Implementação do algoritmo Friends of Friends de complexidade n*logn(n) para classificação de objetos astronômicos

Otávio Migliavacca Madalosso[1], Andrea Schwertner Charão[1], Haroldo Fraga de Campos Velhos[2], Renata Ruiz[2]

[1]Laboratório de Sistemas de Computação – Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brasil

[2]Instituto Nacional de Pesquisas Espaciais – São José dos Campos – SP – Brasil

{andrea,omadalosso}@inf.ufsm.br, {haroldo,renata}@lac.inpe.br

Resumo. *Este artigo apresenta o processo de implementação de uma versão de complexidade n*log(n) do algoritmo Friends-of-Friends. Essa versão é implementada baseada no algoritmo de Barnes-Hut com uma nova verificação para as fronteiras dos semi-cubos do algoritmo. Apresenta também uma comparação de tempo de execução e dos resultados em relação ao algoritmo original, de complexidade* n^2.

1. Introdução

Simulações de n-corpos têm sido utilizadas para promover vários avanços na compreensão de questões relevantes em astrofísica, como por exemplo o processo de formação e evolução de estruturas do Universo. Este tipo de simulação tem um papel fundamental no estudo da evolução cósmica em tópicos como a distribuição de matéria escura em grande escala, a formação de halos de matéria escura, e a formação e evolução de galáxias e aglomerados.

A manipulação e análise da grande quantidade de dados produzidos em simulações também é algo desafiador. Neste contexto, é essencial o desenvolvimento de técnicas computacionais eficientes para extrair informação significativa a partir dessas fontes de dados, em um período apropriado de tempo. Para isso, foi observada a hipótese de melhorar o algoritmo atual, buscando uma complexidade menor para a classificação, o que deve causar um impacto considerável na diminuição de tempo de execução.

O algoritmo FoF tem um alto custo computacional, com complexidade n^2. Nas simulações de n-corpos da atualidade, são usados bilhões de partículas, portanto para aplicar o algoritmo FoF é necessário melhorar seu desempenho. No trabalho de Ruiz et al., apresentou-se uma versão paralela do algoritmo FoF, alcançando uma aceleração significativa quando executada em um cluster com 27 computadores. A implementação paralela, para esta arquitetura com memória distribuída, utilizou o padrão MPI (Message Passing Interface) para comunicação entre os computadores do cluster.

Alguns autores sugerem que o custo computacional deste tipo de método possa ser reduzido com um algoritmo de complexidade n*log(n). Por outro lado, há também que se considerar a hibridez das arquiteturas paralelas existente atualmente, oferecendo múltiplas oportunidades de paralelismo em diferentes dispositivos (por exemplo, clusters de computadores multicore e com GPUs como co-processadores).

2. Desenvolvimento

Para o desenvolvimento e compreensão do algoritmo desenvolvido foi necessário o estudo das seguintes estruturas de dados e casos de exceções.

2.1. Octa-Tree

A implementação da versão n*log(n) do algoritmo Friends of Friends foi dada através da reprodução de um espaço físico de 3 dimensões (representação de um cubo, no qual os corpos celestes serão alocados) através de uma Octa-tree (solução apresentada por J. Barnes e P. Hut). Para cada nó dessa Octa-Tree, podem haver até 8 filhos que representam cada sub-cubo com a metade do tamanho de seu pai, limitando o espaço de busca por outros corpos celestes e aumentando o desempenho do algoritmo.

Nessa implementação, o nó raiz da árvore contém os valores das extremidades do espaço tridimensional que compõe a entrada.

2.2. Agrupamento

O propósito do algoritmo é encontrar grupos de corpos que tem uma distância menor do que um determinado raio entre seus membros.

A verificação de distância entre dois corpos é feita durante a adição de um novo corpo na árvore, conforme ele se aprofunda na árvore, faz verificação entre os nós que são vizinhos do semi-cubo ao qual o novo corpo será alocado. Neste caso encontra-se um problema no que diz respeito a comparação de um corpo com um nó que não é populado, ou seja, um nó que não é folha, para esse caso será observado mais adiante a utilização de zonas de fronteira.

2.3. Zonas de fronteira

A zona de fronteira foi a forma escolhida para lidar com um caso que foi observado durante a implementação do algoritmo. Nesse caso, 2 grupos podem estar em semi-cubos diferentes e mesmo assim participarem do mesmo grupo por estarem ambos muito próximos aos limites físicos desses semi-cubos.

