Estudo e aprimoramento do modela Eta integrando CUDA

Alex L. Mello[1], Henrique G. Flores[2], Marcelo T. Rebonatto [1,2], Carlos A. Holbig[1,2]

[1]Curso de Ciência da Computação, Instituto de Ciências Exatas e Geociências – UPF

[2]Programa de Pós-Graduação em Computação Aplicada (PPGCA) – UPF

122596@upf.bf, 119694@upf.bf, rebonatto@upf.br, holbig@upf.br

Resumo. *Eta é um modelo de previsão meteorológica criado nos anos 70 e utilizado até hoje, passando por diversas atualizações e alterações ao longo dos anos. 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

Os modelos de previsão são softwares que fazem uso de modelos matemáticos com dados da atmosfera e dos oceanos visando prever o clima e a meteorologia de uma região por um determinado período de tempo [Flores 2016].

O Eta é um modelo de previsão regional, que prevê o clima e tempo para uma região determinada, voltado para o uso de pesquisa e decisões operacionais. Este modelo se baseia na utilização de coordenadas verticais, tornando-o adequado para o estudo de regiões topográficas íngremes, como por exemplo, a cordilheira dos Andes[Flores 2016].

Devido à quantidade de informações e a complexidade dos cálculos realizados, a execução do Eta se torna custosa e relativamente demorada. Visando este problema, se torna importante a busca de uma possível solução que possa otimizar o tempo de execução do modelo Eta, realizando o bom uso das tecnologias das GPUs para o processamento do modelo, recurso atualmente não explorado.

O restante deste texto está dividido em seis outras seções. Na seção 2 é explicado como se dá o processamento do Eta. Em seguida, na seção 3, é explicado a implementação realizada. A seção 4 contém os resultados obtidos, enquanto as seções 5 e 6 correspondem às considerações finais e referências do trabalho,respectivamente.

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 MPI (Message Passing Interface).

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 (JNPES) e horizontal (INPES) definidos nos parâmetros do

experimento sendo realizado. A matriz é sobreposta no mapa terrestre, tendo como ponto central a latitude (Lat) e longitude (Lon), também definidas no experimento. O tamanho de cada elemento da matriz é definido pela resolução do modelo (Res) em Kilometros. 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 do modelo, 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, utilizando acesso direto, todas gravam os dados em um mesmo arquivo.

Foi realizada uma análise do código, buscando identificar atividades computacionalmente intensivas, com o intuito de identificar possíveis pontos de paralelismo. Devido à natureza matemática do Eta, foi escolhida a tecnologia CUDA para tentar obter um ganho de desempenho sem alterar a lógica utilizada pelo modelo. É possível que existam melhorias em relação às operações matemáticas e meteorológicas, porém, os envolvidos neste projeto não possuem conhecimento suficiente em tais áreas.

Dos pontos de paralelismo identificados, foram escolhidos 2 pontos para a implementação com CUDA, ambos pertencentes à sub-rotina VTADV, responsável por calcular a contribuição da advecção vertical para as tendências de temperatura, umidade específica, etc. Os pontos escolhidos realizam multiplicação entre valores do índice de 3 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, onde as threads MPI executam tarefas de complexidade elevada são executadas pela GPU. Todas as tarefas responsáveis pelo cômputo do modelo utilizam a GPU, pois executam as mesmas sub-rotinas, com valores diferentes.

3. Implementação

Dentro dos blocos, as threads possuem identificadores únicos (ID) para cada dimensão, denominados threadId, 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.

Dentro do VTADV, sub-rotina sobre a qual se foi trabalhado, foi criado um kernel responsável por conter o código a ser executado no device (GPU). Para efetuar o cálculo, o kernel é chamado, tomando como parâmetros uma cópia das matrizes a serem trabalhadas e as coordenadas referentes aos índices a serem calculados pelo processo. Para cada dimensão da matriz a ser trabalhada existem duas variáveis, uma correspondendo ao índice inicial da dimensão e outra correspondendo ao índice final, como pode ser observado na Figura 1.

Na Figura 1, i e j representam os eixos horizontais e verticais da matriz, MYIS, MYIE, MYJS2 e MYJE2 são, respectivamente, os valores iniciais e finais a serem trabal-

```
!calculando valores a serem utilizados como índices
i = ((blockIdx%x - 1) * blockDim%x) + threadIdx%x
j = ((blockIdx%y - 1) * blockDim%y) + threadIdx%y
!finalizando threads não utilizadas
if(i<1 .OR. i>(MYIE-MYIS +1) .OR. j<1 .OR. J>(MYJE2-MYJS2+1)) return
!cada thread computa e armazena uma posição da matriz
d_a(i,j) = d_b(i,j,1) * d_c(i,j,1) * d_d(1)
```

Figure 1. Exemplo de um kernel utilizado.

hados na matriz pelo processo que originou o kernel, dessa forma é definido o intervalo a ser trabalhado. d_a, d_b, d_c e d_d representam as matrizes envolvidas na operação.

