Paralelização e avaliação de desempenho de um simulador de dispersão de poluentes na atmosfera

Bruno C. de Christo[1], **Andrea S. Charão**[1], **Débora R. Roberti**[2]

[1]Laboratório de Sistemas de Computação
Universidade Federal de Santa Maria (UFSM) - Santa Maria, RS - Brazil

[2]Departamento de Física
Universidade Federal de Santa Maria (UFSM) - Santa Maria, RS - Brazil

{bchristo,andrea}@inf.ufsm.br, d_r_roberti@yahoo.com.br

Resumo. *O programa LAMBDA é utilizado em meteorologia para realizar o cálculo da concentração de partículas lançadas aleatoriamente sobre uma grade, para simular diversos efeitos atmosféricos. Este trabalho tem por objetivo descrever a paralelização do programa LAMBDA, bem como avaliar o desempenho de sua versão paralela.*

1. Introdução

Na área de meteorologia existe a necessidade de simulação de diversos fenômenos atmosféricos. Um estudo recorrente é a simulação do comportamento de partículas que são liberadas na atmosfera. Para se realizar esta simulação, programas como o LAMBDA [Roberti 2004] são utilizados. O LAMBDA calcula as coordenadas no espaço de cada partícula a cada passo no tempo, bem como a concentração de partículas no espaço após um certo número de iterações. É um programa usado em atividades de pesquisa, em que se testam alterações no modelo antes destas serem aplicadas em programas operacionais da mesma natureza. O cálculo da concentração é relativamente rápido, mas o cálculo das coordenadas das partículas, dependendo da quantidade das mesmas, pode ser bastante demorado. Notou-se que o cálculo de coordenadas é um cálculo altamente paralelizável, então neste trabalho tentou-se paralelizar essa porção do código com o fim de reduzir o tempo total de execução. O restante do texto discute a paralelização do LAMBDA (seção 2) e apresenta resultados da análise de desempenho realizada (seção 3).

2. Paralelização do LAMBDA

Esta seção aborda 3 aspectos da paralelização do programa LAMBDA, compreendendo a análise dos códigos iniciais, a divisão das tarefas e o gerador de números aleatórios.

2.1. Análise dos Códigos Iniciais

Este trabalho iniciou com o estudo de uma versão seqüencial e duas versões paralelas do programa LAMBDA [Roberti 2004], que foi escrito em Fortran, assim como um arquivo de entrada de exemplo para o cálculo das concentrações. As versões paralelas, no entanto, eram de uma versão anterior de testes do LAMBDA, onde as variáveis de entrada estavam embutidas no código-fonte. Os códigos foram estudados e compilados no *cluster* do Laboratório de Micrometeorologia da UFSM, que possui 15 máquinas Pentium IV de 2.4GHz e 512MB de RAM. Para compilar os programas, foi utilizado o compilador

Portland, e a biblioteca para o código paralelo MPICH. Após a compilação dos códigos, foram efetuadas diversas medições de tempos de execução utilizando 1, 2, 4 e 8 processadores. A Análise 1 consistiu na análise do primeiro código seqüencial, e a Análise 2 consistiu na análise do segundo código seqüencial. Depois disso, foi analisada a relação desempenho / número de nós, assim como os tempos de execução dos códigos. Posteriormente, calculou-se a aceleração (*speedup*) obtida, dada pela relação entre o tempo de execução seqüencial e o tempo em paralelo com P processadores.

2.2. Divisão das Tarefas

Com o intuito de paralelizar a versão final do LAMBDA, analisou-se o segundo código paralelo do LAMBDA de testes. A estratégia de paralelização consistiu na adaptação de alguns trechos de código, como a divisão do cálculo da próxima posição de cada partícula pelos N nós que irão executar o programa, e as reduções no final do mesmo. Também foram feitos alguns testes nesta versão para determinar o ganho de desempenho em relação ao seqüencial. 3).

2.3. Gerador de números aleatórios

Na versão seqüencial do LAMBDA, existe um gerador de números aleatórios que são usados para calcular as coordenadas da próxima posição de cada partícula. Isto gera um problema, já que se utilizarmos a mesma sequência em cada nó na versão paralela, estaremos realizando o mesmo cálculo de posição em cada nó, obtendo assim várias partículas sobrepostas [Foster 1995]. Como cada nó deverá ter a sua seqüência aleatória para evitar esse problema, um novo problema está sendo criado: o resultado da simulação será sempre semelhante ao seqüencial (na melhor das hipóteses) mas nunca igual, o que torna uma comparação dos códigos seqüencial e paralelo com alto grau de precisão muito difícil. Por enquanto, a versão seqüencial do gerador aleatório de partículas está sendo usada no programa paralelo, mas futuramente pretende-se substituí-la por um gerador de números aleatórios paralelo.

3. Resultados

Tanto o desempenho da primeira análise (com 4 nós) no Jumpshot [LANS 2008] quanto o speedup da primeira e da segunda versão do código são apresentados (Figura 1 e 2). Em relação à nova versão paralela do LAMBDA, O desempenho de vários trechos de código foi analisado: Tempo de cálculo da posição das partículas, de comunicação, de entrada / saída e o tempo total. Foram realizadas 15 medições, desde 1 núcleo até 15 rodando simultaneamente (Figura 3). Como se pode ver neste gráfico, o tempo de E/S ficou praticamente constante, sendo o principal gargaloo tempo de comunicação à medida que se aumenta o número de nós. A concentração também foi calculada, com um outro programa que calcula a concentração a partir do *output* do LAMBDA (Figura 4).

