Utilização de OpenMP na execução do aplicativo Concorde em arquiteturas com memória compartilhada

Guilherme Piegas Koslovski[1], **Gustavo Foletto Mora**[1], **Andrea Charão**[1], **Benhur de Oliveira Stein**[1]

[1]Laboratório de Sistemas de Computação (LSC)
Universidade Federal de Santa Maria (UFSM)
Campus UFSM – 97105-900 – Santa Maria – RS – Brasil

{guilherm, foletto, andrea, benhur}@inf.ufsm.br

Resumo. *Este trabalho descreve uma alteração realizada no aplicativo Concorde TSP Solver, utilizado para cálculo de soluções do problema do caixeiro viajante. Essa caracterização inclui funções e diretivas da API OpenMP, que auxilia no processo de paralelização de código para execução em arquiteturas multiprocessadas, utilizando memória compartilhada. Ao longo do texto, discute-se a análise e as alterações realizadas no código, apresentando-se alguns testes realizados.*

1. Introdução

O desafio de encontrar soluções exatas para o problema do Caixeiro Viajante é comumente referenciado na computação, por ser um problema de complexidade NP-completo, que normalmente demanda um elevado poder computacional.

Diversas implementações que solucionam este problema têm sido utilizadas como *benchmarks* de arquiteturas computacionais. Dentre os aplicativos existentes, Concorde TSP Solver destaca-se por ser disponibilizado com código aberto, e por possuir um elevado número de funções (aproximadamente 700) que permitem ao usuário selecionar o método de solução que será utilizado.

Este trabalho apresenta uma alteração do código original do aplicativo Concorde para execução em arquiteturas multiprocessadas. Para realizar esta alteração, utilizou-se a API OpenMP, que oferece diretivas de compilação e uma biblioteca contendo funções auxiliares para este fim. Através da utilização de OpenMP, é possível paralelizar um código seqüencial de forma rápida, aplicando diretivas de compilação que identificam um fluxo paralelo. [Wilkinson, B. e Allen, M. 1999]

O restante deste trabalho está organizado da seguinte maneira: a seção 2 descreve o problema do caixeiro viajante. Já a seção 3 apresenta o aplicativo Concorde TSP Solver e suas particularidades. A seção 4 apresenta a análise e as alterações que foram efetuadas no código. Posteriormente, na seção 5, descreve-se os testes e os resultados obtidos com a execução do código alterado. Por fim, a seção 6 apresenta as considerações finais.

2. Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) [Wikipédia 2007] é um conhecido problema de otimização combinatória, com complexidade NP-completo [Lewis, H. R. e Papadimitriou, C. H. 2000]. Basicamente, sua descrição

pode ser vista da seguinte forma: um caixeiro viajante deseja visitar um certo número de cidades, passando apenas uma vez em cada cidade, e ao término das visitas, retornar ao ponto de partida. No cálculo de sua rota de viagem, busca-se percorrer a menor distância possível.

3. O aplicativo Concorde

3.1. Informações Gerais

Dentre as implementações existentes para solucionar o PCV, o aplicativo Concorde TSP Solver [Cook 1997] é uma solução escrita utilizando a linguagem de programação C, que oferece algumas particularidades como a possibilidade de implementação de novas funcionalidades e a concordância com um padrão de arquivos de dados para cálculo do problema.

3.2. Estrutura do código original

O código original do aplicativo Concorde possui aproximadamente 60000 linhas implementadas em sua parte principal (cálculo de PCV e funções de troca de dados). Esta implementação compreende diversas funções para cálculo do problema, utilizando diferentes abordagens, permitindo a execução de uma forma adaptável.

Originalmente, o aplicativo Concorde possui um algoritmo de distribuição de trabalho entre computadores interconectados. Neste mecanismo, a árvore de derivação do cálculo é dividida e distribuída, permitindo que a carga de processamento seja distribuída entre outros computadores. Esta distribuição ocorre através de comunicação utilizando *sockets*, e comunicação entre processos trabalhadores utilizando um sistema de compartilhamento de arquivos e diretórios.

