Evento: XXVI Seminário de Iniciação Científica

COMPARAÇÃO DE DESEMPENHO NA PROGRAMAÇÃO PARALELA HÍBRIDA (MPI + OPENMP) NA BUSCA DE TEXTO EM ARQUIVOS[1]
COMPARISON OF PERFORMANCE IN HYBRID PARALLEL PROGRAMMING (MPI + OPENMP) IN SEARCH OF TEXT IN FILES

Matthias Ruben Moeller Trennepohl[2], Edson Luiz Padoin[3], Lori Ronaldo F Machado Filho[4]

[1] Trabalho Realizado no Curso de Ciência da Computação
[2] Aluno do curso de Ciência da Computação da Unijuí
[3] Regional University of Northwest of Rio Grande do Sul (UNIJUI) - Brazil Federal University of Rio Grande do Sul (UFRGS) - Brazil Laboratoire d?Informatique de Grenoble (LIG) - France
[4] Aluno de Pós-Graduação em Engenharia de Software para Dispositivos Móveis pela UNINTER

Resumo

TEMA
O artigo apresenta uma comparação entre dois algoritmos paralelos para busca de palavras dentro de um arquivo de texto.
OBJETIVO e METODOLOGIA
O objetivo é analisar os ganhos de desempenho com o uso da programação paralela Híbrida através do uso de MPI para memória distribuída e OpenMP para memória compartilhada.
RESULTADOS
Os resultados iniciais em sistemas de pequeno porte apresentaram ganhos de desempenho de até 3 vezes se comparado com a versão sequencial.

Introdução

A busca por alto desempenho em rotinas tem fomentado a geração de trabalhos relacionados à processamento paralelo e distribuído, deixando evidenciada sua positividade nos resultados pelos trabalhos de [Garcia et al.2013], [de Oliveira Gressler and Cera 2014] e [Maciel and Duarte 2016]. A programação paralela divide o trabalho que seria realizado por um único processo em diversos processos que podem ser executados concorrentemente em diferentes unidades de processamento, sejam processadores ou cores.

Para a programação de algoritmos paralelos diferentes tecnologias podem ser utilizadas. Algumas são desenvolvidas almejando ganhos de desempenho em ambientes de memória compartilhada, como por exemplo OpenMP. Outras, por sua vez permitem a utilização de recursos distribuídos, como é o caso da biblioteca MPI.

O objetivo deste trabalho é comparar o desempenho de um algoritmo paralelo utilizando apenas memória distribuída, e nova impementação de programação paralela híbrida que utiliza memória

distribuída com memória compartilhada.

Ele está organizado da seguinte forma. A Seção 2 discute o estado da arte das tecnologias de programação a serem utilizadas e os trabalhos relacionados. A Seção 3 descreve a metodologia e o ambiente utilizado na implementação e execução dos testes. Resultados alcançados são discutidos na Seção 4, seguidos das Conclusões.

Programação Paralela

Diferentes abordagens podem ser utilizadas para dividir o processamento de uma aplicação. Uma abordagem é uso de threads que compartilham memória e outra é por meio de processos quando sistema paralelos possuem memória distribuída. Estas abordagens são discutidas nas seções seguintes.

Memória Compartilhada

Open Multi Processing (OpenMP) é uma tecnologia de programação paralela que permite dividir o trabalho em threads. Esta disponível para diversas linguagens, incluindo a linguagem C.

OpenMP é uma tecnologia simples de ser usada. Para paralelizar um código é necessário adicionar ao código textit{#pragma} de paralelização do OpenMP, informando ao compilador as partes que podem ser executadas em paralelo.
Os ganhos podem ser de N vezes no tempo de execução (ou N vezes de textit{speed-up}) quando um programa paralelizado utilizando OpenMP é executado em um processador de N cores.[Chapman 2008]

Memória Distribuída

Message Passing Interface (MPI) é uma tecnologia de programação paralela que permite dividir o trabalho em processos que são executados em diferentes computadores.
No padrão MPI, uma aplicação é constituída por uma ou mais processos dependendo da implementação utilizada. Estes processos se comunicam, através de chamadas de funções para o envio e recebimento de mensagens. Na grande maioria de implementações, um conjunto de processos é criado, porém, cada processo pode executar um programa diferente, por isso o padrão MPI muitas vezes é referido como MPMD (multiple program multiple data).

Algo muito importante em programações paralelas é a comunicação de dados entre processos paralelos e o balanceamento da carga, que seria uma forma de dizer quanto cada processo irá "trabalhar". Os processos podem usar mecanismos de comunicação ponto a ponto ou comunicação em grupo.[Willian Gropp 1999]

O grande objetivo de MPI é prover um amplo padrão para escrever programas com passagem de

mensagens de forma prática, portátil, eficiente e flexível.

Trabalhos Relacionados

A procura por algoritmos de alto desempenho para otimizações de processos tem gerado trabalhos que empregam técnicas de programação paralela e distribuída, como nos trabalhos de [de Oliveira Gressler and Cera 2014] e [Maciel and Duarte 2016]. Contudo, paralelizações relacionados à busca em arquivos de texto pode tem maior foco nos textos de [Gomes and Medina 2016] e [da Silva Júnior et al. 2016].

