SALÃO DO CONHECIMENTO UNIJUÍ 2014
ciência - tecnologia - desenvolvimento social
Câmpus Ijuí, Santa Rosa, Panambi e Três Passos

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

UM ESTUDO COMPARATIVO DE DESEMPENHO UTILIZANDO PROGRAMAÇÃO SEQUENCIAL VS PARALELA APLICADO EM ALGORITMOS GENÉTICOS[1]

Eldair F. Dornelles[2], Henrique A. Richter[3], Miquéias F. M. Trennepohl[4], Taís T. Siqueira[5], Rogério S. M. Martins[6], Edson L. Padoin[7].

[1] Projeto de pesquisa para paralelização dos algoritmos desenvolvidos no grupo GCA
[2] Aluno do Curso de Graduação em Ciência da Computação da UNIJUÍ, eldair.dornelles@unijui.edu.br;
[3] Aluno do Curso de Graduação em Ciência da Computação da UNIJUÍ, henrique.richter@unijui.edu.br;
[4] Aluno do Curso de Graduação em Ciência da Computação da UNIJUÍ, miqueias.trennepohl@yahoo.com.br;
[5] Aluna do Curso de Graduação em Ciência da Computação da UNIJUÍ, taistsiqueira@gmail.com;
[6] Professor Pesquisador, Mestre em Computação Aplicada, Curso de Ciência da Computação, rogerio@unijui.edu.br;
[7] Professor Pesquisador, Mestre em Engenharia da Produção, Curso de Ciência da Computação, padoin@unijui.edu.br;

Introdução

Com a crescente disseminação das arquiteturas computacionais multi-core, as quais proporcionam o processamento de diversas tarefas em paralelo, gerando assim significativos aumentos de desempenho, tem-se gerado um massivo aumento nas pesquisas voltadas para a exploração destes recursos, devido ao fato de que em muitos casos, parte deste poder de processamento disponibilizado não é aproveitado no desenvolvimento das aplicações. É neste contexto que se insere a otimização de algoritmos, a qual faz uso de alguns métodos de paralelização, dentre os quais, pode-se citar a utilização das bibliotecas OpenMP e POSIX Threads.

Neste trabalho é apresentado um estudo comparativo de desempenho entre 3 diferentes métodos de implementação, sendo estes aplicados a um mesmo problema. Inicialmente foi desenvolvido uma implementação sequencial que serviu de base para as 2 seguintes, as quais fizeram uso das bibliotecas OpenMp e POSIX Thread para paralelização de seus algoritmos.

Para realizar as diferentes implementações foi escolhida a meta-heurística Algorítmo Genético, a qual busca uma otimização global e baseia-se nos mecanismos de seleção natural e da genética. Basicamente é gerada uma população com n indivíduos, sendo que cada indivíduo contêm informações que geram uma determinada solução para o problema analisado. Enquanto não for encontrado um individuo que contenha uma solução satisfatória é feito o cruzamento entre os indivíduos, gerando uma nova população, dando assim inicio a uma nova iteração em busca da solução definida, segundo Linden(2012).

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

O problema escolhido para a paralelização dos algoritmos, foi o problema das n rainhas, este, é um problema combinatório exponencial que consiste em colocar n rainhas em um tabuleiro de xadrez com tamanho nxn de forma que não haja colisões, conforme Laguna (1994). Uma colisão ocorre quando duas ou mais rainhas estão posicionadas na mesma linha, coluna ou diagonal. Porém, não é o objetivo deste trabalho encontrar a solução para tal problema, mas sim, medir e analisar o desempenho dos 3 algorítmos implementados, considerando o mesmo número de iterações.

Metodologia

No processo de estudo e desenvolvimento do tema abordado neste trabalho, buscou-se conhecimentos teóricos sobre: problema das n rainhas, a meta-heurística algoritmo genético, os dois métodos de programação paralela (OpenMP e Pthread) e linguagem de programação C++, visando adquirir entendimento suficiente para desenvolver as implementações. Nos parágrafos seguintes são apresentados alguns conceitos sobre threads e explanado de forma rápida os métodos de programação paralela utilizados nesse trabalho.

