UFPR – Departamento de Informática

CI316 – Programação Paralela - Trabalho Prático OpenMP

Prof. Marco Antonio Zanata Alves

Com base na implementação sequencial em linguagem C do algoritmo **K-means**, listados no final deste arquivo. Leia atentamente e elabore um relatório de no máximo 8 páginas com os seguintes itens:

1. Recorte o kernel (parte principal) de cada algoritmo e explique em suas palavras o funcionamento sequencial do trecho.
2. Explique qual a estratégia final (vitoriosa) de paralelização você utilizou.
3. Descreve a metodologia que você adotou para os experimentos a seguir. Não esqueça de descrever também a versão do SO, kernel, compilador, flags de compilação, modelo de processador, número de execuções, etc.
4. Com base na execução sequencial, meça e apresente a porcentagem de tempo que o algoritmo demora em trechos que você não paralelizou (região puramente sequencial).
5. Aplicando a Lei de Amdahl, crie uma tabela com o speedup máximo teórico para 2, 4, 8 e infinitos processadores. Não esqueça de explicar a metodologia para obter o tempo paralelizável e puramente sequencial.
6. Apresente tabelas de speedup e eficiência. Para isso varie o número de threads entre 1, 2, 4 e 8. Varie também o tamanho das entradas, tentando manter uma proporção. Exemplo de tabela:

		1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
Eficiência	N=10.000	1	0,81	0,53	0,28	0,16
	N=20.000	1	0,94	0,80	0,59	0,42
	N=40.000	1	0,96	0,89	0,74	0,58

7. Analise os resultados e discuta cada uma das três tabelas. Você pode comparar os resultados com speedup linear ou a estimativa da Lei de Amdahl para enriquecer a discussão.
8. Seu algoritmo apresentou escalabilidade forte, fraca ou não foi escalável? Apresente argumentos coerentes e sólidos para suportar sua afirmação.

Os trabalhos deverão ser apresentados ao vivo no Zoom pelo aluno com vídeo e áudio. A nota irá considerar domínio do tema, robustez da solução e rigorosidade da metodologia.

Cuidados gerais para efetuar os experimentos

- Para assegurar a corretude da implementação paralela, deve-se verificar se os resultados paralelos batem com os sequenciais executando diferentes entradas.
- Execute pelo menos 20x cada versão para obter uma média minimamente significativa. Ou seja, todo teste, onde mudamos o número de processos ou tamanho de entrada, devemos executar 20x. Mostrar no relatório a **média com desvio padrão**.
  - As métricas deverão ser calculadas encima da média das execuções.
- Sugiro escolher um modelo de máquina e sempre utilizar o mesmo modelo até o final do trabalho.
  - Cuidar para não executar em servidores virtualizados ou que contenham outros usuários (processos ativos) utilizando a mesma máquina. Diversos servidores do DINF são máquinas virtualizadas e os testes de speedup não serão satisfatórios/realísticos.
  - Cuide para que não haja outros processos ou usuários usando a máquina no mesmo momento que você esteja executando seus testes.
  - Sempre execute com as flags máximas de otimização do compilador, exemplo -O3 para o gcc, afinal queremos o máximo desempenho.
- **Teste de escalabilidade forte**: Manter um tamanho de entrada N qualquer, e aumentar gradativamente o número de processos. Sugere-se que escolha-se um N tal que o tempo de execução seja maior ou igual a 10 segundos.
- **Teste de escalabilidade fraca:** Aumentar o tamanho da entrada proporcionalmente com o número de processos. Exemplo: 1xN, 2xN, 4xN, 8xN, 16xN. Atenção, escalar N com o número de threads/processos (não de máquinas no caso do MPI).

Regras Gerais

A paralelização dos códigos deve ser feita em C ou C++ utilizando as primitivas OpenMP. A entrega será feita pelo Moodle dividida em duas partes

- **Relatório em PDF (máximo 8 páginas, fonte verdana 12pts.)**
- **Código fonte paralelo (OpenMP)**

Casos não tratados no enunciado deverão ser discutidos com o professor.

Os trabalhos devem ser feitos individualmente.

Atencão, a cópia do trabalho (plágio), acarretará em nota igual a Zero para todos os envolvidos.

K-means (Complexidade Média)

O agrupamento K-Means é popular na análise de cluster em mineração de dados e consiste em um método que permite a modelagem de funções de densidade de probabilidade pela distribuição de vetores de características.

O método particiona um conjunto de n pontos em k clusters, onde cada ponto será associado ao cluster com a média mais próxima. A distância euclidiana é geralmente a métrica adotada para medir a proximidade.

O agrupamento é um problema NP-difícil, mas existem algoritmos heurísticos eficientes que convergem rapidamente para um ótimo local.

Crie uma versão paralela para o algoritmo kmeans .

O algoritmo:

Nesta implementação, o algoritmo tem como entrada as coordenadas (x, y, z) de k centróides iniciais e um conjunto de pontos (x, y, z).

O algoritmo executa um processo iterativo, onde os pontos são agrupados de acordo com a distância euclidiana mínima entre eles e os centróides. Em seguida, o centróide de cada partição é recalculado com base na média de todos pontos na cluster, e todo o procedimento é repetido até que nenhum centróide seja alterado e nenhum ponto mude de cluster. Após a conclusão, o algoritmo retorna as coordenadas dos k centróides finais

Entrada:

A entrada deve ser lida a partir da entrada padrão.

As primeiras duas linhas contêm o número de centróides k e o número de pontos n. As próximas k + n linhas contém as coordenadas (x, y, z) para os k centróides e n pontos, respectivamente.

Saída:

A saída deve ser gravada na saída padrão.

O programa deve imprimir k linhas com as coordenadas (x, y, z) dos k centróides re-agrupados.

Arquivos:

kmeans.c - Código sequencial em C do algoritmo kmeans