.

Algoritmos

Na versão sequencial, usamos exatamente o mesmo algoritmo descrito no enunciado do trabalho. Na concorrente, dividimos o intervalo de integração em várias partes (mais especificamente, 128 partes, mas esse número é facilmente ajustável dentro do código) e então executamos o algoritmo sequencial em cada uma delas, depois somamos os resultados. Uma variável global guarda qual é a última parte do intervalo que começou a ser calculado, de forma que uma thread pode pegar essa parte para si e incrementar a variável para a próxima parte, que será calculada pela próxima thread que acessar a variável global. Isso garante distribuição de carga entre as threads desde que o intervalo seja dividido em um número de partes grande o suficiente. A sincronização de acesso à variável global é feita com o uso de um mutex.

Essa estratégia de distribuição de carga entre as threads pode parecer um pouco fraca, mas foi a melhor que conseguimos pensar. Para obter a distribuição mais igualitária possível, pensamos numa solução onde cada thread calculava um passo do algoritmo recursivo e então enfileirava os argumentos que seriam usados nas chamadas recursivas do algoritmo em uma lista de tarefas para as threads. O problema dessa ideia é que o tempo que a thread demora para calcular a área dos retângulos e fazer a comparação entre elas é muito pequeno, muito menor do que o tempo que demoraria para fazer a sincronização necessária para uma lista de tarefas. Como são feitas muitas chamadas recursivas, o tempo de sincronização seria gigantesco. Um jeito de amenizar esse problema seria fazer com que cada thread calculasse vários níveis de recursão antes de passar argumentos para a lista de tarefas, mas isso acabou não resolvendo muita coisa, o programa continua muito lento, e é difícil ajustar quantas chamadas recursivas uma thread tem que fazer antes de passar para a próxima tarefa.

Testamos essa última estratégia, mas ela só respondia num tempo razoável para erros muito grandes, onde o cálculo concorrente não teria nenhuma vantagem de qualquer jeito. Nesse algoritmo, era difícil até de se fazer testes. Num evento infortuno, enquanto eram feitos testes no algoritmo, os cpus do computador ficaram tão ocupados enfileirando tarefas e realizando

sincronizações entre threads que o sistema operacional simplesmente não soube lidar com a situação e travou de vez. O computador precisou ser desligado a força, soltando um som de se partir o coração, e então achamos melhor parar com os testes para evitar maiores danos. Nesse ocorrido, o erro que estava sendo testado era grande se comparado com os que estávamos usando no algoritmo sequencial, o que nos fez acreditar que, mesmo se continuássemos ajustando parâmetros e fazendo mais testes em cima dessa ideia, não chegaríamos a lugar nenhum.

Acontece que, já que abandonamos a lista de tarefas, nossa solução acabou não usando variáveis condicionais, que eram explicitamente mencionadas no enunciado do trabalho. Por isso, criamos a pasta *ideia_fracassada*, que contém a solução da lista de tarefas. Nela, usamos uma fila de alocação dinâmica para enfileirar as tarefas e calculamos o número de folhas da árvore de recursão gerada pelo algoritmo recursivo para determinar quando as threads deveriam parar de esperar por tarefas e chamar pthread_exit().

Testes

Nesta seção encontram-se print dos outputs dos testes no terminal. Os testes buscam integrar sobre intervalos onde a função testada varia bastante, de forma que eles acabaram mostrando que a distribuição de carga entre as threads está funcionando bem.

Primeiro testamos a corretude e tempo de execução da integral sequencial. Em seguida, a corretude e tempo de execução para a integral concorrente usando quatro threads (nesses dois testes, estão sendo impressos o intervalo de integração, o erro máximo admitido e o tempo de execução de cada teste). Por último, fazemos a comparação entre o tempo sequencial e o concorrente, onde o ganho é calculado como sendo o tempo gasto pela função concorrente dividido pelo tempo da sequencial. A máquina na qual foram executados os teste possui dois cpu’s, onde cada um consegue executar duas threads, isso se reflete nos testes, onde podemos ver que o incremento de ganho maior acontece quando vamos de uma thread para duas e para de crescer quando vamos de quatro threads para cinco. Pulando para oito thread, vemos que não há grandes diferenças no desempenho concorrente.

