

Modalidade do trabalho: Relatório técnico-científico
Evento: XXIII Seminário de Iniciação Científica

ANÁLISE DE DESEMPENHO DA PARALELIZAÇÃO DO CÁLCULO DE NÚMEROS PRIMOS UTILIZANDO PTHREAD E OPENMP¹

Francisco Berti Da Cruz², Cleber Cristiano Sartorio³, Edson Luiz Padoin⁴, Emilio Hoffmann⁵.

¹ Trabalho realizado no componente curricular de Processamento Paralelo do curso de Graduação em Ciência da Computação da Unijuí

² Aluno do Curso de Graduação em Ciência da Computação da UNIJUI, fbc9911@gmail.com

³ Aluno do Curso de Graduação em Ciência da Computação da UNIJUI, cleber.sartorio@hotmail.com

⁴ Professor Doutorando do Curso de Ciência da Computação da Unijuí, padoin.edson@gmail.com

⁵ Aluno do Curso de Graduação em Ciência da Computação da UNIJUI, emiliohoffmann@gmail.com

1. Introdução

Existem situações onde ocorre a necessidade de encontrar soluções para problemas que exigem uma grande demanda de processamento. Em alguns casos com a utilização da programação sequencial não se obtém resultados satisfatórios em tempos aceitáveis, sendo inviável sua resolução. Nestes casos utiliza-se da programação paralela, envolvendo processos ou threads que assumem funções como subprocessos de um processo mestre.

Pode-se afirmar, conforme [Kumar et al. 2008] e [Almasi and Gottlieb 1988] que Computação Paralela é uma forma de computação em que vários cálculos são realizados ao mesmo tempo, operando sob o princípio de que grandes problemas geralmente podem ser divididos em problemas menores, que então são resolvidos concorrentemente (em paralelo).

Deste modo, neste trabalho será realizada a paralelização de um algoritmo sequencial, com o objetivo de analisar e verificar os ganhos de desempenho com a utilização da paralelização. Será utilizado as API's Pthread e OpenMP, em conjunto com flags de compilação, informações estas que serão apresentadas no decorrer do trabalho.

2. Problema: Cálculo dos números Primos

O estudo consiste em calcular o tempo que se leva para se obter a quantidade de números primos em um intervalo pré-definido. Como se sabe, na formação do conjunto dos números naturais existe um tipo de numeral que possui a propriedade de ser divisível somente por 1 e por ele mesmo, recebendo a denominação de número primo. Segundo [de Oliveira Santos 1998] um número natural P é um número primo quando ele tem exatamente dois divisores distintos: o número 1 e ele mesmo (P). Ex.: 2, 3, 5, 7, 11, 13, 17, 19.

3. API's

Modalidade do trabalho: Relatório técnico-científico
Evento: XXIII Seminário de Iniciação Científica

API(Application Programming Interface) é o conjunto de padrões de programação que permite a construção de aplicativos. De modo geral, é composta por uma série de funções acessíveis somente por programação, e que permitem utilizar características do software menos evidentes ao utilizador tradicional.

3.1 Pthread

Esta API nos disponibiliza maneiras eficientes de expandir o processo de execução em novos processos concorrentes, sendo possível executar com uma maior eficiência sistemas computacionais com múltiplos processadores e/ou processadores com múltiplos núcleos.

Para [Barney et al. 2010] e [Butenhof 1997] pthread é definida como um conjunto de tipos da linguagem C. Existem cerca de 100 procedimentos em pthread, todos com o prefixo “pthread”. As operações com threads incluem a criação, finalização, sincronização, agendamento e sinalização. Todas as threads tem acesso à mesma memória global compartilhada, sendo o programador responsável pela sincronização (proteção) do acesso aos dados.

3.2 OpenMP

Através do processamento paralelo, ocorre frequentemente a utilização de uma API denominada OpenMP(Open Multi-processing), cuja qual é multiplataforma, baseada em memória compartilhada para as linguagens C, C++ e Fortran, possuindo um conjunto de diretivas para o computador pré-definidas e com a finalidade de nos auxiliar na criação de threads para o processamento paralelo.

O OpenMP fornece um modelo portátil e escalável para desenvolvimento de aplicações paralelas com memória compartilhada. Oferece métodos simples para exploração de paralelismo sem interferir no projeto do algoritmo. Um programa que utiliza a API do OpenMP compila e executa corretamente em ambientes sequenciais assim como em ambientes paralelos.[Abinader and Barreto]

4. Metodologia

4.1 Equipamento

Para os testes foi utilizado uma máquina DELL, processador Intel(R) Core(TM) i3-2330M CPU @ 2.20GHz(2 núcleos com duas threads cada), 8gb de RAM DDR3, Sistema Operacional Ubuntu 14-04 LTS, Kernel 3.13.0-49-generic, compilador GCC 4.8.2.

4.2 Algoritmos

A implementação dos algoritmos empregada no trabalho está assim organizada: Inicialmente será analisado o desempenho de um algoritmo sequencial, este que servirá de base para os demais testes. Posteriormente serão testadas flags de compilação com o objetivo de otimizar o algoritmo e, por fim, será realizado a implementação paralela. Vale lembrar que para realizar a verificação da quantidade de números primos existem diversas maneiras, como:

Algoritmo 1: os divisores entre 2 e N-1 são testados;

Modalidade do trabalho: Relatório técnico-científico
Evento: XXIII Seminário de Iniciação Científica

Algoritmo 2: os divisores pares não são testados, a pesquisa se limitando aos divisores ímpares;
Algoritmo 3: os divisores ímpares até a raiz quadrada do N são testados;
Algoritmo 4: parada do programa quando um divisor é encontrado.

