Investigando Técnicas de Otimização e Paralelização Para Simulação de Camada de Mistura de Fluidos Binários

Sherlon Almeida da Silva[1], **Matheus Castro Nicolau da Silva** [2], **Claudio Schepke**[1]

[1]Laboratório de Estudos Avançados – Universidade Federal do Pampa (UNIPAMPA)
97546-550, Alegrete – RS – Brasil

sherlonalmeida@alunos.unipampa.edu.br, claudioschepke@unipampa.edu.br

[2]Grupo de Fenômenos de Transporte Avançado – Universidade Federal do Pampa (UNIPAMPA)
97546-550, Alegrete – RS – Brasil

mathe1s.castro@gmail.com

Resumo. *Simulações numéricas possibilitam aos pesquisadores reproduzir computacionalmente o comportamento de fenômenos naturais. Elas permitem a realização de experimentos muitas vezes inviáveis, porém demandam um alto custo computacional com a realização dos cálculos. Com o objetivo de agilizar a execução de uma simulação de uma camada de mistura de fluidos binários, este trabalho investiga alternativas de otimização e paralelização.*

1. Introdução

Áreas de estudo da matemática e da física deduzem equações diferenciais para representar o comportamento de fenômenos naturais. A partir disso, tais fenômenos podem ser reproduzidos em um computador. As simulações reproduzidas no computador são representações precisas dos fenômenos, e utilizam dados numéricos de entrada para a partir deles poder acompanhar a evolução temporal do problema tratado.

Nos dias atuais a ciência necessita cada vez mais realizar cálculos complexos e simulações que são computadas por máquinas rápidas. Porém, um dos fatores limitantes hoje é a velocidade em que um resultado pode ser retornado. Como exemplo, existe a previsão do tempo, a qual simula o comportamento do clima para descobrir fenômenos como a chuva, furacões, entre outros. Mas esta previsão deve retornar seu resultado antes do fenômeno acontecer, caso contrário seria inútil. Por isso é necessário que os sistemas retornem os resultados em tempo hábil, contornando os problemas de desempenho.

É possível acelerar a execução de uma simulação através de técnicas de programação e utilização de arquiteturas *multi-core* e *many-core*, as quais fornecem grande poder de processamento devido às dezenas, centenas, até mesmo milhares de núcleos de processamento que podem realizar cálculos simultaneamente em paralelo. Uma das alternativas são as unidades de processamento gráfico (GPUs), que estão cada vez mais populares devido ao seu alto desempenho e menor consumo de energia comparado às unidades centrais de processamento (CPUs), suportando alto paralelismo a custos relativamente baixos [Breder et al. 2016]. Ao definir uma arquitetura para executar o programa, também é possível melhorar o desempenho a partir da otimização para memória *cache*, que é uma memória rápida que armazena os dados mais utilizados pela CPU, reduzindo a alta latência de busca por dados e instruções na memória principal.

O objetivo deste trabalho foi investigar o acoplamento das subrotinas mais custosas de um software de simulação de camada de mistura de fluidos binários, e a maneira que o software foi estruturado. A partir disso, foram identificados os principais trechos que podem ser explorados por técnicas de otimização e paralelização, e por fim foi definida a arquitetura alvo e quais técnicas utilizar para o ganho de desempenho.

Este trabalho está organizado da seguinte forma: A Seção 2 apresenta a metodologia deste trabalho. Na Seção 3 são apresentados e discutidos os resultados parciais. Por fim, é exposta na Seção 4 a conclusão deste artigo.

2. Metodologia

Para explorar o potencial paralelo de um programa é necessário identificar as subrotinas que são mais custosas. Neste sentido existem as ferramentas de análise de execução como o GPROF, que coletam dados em tempo de execução, como o tempo, as funções mais requisitadas, entre outros. A partir disso, é possível identificar quais trechos de código são possíveis candidatos a serem otimizados e paralelizados.