[de Oliveira Gressler and Cera 2014] e [Maciel and Duarte 2016] são trabalhos relacionados à computação paralela e distribuída em geral. Nesses textos é evidenciado os resultados positivos provenientes da programação de alto desempenho. Em relação aos resultados obtidos por [de Oliveira Gressler and Cera 2014], sua paralelização com OpenMP utilizando 4 threads gerou um speedup de 2,42 em relação ao algoritmo sequencial para a solução do Problema do Roteamento de Veículos (PRV) aplicando algoritmo genético. Já [Maciel and Duarte 2016], realizou multiplicação de vetores com GPU, obetendo entre seus resultados um speedup de 56,863 em um vetor de 10.000.000 de posições.

Partindo para a especificidade de busca em arquivos texto, o trabalho de [Gomes and Medina 2016] visa paralelizar a busca por indícios de plágio em arquivos acadêmicos. Nele é empregrado a arquitetura Parallel Miss Marple [Arenhardt et al. 2013] aplicando o uso de threads e Java RMI. É relatado um speedup de 14,408 com 24 threads em execução distribuída para duas máquinas.

[da Silva Júnior et al. 2016] proprõem a paralelização de buscas rápidas por palavras de exata similaridade fazendo uso do OpenMP. Em seu trabalho, foram utilizadas palavras com tamanhos entre 10 e 100 caracteres, obtendo um melhor resultado com palavras do tipo DNA (Ácido Desoxirribonucléico), com um speedup de 3,512.

Portanto, este trabalho será fundamentado nas técnicas de otimização utilizadas pelos autores. Para isso, foi aplicado o uso da
tecnologia OpemMP similar ao trabalho de [da Silva Júnior et al. 2016] em conjunto com a computação distribuída do MPI na linguagem de programação C, análogo à ideia de [Gomes and Medina 2016] , onde utiliza RMI.

Metodologia

Nossos experimentos foram realizados em uma plataforma composta de um processador Intel i7-7700 Kaby Lake 7a Geração com 4 núcleos físicos e 4 núcleos virtualizados, que totaliza em 8 núcleos de 3.6GHz.
A plataforma executa em uma máquina virtual com um sistema operacional Ubuntu Linux 18.04 com o kernel 4.15.0-23 que foi virtualizada para ter 4 núcleos.
Todos os aplicativos foram compilados com o compilador GCC versão 2.0.

Evento: XXVI Seminário de Iniciação Científica

Resultados

Nas Figuras 1 e 2 são apresentados os tempos de execução alcançados quando executado a aplicação com MPI.
Os resultados representam uma média das 10 execuções e o speed-up(ganho) comparado com as execuções sequenciais.

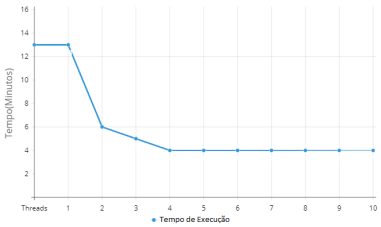

Figura 1. Tempo de Execução com MPI

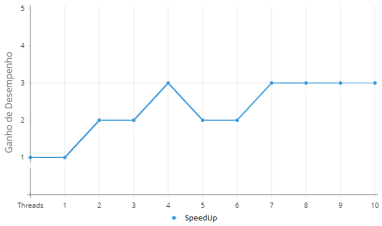

Figura 2. Ganho de Desempenho(SpeedUp) com MPI

Observa-se que o melhor resultado alcançado foi com 4 processos. Isto é justificado, uma vez que o processador utilizado possui apenas 4 núcleos. Assim, nas execuções com o aumento de processos teve-se perda de desempenho.

Por outro lado, quando utilizado programação paralela mista (MPI+OpenMP) para executar a aplicação,
como apresentado nas Figura 3 e 4, obteve-se ganhos com até 10 processos.

Isto se justifica porque a Progamação Paralela Híbrida (MPI + OpenMP) consegue dividir mais o trabalho, aproveitando mais cada core, fazendo com que cada core fique com menos trabalho, e assim tendo-se um melhor ganho de desempenho.

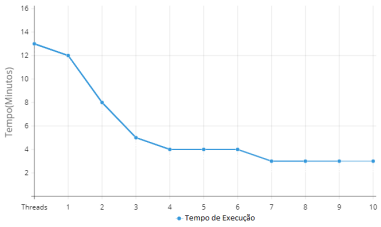

Figura 3. Tempo de Execução com MPI + OpenMP

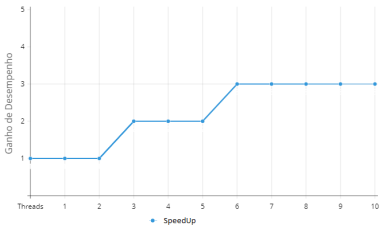

Figura 2. Ganho de Desempenho(SpeedUp) com MPI + OpenMP

Conclusão

Os resultados alcançados, foram conforme o esperado, sendo que a execução com a programação paralela mista alcançou os resultados esperados em menor tempo, e teve um speed-up melhor.

Assim possibilitando concluir que para a busca de um grupo de caracteres em um arquivo de texto, vale a pena ser utilizada uma programação paralela híbrida com MPI e OpenMP ao invés de uma programação paralela somente com MPI para se ter um melhor ganho de tempo.