PRÁTICA 03 – PROGRAMAÇÃO MULTITHREAD (PARTE 01: PARALELISMO)

CONTEÚDO

1. INTRODUÇÃO

1.1. Objetivo

- Fortalecer os aspectos teóricos e práticos relacionados a programação paralela, a partir da implementação de programas envolvendo múltiplas threads.
- Entender as primitivas básicas para manipulação de threads do Padrão Posix Threads (PThreads).
- Estudar com o paralelismo impacta no desempenho dos programas.

1.2. Abordagem Metodológica

- Formação de equipes de trabalho.
- Usar uma linguagem de programação C, compilador padrão C99. Compilar com /usr/bin/gcc, usando opção *–std=c99 -Wall*
- Pesquisar aspectos teóricos e práticos relacionados a programação concorrente com threads disponíveis na linguagem escolhida.
- Realizar a implementação de exemplos simples para entendimento inicial da abordagem de programação, seguida pela implementação de exemplos complexos que permitam consolidar os aspectos técnicos assimilados.
- Promover discussão das diferentes implementações -- considerando decisões de projeto, dificuldades encontradas e soluções implementadas.

1.3. Considerações Gerais

- As atividades devem ser realizadas por equipes com até 4 membros.
- Durante o semestre, serão agendados encontros online nos quais os estudantes realizarão a apresentação das implementações desenvolvidas. Nestas apresentações, os estudantes serão avaliados individualmente.
- As notas serão concedidas de acordo com o desempenho em termos dos aspectos teóricos e práticos de cada estudante nas apresentações.
- Como as apresentações serão feitas a posteriori, é interessante que os relatórios solicitados nas atividades tenham o nível de detalhamento adequado para permitir que os conceitos e decisões de projetos utilizadas sejam recordados para serem discutidos durante as apresentações.

2. PROBLEMAS DE PROGRAMAÇÃO

2.1. Problemas Básicos de Programação

Estatística Multithread [Silberschatz, 2015]

Escreva um programa com múltiplas threads que calcule diversos valores estatísticos para uma lista de números. Esse programa receberá uma série de números na linha de comando e, então, criará três threads de trabalho separados. Um dos threads determinará a média dos números, o segundo thread determinará o valor máximo e o terceiro thread determinará o valor mínimo.

Por exemplo, suponha que seu programa receba os inteiros:

90 81 78 95 79 75 85

O programa apresentará como saída

O valor médio é 82

O valor mínimo é 72

O valor máximo é 96

As variáveis que representam os valores médio, mínimo e máximo serão armazenadas globalmente. Os threads de trabalho estabelecerão esses valores, e quando esses threads de trabalho terminarem, o thread-pai exibirá os valores na tela.

Primos Multithread [Silberschatz, 2015]

Escreva um programa com múltiplos threads que exiba números primos. Esse programa deve funcionar conforme a seguir. O usuário executará o programa e dará entrada em um número na linha de comando. Em seguida, o programa criará um thread separado que exibirá todos os números primos menores ou iguais ao número informado pelo usuário.

2.2. Problemas Avançados de Programação

O objetivo destes problemas é compreender melhor o conceito de thread e como elas podem ser usadas para utilizar melhor os recursos de processamento disponíveis nos computadores modernos.

Cálculo Multithread de π [Mazeiro, 2019]

Neste, será construído um programa em C usando a biblioteca Posix Threads para calcular π usando N threads simultâneos. O valor de π pode ser aproximado pela série de Leibniz-Grégory:

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

Observações:

- Devem ser calculados pelo menos 1 bilhão (10^9) de termos da série; use variáveis reais de dupla precisão (double) nos cálculos;
- O programa deve dividir o espaço de cálculo uniformemente entre as N threads; cada thread efetua uma soma parcial de forma autônoma;
- Para evitar o uso de mecanismos de sincronização, cada thread $T[i]$ deve depositar seu resultado parcial na posição $result[i]$ de um vetor de resultados parciais. Após o término dos threads de cálculo, o programa principal soma os resultados parciais obtidos por elas e apresenta o resultado final na tela;
- Para garantir um bom desempenho, evite operar muito com variáveis globais dentro dos threads; copie as variáveis globais necessárias para variáveis locais (alocadas automaticamente na pilha do thread) e ao final dos cálculos copie os resultados finais de volta para as variáveis globais correspondentes;
- Devem ser medidos os tempos de execução do programa para execuções com 1, 2, 4, 8, 16 e 32 threads, em dois computadores com configuração distinta, como por exemplo: uma máquina monoprocessada; uma máquina dual-processada.
    - Para determinar o tempo de cada execução, use o comando time do UNIX, por exemplo.
- Compilar com os flags *-lm* (biblioteca matemática) e *-lpthread* (threads Posix).
- Ao executar seus experimentos, verifique se a máquina não está muito carregada, o que pode falsear os resultados (use o comando top para ver a carga da máquina);
- Para que os resultados tenham valor estatístico, devem ser feitas pelo menos 5 medidas de cada experimento e calculados o tempo médio e o coeficiente de variação entre elas (coeficiente de variação = desvio-padrão / média, em percentagem); são aceitáveis coeficientes de variação de até 5%; caso contrário, as medições devem ser refeitas.

