Estudo da integração de GPGPU ao modelo Eta sem common blocks

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

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

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

Resumo. *O modelo de previsão numérica do tempo Eta foi criado na década de 1970. No Brasil, o Eta é utilizado deste 1997 para previsão do tempo e desde 2002 para previsão do clima. A implementação atual do modelo Eta utiliza common blocks, recurso incompatível com diversas tecnologias de processamento de alto desempenho. Este trabalho utiliza uma versão do modelo Eta sem common blocks para avaliar o uso de GPGPU no mesmo.*

1. Introdução

Modelos de previsão numérica do tempo (NWP) são modelos compostos por equações matemáticas utilizadas para montar um modelo da atmosfera. Com base em dados atmosféricos coletados, tais modelos conseguem calcular informações como chuva, vento e temperatura, para um determinado período de tempo no futuro. Modelos de NWP podem ser classificados como modelos globais ou regionais. Modelos globais são utilizados para realizar a previsão de condições meteorológicas para todo o planeta, enquanto modelos regionais realizam a previsão para uma determinada região.

O Eta é um modelo de NWP regional descendente do HIBU desenvolvido pelo Instituto meteorológico e Universidade de Belgrado na década de 1970. Ele recebe este nome por utilizar o esquema de coordenada vertical eta [Mesinger et al. 1988], com base na coordenada sigma. A coordenada eta é caracterizada por reduzir erros gerados pelo gradiente de pressão em regiões íngremes, como por exemplo, na Cordilheira dos Andes [Chou and Peters 2018].

A versão operacional do modelo Eta é escrito em Fortran 90, a partir da versão original escrita em Fortran 77 e paralelizado com *Message Passing Interface* (MPI) [Flores 2018], porém foi mantido o uso de *common blocks*, recurso comumente usado em Fortran 77 para comunicação de variáveis entre trechos do programa. Este trabalho tem como objetivo reduzir o tempo de execução do modelo Eta, fazendo uso de GPUs para realizar parte do processamento, em uma versão do modelo sem *common blocks*.

O restante deste texto está dividido em quatro seções. Na seção 2 são explicados os recursos e métodos utilizados. Em seguida, na seção 3, é explicada a implementação realizada. Enquanto as seções 4 e 5 correspondem às considerações finais e referências do trabalho, respectivamente.

2. Materiais e Métodos

Common blocks são blocos de memória compartilhada utilizados em Fortran 77, linguagem originalmente utilizada pelo Eta. Por não existirem variáveis globais em Fortran 77, *common blocks* são utilizados para que não seja necessário comunicar variáveis por

parâmetro entre sub-rotinas [Stanford University 1995]. O uso de *common block* é considerado obsoleto em Fortran 90 e inviabiliza o uso de tecnologias como, por exemplo, CUDA, OpenCL, OpenACC e OpenMP [Flores 2018, Willis 2013], sendo necessário fazer uso de módulos como alternativa.

O guia e referência para programação em Fortran com CUDA [PGI Compilers and Tools 2018] especifica recursos incompatíveis com o uso de *common blocks*, porém compatíveis com módulos, como por exemplo, variáveis com os atributos *device*, *managed*, *constant* e *shared*.

MPI é um padrão para computação paralela que utiliza troca de mensagens para comunicação entre processos. No modelo Eta, MPI é utilizado para dividir a área de processamento entre diferentes tarefas (grão grosso), que podem estar em um mesmo computador, ou em diferentes computadores conectados por rede. Uma versão do Eta, com uso de GPUs foi desenvolvida em [Flores 2018]. Ela utilizou a tecnologia CUDA, com três pontos de paralelização, realizando um paralelismo de execução em dois níveis: MPI+CUDA. A limitação do uso de CUDA em escassos pontos ocorreu devido ao uso de *common blocks*, causando incompatibilidade com CUDA. Além disso, o trabalho de Flores (2018) buscou explorar o uso da tecnologia OpenMP, da mesma forma incompatível o uso com *commom block*.