4. Alteração da implementação original

Para a alteração da implementação original do aplicativo Concorde em arquiteturas multiprocessadas, realizou-se inicialmente uma análise do código, buscando identificar quais os principais pontos de processamento. Para isso, utilizou-se uma versão compilada do aplicativo com suporte ao GNU gprof [Graham, S. L., Kessler, P. B., e McKusick, M. K. 1982].

Na análise do *profile* obtido, observou-se que o maior tempo gasto era nas funções de cálculo do problema propriamente dito. As funções de cálculo são extensas e possuem um alto nível de dependência de dados, o que dificultou o processo de paralelização utilizando OpenMP.

Em uma segunda análise, identificou-se que o gargalo principal do aplicativo era realmente capacidade de processamento, e que a quantidade de memória não constituía um obstáculo, mesmo em problemas com um elevado número de cidades. Desta forma, uma utilização eficiente dos processadores existentes na arquitetura deveria resultar em ganho de desempenho.

Assim, identificou-se qual a função que realizava a inicialização dos processos trabalhadores, e inicializou-se uma para cada processador existente na arquitetura. Esta inicialização é realizada dinamicamente, independente do número de processadores existentes, permitindo que a versão obtida seja utilizada em plataformas com diferente número de processadores.

Nos primeiros testes realizados, notou-se que o desempenho ainda podia ser melhorado. Originalmente no aplicativo Concorde, quando um processo em execução precisa de uma informação ainda não disponível, ele entra em estado de espera, através da realização de chamadas de testes de disponibilidade e, caso não disponível, realizando uma chamada de espera no fluxo. Desta forma, perde-se tempo de processamento em processadores que tenham fluxos neste estado.

Para contornar este obstáculo, foram realizadas alterações no mecanismo de espera do aplicativo, alterando de chamadas com espera ocupada para chamada *sleep*.

5. Testes realizados

5.1. Ambiente de Experimentação

Para testar a eficiência das alterações realizadas no código do aplicativo, elaborou-se um ambiente de experimentação composto por três computadores. Um computador com processador Pentium 4 2.40GHz com 512MB de memória RAM, e dois computadores biprocessados Xeon Quad-Core 2GHz, com 2GB de memória RAM.

5.2. Resultados

Para execução dos testes, selecionou-se um arquivo de dados, contendo 662 cidades, utilizando o padrão TSPLIB. Quanto as opções de configuração do aplicativo Concorde, foram utilizadas as opções padrão, alterando-se apenas as opções para cálculo distribuído, tanto para a versão original, como para a modificada. Foram realizadas 50 repetições para cada ponto e posteriormente calculou-se a média.

