Hilzer's Barbershop Problem

Comparação de Ferramentas de IA Generativa para Resolução de Problemas de Concorrência

Grupo 9
Universidade Federal do Rio de Janeiro
EEL770 - Sistemas Operacionais

6 de julho de 2025

Conteúdo

1 Introdução

Este projeto é baseado no **Little Book of Semaphores** de Allen B. Downey, especificamente focando no **Hilzer's Barbershop Problem** (Seção 5.4). O objetivo principal é adaptar a solução baseada em semáforos do livro para uma versão utilizando **locks** e **condition variables** em C com **pthreads**.

Adicionalmente, este projeto visa investigar e comparar como diferentes ferramentas de IA generativa, especificamente **GPT-4o** e **Claude Sonnet 4**, abordam o mesmo problema de concorrência.

1.1 Objetivos

- **Objetivo Técnico**: Implementar uma solução para o problema Hilzer's Barbershop usando **locks** e **condition variables** em C com pthreads.
- **Comparação de Ferramentas de IA**: Investigar e comparar diferentes ferramentas de IA generativa (GPT-4o e Claude Sonnet 4) para resolver o mesmo problema de concorrência.
- **Documentação e Análise**: Documentar as interações e fornecer uma análise crítica dos resultados obtidos.

2 Metodologia

2.1 Abordagem Inicial

Para começar a solução do problema Hilzer's Barbershop, tentamos a abordagem mais simples e rápida: criar um prompt com o problema inicial e fornecê-lo a quatro LLMs (GPT-4o, Claude Sonnet 4, Gemini 2.5 Pro e DeepSeek-V3)

Logo de cara, as LLMs criaram códigos que apresentavam erros, resultando em deadlocks ou em um funcionamento incorreto. Muitas operações não faziam sentido, como o barbeiro voltar a dormir mesmo com clientes esperando para serem atendidos. Inicialmente, acreditamos que o DeepSeek estava fornecendo uma solução correta, mas logo ela também se mostrou incorreta.

Tentamos modificar nossos prompts e oferecer dicas às LLMs, juntos discutimos em busca da melhor abordagem. No entanto, o primeiro dia de reunião foi frustrante, pois não conseguimos obter resultados satisfatórios. Chegamos à conclusão, naquele momento, de que deveríamos mudar de estratégia para os próximos dias, já que prompts comuns e fragmentados não estavam surtindo efeito.

As principais dificuldades encontradas foram:

- **Falha na Convergência:** Mesmo após múltiplos ciclos de interação e refinamento, os modelos não convergiram para uma solução correta.
- **Inconsistência e Regressão:** O processo de refinamento se mostrou instável, com melhorias pontuais sendo frequentemente revertidas nas próximas iterações.
- **Generalização Excessiva:** Os LLMs tenderam a tratar o desafio como um problema genérico de multi-threading, ignorando que o problema exigia a criação de

uma fila para garantir, por exemplo, que o cliente que vai cortar o cabelo primeiro é o cliente há mais tempo no sofá.

Concluímos que a geração de código para o problema Hilzer's Barbershop por meio de prompt direto são insuficientes e pouco confiáveis, exigindo uma mudança para uma estratégia mais robusta. Acreditamos que isso se deve à escassez de implementações corretas desse problema disponíveis na internet.

2.2 Abordagem Adotada

Devido às limitações encontradas na abordagem inicial, decidimos:

1. **Desenvolver nossa própria solução**: Criar uma implementação completa do problema na pasta `minha/`
2. **Gerar pseudocódigo**: Utilizar o ChatGPT para gerar um pseudocódigo claro a partir da nossa solução
3. **Fornecer pseudocódigo às IAs**: Passar o pseudocódigo gerado para GPT-4o e Claude Sonnet 4 para obter implementações
4. **Comparar resultados**: Analisar as diferenças entre as soluções geradas

2.3 Ferramentas Utilizadas

- **GPT-4o**: Para geração de pseudocódigo e implementação baseada no pseudocódigo.
- **Claude Sonnet 4**: Para implementação baseada no pseudocódigo
- **GitHub**: Para armazenamento e documentação das interações

3 Implementação

O código fonte dos problemas pode ser encontrado no link do gitHub: `https://github.com/MariaLuizaCw/Operating-Systems/tree/master`. Além disso, o código fonte e o MakeFile foram compacatados e enviados no moodle com o nome "grupo9.zip". Abaixo detalhamos a estrutura do repositório e algumas estruturas fundamentais comuns a todas as implementações.

