QuickSort Concorrente em MPI e OpenMP

Trabalho Prático de Programação Concorrente

Danilo Nery	8602430
Frederico de Azevedo Marques	8936926
Isadora Maria Mendes de Souza	8479318
Roberto Pommella Alegro	8936756

Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação
São Carlos, SP — Novembro/2016

Índice

Introdução

Este trabalho propõe o desenvolvimento de uma aplicação concorrente que ordene uma sequência n de números inteiros usando o Algoritmo Parallel Sorting by Regular Sampling (PSRS) e p processos concorrentes (processos e/ou threads).

Este algoritmo é uma versão concorrente do Quicksort caracterizado por Michael QUINN (2003). Para a implementação do mesmo, foi considerado tanto o paradigma de interação com espaço de endereçamento distribuído quanto o compartilhado, em C POSIX no Linux e foi usado MPI (OpenMPI) e OpenMP.

Uma breve explicação sobre o algoritmo

O algoritmo PSRS possui basicamente quatro fases distintas que podem ser explicadas assim:

Existe uma lista de n valores e p processos que compartilham memória. Os *n* valores são divididos igualmente entre os *p* processos — isto é, haverá *p* processos com não mais que *n/p* valores e o quicksort sequencial é executado para cada parte os valores que cada um desses processos ficou. Depois disso, são escolhidos alguns valores de amostra de cada um desses processos, ordena-se esses valores e são selecionados *p - 1* valores de pivô dessa lista de valores amostrais. Desta primeira parte do algoritmo até a escolha dos pivôs, a implementação é em OpenMP.

A partir deste ponto, são criados *p* processos MPI que particionam sua "lista de valores" em *p* partes e usam os *p - 1* valores de pivô como "separadores". Assim, cada processo i (com i = 1, 2, …, p) mantém sua respectiva parte e envia a que não lhe pertence.

Para concluir o algoritmo, a implementação volta a ser em OpenMP. Nesta fase, cada processo agrupa suas partições numa lista só, o que resulta em todos os valores ordenados.

Para ilustrar o algoritmo, vamos supor como seria ordenar 30 elementos não ordenados: Eles seriam divididos em 3 processos, cada processo com 10 valores. Cada processo ordenará a sua respectiva lista de 10 valores usando quick sort sequencial. Cada processo selecionará amostras de valores e então, haverá uma lista ordenada de amostras originada através de todos os processos OpenMP. Com isso, será selecionado os pivôs, neste caso, 2. Os 10 valores de cada processo serão separados em no máximo 3 grupos (o grupo de valores que são menores que o primeiro pivô, os grupo de valores que que estão entre o valor do primeiro e segundo pivô, e o grupo de valores que são maiores que o segundo pivô). Com isso, cada processo manterá os valores que pertencem a ele haverá troca de mensagens para que cada ele envie valores que não.

Execução

A execução da aplicação deve ser feita pelas linhas de comando a seguir:

$ make

$make run <n> <p>

onde <p> é o numero de processos e <n> é o numero de valores a serem ordenados.

Referências

[1] QUINN, M.J. Parallel Programming in C with MPI and OpenMP, McGraw-Hill, Published, capítulo 14, pp: 338-352, ano 2003, ISBN 0072822562.