13º SIEPE SALÃO INTERNACIONAL DE ENSINO, PESQUISA E EXTENSÃO
CIÊNCIA, TECNOLOGIA E INOVAÇÃO PARA UM PLANETA + HUMANO
17 a 19 nov.21

ACELERANDO UMA APLICAÇÃO DE SIMULAÇÃO DE CÂMARA DE COMBUSTÃO UTILIZANDO PROGRAMAÇÃO PARALELA

Glener Lanes Pizzolato, discente de graduação, Universidade Federal do Pampa, Campus Alegrete
Claudio Schepke, docente, Universidade Federal do Pampa

glenerpizzolato.aluno@unipampa.edu.br, claudioschepke@unipampa.edu.br.

A simulação computacional é um grande benefício da programação, pois permite representar ambientes reais de forma discreta, disponibilizando benefícios financeiros e sem risco de segurança. No lugar de utilizar materiais, ambientes e estruturas reais específicas, utiliza-se um ambiente computacional, para o qual existe um custo de implementação de uma solução verdadeira e o custo de tempo de execução que a simulação leva. O algoritmo estudado neste trabalho simula uma câmara de combustão, o mesmo foi implementado em *Fortran90* de forma sequencial e exige aproximadamente 3 horas para completar a execução de um exemplo real. Visto isto, o trabalho tem como objetivo acelerar este algoritmo em granularidade fina através da API de programação paralela *OpenMP*. O algoritmo possibilita uma representação mais ampla de fluidos com diferentes características físicas. A camada de mistura compreensível serve como um modelo para a análise e a capacidade de simular problemas como por exemplo: propulsão de ar de alta velocidade; mistura de reagentes; geração de ruído em bocais de exaustão, etc. A aplicação utiliza o método de *Runge-Kutta* de sexta ordem para a solução numérica das derivadas pois garante que erros de precisão não impactem nos valores obtidos, e para realizar a mistura de oxidante e combustível utiliza o modelo de instabilidade de *Kelvin-Helmholtz*. Primeiramente a ferramenta *gprof* foi executada no algoritmo sequencial e dessa forma obteve-se o custo computacional que cada rotina do programa separadamente tinha na execução total, dessa forma a implementação e otimização começou nas rotinas mais custosas para as menos custosas (correspondente a valores abaixo de 5% da execução total). Através dos testes realizados foi possível medir o tempo de inicialização (correspondente a leitura do arquivo de entrada, alocação de memória e preenchimento de estruturas de dados) e custo por iteração separadamente e para avaliar e comparar o algoritmo sequencial com o paralelo foi utilizado o conceito de *speed up* e eficiência. Desta forma é possível saber quantas vezes o algoritmo paralelo foi mais rápido e a eficiência que cada thread teve no experimento. O melhor caso para a implementação paralela foi ao utilizar 16 *threads*, onde se obteve o *speed up* de 3,85 para a inicialização e 5,45 para o custo por iteração. Levando em consideração que cada iteração tem o mesmo custo computacional (número de instruções/operações) é possível medir o tempo aproximado de uma execução real, dessa forma a redução de tempo foi bem expressiva, reduzindo as 3 horas anteriores para aproximadamente 34 minutos. O objetivo deste trabalho foi concluído quando o

resultado da versão paralela superou a versão original sequencial escrita em *Fortran90*. Para trabalhos futuros busca-se implementar uma versão em *Many-Core* em *GPU*, utilizando a API *OpenACC*. São esperados alguns desafios, visto que o código é muito extenso e esta implementação exige uma grande movimentação de dados entre *host* e *GPU* e as saídas do programa. Tanto da versão sequencial quanto paralela devem ser 100% fiel uma à outra em termos de saída de dados processados. Também busca-se aumentar o tamanho da entrada do programa. Dessa forma será possível avaliar as duas APIs com diferentes tipos e tamanhos de entrada. Uma arquitetura heterogênea também será estudada, visto que a mesma mistura programação paralela *Multi-core* e *Many-core (GPU)*, podendo dessa forma aproveitar o máximo das duas APIs para se obter um resultado ainda melhor desta última versão implementada.

Agradecimentos: Agradeço primeiramente a UNIPAMPA, campus Alegrete por disponibilizar seu espaço para que este trabalho possa ter sido realizado, também às instituições CNPq e FAPERGS que me apoiaram durante o desenvolvimento do projeto na graduação.
Palavras-chave: Simulação; Câmara de Combustão; OpenMP; Desempenho.