MAC 0219/5742 – Introdução à Computação Concorrente, Paralela e Distribuída

Prof. Dr. Alfredo Goldman

3º Exercício Programa `versão 1.2`

Balanceamento de Carga em Aplicações Híbridas

Monitores: Giuliano Belinassi e Marcos Amaris

1 Introdução

Diversas áreas das ciências naturais utilizam equações diferenciais para modelar eventos de seu domínio, entretanto muitas vezes tais equações não têm solução analítica, sendo necessário empregar técnicas de cálculo numérico para obter suas soluções. Um passo recorrente em tais métodos é a computação de integrais numéricas; um dos assunto deste Exercício Programa.

Muito embora aproximar uma integral numericamente não seja uma tarefa de custo computacional elevado, aproximá-la com precisão arbitrária pode ser uma tarefa custosa e necessária, requisitando alocar diversas máquinas para o trabalho. Tais máquinas podem não ter o mesmo poder computacional e portanto a divisão de carga deve ser feita da melhor forma possível.

Neste EP, abordaremos não apenas uma técnica de integração nominada *Integração de Monte Carlo* – um modelo estatístico para computar integrais numericamente – mas também técnicas de computação distribuída, e vocês deverão aplicá-las para acelerar seu cálculo. Aqui o desafio não é o cálculo em si, mas sim como distribuir a carga de maneira a minimizar o tempo de execução do programa.

2 Integração de Monte Carlo em uma Dimensão

Embora a definição disponível no Wolfram seja genérica a ponto de abordar funções de várias variáveis, aqui trabalharemos em condições simplificadas, com funções de uma variável.

Seja $f : \mathbb{R} \longrightarrow \mathbb{R}$ uma função Riemann-Integrável no intervalo $]a, b[$. A *Integração de Monte Carlo* procura aproximar:

$$\int_a^b f(x)\mathrm{d}x \approx (b-a)\left(\langle f\rangle \pm \sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}}\right) \tag{1}$$

onde N é o número de pontos amostrado aleatoriamente, $\langle f\rangle$ e $\langle f^2\rangle$ são definidos por:

$$\langle f\rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \qquad \langle f^2\rangle = \frac{1}{N}\sum_{i=1}^{N} f^2(x_i) \qquad f^2(x) = f(x)f(x)$$

e cada $x_i \in]a, b[$ é amostrado aleatoriamente seguindo uma distribuição uniforme.

3 Integral a ser Calculada

Considere $f : \mathbb{Z} \times \mathbb{Z} \times \mathbb{R} \longrightarrow \mathbb{R}$ dada por

$$f(M, k, x) = \frac{\sin([2M + 1]\pi x) \cos(2\pi k x)}{\sin(\pi x)}$$

Vocês deverão calcular, usando *Integração de Monte Carlo*:

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} f(M, k, x)\mathrm{d}x = 2\int_{0}^{\frac{1}{2}} f(M, k, x)\mathrm{d}x = \begin{cases} 1 & \text{se } |k| \leq |M| \text{ e } M \geq 0 \\ -1 & \text{se } |k| \leq |M| \text{ e } M < 0 \\ 0 & \text{caso contrário} \end{cases} \quad (2)$$

Note que a variável de integração é x, e portanto os inteiros M e k são tratados como constantes na integração. Esta integral foi escolhida por três razões: O Wolfram não consegue computá-la algebricamente; há problemas de precisão quando M e k tornam-se suficientemente grandes; e é muito difícil resolvê-la algebricamente.

4 Balanceamento de Carga

Neste EP, deve ser usado o `MPI` para realizar a comunicação entre diversas máquinas. O `MPI` (*Message Passing Interface*) é um padrão que permite a programação paralela e distribuída de aplicações computacionais. Assim problemas são resolvidos usando *divisão e conquista*. Para usar essas soluções distribuídas um balanceamento de carga deve ser feito e mais ainda quando se tem recursos computacionais heterogêneos tais como CPUs distintas, GPUs e FPGAs. Em sua implementação com `MPI`, o processo 0 (*master*) deve ser encarregado de enviar o trabalho para a GPU, e o processo 1 deve ficar encarregado de distribuir a carrega usando `OpenMP` ou `Pthreads`.

Para usar esses recursos computacionais como GPUs, os dados devem ser transferidos para a memória principal GPU usando o barramento PCI Express. Esse envio tem um custo computacional que as vezes não compensa a utilização desses aceleradores. Nesse trabalho deve ser feito o balanceamento de carga durante a execução da técnica de integração de Monte Carlo. Um processo de tuning ou treinamento deve ser feito para fazer esse balanceamento.