3.1 Estrutura do Repositório

O projeto foi organizado em três pastas principais:

- `minha/`: Solução desenvolvida pelo grupo
- `gpt/`: Solução gerada pelo GPT-4o
- `claude/`: Solução gerada pelo Claude Sonnet 4

As pastas `claude/` e `gpt/` contém:

- `barbershop.c`: Código fonte
- `Makefile`: Arquivo de compilação
- `link_prompt.txt`: Link para as interações com a IA.

A pasta `minha/` contém:

- `barbershop.c`: Código fonte
- `Makefile`: Arquivo de compilação
- `pseudocodigo.txt`: Pseudocódigo gerado a partir do código fonte.
- `link_prompt_pseudo.txt`: Link para a interação com o GPT pedindo a conversão do código fonte em pseudocódigo.

3.2 Componentes Principais das Implementações

Todas as implementações compartilham estruturas similares:

3.2.1 Estrutura Customer

```
typedef struct {
    int id;
    pthread_cond_t waiting_for_service;
    pthread_cond_t waiting_for_sofa;
} Customer;
```

3.2.2 Estrutura Queue

```
typedef struct {
    int *items;
    int front;
    int rear;
    int count;
    int capacity;
} Queue;
```

3.2.3 Variáveis Globais de Sincronização

- `pthread_mutex_t shop_mutex`: Controle de acesso à barbearia
- `pthread_cond_t sofa_cond`: Condição para disponibilidade do sofá
- `pthread_cond_t barber_cond`: Condição para disponibilidade de barbeiro
- Filas para gerenciamento de clientes

3.3 Compilação e Execução

1. Navegue até a pasta da solução desejada:

   ```
   cd cloude
   # ou
   cd gpt
   # ou
   cd minha
   ```

2. Compile o programa:

   ```
   make
   ```

3. Execute o programa com os parâmetros desejados:

   ```
   ./barbershop MAX_CUSTOMERS NUM_BARBERS MAX_PEOPLE_IN_SHOP SOFA_CAPACITY
   ```

Exemplo

`./barbershop 12 3 10 4`

Significado:

- `12` clientes (threads) serão criados.
- `3` barbeiros disponíveis para atendimento.
- A barbearia comporta no máximo `10` pessoas simultaneamente.
- O sofá comporta até `4` pessoas sentadas.

4 Comparação das Ferramentas de IA

4.1 Documentação das Interações

As interações com ambas as ferramentas estão documentadas através de links armazenados nos arquivos `link_prompt.txt` de cada pasta:

- **GPT-4o**: `https://chatgpt.com/share/686a8f07-c40c-800e-b16a-d3b46145f90b`
- **Claude Sonnet 4**: `https://claude.ai/public/artifacts/bb5458c5-eb2f-4c9e-bbed-f41a25e4fa7e`

4.2 Análise Comparativa

Ambas as IAs foram capazes de implementar corretamente o código com base no pseudocódigo fornecido, e com poucas interações. No entanto, observamos uma diferença relevante no estilo de resposta de cada ferramenta:

Cloud apresentou uma implementação mais fiel ao pseudocódigo original, respeitando a ordem das operações e a lógica definida previamente.

O GPT, por outro lado, embora tenha gerado um código funcional, realizou modificações na ordem das operações. Um exemplo crítico foi a movimentação dos clientes da fila em pé para o sofá: apesar do pseudocódigo indicar claramente que essa movimentação deve ocorrer assim que um cliente do sofá é chamado para o corte, o GPT posicionou essa transição apenas após o pagamento, o que altera o comportamento do sistema e pode gerar inconsistências na simulação. Após um novo prompt, no entanto, a ordem foi corrigida e o código passou a refletir fielmente a lógica esperada.

Essa diferença de comportamento pode indicar que o Cloud prioriza mais a aderência literal ao enunciado, enquanto o GPT tenta "otimizar"ou reinterpretar com base em padrões aprendidos.

5 Conclusões e Reflexões Finais

Este trabalho foi mais difícil do que imaginávamos, achávamos que bastaria passar o enunciado para as IAs e obter soluções funcionais em uma ou poucas interações. Dada a escassez de implementações corretas desse problema na internet, faz sentido que as IAs não conseguissem desenvolver a solução sem um guia claro como o pseudocódigo. Mesmo com o pseudocódigo em mãos, o GPT alterou a ordem das operações, enquanto o Cloud foi mais fiel, o que nos faz confiar mais nele para problemas de código.

6 Referências

- Downey, Allen B. *The Little Book of Semaphores*. Green Tea Press, 2008.
- POSIX Threads Programming. `https://computing.llnl.gov/tutorials/pthreads/`
- OpenAI GPT-4o. `https://openai.com/gpt-4`
- Anthropic Claude Sonnet 4. `https://www.anthropic.com/claude`
- DeepSeek - V3 `https://chat.deepseek.com/`
- Gemini 2.5 Pro `https://gemini.google.com/app?hl=pt-BR`