4. Conclusão

O speedup da primeira versão foi muito baixo, indicando que existia algum problema no ambiente de execução ou no código. A operação MPI_REDUCE era o gargalo da execução, e por isso, quanto mais nós, mais tempo a execução estava demorando. Na nova versão do código, o problema estava corrigido, já speedups maiores que 1 foram obtidos nas execuções paralelas. Como se pode observar no gráfico de desempenho do novo

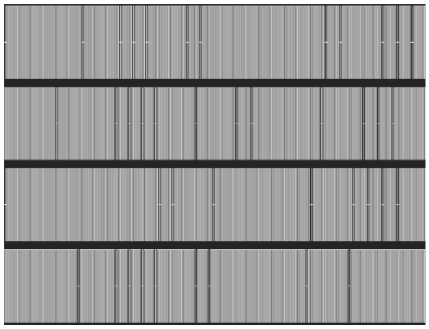

Figura 1. Análise 1 no Jumpshot, com 4 nós. Operações MPI_BROADCAST (pontos brancos) e MPI_REDUCE (linhas claras). Para referência, o tempo de execução em 4 nós foi de 55.67 segundos, enquanto de um único nó foi de 6.50 segundos.

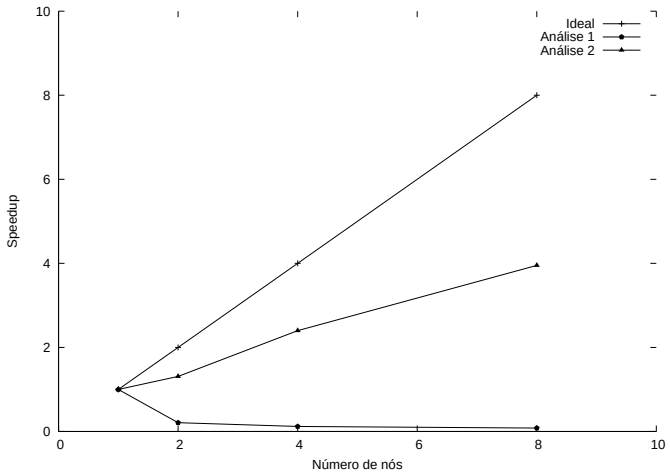

Figura 2. Speedup das análises 1 e 2 de cada código paralelo inicial.

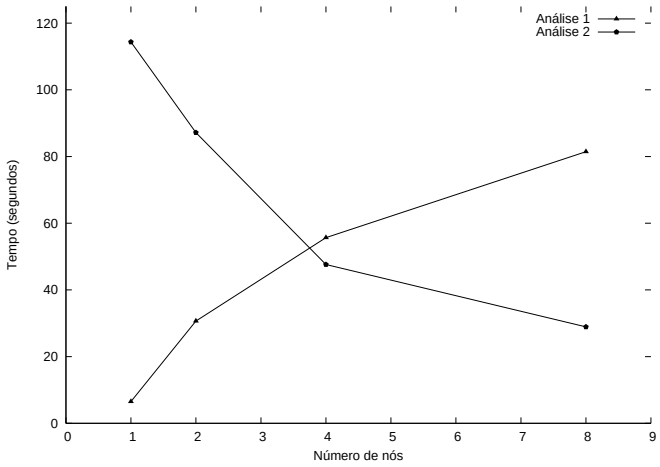

Figura 3. Tempo de execução detalhado do novo LAMBDA paralelo. Dividido em tempo de cálculo, comunicação, entrada/saída e total.

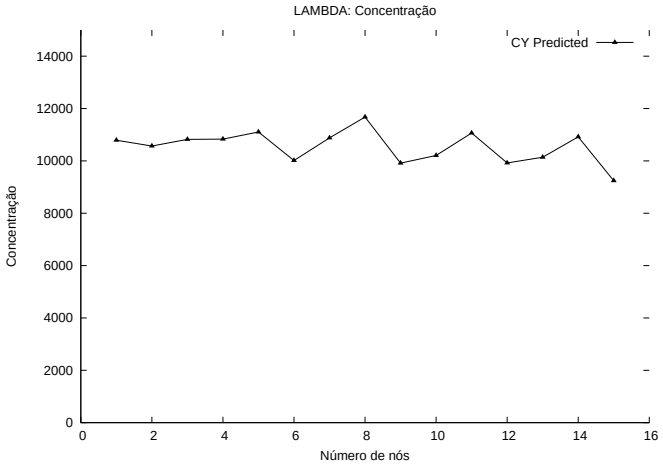

Figura 4. Concentração de partículas pelo número de nós no LAMBDA paralelo novo.

código, o ideal é executar este código entre 6 a 8 processadores simultaneamente neste cluster, já que com mais do que 8 processadores o desempenho cai devido ao aumento da comunicação entre os núcleos [Wilkinson 1998]. Planeja-se reimplementar o gerador de números aleatórios em paralelo, utilizando um número multiplicado pelo ID do núcleo, por exemplo, de modo a não obter as mesmas coordenadas em cada núcleo. Também se pretende realizar novamente parte do experimento de Copenhagen [Gryning 2002] no novo LAMBDA paralelo, para se determinar se o paralelismo foi bem sucedido por completo, e então projetar uma nuvem tridimensional com a dispersão das partículas, como na figura 1 do artigo referenciado [Roberti 2004].

Referências

Foster, I. (1995). Designing and building parallel programs.

Gryning, S.; Lyck, E. (2002). The copenhagen tracer experiments: Reporting of measurements. Risø National Laboratory, Roskilde, Denmark.

LANS (2008). Jumpshot: Performance visualization for parallel programs. Site: http://www-unix.mcs.anl.gov/perfvis/software/viewers/index.htm. Acessado em Janeiro/2008.

Roberti, D. e. a. (2004). Parallel implementation of a lagrangian stochastic model for pollution dispersion. Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing.

Wilkinson, B.; Allen, M. (1998). Parallel programming: techniques and applications using networked workstations and parallel computers.