Nesta atividade devem ser entregues o código-fonte desenvolvido (um arquivo calcpi.c) e um relatório contendo as seguintes informações:

- Tabelas com os tempos de execução obtidos, valores médios e coeficientes de variação;
- Gráficos separados para as curvas de tempo médio de execução em cada plataforma e sua análise. Nos gráficos, o eixo X indica o número de threads e o eixo Y o tempo de execução;
- Um gráfico consolidado (todas as curvas no mesmo gráfico) com as curvas de tempo normalizadas e sua análise crítica. Uma curva normalizada guarda apenas as proporções entre os valores de tempo; nela, os tempos de execução com N threads são divididos pelo tempo de execução obtido com apenas um thread, na mesma plataforma;
- O diagrama de tempo da execução (simplificado, não precisa representar TODOS os threads);
- Respostas às seguintes questões:
    - Os threads implementados são preemptivas ou cooperativas? Explique sua resposta.
    - Que modelo de threads o sistema operacional que você usou implementa (N:1, 1:1 ou N:M)? Como isso pode ser deduzido a partir dos experimentos?

Multiplicação Multithread de matrizes

Neste projeto, será construído um programa em C usando a biblioteca Posix Threads para realizar a multiplicação de duas matrizes: $C_{m\times n} = A_{m\times p} \times B_{p\times n}$, usando até $N = m * n$ threads simultâneos.

Observações:

- O usuário deve definir por linha de comando os valores a ordem das matrizes de entrada. Os valores de cada posição das matrizes de entrada serão definidos aleatoriamente pelo seu programa. Use variáveis reais de dupla precisão (*double*) nos cálculos;
- O programa deve dividir o espaço de cálculo uniformemente entre as N threads; cada thread efetua

parte da multiplicação entre as matrizes de forma autônoma;

- Devem ser medidos os tempos de execução do programa para execuções com 1, 2, 4, 8, 16 e 32 threads, para realizar multiplicações de matrizes 1000×1000, em dois computadores com configuração distinta, como por exemplo: uma máquina monoprocessada; uma máquina dual-processada
    - Para determinar o tempo de cada execução, use o comando time do UNIX, por exemplo.
- Compilar com os flags *-lm* (biblioteca matemática) e *-lpthread* (threads Posix).
- Ao executar seus experimentos, verifique se a máquina não está muito carregada, o que pode falsear os resultados (use o comando *top* para ver a carga da máquina);
- Para que os resultados tenham valor estatístico, devem ser feitas pelo menos 5 medidas de cada experimento e calculados o tempo médio e o coeficiente de variação entre elas (coeficiente de variação = desvio-padrão / média, em percentagem); são aceitáveis coeficientes de variação de até 5%; caso contrário, as medições devem ser refeitas.

Nesta atividade devem ser entregues o código-fonte desenvolvido (um arquivo multmatriz.c) e um relatório contendo as seguintes informações:

- Tabelas com os tempos de execução obtidos, valores médios e coeficientes de variação;
- Gráficos separados para as curvas de tempo médio de execução em cada plataforma e sua análise. Nos gráficos, o eixo X indica o número de threads e o eixo Y o tempo de execução;
- Um gráfico consolidado (todas as curvas no mesmo gráfico) com as curvas de tempo normalizadas e sua análise crítica. Uma curva normalizada guarda apenas as proporções entre os valores de tempo; nela, os tempos de execução com N threads são divididos pelo tempo de execução obtido com apenas um thread, na mesma plataforma;
- O diagrama de tempo da execução (simplificado, não precisa representar TODOS os threads);
- Respostas às seguintes questões:
    - Os threads implementados são preemptivas ou cooperativas? Explique sua resposta.
    - Que modelo de threads o sistema operacional que você usou implementa (N:1, 1:1 ou N:M)? Como isso pode ser deduzido a partir dos experimentos?

3. Entregáveis e Datas

3.1. Entregáveis

- Códigos fontes devidamente comentados.
- Relatório contendo documentação dos códigos-fontes.
- Relatório técnico descrevendo todos os passos do desenvolvimento, incluindo descrições dos conceitos relacionados, testes realizados, procedimentos de compilação e implantação do código etc.

3.2. Datas

Entregável	Data
Upload no Moodle de Arquivos fontes e relatórios	06/10/2020
Encontro online para defesas dos aspectos de implementação	13/10/2020

4. Pontuação

Item	Pontuação
Implementação e documentação dos programas básicos (Estatística e Primos)	20%
Implementação e documentação dos programas avançados (Cálculo de π e Multiplicação de matrizes)	70%
Verificação e manipulação correta de todas as condições de erro	5%
Bom estilo de codificação (e.g., boa formatação do código, nomes de variáveis significativos, comentários adequados etc.)	5%

5. Referências Bibliográficas

Barney, B. **POSIX Threads Programming**. *Lawrence Livermore National Laboratory*. Disponível em: https://computing.llnl.gov/tutorials/pthreads/. Acesso em: 21/09/2020

Silberschatz, A.; Galvin, P. B.; Gagne, G. **Fundamentos de Sistemas Operacionais**. 9a ed., 2015, Willey.

Tanenbaum, A. S. Sistemas Operacionais Modernos. Prentice-Hall, 3ª edição, 2008.

Stallings, W. **Operating System: Internals and Design Principles**. 7th ed., 2012, Prentice Hall.

Mazeiro, C. A. **Sistemas Operacionais: Conceitos e Mecanismos**. 2019, Disponível em: http://wiki.inf.ufpr.br/maziero/doku.php?id=so:start. Acessado em: 17/09/2019