O software utilizado é uma continuação do trabalho de [Quirino and Mendonça 2007] e de [Manco 2014] e calcula as equações de Euler utilizando uma matriz como estrutura de dados para representação em 2 dimensões (2D) da camada de mistura de fluidos, representando o domínio físico através de uma malha computacional. Fluidos inseridos com diferentes velocidades criam um ponto de inflexão no perfil vertical de velocidade, o que somado com a perturbação feita por um pulso acústico, propicia a criação de vórtices de Kelvin-Helmholtz no escoamento. Os vórtices aumentam a área de contato entre os fluidos, acelerando o processo de difusão entre os fluidos. Sendo estes um oxidante e um combustível, pode-se realizar experimentos numéricos de combustão. Uma aplicação pode ser verificada no sistema de propulsão Scramjet, da NASA, em que um veículo não tripulado atingiu a velocidade de $12.144 km/h$ por 300 segundos. Cada imagem da simulação é uma representação adimensional da vorticidade no instante de tempo em que se encontra. Por exemplo, na Figura 1 o eixo y representa a vorticidade adimensional, e o eixo x representa o domínio físico da câmara de mistura. O método numérico utilizado no software é denominado

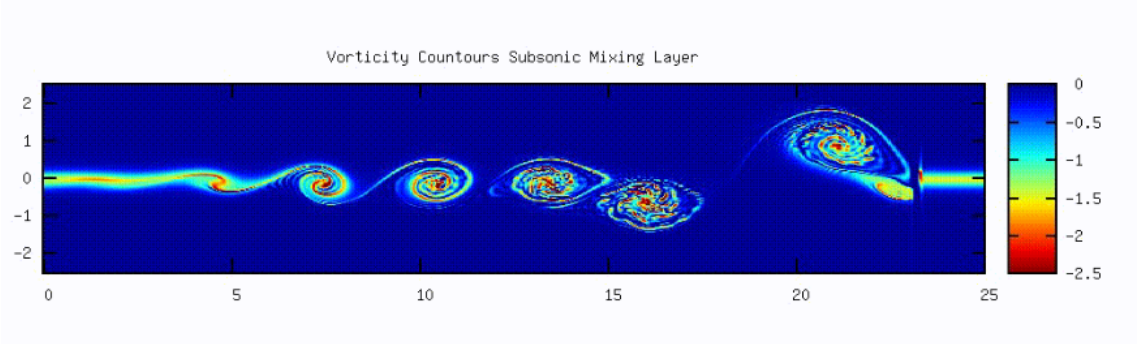

Figura 1. Simulação de Mistura de Fluidos - Vórtices.

Euler Explícito, o que significa que os valores futuros são calculados diretamente a partir dos valores atuais. São gerados arquivos de dados contendo os valores das propriedades físicas medidas na simulação. Com estes dados são geradas imagens para interpretar

estes resultados. A otimização do desempenho para esta aplicação é necessária uma vez que os dados levam horas para serem obtidos.

Outro fator a ser salientado é que o software ainda está em desenvolvimento. Novas equações serão adicionadas em forma de subrotinas, como as equações para reações químicas e combustão. A demora na execução atrasa os testes das novas subrotinas, além do programa ficar mais lento conforme novas funcionalidades forem inseridas, e para obter dados ainda mais precisos, o custo computacional se torna ainda maior.

3. Resultados Parciais e Discussão

A análise detalhada do programa juntamente com a análise de execução do GPROF forneceram informações relevantes para futuras otimizações e paralelizações. Como é possível ver na Tabela 1, foram identificadas as 12 subrotinas mais custosas de um total de 78 subrotinas. Juntas, estas 12 subrotinas somam aproximandamente 86% da execução do programa. Com a coleta das subrotinas mais custosas, tornou-se necessário

Tabela 1. Custo de execução por Subrotina coletados com GPROF.

Subrotina	Código	Perc(%)
eulernonlinear(...)	eqeuler.f90	11.13
lddfiltery(...)	filtering.f90	11.02
strechingx(...)	eqeuler.f90	10.57
lddfilterx(...)	filtering.f90	8.55
dudx(...)	diff.f90	8.36
dudytau(...)	diff.f90	7.68
dudy(...)	diff.f90	7.33
dudxtau(...)	diff.f90	7.27
rhs_euler(...)	rhs.f90	6.18
rk_steep(...)	rk.f90	2.96
dy4(...)	diff.f90	2.46
dx4(...)	diff.f90	2.35

compreender o fluxo de execução do programa de forma detalhada, como por exemplo a comunicação entre as subrotinas, a passagem das variáveis e parâmetros, e o que cada uma computa. Todo o fluxo de execução das subrotinas mais custosas estão na Figura 2. A Figura 2 contém um diagrama que apresenta o início da execução no programa euler.f90, o qual chama as demais subrotinas do programa. Cada nó do diagrama possui o nome do código e a subrotina chamada, bem como a porcentagem de execução. Após esta análise foi possível observar o comportamento do programa e os possíveis trechos de código a serem otimizados e paralelizados. O programa conta com um grande número de laços de repetição que operam sobre matrizes com cálculos independentes, viabilizando a paralelização. Estes laços de repetição também podem ser otimizados para varredura contígua na memória, para aproveitar a localidade espacial e temporal da memória *cache* a partir de técnicas como *Loop Interchange*, que reordena os laços aninhados.

