MAC0438 – Programação concorrente – 1s2015
EP2
Data de entrega: 25/5/2015

Prof. Daniel Macêdo Batista

1 Introdução

Da wikipedia: "*Em matemática, uma **série de Taylor** é uma série de funções da seguinte forma:*

$$f(x) = \sum_{n=0}^{\infty} a_n (x-a)^n \tag{1}$$

na qual $a_n = \frac{f^{(n)}(a)}{n!}$ (`http://pt.wikipedia.org/wiki/S%C3%A9rie_de_Taylor`)

2 Problema

Funções trigonométricas podem ser calculadas a partir da série de Taylor. Por exemplo, o cosseno pode ser calculado, para qualquer x, como sendo:

$$cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \tag{2}$$

Somatórios infinitos como este acima costumam ser implementados de forma paralela com o objetivo de encontrar a melhor aproximação para a função em um dado x. Quanto maior a quantidade de termos do somatório, mais preciso o valor da função, desde claro que o hardware e a linguagem de programação utilizada tenham a precisão necessária para representar o número sendo calculado. Encontrar valores mais precisos de somatórios como este acima costuma ser considerado como um bom benchmark para avaliar novas arquiteturas de super computadores. Quanto mais preciso e mais rápido o cálculo das aproximações, melhor tende a ser visto o computador. Claro que de nada adianta um hardware poderoso se a implementação do código não for tão boa quanto e não souber tirar proveito do hardware.

Em implementações paralelas de somatórios (na verdade de séries infinitas de um modo geral) cada termo pode ser calculado de forma independente por uma thread/processo. De tempos em tempos, os termos são avaliados a fim de se verificar se o critério de parada foi atendido. O critério de parada pode ser por exemplo a diferença entre dois valores consecutivos ou o mínimo valor de um termo. O critério de parada pode ser verificado através de uma barreira de sincronização. No caso de somatórios, essa barreira pode por exemplo ter também o papel de somar todos os termos encontrados naquela iteração (rodada) em uma variável global.

Sua tarefa neste EP será implementar um algoritmo paralelo que calcule uma aproximação para a função cosseno. Seu programa deve ser implementado de modo a parar em duas condições (o usuário deve informar na linha de comando qual critério de parada ele quer): quando o módulo da diferença entre

o valor de $cos(x)$ em duas rodadas consecutivas for menor do que um valor `f` passado como entrada para o programa ou quando alguma thread calcular um termo menor, em módulo, do que um valor `m` passado como entrada para o programa.

3 Requisitos

3.1 Threads e barreira de sincronização

O seu EP terá dois modos de operação. No primeiro ele deverá criar uma quantidade de threads igual à quantidade de núcleos do computador (q) e essas threads deverão se sincronizar sempre que todas elas terminarem de encontrar q novos termos da série infinita (a descoberta da quantidade de núcleos do computador deve ser implementada no seu código). Ao fim de cada rodada, uma variável global contendo o valor do $cos(x)$ até o momento deve ser atualizada. O programa deve parar (i) quando o módulo da diferença entre dois valores de $cos(x)$ calculados por rodadas consecutivas for menor do que o valor `f` passado como parâmetro para o seu programa; ou (ii) quando alguma thread calcular um termo menor, em módulo, do que um valor `m` passado como parâmetro para o seu programa.

O segundo modo de operação difere do primeiro porque o usuário precisa passar a quantidade de threads que ele deseja criar. Ou seja, nesse modo o seu programa não precisa descobrir o valor de q já que o usuário vai informar quantas threads ele quer.

A barreira de sincronização utilizada no seu programa pode vir pronta de uma biblioteca ou de forma nativa da linguagem de programação que você utilizar. Você deve informar no LEIAME do seu programa qual barreira você utilizou. Ou seja, mesmo que você utilize uma barreira pronta você precisa ler a documentação da barreira e informar qual barreira é implementada na biblioteca.

IMPORTANTE: Seu programa não pode utilizar informações pré-calculadas que facilitem o cálculo do valor do cosseno. Por exemplo, você deve encontrar os fatoriais em tempo de execução. Utilizar uma tabela com fatoriais pré-calculados ou qualquer outra tabela com números pré-calculados que facilitem o cálculo dos termos implicará em nota ZERO no EP.

IMPORTANTE: Você não pode utilizar nenhuma biblioteca que já faça o cálculo do cosseno sozinha, ou que faça o cálculo de alguma outra função trigonométrica que facilite o cálculo do cosseno. O cálculo deve ser feito no código implementando as operações da fórmula acima. Programas que utilizem funções já prontas para calcular o cosseno implicarão em nota ZERO no EP.

3.2 Linguagem

Seu programa pode ser escrito em qualquer linguagem de programação que tenha compilador ou interpretador de código aberto disponível para o GNU/Linux.