Thread é uma forma de um processo dividir a si mesmo em várias tarefas que podem ser executadas paralelamente, fazendo com que o sistema gerencie cada tarefa de forma independente. Diferente dos processos, que são independentes e a memória não é compartilhada, as threads de um mesmo processo podem utilizar as mesmas instruções e variáveis. Além de ocupar menos memória, a troca de contexto entre threads é mais rápida do que entre processos, segundo Edward (2006).

Sendo assim, as threads são importantes elementos no processamento paralelo, pois através da sua utilização é possível fazer com que um código sequencial (executado por apenas uma thread) seja dividido entre várias threads e tenha o seu desempenho melhorado. Para realizar essa divisão de processamento entre múltiplas threads, os programadores contam com a ajuda de bibliotecas que auxiliam na paralelização dos códigos sequenciais desenvolvidos. Dentre as bibliotecas oferecidas pode-se destacar a PThread e a OpenMP, as quais seguem descritas abaixo.

Pthread são bibliotecas para as linguagens C e C++ que permitem gerar um conjunto de threads que poderão rodar em simultâneo. Implementações feitas com essas bibliotecas são mais eficientes em sistemas multi-processador e/ou multi-core onde os threads criados serão executados em múltiplos processadores/núcleos, possibilitando que o tempo de execução do programa diminua, ganhando mais velocidade através do processamento paralelo, este maior desempenho também pode ser obtido em sistemas com um único processador. YOLINUX (2014).

A biblioteca Pthread trabalhada com programação paralela com threading explícito, ou seja, o programador é responsável por criar explicitamente as múltiplas threads e definir o trabalho que

será realizado pelas mesmas. Além da criação das threads, o programador também deve gerenciar a sincronização entre as mesmas, para que a paralelização do código sequencial não corrompa os resultados obtidos.

Para que a biblioteca Pthread possa ser utilizada na linguagem de programação C++, que foi utilizada para o desenvolvimento da aplicação deste trabalho, é necessário realizar a sua importação através do comando "#include <pthread.h>". Após realizar a importação já é possível utilizar os recursos e funções disponibilizadas pela biblioteca Pthread, como por exemplo, a criação de uma nova thread a partir do comando "pthread_create" e a sicronização através do comando "pthread_join".

OpenMP é um padrão utilizado para a paralelização de aplicações. Este padrão foi desenvolvido e é mantido pelo grupo OpenMP Architecture Review Board que é formado por empresas com grande participação no desenvolvimento de software e hardware, tais como IBM, Intel, SUN, dentre outras. Surgiu pela necessidade que estas grandes empresas encontraram em formular um padrão para a programação paralela em ambientes com memória compartilhada. Nestes ambientes, os vários núcleos de processamento existentes fazem uso de um mesmo recurso de memória, sendo assim, não há necessidade de realizar troca de informações entre os núcleos de processamento, porém, é necessário realizar uma sincronização entre os mesmos para que não ocorram problemas no acesso a memória compartilhada, segundo CALDAS e SENA (2008).

Ao contrário do que algumas pessoas podem pensar, o OpenMP não trata-se de uma linguagem de programação, mas sim de um padrão que define uma API para programação paralela em memória compartilhada. Ele reúne um conjunto de diretivas de compilação, rotinas e variáveis de ambiente. As diretivas de compilação do padrão OpenMP podem ser definidas como linhas de código diferenciadas, que tem o objetivo de informar ao compilador como deve ser realizada a paralelização de um determinado trecho de código.

Na linguagem de Programação C++, que foi a linguagem utilizada para o desenvolvimento da aplicação apresentada neste trabalho, a paralelização através do OpenMP ocorre através da utilização da diretiva de compilação #pragma omp. Esta diretiva deve ser definida logo acima do código que se deseja paralelizar.

Para que o compilador do C++ possa realizar corretamente a paralelização dos códigos marcados com as diretivas de compilação do OpenMP deve-se adicionar a opção -fopenmp ao seu comando de execução. Além disso, é necessário a importação da biblioteca "omp.h" para que seja possível utilizar as funções e as diretivas de compilação existentes na API do OpenMP.