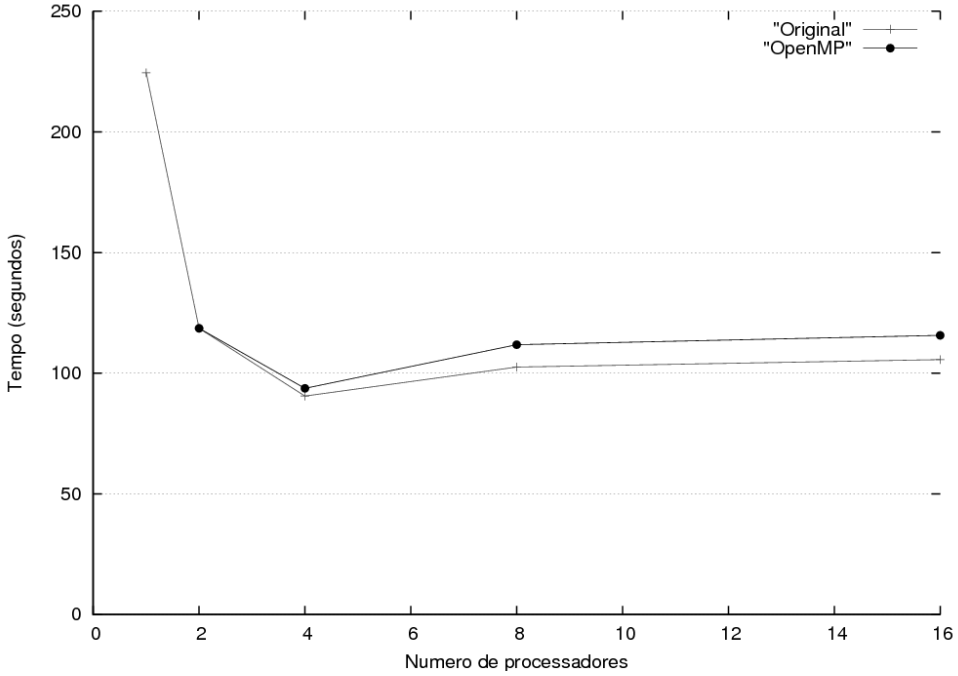

Figura 1. Tempo de cálculo para 662 cidades

Em um primeiro teste, comparou-se o desempenho da versão original do programa com a versão alterada, utilizando-se uma máquina bi-processada. O aumento de desempenho obtido pareceu promissor. Resolveu-se então realizar um teste em maior escala, comparando-se a versão alterada com o OpenMP, com a versão original, utilizando-se a opção de execução distribuída, para ocupar as CPUs disponíveis. Para a utilização de mais CPUs com a versão original, foram lançados vários processos, cada um ocupando um núcleo.

A figura 1 apresenta o segundo teste realizado, com o tempo de execução da aplicação em segundos. A comparação realizada é entre o código original do Concorde variando o número de processos, e o código alterado variando o número de threads.

Observa-se nesta figura que comparando a execução sequencial e as execuções em paralelo utilizando OpenMP, foi obtido uma diminuição dos tempos de execução. Porém em todos os testes comparando-se o número de processadores, a versão com OpenMP obteve um desempenho pior do que a versão original do concorde.

Um possível motivo para a perda de desempenho seria a comunicação entre threads que é utilizada.

6. Considerações Finais

Este trabalho descreveu a realização de uma alteração no código original do aplicativo Concorde, utilizado para cálculo de soluções exatas do problema do caixeiro viajante. As alterações foram realizadas utilizando a API OpenMP, incluindo algumas diretivas de compilação que permitiram a execução da aplicação em máquinas biprocessadas. Os testes realizados apresentaram gráficos e números que ajudam a observar que por mais que tenha sido obtido um ganho em relação a versão original sequencial, não foi suficiente para superar o ganho da versão original paralela, comparando o número de processadores utilizados. Esta esperiência reforça a dificuldade da paralelização de programas complexos utilizando o OpenMP.

Como trabalhos futuros, sugere-se realizar uma análise nas complexas funções de cálculo, realizando primeiramente uma alteração em suas estruturas, e assim possibilitando o uso das diretivas OpenMP. Posteriormente, será possível diminuir os trechos de código seqüenciais executados.

Referências

Cook, W. (1997). Concorde TSP Solver. http://www.tsp.gatech.edu/concorde.html.

Graham, S. L., Kessler, P. B., e McKusick, M. K. (1982). gprof: a Call Graph Execution Profiler. In *SIGPLAN Symposium on Compiler Construction*, pages 120–126.

Lewis, H. R. e Papadimitriou, C. H. (2000). *Elementos de Teoria da Computação*. Prentice Hall.

Wikipédia (2007). Problema do Caixeiro Viajante. http://pt.wikipedia.org/wiki/Problema_do_caixeiro_viajante.

Wilkinson, B. e Allen, M. (1999). *Parallel Programming: Techniques and Applications Using Workstation and Parallel Computers*. Prentice Hall.