Para os testes da implementação do modelo Eta utilizando CUDA foi utilizado um computador com CPU Intel Core i7 920 de 64 bits, com um clock de 2,66GHz, 4 cores e 8 threads; 8GiB de RAM (Random-access memory); Sistema operacional Ubuntu versão 16.04 LTS para processadores de 64 bits; GPU GeForce GTX 770 da Nvidia, possuindo 16384 CUDA cores e 2GiB de memória de interface GDDR5, com largura de 256 bits e banda de 224.3 Gigabytes por segundo.

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 número de threads utilizadas será sempre múltiplo de 32[NVIDIA 2016b].

Foi utilizada a versão 16.5 do compilador PGI para Fortran 90, que pode ser obtida de forma gratuita no pacote disponibilizado pela NVIDIA denominado “NVIDIA OPENACC TOOLKIT”, com licença para 90 dias, ou por um ano para estudantes de áreas relacionadas.[NVIDIA 2016a]

4. Resultados

Com o objetivo de analisar o ganho de desempenho da implementação realizada, foram realizados testes com 2 áreas diferentes, uma com 201x259 elementos e outra 101x159, mantendo a resolução de 10Km nas duas. Para ambas as áreas, foi realizado um teste com 7 processos MPI (6 tarefas de processamento e um servidor de I/O) e outro com 4 processos (3 + 1). Os testes com diferentes números de processos permitem observar a diferença entre o desempenho quanto a utilização de um número de threads maior que a quantidade de núcleos físicos do processador, enquanto os diferentes tamanhos de área possibilitam observar como a GPU se comporta com o aumento no número de elementos a serem processados. O período realizado foi de 6 horas de previsão, compreendido a partir de zero horas do dia 01/01/2009. Cada um dos testes foi repetido cinquenta vezes e, o tempo médio foi então calculado, de cada conjunto sendo excluídos do cálculo o maior e o menor tempo encontrado. Os testes se dividem em 2 grupos: A implementação original do modelo Eta e a implementação com CUDA. A Tabela 1 compara o tempo de execução dos testes realizados.

A Tabela 1 apresenta o resultado dos testes correspondentes à execução total do modelo. Para a execução utilizando 4 processos, os testes implementados com CUDA executaram em tempo menor para os dois tamanhos de áreas testados. Para a área de 101x159 o tempo médio sem CUDA foi aproximadamente 1,24% menor que o tempo

Table 1. Tempo médio de execução em função da área coberta, em segundos.

Área/Execução	7 processos	7 processos com CUDA	4 processos	4 processos com CUDA
Área de 101×159	218,43	214,37	790,40	795,82
Área de 201×259	217,51	215,16	805,32	788,16

de execução sem CUDA. Para os testes com a área maior (201x259) a diferença foi de 17,16 segundos, representando um ganho de desempenho de aproximadamente 2,13% na execução com uso de CUDA.

Para a execução utilizando 7 processos, os testes implementados com CUDA apresentam um ganho de desempenho de aproximadamente 1,85% para a área menor. Para o teste com a área maior, os tempos foram 790,40 segundos sem CUDA e 795,82 com CUDA, um aumento de 0,68%. A perda de desempenho pode ser explicada uma vez que quantidade de elementos da área total a ser processada é maior que o número de CUDA cores existentes na GPU utilizada e pelo maior número de processos competindo pelo seu uso. Mesmo que a CPU da máquina tenha 8 threads, apenas 4 núcleos físicos são disponibilizados, ou seja, num ambiente real seriam usados no máximo o número de núcleos físicos da CPU.

5. Considerações finais

Com os resultados obtidos é possível afirmar que CUDA é uma estratégia promissora e eficiente, sendo assim vantajosa sua integração ao Eta. O ganho de desempenho é possível não só na seção analisada, mas em todo o modelo, nas partes onde ocorra uma complexidade elevada na utilização de processamento lógico e aritmético. Ou seja, a implementação do modelo Eta com CUDA pode produzir uma redução significativa em seu tempo de execução. Por exemplo, apenas no VTDAV, sub-rotina onde foram realizados os testes, é possível usar CUDA em 54 pontos, com códigos variando de complexidade O(m*n) até O(n4). Além disso, outros 5 fontes, que consomem 47% do tempo total podem ser explorados.

Para a realização de trabalhos futuros pode ser investigada a utilização de outras tecnologias de paralelismo utilizando a GPU, como por exemplo OpenACC. 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.

6. References

References

Flores, H. G. (2016). Estudo e aprimoramento do modelo eta utilizando recursos de computação paralela e distribuída. *Proposta de trabalho de conclusão (Mestrado em Computação Aplicada)*.

NVIDIA (2016a). Openacc: More science less programming. Disponível em: https://developer.nvidia.com/openacc. Acesso em: Novembro, 2016.

NVIDIA (2016b). Programming guide: Cuda toolkit documentation. Disponível em: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html. Acesso em: Novembro, 2016.