UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
SISTEMAS OPERACIONAIS I
PROFESSOR: LEANDRO MARZULO
CURSO: CIÊNCIA DA COMPUTAÇÃO

Relatório OpenMP e Pthreads

INTEGRANTES:

CAMILA ELEUTÉRIO GUSMÃO
MURILO DE JESUS SANTOS SILVA
THAINÁ LOPES FIGUEIREDO

ENUNCIADO:

Elaborar duas versões do programa contador de primo (o mesmo do trabalho 1): uma versão usando pthreads e outra usando OpenMP. Nas duas versões, o programa deverá receber, por linha de comando, os limites inferior e superior e o número de threads desejadas. No caso da versão OpenMP, você deverá usar compilação condicional (com #ifdef _OPENMP) conforme visto nos exemplos em sala de aula. A implementação em OpenMP deve usar a diretiva parallel for com redução. Você deverá entregar 1 zip com os dois programas e os tempos usando oversubscription, escalomento "static", "dynamic" e "guided".

EXPLICAÇÃO DA IMPLEMENTAÇÃO:

O programa foi implementado em um ambiente de testes composto por uma CPU Intel Core i5 2450M (quatro núcleos) 2,5 GHz, RAM de 4GB, Windows 10 Pro versão 1607 (Build 14393.321) e GCC 4.4.5. Tanto na implementação dos primos na OpenMP quanto na versão de PThreads, encontramos um total de 664580 números primos na faixa fornecida de 1 a 10 milhões de números. Verificamos a medição do tempo durante a compilação utilizando o comando “time” para ambas as versões, seus valores estão descritos na tabela abaixo. É possível verificar que na comparação entre os tempos Real e Sys, a versão de Pthreads apresentou uma melhora no tempo em relação as versões do OpenMP, devido a utilização das threads criadas durante sua execução. Porém o tempo de Users é menor na versão OpenMP, a partir da utilização de 4 threads. Isso confirma que a versão de Ptheads aperfeiçoa a paralelização porque a comunicação entre as threads é mais eficiente, devido ao espaço de endereçamento compartilhado, e que o OpenMP é uma técnica de paralelização menos trabalhosa, visto que a versão Static antes do loop todas as interações já são atribuídas para todos os threads, Dynamic durante sua execução que vai determinando o escalonamento e o Guided as interações sequenciais são atribuídas a cada thread dinamicamente.

OpenMP Static

Quantidade de Threads	**Real**	**Users**	**Sys**
2	0m16.715s	0m27.500s	0m0.031s
4	0m12.065s	0m38.969s	0m0.031s
8	0m11.493s	0m43.469s	0m0.016s
16	0m11.258s	0m43.891s	0m0.031s

OpenMP Dyn

Quantidade de Threads	**Real**	**Users**	**Sys**
2	0m18.214s	0m30.391s	0m0.016s
4	0m12.324s	0m38.969s	0m0.031s
8	0m11.373s	0m43.297s	0m0.047s
16	0m11.266s	0m44.063s	0m0.031s

OpenMP Gui

Quantidade de Threads	**Real**	**Users**	**Sys**
2	0m17.505s	0m28.125s	0m0.047s
4	0m12.282s	0m39.172s	0m0.016s
8	0m11.550s	0m43.375s	0m0.031s
16	0m11.235s	0m44.234s	0m0.031s

PThreads

Quantidade de Threads	**Real**	**Users**	**Sys**
2	0m15.706s	0m25.797s	0m0.031s
4	0m12.202s	0m38.984s	0m0.016s
8	0m11.467s	0m43.703s	0m0.016s
16	0m11.351s	0m44.422s	0m0.063s

Considerando que o processador utilizado nos casos de teste possui 4 núcleos, e que o oversubscription é justamente quando há um número superior de threads em relação ao número de cores, constatamos que o estes casos foram alcançado para todos os testes com número de threads superior a 4.