Testes para a versão sequencial:

Passou no teste "cosseno"
Intervalo: [0.000000, 1.570796]
Erro: 0.0000010000
Tempo: 0.8256462690

Passou no teste "exponencial"
Intervalo: [1.000000, 0.000000]
Erro: 0.0000100000
Tempo: 0.1512426680

Passou no teste "função a"
Intervalo: [2.000000, 7.000000]
Erro: 0.0000100000
Tempo: 1.2493693480

Passou no teste "função b"
Intervalo: [-1.000000, 1.000000]
Erro: 0.0000100000
Tempo: 0.0866539050

Passou no teste "função c"
Intervalo: [-20.000000, 100.000000]
Erro: 1.0000000000
Tempo: 0.2310330660

Passou no teste "função d"
Intervalo: [-8.000000, -0.550000]
Erro: 0.0000010000
Tempo: 7.3222146190

Passou no teste "função e"
Intervalo: [-8.000000, -0.550000]
Erro: 0.0000100000
Tempo: 0.9960612210

Passou no teste "função f"
Intervalo: [-10.000000, 20.000000]
Erro: 0.0010000000
Tempo: 0.4210764360

Passou no teste "função g"
Intervalo: [-2.000000, 1.000000]
Erro: 0.0000010000
Tempo: 3.6855562840

Testes para versão concorrente:

Passou no teste "cosseno"
Intervalo: [0.000000, 1.570796]
Erro: 0.0000010000
Numero de threads: 4
Tempo: 0.3481820900

Passou no teste "exponencial"
Intervalo: [1.000000, 0.000000]
Erro: 0.0000100000
Numero de threads: 4
Tempo: 0.0599138310

Passou no teste "função a"
Intervalo: [2.000000, 7.000000]
Erro: 0.0000100000
Numero de threads: 4
Tempo: 0.5614694710

Passou no teste "função b"
Intervalo: [-1.000000, 1.000000]
Erro: 0.0000100000
Numero de threads: 4
Tempo: 0.0397054320

Passou no teste "função c"
Intervalo: [-20.000000, 100.000000]
Erro: 1.0000000000
Numero de threads: 4
Tempo: 0.0998690200

Passou no teste "função d"
Intervalo: [-8.000000, -0.550000]
Erro: 0.0000010000
Numero de threads: 4
Tempo: 2.8109572310

Passou no teste "função e"
Intervalo: [-8.000000, -0.550000]
Erro: 0.0000100000
Numero de threads: 4
Tempo: 0.3542487870

Passou no teste "função f"
Intervalo: [-10.000000, 20.000000]
Erro: 0.0010000000
Numero de threads: 4

Tempo: 0.1636672020

Passou no teste "função g"
Intervalo: [-2.000000, 1.000000]
Erro: 0.0000010000
Numero de threads: 4
Tempo: 1.3327010400

Testes comparativos usando duas threads:

Resultado sequencial: 1.0000000000
Resultado concorrente: 1.0000000000
Tempo sequencial: 0.8251964300
Tempo concorrente: 0.4204834110
Ganho concorrente: 1.96

Teste "exponencial"
Resultado sequencial: -1.7182818285
Resultado concorrente: -1.7182818285
Tempo sequencial: 0.1533825610
Tempo concorrente: 0.0827549530
Ganho concorrente: 1.85

Teste "função a"
Resultado sequencial: 27.5000000000
Resultado concorrente: 27.5000000000
Tempo sequencial: 1.2779346610
Tempo concorrente: 0.6485840730
Ganho concorrente: 1.97

Teste "função b"
Resultado sequencial: 1.5707964928
Resultado concorrente: 1.5707964928
Tempo sequencial: 0.0929230570
Tempo concorrente: 0.0496464610
Ganho concorrente: 1.87

Teste "função c"
Resultado sequencial: 336002.4393136707
Resultado concorrente: 336002.4393136707
Tempo sequencial: 0.2398610380
Tempo concorrente: 0.1251020080
Ganho concorrente: 1.92

Teste "função d"
Resultado sequencial: 0.5466247424
Resultado concorrente: 0.5466247424

Tempo sequencial: 7.3801904520
Tempo concorrente: 3.7670339830
Ganho concorrente: 1.96

Teste "função e"
Resultado sequencial: -0.4640199794
Resultado concorrente: -0.4640199794
Tempo sequencial: 0.9973953720
Tempo concorrente: 0.5017521860
Ganho concorrente: 1.99

Teste "função f"
Resultado sequencial: 200.2441143792
Resultado concorrente: 200.2441143792
Tempo sequencial: 0.4223253370
Tempo concorrente: 0.2163575890
Ganho concorrente: 1.95

Teste "função g"
Resultado sequencial: 0.5626533126
Resultado concorrente: 0.5626533126
Tempo sequencial: 3.6745125180
Tempo concorrente: 1.8515765090
Ganho concorrente: 1.98

Três threads:

Teste "cosseno"
Resultado sequencial: 1.0000000000
Resultado concorrente: 1.0000000000
Tempo sequencial: 0.8272035870
Tempo concorrente: 0.3671561990
Ganho concorrente: 2.25

Teste "exponencial"
Resultado sequencial: -1.7182818285
Resultado concorrente: -1.7182818285
Tempo sequencial: 0.1583041120
Tempo concorrente: 0.0642422850
Ganho concorrente: 2.46