3.3 Entrada e saída

O seu programa receberá como entrada obrigatoriamente na linha de comando e **nesta ordem**:

- primeiro argumento: um valor inteiro maior ou igual a zero. Quando for igual a zero significa que seu programa deve criar uma quantidade de threads igual à quantidade de núcleos do computador. Quando for diferente de zero significa que seu programa deve criar uma quantidade de threads igual a esse valor;

- segundo argumento: um caracter `f` ou `m`. Se for `f` significa que o EP deve parar quando o módulo da diferença entre dois valores de $cos(x)$ calculados por rodadas consecutivas for menor do que o valor do terceiro argumento. Se for `m` significa que o EP deve parar quando alguma thread calcular um termo menor, em módulo, do que o valor do terceiro argumento;
- terceiro argumento: um valor inteiro que definirá a precisão do cálculo e que terá significado a depender do segundo argumento passado pelo usuário. Se esse valor for 110 por exemplo e o segundo argumento tiver sido `f`, significa que o código precisa parar quando o módulo da diferença entre dois valores consecutivos calculados for menor do que 10^{-110};
- quarto argumento: o valor de x;
- quinto argumento (opcional): um caracter `d` ou um caracter `s` (ou nenhum desses dois parâmetros será passado ou apenas um será passado). Quando nenhum parâmetro opcional for passado na linha de comando o seu programa deverá imprimir apenas ao término da execução:
  - Número de rodadas, que vai ser um contador que armazena quantas vezes as threads se encontraram na barreira;
  - Valor encontrado do $cos(x)$ com alta precisão (não tem problema se a saída ocupar a tela inteira).

  Quando a opção `d` for passada na linha de comando o seu programa deverá imprimir:
  - A cada rodada, ordem com que as threads chegaram na barreira. Cada thread precisa de um identificador;
  - A cada rodada, valor parcial do $cos(x)$;
  - Ao término da execução, o número de rodadas, que vai ser um contador que armazena quantas vezes as threads se encontraram na barreira;
  - Ao término da execução, o valor encontrado do $cos(x)$.

  Quando a opção `s` for passada na linha de comando o seu programa deverá calcular o cosseno de forma sequencial **sem a utilização de threads ou processos paralelos**. Nesse caso o programa deverá imprimir:
  - Valores parciais do $cos(x)$ calculados **a cada novo termo calculado**;
  - Ao término da execução, o valor encontrado do $cos(x)$.
  - Ao término da execução, a quantidade de termos calculados.

3.4 Precisão dos números

O seu código deve obrigatoriamente utilizar tipos com maior precisão do que os tipos nativos da linguagem utilizada. Por exemplo, no caso de você escrever seu código em C, você pode utilizar a biblioteca GMP (`https://gmplib.org/`). Escolher uma biblioteca que permita cálculos com alta precisão faz parte da sua tarefa.

4 Sobre a entrega

Junto com o seu programa você deve entregar um relatório em .pdf apresentando uma análise de desempenho que você realizou no seu programa. O formato do relatório é livre mas deve conter obrigatoriamente:

- Configuração da(s) máquina(s) onde você executou o seu programa;
- Entradas (valores de `x`, `f` e `m`) que foram passadas para o seu programa;
- Quantidade de repetições que foram executadas para cada entrada;
- Gráficos comparando o desempenho, em termos do tempo de execução, do seu programa paralelo com diversas quantidades de threads e da versão sequencial para diversas entradas. Esses gráficos precisarão conter médias aritméticas e alguma informação de dispersão como desvio padrão, intervalo de confiança, etc...;
- Explicação de como o tempo de execução foi medido.
- Explicações sobre os resultados obtidos: foi o esperado? Não foi o esperado? etc...
- Comentários a respeito da biblioteca para manipulação dos números com alta precisão: como ela faz para obter a alta precisão, qual a precisão, se ela utiliza internamente tipos simples para representar as variáveis, quais são esses tipos, etc...

O relatório deve ser entregue junto com o programa no arquivo .tar.gz. Ou seja, este .tar.gz deve conter os seguintes itens:

- fonte, Makefile (ou similar), arquivo LEIAME;
- relatório em .pdf.

O nome do arquivo .tar.gz deve ser `ep2-membros_da_equipe.tar.gz` e quando desempacotado ele deve produzir um diretório contendo os itens. O nome do diretório deve ser `ep2-membros_da_equipe`. Por exemplo: `ep2-joao-maria`.

Obs.: O EP pode ser feito individualmente ou em dupla.

5 Bônus para o programa mais eficiente

Os EPs que receberem nota 10,0 passarão por uma avaliação extra para verificar qual é o mais eficiente. Os códigos serão executados em diversos cenários definidos pelo professor e aquele que rodar, em média, mais rápido devolvendo os valores mais precisos do $cos(x)$ será considerado o mais eficiente. Os autores deste EP ganharão 1,0 extra na média final.