O kernel de CUDA desenvolvido deve ser testado com diferentes tamanhos do problema, medindo em cada execução o tempo gasto pela transferência dos dados, execução do Kernel e a transferência da resposta. Também a função desenvolvida para CPU deve ser testada usando `OpenMP` ou `Pthreads` será paralelizada em um conjunto de threads. Finalmente, o balanceamento vai responder para o método de Monte Carlo quando vale a pena usar uma pralelização usando só CPU em memória compartilhada, quando quando usar só GPU e quando usar os dois dispositivos CPUs e GPUs.

5 Entrada/Saída

O programa deve receber três argumentos: O tamanho do problema N (número de amostras), e os parâmetros inteiros k e M, da seguinte forma:

```
./main N k M
```

A saída deve conter as seguintes informações:

1. Tempo gasto para resolver o problema utilizando o seu algoritmo de distribuição de carga. Aqui é permitido usar CPU com várias threads, GPU, ou ambos. O que importa é que aqui o tempo seja o menor possível.
2. Tempo gasto para resolver o problema na GPU usando apenas uma thread na CPU.
3. Tempo gasto para resolver o problema usando T threads, onde T pode ser obtido em função do número de processadores da máquina
4. Tempo sequencial usando apenas uma CPU e uma thread.
5. Os erros da integral com o valor calculado algebricamente, conforme exibido na Eq (2). Note que a Eq. (1) admite dois valores, e portanto deverão ser impressos ambos os erros.

e ela deve seguir:

```
Tempo com balanceamento de carga em segundos: t1
Erro no calculo com a soma: e1
Erro no calculo com a subtracao: e2
-----------------------------------------------------
Tempo na GPU com uma thread na CPU em segundos: t2
Erro no calculo com a soma: e3
Erro no calculo com a subtracao: e4
-----------------------------------------------------
Tempo na CPU com T threads em segundos: t3
Erro no calculo com a soma: e5
Erro no calculo com a subtracao: e6
-----------------------------------------------------
Tempo sequencial em segundos: t4
Erro no calculo com a soma: e7
Erro no calculo com a subtracao: e8
```

Onde `t1`,`t2`,`t3`,`t4`, `e1`, `e2`, `e3`, `e4`, `e5`, `e6`, são números em ponto flutuante.

6 Restrições

1. A integral de $f(M, k, x)$ deve ser computada usando *Integração de Monte Carlo*, embora existam maneiras mais eficientes para calcular integrais numericamente (veja Quadratura Gaussiana).
2. O número de amostras N cabe em um inteiro de 64-bits, mas pode não ser possível alocar um vetor de tamanho N.

3. O programa deve ser implementado em `CUDA` C/C++ e deve usar `MPI` para estabelecer uma conexão entre diversas máquinas, `pthreads` ou `OpenMP` para um uso eficiente da CPU, e `CUDA` para o uso da GPU.

4. Parte da integração deve ser feita na GPU quando valer a pena executá-la nesta.

5. Não testaremos com mais de uma GPU. Seu programa não precisa tratar isto.

7 Dicas

- Note que $f(M, k, 0)$ não está definido e calcular f neste ponto levantará um `NaN`. A probabilidade de se sortear 0 amostrando entre $[0, \frac{1}{2}]$ em ponto flutuante é baixa, mas pode ocorrer. Você pode evitar isto simplesmente retirando o 0 do intervalo. Será que 10^{-320} é próximo o suficiente de 0?

- Escreva um teste automatizado que compara o resultado obtido na GPU com uma implementação sequencial do mesmo algoritmo.

- O balanceamento de carga pode ser feito em três fases, em toda as fases é necessário que usem MPI para ou envio e recebo de dados mais esses tempos não serão considerados. Em uma fase, se faz o Tuning de diferentes tamanhos de N sobre a GPU considerando só a transferência de dados de ida e volta CPU-GPU e a execução do kernel. Em outra fase, a execução é feita usando só CPU com diferentes tamanhos de N e finalmente em outra fase a versão hibrida usando CPU + GPU.

8 Entrega

Deverá ser entregue um pacote no sistema PACA com uma pasta com o nome e o sobrenome do estudante que o submeteu no seguinte formato: `nome.sobrenome.zip`. Se o EP for feito em dupla, o formato deve ser `nome1.sobrenome1.nome2.sobrenome2.zip`. Somente um estudante da dupla submeterá a tarefa. Essa pasta deve ser comprimida em formato `ZIP` e deve conter dois itens:

- Os códigos fonte do programa, em conjunto com um `Makefile` que o compila, gerando um executável `main`.

- Um relatório em `.txt` ou `.pdf`

Em caso de dúvidas, use o fórum de discussão do Paca. A data de entrega deste Exercício Programa é até às **23:55 horas da Quinta-Feira, 28 de Junho**.