Teste "função a"
Resultado sequencial: 27.5000000000
Resultado concorrente: 27.5000000000
Tempo sequencial: 1.2642772890
Tempo concorrente: 0.5831060040
Ganho concorrente: 2.17

Teste "função b"
Resultado sequencial: 1.5707964928
Resultado concorrente: 1.5707964928
Tempo sequencial: 0.0877327800
Tempo concorrente: 0.0364627960
Ganho concorrente: 2.41

Teste "função c"
Resultado sequencial: 336002.4393136707
Resultado concorrente: 336002.4393136707
Tempo sequencial: 0.2347238160
Tempo concorrente: 0.1002100250
Ganho concorrente: 2.34

Teste "função d"
Resultado sequencial: 0.5466247424
Resultado concorrente: 0.5466247424
Tempo sequencial: 7.3600014550
Tempo concorrente: 3.1726835070
Ganho concorrente: 2.32

Teste "função e"
Resultado sequencial: -0.4640199794
Resultado concorrente: -0.4640199794
Tempo sequencial: 0.9930422680
Tempo concorrente: 0.4044015580
Ganho concorrente: 2.46

Teste "função f"
Resultado sequencial: 200.2441143792
Resultado concorrente: 200.2441143792
Tempo sequencial: 0.4220305830
Tempo concorrente: 0.1807259390
Ganho concorrente: 2.34

Teste "função g"
Resultado sequencial: 0.5626533126
Resultado concorrente: 0.5626533126
Tempo sequencial: 3.6880606760
Tempo concorrente: 1.5463759060
Ganho concorrente: 2.38

Quatro threads:

Teste "cosseno"
Resultado sequencial: 1.0000000000
Resultado concorrente: 1.0000000000
Tempo sequencial: 0.8251692310
Tempo concorrente: 0.3370019210

Ganho concorrente: 2.45

Teste "exponencial"
Resultado sequencial: -1.7182818285
Resultado concorrente: -1.7182818285
Tempo sequencial: 0.1531522890
Tempo concorrente: 0.0576375260
Ganho concorrente: 2.66

Teste "função a"
Resultado sequencial: 27.5000000000
Resultado concorrente: 27.5000000000
Tempo sequencial: 1.2654011490
Tempo concorrente: 0.5626989960
Ganho concorrente: 2.25

Teste "função b"
Resultado sequencial: 1.5707964928
Resultado concorrente: 1.5707964928
Tempo sequencial: 0.0886634990
Tempo concorrente: 0.0325809520
Ganho concorrente: 2.72

Teste "função c"
Resultado sequencial: 336002.4393136707
Resultado concorrente: 336002.4393136707
Tempo sequencial: 0.2358121480
Tempo concorrente: 0.0908787480
Ganho concorrente: 2.59

Teste "função d"
Resultado sequencial: 0.5466247424
Resultado concorrente: 0.5466247424
Tempo sequencial: 7.3403558640
Tempo concorrente: 2.8066495700
Ganho concorrente: 2.62

Teste "função e"
Resultado sequencial: -0.4640199794
Resultado concorrente: -0.4640199794
Tempo sequencial: 1.0010807870
Tempo concorrente: 0.3391657370
Ganho concorrente: 2.95

Teste "função f"
Resultado sequencial: 200.2441143792
Resultado concorrente: 200.2441143792
Tempo sequencial: 0.4302029060
Tempo concorrente: 0.1595163250
Ganho concorrente: 2.70

Teste "função g"
Resultado sequencial: 0.5626533126
Resultado concorrente: 0.5626533126
Tempo sequencial: 3.6796717480
Tempo concorrente: 1.3171634610
Ganho concorrente: 2.79

Cinco threads:

Teste "cosseno"
Resultado sequencial: 1.0000000000
Resultado concorrente: 1.0000000000
Tempo sequencial: 0.8297662800
Tempo concorrente: 0.3408959740
Ganho concorrente: 2.43

Teste "exponencial"
Resultado sequencial: -1.7182818285
Resultado concorrente: -1.7182818285
Tempo sequencial: 0.1519979480
Tempo concorrente: 0.0583023270
Ganho concorrente: 2.61

Teste "função a"
Resultado sequencial: 27.5000000000
Resultado concorrente: 27.5000000000
Tempo sequencial: 1.2625393310
Tempo concorrente: 0.5553058980
Ganho concorrente: 2.27

Teste "função b"
Resultado sequencial: 1.5707964928
Resultado concorrente: 1.5707964928
Tempo sequencial: 0.0873817090
Tempo concorrente: 0.0321420390
Ganho concorrente: 2.72

Teste "função c"
Resultado sequencial: 336002.4393136707
Resultado concorrente: 336002.4393136707
Tempo sequencial: 0.2365637790
Tempo concorrente: 0.0921207650
Ganho concorrente: 2.57