A necessidade de tratar esse caso resultou em um vetor de referências aos corpos que se encontram nos limites de cada semi-cubo. Esses limites são delimitados por uma diferença entre os limites de cada semi-cubo e o raio no qual o algoritmo está sendo executado. Essa solução gerou um novo caso a ser tratado posteriormente que lida com a situação de que essa zona de fronteira do cubo é maior ou igual que a área do próprio cubo em si, logo, todo o cubo pode ser considerado uma zona de fronteira e a árvore pode parar de crescer e simplesmente alocar todos os nós que ali estiverem em um novo vetor.

2.4. Função *Relabel*

Um caso que deve ser tratado durante a verificação de agrupamento é o de um corpo que está sendo adicionado já pertence a algum grupo, e durante a verificação o algoritmo descobre que ele já pertence a outro grupo, ou seja, esse novo corpo é um elo entre dois grupos que até o momento eram distintos.

Essa situação é resolvida através de uma função que efetua uma atualização dos identificadores em todos os objetos dos grupos para que os dois se unam e passem a ser

tratados como apenas um. Isso é feito através do vetor de grupos que o programa mantém, tornando mais fácil o processo de *relabel*.

Foi observado que o algoritmo original Friends-of-Friends de complexidade n^2 não trata esse tipo de comportamento, isso causou um atraso e confusão durante a implementação do algoritmo n*log(n), além de impossiblitar uma comparação precisa entre os resultados dos algoritmos (pois o resultado correto, obtido na versão em desenvolvimento, é diferente do resultado obtido pelo algoritmo original).

3. Comparação e Algoritmo Original

Durante o desenvolvimento do projeto, era necessária uma comparação entre os resultados esperados (obtidos via execução da versão original do algoritmo) e os resultados encontrados através da versão em desenvolvimento. Essa comparação mostrou resultados divergentes levantando suspeitas relacionadas ao funcionamento do algoritmo original. Observou-se que o algoritmo original não trata uma situação específica que causa uma perda de precisão de seu resultado final, uma vez que um corpo pertence a algum grupo, esse corpo nunca mais é verificado, o que exclui a verificação discutida na sessão Função *relabel*, que lida com o caso de um corpo ser o elo entre dois grupos diferentes. Isso torna possível que o algoritmo original gere um resultado incorreto dependendo do arquivo de entrada do problema.

4. Experimentação e Resultados

Para comparar o tempo de execução dos algoritmos foi selecionada uma amostra de dados na qual o algoritmo original não perde a precisão em seu resultado, pois foram excluídos os dados que geravam o erro.

Os experimentos foram realizados em um servidor com um processador Intel Xeon E5620 2,4Ghz, 12 GB DDR3 de RAM, SO Debian 6 (Squeeze). Os dados de entrada consistem em informações do posicionamento e identificação de 39761 corpos. O resultado esperado dos algoritmos é identificar quantos grupos com mais de um elemento existem na amostra.

	numero de Grupos	Tempo de Execução(ms)
Algoritmo Original	226	29496
Algoritmo Novo	226	149

Pode-se notar que a vantagem no tempo de execução é significativa, porém é importante observar também que o algoritmo novo contém propriedades em sua estrutura que podem gerar overheads dependendo da amostra de dados de entrada e da distância que será utilizada para o algoritmo considerar dois corpos próximos o suficiente para pertencerem ao mesmo grupo.

5. Conclusão

Com base nos resultados obtidos através da execução e medição do tempo de execução dos algoritmos, conclui-se que o ganho de desempenho e precisão dos resultados da versão n*log(n) pode trazer grandes benefícios para aplicações que utilizem essa nova versão.

O projeto agora pretende explorar o uso de arquiteturas paralelas para sistemas heterogêneos utilizando as interfaces OpenACC e OpenMP.

6. Referências

Huchra, J. P. e Geller, M., J. (1982). Groups of Galaxies I. Nearby groups. The astrophysical, v. 257, p. 423 - 437.

Ruiz, R. S. R. ; Campos Velho, H. F. ; Caretta, C. A. Paralelizacao do Algoritmo Friends- of-Friends para identificar Halos de Materia Escura. In: IX Workshop do Curso de Computacao Aplicada, 2009, Sao Jose dos Campos.

Vasconcellos, E. C. ; Carvalho, R. R.; Gal, R. R.; La Barbera, F. ; Capelato, H. V. ; Campos Velho, H. F. ; Trevisan, M. ; Ruiz, R. S. R. - Decision Tree Classifiers for Star/Galaxy Separation. The Astronomical Journal (New York, N.Y.), v. 141, p. 189, 2011.

Springel, Volker; Yoshida, Naoki; White, Simon D. M. GADGET: a code for collisionless and gasdynamical cosmological simulations. New Astronomy 6 (2001) 79 – 117.

J. Barnes and P. Hut (December 1986). "A hierarchical O(N log N) force-calculation algorithm". Nature 324 (4): 446–449