5º Trabalho Prático de Programação Concorrente – 2016
QuickSort Concorrente em *MPI (OpenMP)* e *OpenMP*

Desenvolva uma aplicação concorrente que ordene uma sequência ***n*** de números inteiros usando o Algoritmo PSRS (***Parallel Sorting by Regular Sapling***) e ***p*** processos concorrentes (processos e/ou *threads*). O algoritmo PSRS é uma versão concorrente do ***quicksort***, proposta no Capítulo 14 de Quinn, 2003, Seção 14.5.1, páginas 346 e 347. A sua implementação neste trabalho prático deve considerar **tanto** o paradigma de interação com espaço de endereçamento distribuído **quanto** o compartilhado, em ***C POSIX no Linux***. Use o ***MPI (OpenMPI)*** e o ***OpenMP***.

O algoritmo ***PSRS*** tem 4 fases distintas. Implemente a 1ª fase e parte da 2ª fase em ***OpenMP*** (na 2ª fase, implemente em ***OpenMP*** até a escolha dos pivôs). Implemente a parte final da 2ª fase (ordenar as sublistas em ***p*** pedaços disjuntos), a 3ª fase e parte da 4ª fase em ***MPI*** (até juntar as ***p*** partições em uma única lista). A parte final da 4ª fase (ordenar todos os elementos da lista única) deve ser implementada em ***OpenMP*** novamente.

Veja o livro ou o arquivo ***pdf*** disponível no *moodle*, contendo uma descrição detalhada dessas fases e do que deve ser implementado em ***MPI (OpenMPI)*** e ***OpenMP***.

A execução da aplicação deve ser feita pela linha de comando (***shell***) seguindo:

GGG_psrs <n> <p>

GGG faz parte do nome do executável (binário) e indica o grupo. Deve ser substituído, por exemplo, por G01 para o grupo 01 ou G20 para o grupo 20;

<n> é um argumento de entrada e determina a quantidade de números inteiros a ordenar;

<p> é um argumento de entrada e determina o número de processos e *threads* que executarão concorrentemente, respectivamente no ***MPI (OpenMPI)*** e ***OpenMP***.

Os grupos criados para o trabalho anterior (Crivo Erathsthenes) devem ser mantidos.

Os números presentes na relação desordenada devem ser gerados aleatoriamente pelo algoritmo. Não são fornecidos na linha de comando.

A saída do algoritmo deverá obrigatoriamente ser apenas uma relação ordenada com ***n*** números separados por uma vírgula e um espaço em branco, com exceção do último número que será seguido por um ponto final apenas. Não se preocupe com a quebra da linha nesta saída.

Siga rigorosamente esta especificação. Em uma das etapas da correção deste trabalho a saída será redirecionada para um arquivo e o conteúdo deste será comparado usando o utilitário ***diff***. Para a correção nós escolheremos sequências de números para serem testadas nos algoritmos.

Desenvolva a aplicação concorrente segundo as diretrizes passadas aqui e em aula.

O trabalho deverá ser entregue no *Moodle*, em um ***zip***, contendo:

(1) código-fonte na linguagem C (siga o nome solicitado acima);

(2) ***pdf*** contendo a descrição do desenvolvimento da aplicação segundo os critérios acima.

(3) a linha de comando para compilar e a linha de comando para executar o seu programa. Se o grupo criou algum ***Makefile*** para a compilação, por favor, anexe-o junto à submissão.

Coloque **no código fonte e no pdf** o nome dos integrantes do grupo que ***efetivamente participaram*** do trabalho. Os nomes que não aparecerem nessa relação ficarão com nota "zero" no trabalho.

A data de entrega deste trabalho será especificada em sala de aula e estará visível no *Moodle*.

As questões omissas e/ou ambíguas serão fixadas pelo professor.

Qualquer dúvida (ou se encontrar algum problema na especificação), por favor, entre em contato com o professor para a orientação/correção adequada.

Referência

[1] QUINN, M.J. Parallel Programming in C with MPI and OpenMP, McGraw-Hill,Published, capítulo 14, pp: 338-352, ano 2003, ISBN 0072822562. O algoritmo a ser implementado está descrito especificamente na Seção 14.5.1, páginas 346 e 347.