INSTITUTO FEDERAL
Paraná
Campus Pinhais

Bacharelado em Ciência da Computação
Projeto de Sistemas Distribuídos — 27 de agosto de 2026
Trabalho 1 — v2.0

1 Introdução

Seu trabalho consiste em implementar um algoritmo para resolver um problema de maneira sequencial, transformá-lo em um programa paralelo e fazer o cálculo de *speedup*.

1.1 Problemas disponíveis

- Método de Gauss-Jacobi para resolução de sistemas lineares, usado em vários sistemas de simulação;
- Métodos para resolução de sistemas lineares esparsos;
- Algoritmo de Dijkstra para distância em grafos;
- Transformada Rápida de Fourier (FFT), usada em imagens, áudio, telecomunicações, ...;
- Equação do calor 2D (*2D Steady State Heat Equation*);
- Remoção de ruído em imagens;
- Simulação dinâmica de partículas/moléculas (velocidade, aceleração, ...);
- Detectar todas as entradas de um circuito que geram uma saída verdadeira (Problema da Satisfazibilidade Booleana);
- Imagem PPM do *Mandelbrot set*;

2 O trabalho

Você deve encontrar um problema do mundo real que seja resolvido com algum dos problemas acima, modelar e implementar a versão sequencial do programa em C/C++ para resolver o problema.

Escolha um problema interessante e contextualize o problema no relatório. Faça implementações corretas e justifique suas decisões de projeto.

Depois, implemente a versão paralela, também em C/C++, meça o tempo e calcule as métricas de *speedup* e eficiência.

2.1 Como medir o tempo

Você deve medir o tempo apenas das regiões que considera paralelizáveis (usando funções do próprio OpenMP e o comando *time*, do Linux) e a partir desses dados, deve calcular o *speedup* máximo teórico para o seu problema.

Crie várias entradas de teste, de diferentes tamanhos. Por exemplo, para o Djikstra, crie vários grafos que você usará para testar seu programa: grafos pequenos, médios, grandes, muito grandes, ... Crie entradas grandes o suficiente para que o programa sequencial demore vários minutos para executar.

Para cada entrada de teste, execute seu programa ao menos 5 vezes para calcular a média dos tempos.

Ao testar a versão paralela, execute os testes com várias quantidades de *threads*: 1, 2, 4, 8, ... Faça todos os testes no mesmo computador, sem executar outros programas ao mesmo tempo, para não interferir nas suas medições. Se usar um *laptop*, use o carregador.

3 Entrega

Você deverá enviar para o email do professor, até 30/setembro, 23h59, um arquivo *.zip* contendo uma pasta, que por sua vez contém:

- Os arquivos `.c` ou `.cpp` da versão sequencial E da versão paralela;
- Um arquivo de texto chamado `README.md` contendo o nome completo dos integrantes do grupo e informações sobre como compilar os programas;
- Um relatório chamado `relatorio.pdf` contendo o nome completo dos integrantes do grupo e o conteúdo descrito na seção 3.1.

O arquivo compactado descrito acima deve ser nomeado com as iniciais do nome de cada integrante separados por um hífen. Por exemplo: se o Fulano da Silva Sousa fez o trabalho com o João de Souza, o arquivo deve ser nomeado `fss-js.zip`. A pasta, dentro do arquivo compactado, deve ter o mesmo nome, a não ser pela extensão.

O trabalho pode ser feito em grupos de até 3 pessoas.

3.1 Relatório

A equipe deve entregar um relatório no formato `.pdf`, explicando como resolveu os desafios do trabalho. O relatório não deve conter o código completo! Só adicione código no relatório se precisar explicar/justificar algo muito específico da sua solução. Ainda assim, adicione apenas o trecho relevante.

O relatório deve explicar o problema escolhido, ideia de funcionamento da solução sequencial, discussão das regiões que o grupo julgou que são paralelizáveis, discussão das métricas estimadas e dos resultados reais. A discussão deve mostrar a escalabilidade do problema, relacionar os resultados obtidos com a estimativa esperada, e discutir se a solução proposta é interessante.

O relatório deve conter todas as informações necessárias para reproduzir a sua solução. Inclua no seu relatório as informações do hardware utilizado, e de cada medição de tempo realizada, bem como gráficos para auxiliar a análise de desempenho.

Inclua as referências de pesquisa e código que utilizar para seu trabalho no final de seu relatório.

3.2 Critérios de correção

Seu trabalho será corrigido considerando:

- Seus programas devem funcionar corretamente!
- Objetividade e clareza do relatório.
- Uso dos métodos e construções adequadas de programação e de OpenMP.
- Domínio do trabalho na defesa/acompanhamento.
- Adequação à esta especificação.

Não serão aceitos trabalhos entregues fora do prazo. Trabalhos sem relatório ou que não sejam defendidos/acompanhados em sala não serão considerados!

Plágio não será tolerado. Se isto acontecer, os trabalhos envolvidos serão desconsiderados e o caso será encaminhado para as Coordenações do campus.

O uso de inteligência artificial pode acarretar na redução de nota parcial ou totalmente, se o professor identificar que não há domínio do trabalho proposto por parte do estudante.

Histórico das Revisões:

- 27/ago/2026 - v2.0: adição de problemas e reescrita da seção de medição de tempo e de entrega.
- 19/set/2025 - v1.0: primeira versão.