Teste "função d"
Resultado sequencial: 0.5466247424
Resultado concorrente: 0.5466247424
Tempo sequencial: 7.3442540780

Tempo concorrente: 2.8660352990
Ganho concorrente: 2.56

Teste "função e"
Resultado sequencial: -0.4640199794
Resultado concorrente: -0.4640199794
Tempo sequencial: 0.9883114760
Tempo concorrente: 0.3393059330
Ganho concorrente: 2.91

Teste "função f"
Resultado sequencial: 200.2441143792
Resultado concorrente: 200.2441143792
Tempo sequencial: 0.4220877020
Tempo concorrente: 0.1573352050
Ganho concorrente: 2.68

Teste "função g"
Resultado sequencial: 0.5626533126
Resultado concorrente: 0.5626533126
Tempo sequencial: 3.7034608970
Tempo concorrente: 1.4074590660
Ganho concorrente: 2.63

Oito threads:

Teste "exponencial"
Resultado sequencial: -1.7182818285
Resultado concorrente: -1.7182818285
Tempo sequencial: 0.1598445130
Tempo concorrente: 0.0608158400
Ganho concorrente: 2.63

Teste "função a"
Resultado sequencial: 27.5000000000
Resultado concorrente: 27.5000000000
Tempo sequencial: 1.2876495040
Tempo concorrente: 0.6012197390
Ganho concorrente: 2.14

Teste "função b"
Resultado sequencial: 1.5707964928
Resultado concorrente: 1.5707964928
Tempo sequencial: 0.0907702800
Tempo concorrente: 0.0350728200
Ganho concorrente: 2.59

Teste "função c"
Resultado sequencial: 336002.4393136707
Resultado concorrente: 336002.4393136707
Tempo sequencial: 0.2500945230

Tempo concorrente: 0.0945207190
Ganho concorrente: 2.65

Teste "função d"
Resultado sequencial: 0.5466247424
Resultado concorrente: 0.5466247424
Tempo sequencial: 7.5087801440
Tempo concorrente: 2.9431480700
Ganho concorrente: 2.55

Teste "função e"
Resultado sequencial: -0.4640199794
Resultado concorrente: -0.4640199794
Tempo sequencial: 1.0116602560
Tempo concorrente: 0.3482843100
Ganho concorrente: 2.90

Teste "função f"
Resultado sequencial: 200.2441143792
Resultado concorrente: 200.2441143792
Tempo sequencial: 0.4342664340
Tempo concorrente: 0.1602538910
Ganho concorrente: 2.71

Teste "função g"
Resultado sequencial: 0.5626533126
Resultado concorrente: 0.5626533126
Tempo sequencial: 3.7435626600
Tempo concorrente: 1.3567158930
Ganho concorrente: 2.76

Uma thread:

Teste "cosseno"
Resultado sequencial: 1.0000000000
Resultado concorrente: 1.0000000000
Tempo sequencial: 0.8333844820
Tempo concorrente: 0.8219937510
Ganho concorrente: 1.01

Teste "exponencial"
Resultado sequencial: -1.7182818285
Resultado concorrente: -1.7182818285
Tempo sequencial: 0.1520503900
Tempo concorrente: 0.1525653140
Ganho concorrente: 1.00

Teste "função a"
Resultado sequencial: 27.5000000000

Resultado concorrente: 27.5000000000
Tempo sequencial: 1.2612289210
Tempo concorrente: 1.2603851640
Ganho concorrente: 1.00

Teste "função b"
Resultado sequencial: 1.5707964928
Resultado concorrente: 1.5707964928
Tempo sequencial: 0.0878766250
Tempo concorrente: 0.0859457490
Ganho concorrente: 1.02

Teste "função c"
Resultado sequencial: 336002.4393136707
Resultado concorrente: 336002.4393136707
Tempo sequencial: 0.2347668970
Tempo concorrente: 0.2386796170
Ganho concorrente: 0.98

Teste "função d"
Resultado sequencial: 0.5466247424
Resultado concorrente: 0.5466247424
Tempo sequencial: 7.4586255440
Tempo concorrente: 7.5911088940
Ganho concorrente: 0.98

Teste "função e"
Resultado sequencial: -0.4640199794
Resultado concorrente: -0.4640199794
Tempo sequencial: 0.9964058950
Tempo concorrente: 0.9880860340
Ganho concorrente: 1.01

Teste "função f"
Resultado sequencial: 200.2441143792
Resultado concorrente: 200.2441143792
Tempo sequencial: 0.4234160040
Tempo concorrente: 0.4194235950
Ganho concorrente: 1.01

Teste "função g"
Resultado sequencial: 0.5626533126
Resultado concorrente: 0.5626533126
Tempo sequencial: 3.6813626110
Tempo concorrente: 3.6582570990
Ganho concorrente: 1.01