No ano de 2018 foi desenvolvida uma nova versão do Eta, sem o uso de *common blocks*. Esta nova versão, atende a demandas de atualização do código fonte do modelo e abre possibilidade para exploração de tecnologias incompatíveis com a versão anterior, tais como CUDA e OpenMP. Ela será utilizada neste trabalho para ampliar o uso de GPUs na execução do modelo Eta [GOMES 2018].

As tecnologias de computação de alto desempenho estudadas para uso nesse trabalho foram OpenACC, CUDA e OpenCL. Enquanto CUDA e OpenACC possuem ampla compatibilidade com o compilador da Portland Group INC (PGI) utilizado no modelo Eta, OpenCL não é atualmente suportado pelo mesmo.

OpenACC é uma API desenvolvida em conjunto por Cray e Nvidia para programação em sistemas heterogêneos. OpenACC pode ser utilizado tanto em CPUs quanto em GPUs, e funciona através de diretivas de compilador [Nvidia Corporation 2015].

Utilizando OpenACC o programador indica ao compilador trechos de códigos a serem paralelizados, desta forma, a otimização é dependente do compilador, visto que o programador não possui controle sobre o código a ser executado na GPU [Larkin 2018].

Este processo tem como vantagem a fácil e rápida implementação de paralelismo, no entanto, não é possível garantir o máximo ganho de desempenho, sendo possível que soluções de baixo nível apresentem melhor desempenho.

Tecnologias de GPGPU são ideais para implementação em modelos de NWP devido a grande quantidade de *threads* encontradas em uma GPU [Michalakes and Vachharajani 2008], além disso, as *threads* podem ser organizadas em blocos tridimensionais. Modelos de NWP são compostos em grande parte de operações sobre matrizes de duas ou três dimensões, desta forma, cada *thread* pode ser usada para representar um elemento das matrizes utilizadas no modelo.

Por serem recursos especializados para processamento em matrizes, os processadores das GPUs possuem unidades de controle reduzidas em relação às CPUs, e como consequência, as *threads* executam em *lockstep*, ou seja, toda *thread* executa o mesmo conjunto de instruções, sobre dados diferentes. Tal formato de execução é nomeado *Single Instruction, Multiple Thread* (SIMT) no *white paper* da arquitetura Fermi [Nvidia 2009].

Devido à capacidade de processamento paralelo de GPUs, o desempenho obtido geralmente é limitado pelo acesso de memória, considerando que para execução de um *kernel* (rotina executada na GPU) é necessário copiar os dados necessários para a GPU, e ao termino do processamento, copiar os resultados de volta para a memória acessível pela CPU. Para mitigar esta limitação busca-se manter os dados na GPU quando possível, e copiar para a CPU somente o necessário, desta forma melhorando a relação entre cópia de memória e processamento.

Como as operações do modelo Eta envolvem um grande número de matrizes, busca-se executar todo o processamento de uma sub-rotina em GPU de forma contínua, dessa forma reduzindo cópias de resultados anteriores para a CPU devido a dependências.

3. Implementação

Para a implementação foi escolhida inicialmente a sub-rotina VTADV, responsável por calcular a contribuição da advecção vertical para as tendências de temperatura e umidade específica, entre outras variáveis. Foi criada uma versão do VTADV com o objetivo de executar todo seu processamento em GPU.

Cada iteração da fase de cálculos do modelo corresponde a 20 segundos de previsão, e a cada duas iterações a rotina VTADV é executada. Como exemplo, utilizando o modelo eta para realizar uma previsão de 12 horas (43200 segundos), são realizadas 2161 iterações, e o VTADV é executado 1081 vezes, consumindo 16% do tempo total do modelo.

O modelo Eta é composto por aproximadamente 72 mil linhas de código em Fortran. Destas, 934 linhas correspondem à versão original da sub-rotina VTADV, enquanto a versão com CUDA contém atualmente duas mil linhas.

