**Lista de Sistemas Distribuídos e Programação Concorrente**

Prof. MSc. Rodrigo D. Malara

OpenMP - Baixe os arquivos do Google Classroom para realizar os exercícios

1. omp\_reduction.c

Redução (totalização) de variável com loop paralelo

* + Compile e execute o programa.
  + Verifique a saída e o resultado
  + Remova a diretiva de redução e teste o programa novamente. Verifique se a somatória muda.

Ao término desse exercício enviar um print da execução do código original com a diretiva de redução (reduction) e um print da execução do código sem a diretiva de redução mas que apresente um resultado diferente do primeiro printscreen. Nomear os arquivos como ex5-1.jpg e ex5-2.jpg

Explique o motivo de ocorrer a diferença entre a versão com a diretiva de redução e a versão sem essa mesma diretiva.

1. omp\_workshare0.c

Paralelização simples.

* Compile e execute o programa.
* Execute o programa mais algumas vezes e analise a saída. O que você vê? Normalmente, o escalonamento de threads não é determinístico. Toda vez que você executa o programa, diferentes ordens de execução são exibidas.
* Remove a diretiva #pragma omp for. Observe a diferença na saída em comparação ao programa anterior. Todo o trabalho será executado totalmente por cada um dos threads.

Ao término desse exercício enviar o print de ao menos 2 execuções mostrando diferenças no escalonamento de threads. Nomear os arquivos como ex1-1.jpg e ex1-2.jpg

1. omp\_workshare1.c

Este exemplo demonstra o uso da construção de compartilhamento de trabalho do loop OpenMP. Observe que ele especifica o agendamento dinâmico de encadeamentos e atribui um número específico de iterações a serem feitas por cada encadeamento.

* Compile e execute o programa.
* Execute o programa mais algumas vezes e analise a saída. O que você vê? Normalmente, o agendamento dinâmico (dynamic scheduling) não é determinístico. Toda vez que você executa o programa, diferentes threads podem executar diferentes partes do trabalho. É até possível que um encadeamento não funcione porque outro encadeamento é mais rápido e requer mais trabalho. Na verdade, pode ser possível que um thread faça todo o trabalho.
* Edite o arquivo de origem workshare1 e altere o agendamento dinâmico para agendamento estático (static scheduling).
* Recompile e execute o programa modificado. Observe a diferença na saída em comparação com o agendamento dinâmico. Especificamente, observe que o thread 0 obtém o primeiro bloco, o thread 1 o segundo bloco e assim por diante.
* Execute o programa mais algumas vezes. A saída muda? Com o planejamento estático, a alocação de trabalho é determinística e não deve ser alterada entre as execuções, e todo segmento tem trabalho a fazer.
* Reflita sobre possíveis diferenças de desempenho entre o agendamento dinâmico e estático.

Ao término desse exercício enviar um print da execução do código original com agendamento dinâmico (dynamic scheduling) e um print da execução do código com agendamento estático (static scheduling). Nomear os arquivos como ex2-1.jpg e ex2-2.jpg

Também explique o motivo do agendamento estático ser mais rápido e preferido do que o agendamento dinâmico

1. omp\_workshare2.c

Este exemplo demonstra o uso da construção de compartilhamento de trabalho OpenMP SECTIONS Observe como a região PARALELA é dividida em seções separadas, cada uma das quais será executada por um encadeamento.

* Compile e execute o programa.
* Execute o programa várias vezes e observe as diferenças na saída. Como há apenas duas seções, você deve notar que alguns segmentos não fazem nenhum trabalho. Você pode / não perceber que os threads que estão fazendo o trabalho podem variar. Por exemplo, a primeira vez que o encadeamento 0 e o encadeamento 1 podem fazer o trabalho e a próxima vez que ele pode ser o encadeamento 0 e o encadeamento 3. É até possível que um encadeamento faça todo o trabalho. Qual encadeamento funciona é não-determinístico neste caso.

Ao término desse exercício enviar um print da execução do código original. Nomear o arquivo como ex3.jpg

1. omp\_mm.c

Este exemplo executa um múltiplo de matriz distribuindo as iterações da operação entre os encadeamentos disponíveis.

* Depois de revisar o código-fonte, compile e execute o programa.
* Revise a saída. Ele mostra qual thread fez cada iteração e a matriz com o resultado.

Ao término desse exercício enviar um print da execução do código original. Nomear o arquivo como ex4.jpg