Implementação do Modelo Eta com CUDA

Alex Lima de Mello[1],Henrique Gavioli Flores[2], Marcelo Trindade Rebonatto[12], Carlos amaral Holbig[12]

[1]nstituto de Ciências Exatas e Geociências – Universidade de Passo Fundo (UPF)
Caixa Postal XXXXXXX – XXXXXXX – Passo Fundo – RS – Brasil

[2]Programa de Pós-Graduação em Computação Aplicada
Universidade de Passo Fundo (UPF) – Passo Fundo, RS, Brasil

{122596,119694,rebonatto,holbig}@upf.br

Resumo. *Eta é um modelo de previsão meteorológica e climática criado nos anos 70 e utilizado até hoje. Modelos de previsão necessitam de uma grande quantidade de dados de entrada e tendem a possuir um tempo de execução consideravelmente grande. O intuito deste trabalho é avaliar a viabilidade de CUDA como aprimoramento ao modelo Eta, reduzindo o tempo de processamento e mantendo a integridade dos resultados do modelo.*

1. Introdução

Modelos atmosféricos são modelos matemáticos construídos com base no conjunto de equações dinâmicas que governam os movimentos atmosféricos [Flores 2016]. Tais modelos podem ser usados para prever o clima e a meteorologia de uma determinada área

O Eta é um modelo de previsão atmosférica voltado para o uso de pesquisa e decisões operacionais. Este modelo é descendente do modelo Hydrometeorological Institute and Belgrade University (HIBU), após várias atualizações foi reescrito em 1988 para usar a coordenada vertical Eta. [INPE/CPTEC]

Devido à quantidade de informações e a complexidade matemática do modelo, a execução do Eta implica em um alto custo, tanto em tempo quanto em recursos computacionais. Uma das formas de diminuir o tempo de execução do modelo é explorar outros recursos computacionais, como por exemplo GPUs.

2. Materiais e Métodos

O modelo Eta é atualmente escrito em Fortran 90, contando com alguns trechos de bibliotecas escritos em Fortran 77. A paralelização do modelo é realizada utilizando MPICH, uma implementação da tecnologia Message Passing Interface (MPI).

Os processos criados para a execução do modelo são definidos como tarefas de previsão ou servidores de entrada e saída (servidores de I/O), a quantidade de cada uma pode ser definida individualmente pelo usuário. Os servidores de I/O serão responsáveis por armazenar os resultados obtidos pela execução do modelo, enquanto as tarefas de previsão são responsáveis pelo cômputo do modelo.

A área a ser processada pelo Eta é definida por uma matriz, com a quantidade de elementos no eixo vertical e horizontal definidos nos parêametros do experimento sendo realizado. A matriz é sobreposta no mapa terrestre, tendo como ponto central a latitude

e longitude, também definidas no experimento. O tamanho de cada elemento da matriz é definido pela resolução do modelo em quilômetros. A área total a ser processada é dividida em subáreas, das quais cada tarefa de previsão é responsável por uma delas.

Após cada fase de cálculos, os dados são enviados para o servidor de I/O, que ficará responsável por gravar as informações em disco enquanto as tarefas de previsão prosseguem para a próxima iteração, se disponível, na próxima vez será utilizado um servidor de I/O diferente. Caso seja definido zero servidores de I/O, a gravação será efetuada pelas próprias tarefas de previsão, todas gravam os dados em um mesmo arquivo.

Foi realizada uma análise do código, buscando identificar atividades computacionalmente intensivas, com o intuito de identificar pontos de paralelismo. Foi escolhida a tecnologia CUDA para tentar obter um ganho de desempenho sem alterar a lógica utilizada pelo modelo. Apesar de acreditarmos que existam melhorias em relação as operações matemáticas e meteorológicas, neste trabalho nos concentramos apenas no aspecto computacional.

Dos pontos de paralelismo identificados, foram escolhidos 3 pontos para a implementação com CUDA, os pontos escolhidos realizam multiplicação entre valores de três diferentes matrizes, tarefa favorável à execução paralela pela GPU.

Com a utilização de CUDA, a paralelização no Eta agora é composta por 2 níveis: O primeiro MPI, dividindo a área a ser computada entre diferentes processos; o segundo CUDA, paralelizando os cálculos de complexidade elevada.

3. Implementação

Dentro dos blocos, as threads possuem identificadores únicos (ID) para cada dimensão, são eles threadIdx%x, threadIdx%y e threadIdx%z, para as dimensões X, Y e Z respectivamente. Utilizando a ID da thread, a ID do bloco (blockIdx%x, blockIdx%y e blockIdx%z) a que pertence e conhecendo o tamanho do bloco, definido por blockDim em cada direção, é possível acessar as threads de forma contínua, independentemente do bloco a que pertencem.

Os índices da matriz são calculados com base na ID de cada thread. As threads que possuem valor do índice menor que o início, ou maior que o fim de cada dimensão da matriz são finalizadas, restando uma para cada elemento a ser trabalhado.

Dentro do arquivo fonte VTADV foram criados 3 kernels, responsáveis por conter os códigos a serem executados no device (GPU). As threads criadas pela mesma CPU competem pelo uso da GPU, sendo que, em cada loop da fase de cálculos, dois dos kernels são executados apenas uma vez, enquanto o outro é chamado duas vezes.

