Universidade Federal do ABC
MCZA020-13 - Programação Paralela
2019.Q1

Lista de Exercícios 4

Prof. Emílio Francesquini

11 de março de 2019

Lista de termos cuja definição você **deve** saber:

- Threads vs. processos
- Condição de corrida (*race condition*)
- Seção crítica
- Espera ocupada (*busy-waiting*)
- Mutexes
- Semáforos
- Variáveis de condição
- Produtor consumidor
- Pipeline
- Read-write locks
- Funções *thread-safe* /reentrantes.

Exercícios

1. [PP] Se um programa utiliza mais de um mutex e os mutexes podem ser travados em ordens arbitrárias o programa pode entrar em um impasse (*deadlock*). Ou seja, threads podem ficar bloqueados indefinidamente esperando pelo uso de um dos mutexes. Por exemplo, suponha que um programa utilize duas estruturas de dados distintas que precisem ser protegidas (ex. duas arrays ou duas listas ligadas) e que cada uma

destas estruturas possua um mutex associado a elas. Suponha também que cada estrutura de dados possa ser acessada (lida ou escrita) após o programa obter o acesso ao mutex associado à ela.

(a) Considere que o programa seja executado com dois threads e que a seguinte sequência de eventos ocorra:

Time	Thread 0	Thread 1
0	`pthread mutex lock(&mut0)`	`pthread mutex lock(&mut1)`
1	`pthread mutex lock(&mut1)`	`pthread mutex lock(&mut0)`

O que acontece com o programa?

(b) Isto seria um problema caso programa utilizasse *busy-waiting* com duas variáveis de flag em vez de mutexes?

(c) Isto seria um problema caso o programa utilizasse semáforos em vez de mutexes?

2. Dê um exemplo de uma sequência de acessos à memória pertencente à uma lista ligada na qual os seguintes pares de operações podem resultar em problemas:

(a) Duas remoções executando simultaneamente.

(b) Uma inserção e uma remoção executando simultaneamente.

(c) Uma busca e uma remoção executando simultaneamente.

(d) Duas inserções executando simultaneamente.

(e) Uma inserção e uma busca executando simultaneamente.

3. [PP] As operações de inserção e remoção de uma lista ligada ordenada (conforme vimos em aula) consistem em duas fases. Na primeira fase ambas as operações buscam a posição correta na lista onde devem efetuar as modificações. Caso encontrem, na segunda fase um novo nó da lista ligada é inserido ou removido. Na verdade é bem comum que programas que fazem uso de lista ligada dividam essa tarefa em duas chamadas de função distintas. Em ambos os casos a primeira fase envolve apenas acesso de leitura à lista. Seria seguro, então, utilizar um read-lock para travar a lista da primeira fase e então travar a lista usando um lock de escrita durante a segunda fase? Justifique sua resposta.

4. [PP] Baixe o arquivo fonte `pth_mat_vect_rand_split.c` do livro do Peter Pacheco (link disponível na página da disciplina). Utilizando o `perf stat` ou ainda algum programa que faça o perfilamento dos acessos à memória cache (ex. Valgrind) compile o programa e o execute

com os tamanhos: $k \times (k \cdot 10^6)$, $(k \cdot 10^3) \times (k \cdot 10^3)$ e $(k \cdot 10^6) \times k$. Escolha k de modo que pelo menos uma das entradas cause falhas de cache na ordem de 10^6.

   (a) Quantos write-misses da cache L1 ocorrem com cada uma das três entradas?

   (b) Quantos write-misses da cache L2 ocorrem com cada uma das três entradas?

   (c) Onde ocorrem a maior parte dos write-misses? Para qual entrada ocorre o maior número de write-misses? Você é capaz de explicar o motivo?

   (d) Quantos read-misses da cache L1 ocorrem com cada uma das três entradas?

   (e) Quantos read-misses da cache L2 ocorrem com cada uma das três entradas?

   (f) Onde ocorrem a maior parte dos read-misses? Para qual entrada ocorre o maior número de read-misses? Você é capaz de explicar o motivo?

   (g) Rode o programa com cada uma das três entradas sem utilizar o perfilamento das caches. Qual é a entrada mais rápida? Qual é a entrada mais lenta? Suas observações sobre o número de misses (read e write) ajudam a explicar o que está ocorrendo? Justifique sua resposta.

5. Embora a função `strtok_r` seja thread-safe, ela tem a péssima característica de modificar a string de entrada. Escreva uma função equivalente que seja thread-safe e que não modifique a string de entrada.

6. Recupere o programa de cálculo de áreas baseado em trapézios que vimos em aula e implementamos utilizando MPI. Modifique aquele programa para que utilizar Pthreads em lugar de MPI. Faça-o utilizando mutexes, semáforos e *busy-waiting*. Quais são as vantagens e desvantagens que você observa em cada abordagem?

7. Implemente uma tabela de Hash thread-safe e paralela. Sua tabela de hash deve ter um bom desempenho e deve aceitar que hajam operações de escrita e leitura ocorrendo concomitantemente.

8. Escreva um programa que estime o tempo gasto em cada uma das seguintes operações envolvendo Pthreads:

   (a) Criação de um thread.

(b) Thread-join.

(c) Obtenção de um lock (mutex).

(d) Liberação de um lock (mutex).

(e) Wait em uma variável de condição.

(f) Signal em uma variável de condição.

(g) Broadcast em uma variável de condição.

Para cada um dos casos descreva com o método utilizado para medir o tempo de cada uma das funções. Não esqueça de documentar a máquina na qual as medidas foram efetuadas.

9. Implemente uma fila thread-safe com múltiplos produtores e múltiplos consumidores. Utilize primeiro (apenas) mutexes e meça o desempenho para 1000 inserções e para 1000 remoções utilizando 64 threads. Altere a sua fila para utilizar mutexes e variáveis de condição. Meça o desempenho para o mesmo conjunto de operações e o mesmo número de threads. Como os resultados se comparam? Você é capaz explicar as diferenças encontradas?

10. Implemente um pipeline de 3 estágios onde a comunicação entre cada um dos estágios é feita através de uma fila. Considere primeiro o caso onde cada estágio do pipeline é executado por um único thread. Em seguida altere o seu código para que ele seja capaz de trabalhar com um número arbitrário de threads em cada estágio. Que mecanismos você utilizou para a sincronização? O que o fez escolher um mecanismo dentre os demais disponíveis?