MAC 0219/5742 - Introdução à Computação Paralela e Distribuída

Mini EP 7 - Contenção

Alfredo Goldman e Vitor Terra

1. Introdução

Em computação paralela, contenção (ou *thread contention*) é o nome dado para a situação em que uma ou mais *threads* se encontram impossibilitadas de continuar pois aguardam um recurso compartilhado que atualmente está sendo utilizado por outra thread. O tempo que uma *thread* passa em contenção é essencialmente perdido, em vez de ser usado para alguma tarefa útil.

Um exemplo de contenção é o caso em que muitas *threads* chegam em uma seção crítica ao mesmo tempo. Como apenas uma pode executar a região, simultaneamente, todas as demais ficarão aguardando até que a região seja liberada. Este caso também é chamado de *lock contention*.

Neste mini EP, tentaremos empregar uma técnica para redução de contenção em um programa contendo uma seção crítica.

2. Código fornecido

Foi fornecido no e-Disciplinas o arquivo `src_miniep7.zip`, contendo os arquivos `contention.c` e um `Makefile`. Este código aloca e preenche aleatoriamente um vetor com M `doubles`, aplica uma função custosa[1] (em termos de tempo) sobre cada elemento, e então calcula o valor máximo do vetor. Todo o trabalho é dividido entre N *threads* da forma mais uniforme possível.

Para compilar o código, use o comando

```
$ make
```

Para executar, use o comando

```
$ ./contention <num_threads> <array_size>
```

onde `num_threads` é a quantidade de *threads* a serem criadas para o trabalho (que deve ser ≥ 1) e `array_size` é o tamanho do vetor. A saída do programa será o

[1] Esta função tem a única finalidade de ocupar as *threads* por mais tempo, para que melhor possamos observar os efeitos das mudanças que vamos aplicar.

tempo real transcorrido (*wall time*[2]) desde a criação das *threads* até a finalização, em segundos.

Para remover o arquivo binário gerado na compilação, use o comando

```
$ make clean
```

Certifique-se de que você entendeu o código e execute algumas vezes antes de prosseguir.

3. Técnica para reduzir contenção

Note, que para cada iteração na função `thread_work`, as *threads* precisam adquirir o `lock` antes de verificar se o elemento daquela iteração é maior que o máximo encontrado até o momento. No entanto, essa verificação resulta em `true` relativamente poucas vezes (seriam muitas vezes se os valores estivessem de alguma forma em ordem crescente, o que não é o caso). Sendo assim, muitas `threads` acabam tendo que esperar, na contenção, sendo que sequer irão alterar a variável global `max`.

Por outro lado, podemos fazer a verificação antes de entrar na seção crítica, já que não há restrição para múltiplas leituras simultâneas em `max`. Dessa forma, não é preciso entrar na seção crítica quando o resultado for falso. Assim, evitamos que a *thread* fique na contenção desnecessariamente e ela pode prosseguir para a próxima iteração. É possível inserir essa checagem compilando com a seguinte opção:

```
$ make IF=1
```

Essa opção inserirá um `if`, tal como aquele presente na seção crítica, imediatamente antes da chamada de `pthread_mutex_lock`. Efetivamente, o código na linha 70 em `thread_work` é modificado para algo como:

```
if (t->arr[i] > max)
{
    pthread_mutex_lock(&lock);
    if (t->arr[i] > max)
        max = t->arr[i];
    pthread_mutex_unlock(&lock);
}
```

[2] Tempo real decorrido – Wikipédia, a enciclopédia livre (wikipedia.org)

Rode com e sem o `IF=1`. Houve diferença de tempo de execução? E se inseríssemos mais `if`s encadeados antes da seção crítica? Será que o tempo de contenção seria mitigado mais ainda? Você pode compilar com `IF=K`, trocando `K` pelo número de `if`s que deseja inserir.

4. Entrega

Você deverá produzir um relatório em formato .txt ou .pdf contendo os seguintes itens:

1. Faça testes variando o tamanho do vetor, a quantidade de *threads* e a quantidade de `if`s encadeados, mostrando médias e intervalos de confiança dos tempos impressos na saída. Você pode utilizar gráficos, imagens e/ou tabelas para apresentar os resultados obtidos.

2. Explique os resultados observados nos testes do item anterior. Por que você acha que ocorreu o observado?

3. Explique por que não podemos eliminar o `if` de dentro da seção crítica quando adicionamos o `if` de fora.

Entregue pelo e-Disciplinas o relatório em .txt ou .pdf com o seu nome e sobrenome. Caso deseje anexar imagens fora do relatório, entregue-as junto com o relatório em uma pasta compactada com o seu nome e sobrenome no seguinte formato: `miniep7_nome_sobrenome.zip`. Relatórios em .doc, .docx ou .odt não serão aceitos.

Em caso de dúvidas, use o fórum de discussão do e-Disciplinas ou entre em contato diretamente com o monitor (vitortterra@ime.usp.br) ou o professor (gold@ime.usp.br).

5. Agradecimentos

A Giuliano Belinassi e Matheus Tavares, monitores da disciplina em anos anteriores, pela elaboração do enunciado do mini EP, e a William Razente pela correção de um *bug* na versão anterior do `Makefile`.