Neste trabalho será realizada a implementação do pior caso, algoritmo 1, visto que não se busca desempenho, e sim, medição do tempo de execução.

A seguir será apresentado a implementação sequencial, paralela com Pthread e por fim, paralela com OpenMP, salienta-se que em todas as execuções foram utilizadas a flag de compilação -O3 e que os testes foram executados 10 vezes para se obter uma média aritmética. Vale lembrar que na programação paralela foram executados os testes com 1,2,4,5,10 e 20 threads.

4.2.1 Algoritmo Sequencial

O algoritmo sequencial consiste em verificar se o número testado é primo ou não, se for, é somado 1(um) a uma variável de controle. Vale lembrar que para mensuração do tempo de execução foi utilizada uma system call do kernel do linux.

4.2.2 Algoritmo Paralelo com Pthread

Nesta etapa, o algoritmo antes sequencial sofre algumas modificações para suportar referida característica. Deve-se observar que com o uso de threads ocorre problemas quanto a integridade da informação, visto que, vários processos em execução simultaneamente podem comprometer o resultado final.

4.2.3 Algoritmo Paralelo com OpenMP

Quanto a utilização da API OpenMP destaca-se sua facilidade de utilização, sendo necessário basicamente inserir diretivas no algoritmo sequencial onde deve ser executado paralelamente. Em relação a implementação sequencial, foi utilizado apenas 3 diretivas, sendo estas necessárias para realizar a paralelização e integridade do cálculo de referido método.

5. Resultados

5.1 Algoritmo Sequencial

No algoritmo sequencial foi realizada uma comparação dos tempos de execução do código compilado com e sem flag, onde no primeiro algoritmo foi executado em 810,41 segundos e no segundo em 799,02. Há um ganho de praticamente 11 segundos no código compilado com flag.

5.2 Algoritmo Paralelo com Pthread

Neste teste ocorreu um ganho de tempo significativo na execução com duas threads, posteriormente o tempo torna-se praticamente estático. Vale ressaltar que apesar de ocorrer a criação de 20 threads, não houve um aumento no período de execução, o que tende a ser verdadeiro, devido ao tempo gasto para criação das threads.

Modalidade do trabalho: Relatório técnico-científico
Evento: XXIII Seminário de Iniciação Científica

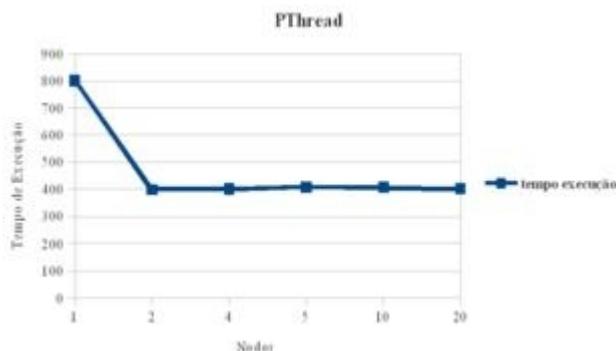


Figura 1. Gráfico com os tempos de execução da implementação com pthread

5.2 Algoritmo Paralelo com OpenMP

No algoritmo paralelo com OpenMP o tempo foi reduzindo gradativamente conforme a criação das threads. Mais detalhes com a duração das execuções na figura 4.

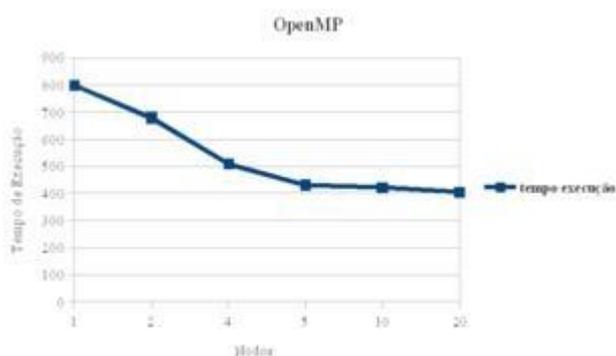


Figura 2. Gráfico com os tempos de execução da implementação com OpenMP

Percebe-se que em comparação com a API Pthread ocorre uma equivalência no início(1 thread) e no final(20 threads), sendo que as demais coletas demoraram mais tempo para serem executadas no OpenMP.

6. Conclusão

Dado o exposto, pode-se observar através dos resultados obtidos que, a utilização da programação paralela corresponde a um ganho significativo de tempo em relação a programação sequencial, e que ao ser utilizado as APIs Pthreads e OpenMP pode-se ocorrer variações em um mesmo algoritmo.

Modalidade do trabalho: Relatório técnico-científico
Evento: XXIII Seminário de Iniciação Científica

Por fim, como resultado final deste trabalho, a análise e pesquisa realizada nos leva a perceber que a escolha de determinada API deve ser fruto de um estudo aprofundado sobre a necessidade da aplicação, hardware disponível e flags específicas para o processador que será utilizado.

Palavras-chave: processamento paralelo; paralelo; sequencial;

Referências

- Abinader, B. and Barreto, R. Comparativo das ferramentas de código aberto pthreads e gomp para criação de threads com o compilador gcc.
- Almasi, G. S. and Gottlieb, A. (1988). Highly parallel computing.
- Barney, B. et al. (2010). Introduction to parallel computing. Lawrence Livermore National Laboratory, 6(13):10.
- Butenhof, D. R. (1997). Programming with POSIX threads. Addison-Wesley Professional.
- de Oliveira Santos, J. P. (1998). Introdução teoria dos números. Instituto de Matemática Pura e Aplicada.
- Kumar, R., Marinov, D., Padua, D., Parthasarathy, M., Patel, S., Roth, D., Snir, M., and Torrellas, J. (2008). Parallel computing research at Illinois.