Esta nova versão da sub-rotina apresentou diversos erros de execução, porém os mesmos foram solucionados ao usar a implementação do modelo Eta sem uso de *common blocks*. É conveniente salientar que os resultados obtidos pela nova versão com CUDA são validados, por meio de comparações binárias dos arquivos de resultados com a versão enviada pelo INPE.

A sub-rotina VTADV com CUDA ainda está em desenvolvimento e atualmente composta por mais de 40 *kernels* CUDA. A paralelização de mais trechos de código com o uso de CUDA permite que certos dados sejam mantidos na memória da GPU, possibilizando um aumento no desempenho do modelo. Esse número com certeza será alterado, agrupando em *kernels* rotinas similares na fase de otimização dos acessos a memória.

4. Considerações finais

A atual versão operacional do modelo Eta inviabiliza a implementação de novas tecnologias de processamento de alto desempenho. Após adaptar o modelo para utilizar módulos, retirando os *common blocks*, é possível prosseguir com testes para o emprego de GPUs,

com as tecnologias CUDA ou OpenACC. Além disso, outros padrões de alto desempenho, como OpenMP poderiam no futuro ser explorados.

Os problemas identificados com o uso de CUDA no Eta foram resolvidos. A versão atual do novo código desenvolvido, já realiza o processamento de toda a rotina VTADV com o uso de GPUs, permitindo que CUDA possa ser explorada em aproximadamente 16% do tempo de execução do modelo Eta. Os resultados preliminares estão corretos, porém ainda não foi finalizada a otimização dos acessos a memória, elemento essecial para um bom desempenho em GPGPU.

Dando continuidade a pesquisa, é necessário avaliar a diferença de desempenho entre as versões. Outros objetivos são aumentar a área de código paralelizada em GPU, com o objetivo de reduzir cópia de memória entre CPU e GPU, e verificar a viabilidade de outras soluções.

Referências

Chou, S. C. and Peters, C. A. (2018). Comparison between eta vs sigma coordinate model runs over south america.

Flores, H. G. (2018). Integração da tecnologia cuda ao modelo de previsão do tempo eta. Dissertação de mestrado, PPGCA - Universidade de Passo Fundo.

GOMES, J. L. (2018). Re: Eta sem common blocks [mensagem pessoal]. Mensagem recebida por `a.lexii@hotmail.com`.

Larkin, J. (2018). Openacc online course 2018. `http://info.nvidia.com/rs/156-OFN-742/images/OpenACC%20Course%202018%20Week%201.pdf`. Acessado em 21 Out. 2018.

Mesinger, F., Janjić, Z. I., Ničković, S., and Gavrilov, D. (1988). The step-mountain coordinate: Model description and performance for cases of alpine lee cyclogenesis and for a case of an appalachian redevelopment. *Monthly Weather Review*, 116(7):1493–1518.

Michalakes, J. and Vachharajani, M. (2008). Gpu acceleration of numerical weather prediction. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–7.

Nvidia (2009). *Whitepaper: NVIDIA's Next Generation CUDA Compute Architecture: Fermi*. Acessado em 21 Out. 2018.

Nvidia Corporation (2015). Openacc. `https://www.pgroup.com/lit/brochures/openacc_sc15.pdf`. Acessado em 22 Out. 2018.

PGI Compilers and Tools (2018). Cuda fortran programming guide and reference. `https://www.pgroup.com/resources/docs/18.10/pdf/pgi18cudaforug.pdf`. Acessado em 15 Fev. 2019.

Stanford University (1995). Fortran 77 tutorial. `https://web.stanford.edu/class/me200c/tutorial_77/`. Acessado em 14 Jan. 2019.

Willis, J. S. (2013). *Investigating the use of GPUs with a Monte Carlo Astrophysical Simulation*. Tese de Doutorado, University of Edinburgh.