*PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS*

*Instituto de Ciências Exatas e Informática*

*Departamento de Ciência da Computação - Curso de Ciência da Computação*

*Disciplina: Seminários II Segundo semestre de 2019*

*Professor: Saulo Augusto de Paula Pinto —* [***saulo@pucminas.br***](mailto:saulo@pucminas.br)

***Quarto Roteiro***

***Seções críticas. Balanceamento de carga.***

***ATENÇÃO****: poste apenas um arquivo no SGA com as respostas daquilo que você conseguir terminar de fazer* ***ATÉ O FINAL DESTA AULA (dia 16/10/2019, 12:20h)****. Duvido que consigam terminar tudo! :-p ☺.*

*Em CADA resposta, se você teve que alterar o código em C, poste o código logo após sua resposta.*

*A nota desta atividade é proporcional à quantidade de respostas corretas entregues. Mas não se preocupe com isso. Preocupe-se em entender e aprender o que está fazendo no seu ritmo! Posteriormente, você deverá postar o restante das atividades não concluídas hoje. Mãos à obra!...*

Faça (ou refaça) o roteiro anterior e responda:

1. Como ficou o tempo de execução da versão paralelizada com *reduction* em relação à versão anterior (sequencial)? Use seu código para imprimir o *speedup* alcançado.
2. Como ficou o tempo de execução da versão paralelizada com a seção crítica em relação ao que usou a redução (*reduction*)? Rode pelo menos 10 vezes e compare o tempo médio (computado e impresso em seu código, como na tarefa do cálculo de Pi).
3. Explique porque o tempo com a seção crítica ficou “bom” (“comparável” ao tempo com a redução). Pense na carga de trabalho que cada *thread* recebe ao serem divididas as iterações do *loop* paralelizado.
4. Em relação ao balanceamento de carga, compare os tempos de execução usando a redução (sem seção crítica) e cada uma das três políticas (*static, dynamic e guided*, estas duas últimas com *chunk size* igual a 100). Use LIMITE\_MAX igual a 2 milhões e 4 milhões. Novamente, rode pelo menos 10 vezes com cada parâmetro e reporte algo parecido com:

**Número de execuções:**

**Tempo médio nas execuções:**

**LIMITE\_MAX:** 2000000

|  |  |  |  |
| --- | --- | --- | --- |
| **Política** | **static** | **dynamic** | **guided** |
| **Tempo médio** |  |  |  |

**Número de execuções:**

**Tempo médio nas execuções:**

**LIMITE\_MAX:** 4000000

|  |  |  |  |
| --- | --- | --- | --- |
| **Política** | **static** | **dynamic** | **guided** |
| **Tempo médio** |  |  |  |

1. Varie o *chunk* de 50 em 50 até um valor máximo, MAX\_CHUNK, que você definir, e tente determinar o *chunk* ótimo para as políticas *dynamic* e *guided*. Novamente, reporte o tempo médio para, pelo menos, 10 execuções. Ou seja, para cada valor de *chunk* seu código deve rodar pelo menos 10 vezes e reportar o tempo médio para verificação. Mantenha variáveis para armazenar o “tempo mínimo global”, isto é, o menor tempo entre todas as execuções com todas as políticas com todos os valores de *chunk*, e o “tempo mínimo global médio”, isto é parecido com o anterior, mas considera as 10 (pelo menos) execuções. Use LIMITE\_MAX de 4 milhões.