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

A paralelização através de OpenMP foi utilizada neste trabalho através da diretiva de compilação "#pragma omp parallel for" para paralelizar o laço de repetição responsável pela avaliação da população gerada através do algoritmo genético utilizado.

Resultados e discussão

Para a realização dos testes e a coleta de resultados para o estudo de caso abordado neste trabalho foram utilizados: um tabuleiro de 128X128 (capacidade para 128 rainhas), população com 120 indivíduos e definido 5000 iterações. Segue abaixo a exibição dos resultados obtidos.

Sequêncial

Threads	Teste1	Teste2	Teste3	Média
1	83202.23 ms	83145.61 ms	83984.61 ms	83444.15 ms

Pthread

Threads	Teste1	Teste2	Teste3	Média
2	112914.80 ms	111533.48 ms	111747.01 ms	112065.1 ms
4	84391.32 ms	85007.45 ms	86083.19 ms	85160.65 ms
6	91306.86 ms	91641.80 ms	91763.69 ms	91570.78 ms
8	85769.31 ms	82212.87 ms	86247.39 ms	84743.19 ms

OpenMP

Threads	Teste1	Teste2	Teste3	Média
2	61249.57 ms	67713.92 ms	65786.19 ms	64916.56 ms
4	61128.40 ms	61213.21 ms	60840.31 ms	61060.64 ms
6	74852.33 ms	74750.94 ms	74727.85 ms	74777.04 ms
8	66250.70 ms	64511.58 ms	66184.75 ms	65649.01 ms

Fazendo uma análise dos dados extraídos através dos testes realizados, pode se perceber que ao utilizar a biblioteca OpenMP, conseguiu-se atingir os melhores resultados de desempenho da aplicação. A implementação sequencial obteve o segundo melhor desempenho e a implementação fazendo uso da biblioteca Pthread foi a que obteve o pior desempenho. Um fato interessante a ser destacado, e que talvez possa justificar o baixo desempenho da biblioteca Pthread é que, ao executar o algoritmo sequencial, o mesmo utilizava 100% de um núcleo, o OpenMP utilizava próximo de 80% de cada núcleo, já o Pthread utilizava em torno de 60% de cada núcleo alocado.

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

Conclusões

Ao concluir o trabalho e analisar os resultados dos testes que foram realizados, pode-se perceber a capacidade de desempenho superior do algoritmo que foi implementado utilizando a biblioteca OpenMP, embora ambos os algoritmos paralelos possam ser melhorados ainda mais visando otimizá-los para uma determinada arquitetura multi-core, e talvez assim possam chegar a resultados semelhantes. Pode-se perceber também no presente trabalho, o quanto as bibliotecas aqui estudadas auxiliam na prática da programação paralela, principalmente a OpenMP, a qual se encarrega da maior parte da programação paralela, bastando ao programador definir apenas qual o trecho de código que o mesmo deseja paralelizar que a API se encarrega de criar as threads e sincronizá-las.

Palavras-Chave: Processamento Paralelo, OpenMP, PThread, Algoritmos Genético, N Rainhas.
Referências Bibliográficas

CALDAS, José A. e SENA, Maria C. Tutorial OpenMp C/C++. Anais eletrônicos. [Maceió, AL], 2008. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/erad-rs/2010/006.pdf>. Acesso em: 06 jun. 2014.

EDWARD A. Lee, The Problem with Threads. Anais eletrônicos. [Berkeley, CA], 2006. Disponível em : <http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.pdf >. Acesso em: 06 jun 2014

LAGUNA, M. A Guide to Implementing Tabu Search. Investigacion Operativa, 4:159-178, 1994.

LINDEN, Ricardo. Algoritmos Genéticos. 3ª ed. Ciência Moderna, 2012.

YOLINUX. Site. POSIX Thread (pthread) libraries. Disponível em: <http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html>. Acesso em: 06 jun 2014