4. Conclusão e Trabalhos Futuros

Simulações computacionais demandam um alto poder de processamento. Tal recurso é encontrado em arquiteturas *multi-core* e *many-core*, uma vez que estas máquinas

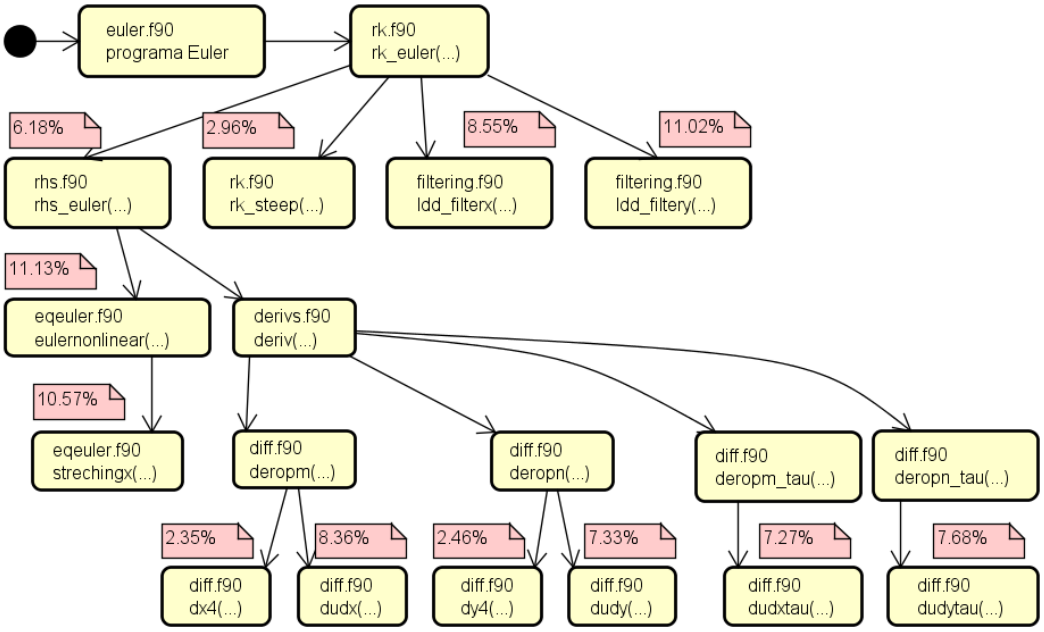

Figura 2. Fluxo de execução do programa.

dispõem de diversos núcleos de processamento. Uma das principais etapas para o aumento do desempenho de um programa é identificar o foco de maior custo computacional e aplicar técnicas de otimização e paralelização. Esta identificação de trechos de código custosos pode ser feita com ferramentas de análise de execução, como o GPROF.

Devido à análise detalhada do código, como trabalhos futuros serão implementadas versões paralelas com a utilização de programação híbrida com as interfaces de programação OpenMP (*Open Multi-Processing*) para CPUs (*Central Processing Units*) e CUDA (*Compute Unified Device Architecture*) para GPUs (*Graphics Processing Units*). Também serão implementadas otimizações para memória *cache* a partir da reordenação da varredura de laços de repetição, como a técnica *Loop Interchange*.

Agradecimentos

Este trabalho foi realizado com recursos do PROBIC/FAPERGS 2017 e pela agência de fomento CNPq via Edital Universal Processo N. 457684/2014-3.

Referências

Breder, B., Charles, E., Cruz, R., Clua, E., Bentes, C., and Drummond, L. (2016). Maximizando o uso dos recursos de gpu através da reordenação da submissão de kernels concorrentes. In *Anais do WSCAD 2016 (XVII Simpósio em Sistemas Computacionais de Alto Desempenho)*, pages 98–109, Aracajú. SBC.

Manco, J. A. A. (2014). Condições de contorno não reflexivas para simulação numérica de alta ordem de instabilidade de kelvin-helmholtz em escoamento compressível. Master's thesis, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos.

Quirino, S. F. and Mendonça, M. T. (2007). Direct numerical simulation of compressible shear layer with heat source. In *19th International Congress of Mechanical Engineering*, Brasília.