Para efetuar os cálculos, cada kernel é chamado, tomando como parêametros uma cópia das matrizes a serem trabalhadas e as coordenadas referentes aos índices a serem calculados pelo processo. Para cada dimensão da matriz existem duas variáveis, uma correspondendo ao índice inicial da dimensão e outra correspondendo ao índice final.

Para os testes da implementação do Eta utilizando CUDA foram utilizados 20 computadores com CPU Intel Core i7-3770 de 64 bits, com um clock de 3,40 GHz, com 4 nécleos físicos e 4 lógicos; 8GiB de RAM (Random-access memory); Sistema operacional Ubuntu versão 16.04 LTS para processadores de 64 bits; GPU GeForce GT 630 da Nvidia,

possuindo 384 CUDA cores, 2 GiB de memória, interface 64 bit-DDR3, com 14.4 Gigabytes de largura de banda por segundo [NVIDIA a]. Como servidor de I/O foi utilizado um computador com CPU Intel Core i7 920 de 64 bits, com um clock de 2,66 GHz, 4 cores e 8 threads; 8 GiB de RAM; Sistema operacional Linux, distribuição Ubuntu versão 16.04 LTS para processadores de 64 bits; GPU GeForce GTX 770 da Nvidia, possuindo 1536 CUDA cores e 2 GiB de memória, interface 256-bits GDDR5 e largura de banda de 224.3 Gigabytes por segundo [NVIDIA c]. As máquinas se comunicam usando uma rede FastEthernet.

A GPU utilizada possui limite de 1024 threads por bloco, dimensões máximas de 1024,1024,64 para x, y e z respectivamente, e warp size de 32, significando que independentemente do tamanho do bloco, o némero de threads utilizadas será sempre méltiplo de 32. [NVIDIA d]

Foi utilizada a versão 17.10 do compilador PGI edição comunitária, que pode ser obtida de forma gratuita no pacote disponibilizado pela NVIDIA , com licença para 90 dias, ou por um ano para estudantes de áreas relacionadas. [NVIDIA b]

Para a compilação do modelo Eta utilizando CUDA, é importante verificar que esteja instalado o driver CUDA, disponibilizado pela Nvidia. Deve-se acrescentar a flag "-Mcuda" ao arquivo responsável por controlar as flags de compilação (makefile).

4. Resultados

Com o objetivo de analisar o ganho de desempenho da implementação realizada, foram realizados testes com três áreas diferentes, denominadas: GRANDE(251x581), MÉDIA(181x349) e PEQUENA(101x159), mantendo a resolução de 10Km em todas. Para os testes foram utilizados até 80 processos e um servidor de I/O. O período realizado foi de 12 horas de previsão, compreendido a partir de zero horas do dia 01/01/2009. Cada um dos testes foi repetido 20 vezes, o tempo médio e o coeficiente de variação foram então calculados. Os coeficientes de variação variam de aproximadamente 0,72% à 3.9%, mostrando que não ocorreu interferência significativa de valores externos durante a execução dos testes. Como se pode esperar, o coeficiente de variação tende a aumentar para os menores tempos de execução.

Os testes se dividem em dois grupos: A implementação original do modelo Eta e a implementação com CUDA. A Tabela 1 compara o tempo de execução em segundos dos testes realizados.

Table 1. Tempo médio de execução em função do número de Processos

Processos/horas de previsão	PEQUENA	MÉDIA	GRANDE
80 processos	191,67	357,67	582,87
80 processos com CUDA	194,65	361,07	588,05
60 processos	169,56	332,20	577,09
60 processos com CUDA	166,32	330,77	573,35
40 processos	144,49	285,24	544,04
40 processos com CUDA	143,15	283,60	543,23

A Tabela 1 apresenta o resultado dos testes preliminares. é possível observar que a

integração de CUDA em relação a execução do modelo sem o uso dessa tecnologia afetou de forma pouco significativa os resultados, tanto positivamente quanto negativamente.

é possível analisar que com a diminuição do número de processos obteve-se um tempo de execução menor que em relação a execução com 80 processos, este resultado se dá pelo overhead dos processos ser maior que o ganho de desempenho obtido.

5. Considerações finais

Os resultados obtidos mostram que a integração de CUDA na implementação atual do modelo ETA resulta em uma mudança pouco significativa nos tempos de execução, em alguns casos resultando em um tempo maior que o da implementação com MPI.

Outro possível ponto de estudo é utilização de outras tecnologias de paralelismo utilizando a GPU, como por exemplo OpenACC, que possui uma forma de utilização similar ao OpenMP. Futuramente, pode-se desenvolver um trabalho com objetivo de integrar a tecnologia CUDA em todos os pontos do modelo Eta que se identifique o benefício do uso da ferramenta, ou até mesmo a criação de um modelo paralelizado apenas com recursos de GPU.

6. Referências

References

Flores, H. G. (2016). Estudo e aprimoramento do modelo eta utilizando recursos de computação paralela e distribuída.

INPE/CPTEC. Model | Eta Model. http://etamodel.cptec.inpe.br/history.

NVIDIA. NVIDIA GeForce GT 630. http://www.nvidia.com.br/object/geforce-gt-630-br.html#pdpContent=2.

NVIDIA. OpenACC: More Science Less Programming. https://developer.nvidia.com/openacc.

NVIDIA. Placa de vídeo GTX 770 com GPU Boost 2.0. http://www.nvidia.com.br/object/geforce-gtx-770-br.html#pdpContent=2.

NVIDIA. Programming Guide :: CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#compute-capabilities.