Título: Projeto 1 — DGEMM Sequencial e Paralela com OpenMP
Autores: João Manoel Fidelis Santos e Maria Eduarda Guedes Alves
Disciplina: DEC107 — Processamento Paralelo
Data: 29/09/2025

1. **Introdução**

    ● **Objetivo do projeto**

O primeiro projeto da disciplina tem como objetivo principal aplicar os conceitos estudados sobre paralelismo em arquiteturas de memória compartilhada utilizando diretivas de compilação do OpenMP. Os objetivos específicos são implementar e comparar versões da multiplicação de matrizes em formato sequencial e paralelo usando OpenMP.

    ● **Conceitos teóricos relevantes (multiplicação de matrizes, paralelismo, OpenMP)**

Para realizar o trabalho, foi necessário caminhar por alguns conceitos teóricos relevantes. Dentre vários problemas computacionais, a multiplicação de matrizes é considerada um problema clássico evidente em vários cenários, desde sistemas de segurança e criptografia até fotos digitais e redes sociais (RIBEIRO, 2023)... (Falar sobre a complexidade)... (Falar sobre o paralelismo como uma forma de contornar esse problema)... (Falar do OpenMP como ferramenta para isso)...

    ● **Importância da análise de desempenho**

O objetivo principal dos algoritmos é a resolução de problemas do mundo real, porém, de nada adianta se ele não for eficiente, ou em outras palavras, desperdiça recursos computacionais. Quanto maior o número de operações realizadas, maior é o tempo de execução do algoritmo, gerando sua inviabilidade. A análise de desempenho dos algoritmos é essencial para… Os FLOPs são uma unidade de medida para operações de ponto flutuante por segundo e uma estratégia útil para determinar o desempenho computacional de um sistema…

2. **Metodologia**

    ● **Descrição da implementação sequencial**

A versão sequencial da multiplicação de matrizes é composta pela função principal (main), por uma função de alocação dinâmica de matrizes (matriz_alloc) e a função dgemm sequencial (dgemm_seq). A escolha pela alocação dinâmica de memória foi necessária para possibilitar o uso de matrizes de dimensões arbitrárias, definidas em tempo de execução, além de evitar o risco de estouro da memória da stack. Outro ponto importante é que as matrizes foram implementadas como vetores unidimensionais (lineares), armazenados em ordem por linha. Essa abordagem melhora a localidade de memória e o aproveitamento do cache, uma vez que todos os elementos estão contíguos na memória.

Além disso, evita os problemas de fragmentação e indireção que surgem ao utilizar arrays bidimensionais alocados dinamicamente (double**), nos quais cada linha pode estar em regiões distintas da memória.

A função dgemm recebe as matrizes representadas como vetores alocados dinamicamente e realiza a multiplicação utilizando três loops aninhados, seguindo a versão clássica do algoritmo de multiplicação de matrizes. Foram investigadas alternativas para reduzir a complexidade assintótica de O(n^3), porém tal redução só é possível com algoritmos mais sofisticados que fogem ao escopo deste trabalho. Assim, optou-se por manter a implementação tradicional. Como as matrizes foram implementadas como vetores lineares em ordem por linha, isso otimiza o acesso à memória. Na função principal, os vetores de precisão dupla são criados, inicializados com valores numéricos, utilizados como entrada para a função dgemm sequencial, onde é realizada uma comparação de tempo.

- **Descrição da implementação paralela (quais diretivas OpenMP foram usadas e por quê)**

- **Descrição do hardware utilizado nos testes**

Os experimentos foram realizados em um notebook Dell Inspiron 15 3520 com as seguintes configurações:

- Notebook Dell Inspiron 15 3520
- Sistema Operacional Ubuntu 24.04.2 LTS (GNU/Linux 6.6.87.2-microsoft-standard-WSL2 x86_64) rodando via WSL versão 2.5.9.0
- Processador Intel Core i5 1135G7 (11ª geração) com 4 núcleos físicos e 8 threads, e velocidade de 2,433 GHz
- Memória RAM tipo DDR4 com capacidade máxima e total de 18 GB e velocidade de 3,2 GHz
- Cache L2 de 5 MB e cache L3 de 8 MB
- Compilador GCC 13.3.0 (Ubuntu 13.3.0-6ubuntu2~24.04)
- ? Possíveis flags a serem utilizadas: -O3 -fopenmp -march=native (deixa o código mais rápido; ativa o paralelismo; aproveita instruções especiais da CPU)

Essas informações foram obtidas a partir das configurações do sistema e do site oficial da Dell.

- **Procedimentos para medição de tempo**

A medição do tempo foi realizada utilizando a função omp_get_wtime() da OpenMP, que retorna o tempo em segundos. Cada experimento foi repetido com os valore indicados nas especificações dp trabalho. Antes de cada execução, a matriz de saída foi reinicializada

para garantir condições idênticas de teste. Para avaliar a diferença entre um simples algoritmo O(n^3) e outro que faz a implementação altamente otimizada, utilizamos uma das rotinas da BLAS.

- **Métricas utilizadas para avaliação**

A métrica escolhida para avaliar o desempenho da versão sequencial foi o tempo de execução em segundos, pois representa diretamente o custo computacional do algoritmo e permite comparações objetivas entre diferentes abordagens de implementação. Além disso, das métricas citadas ela é a única viável para a versão sequencial.

3. Resultados

- **Tabelas e/ou gráficos com os tempos obtidos para cada tamanho de matriz e configuração de threads**

- **Cálculos de speedup e eficiência (se aplicável)**

- **Observações sobre balanceamento de carga e escalabilidade**

4. Discussão

- **Comparação entre as versões sequencial e paralela**

- **Impacto do aumento do número de threads**

- **Discussão de gargalos e limitações encontradas**

- **Sugestões de melhorias**

5. Conclusão

● Principais aprendizados do projeto

Em relação à versão sequencial, ficou evidente que, embora o algoritmo clássico de multiplicação de matrizes seja conceitualmente simples, sua complexidade de O(n^3) o torna inviável para dimensões muito grandes. Isso demonstra que, em problemas de alto custo computacional, não basta apenas implementar corretamente o algoritmo, mas é necessário recorrer a estratégias de otimização e, em alguns casos, a algoritmos avançados que permitam reduzir o tempo de execução e tornar a solução prática.

● Considerações sobre desempenho obtido

6. Referências

● Material de aula utilizado

ORELLANA, Esbel Tomas Valero. CET107-Processamento Paralelo Aula 02 - Introdução. Material da disciplina Processamento Paralelo - Universidade Estadual de Santa Cruz, Ilhéus, 2025. Disponível em: Google Sala de Aula.

● Documentação oficial do OpenMP

● Outras fontes relevantes