Área: Processamento Paralelo e Distribuído

Processamento paralelo aplicado à problemas de filogenia

Elias Batista FERREIRRA, Wellington Santos MARTINS,
Instituto de Informática (INF)
Universidade Federal de Goiás (UFG)
{eliasferreira, wellington}@inf.ufg.br

Palavras-chave : Computação Paralela, GP-GPU, CUDA, Filogenia

1. Introdução

Na Biologia, a construção de árvores filogenéticas é utilizada para explicar os possíveis relacionamentos entre as espécies atuais e seus possíveis históricos evolutivos - estas relações evolutivas podem facilmente ser visualizadas entre as espécies mostradas na árvore. A partir das sequências de DNA de uma espécie, é possível inseri-la na árvore com um grau de incerteza muito baixo sobre a posição que ela deve ocupar. No entanto, o que fazer quando não temos à disposição o DNA de todas as espécies que nos interessam? Especialmente os antepassados mais

longínquos. Motivos como à falta de pesquisadores, de recursos, interesse ou mesmo por uma espécie ser muito rara são justificativas para procurar alternativas à construção da árvore filogenética pelo DNA.

O uso de técnicas alternativas, como a analise de fenótipos, não permite ter certeza onde essa espécie se encaixa na árvore que temos. Apesar de não podermos dar esta garantia, que ocorre quando temos o DNA de um indivíduo dessa espécie, a análise de fenótipos pode nos dar uma boa ideia de onde ela, provavelmente, estará na árvore. E, através de cálculos estatísticos, podemos ter uma ideia da incerteza envolvida em cada possibilidade de lugar para esse espécie na árvore.

Quando tratamos árvores com poucas espécies, o tempo de processamento é aceitável, mas nem sempre é o caso. À medida que o número de espécies aumenta o problema cresce de forma exponencial, o que torna o problema NP-Completo. Diversas soluções na área de computação paralela são utilizadas para melhorar o speedup destes problemas.

O uso de clusters, computação distribuída e outros modelos de processamento paralelo são mecanismos comumente utilizados para resoluções de problemas relacionados a filogenia. Apesar de ser uma solução facilmente escalável torna-se cara e com grande dificuldade para sincronização e comunicação entre as threads.

Nesse trabalho, procuramos com o uso de GPGPU (General Purpose Graphical Processing Unit – Unidade de Processamento Gráfico de Propósito Geral) aproveitar a grande capacidade de processamento destas placas gráficas para conseguirmos um tempo de processamento aceitável na descoberta das relações evolutivas e classificação nas árvores filogenéticas. Além do que o uso destas placas reduz consideravelmente o gargalo existente nessa comunicação e sincronização dos processos.

2. Material e Métodos

A pesquisa foi conduzida de acordo com as seguintes etapas: pesquisa bibliográfica, estudo de algoritmos, implementação de algoritmos, testes das implementações, testes com dados reais, análise dos resultados, construção de soluções através de uma abordagem em GPGPU, escrita e apresentação de trabalhos.

A bibliografia base adotada para a implementação dos algoritmos é [1]. Como material de referência para a linguagem e para o ambiente de desenvolvimento em CUDA foram utilizados [4, 5, 6, 7]. Como material adicional para o estudo de filogenia foram utilizados [2,3].

No primeiro momento foi realizado estudo de todos os algoritmos apresentados em [1] bem como o estudo da justificativa teórica por trás dos algoritmos. Os seguintes algoritmos foram estudados e implementados em linguagem C/C++ (a notação Algoritmo X é usada como referência para o algoritmo X em [1]).

- Algoritmo 2: algoritmo para gerar uma imagem da árvore filogenética.
- Algoritmo 3: algoritmo para verificar se o problema é uma filogenia perfeita ou não.
- Algoritmo 4, 5, 6, 7, 8 e 10: usados na construção de uma filogenia perfeita.
- Algoritmo 9: gera uma string no formato Newick representando a árvore filogenética.
- Algoritmo 11: calcula a parcimônia de uma sub-árvore.
- Algoritmo 12: calcula a parcimônia de uma árvore filogenética.
- Algoritmo 13: verifica se uma determinada árvore é viável.
- Algoritmo 14: gera configurações iniciais para o problema.
- Algoritmo 15: gera configurações aleatórias para o problema.
- Algoritmo 16 e 17: constrói a árvore quando a filogenia não e perfeita.

A implementação da rotina de “heurística_serial” (algoritmo 16) não pode ser concluída porque ela utiliza outra rotina chamada “crossover_case” que não consta na tese ou na bibliografia ciada. Tentamos obter colaboração por parte do autor do algoritmo, porém não obtivemos sucesso.

A rotina “heurística_serial” é o último algoritmo da etapa sequencial do trabalho proposto por [1]. A saída desta rotina é utilizada como entrada para os módulos paralelos propostos pelo autor.

O algoritmo 1 é apenas um exemplo e não foi implementado. Os algoritmos 18, 19, 20, 21, 22 não foram implementados, pois se tratam de outra tecnologia de paralelismo, especificamente MPI.

Três algoritmos foram codificados em CUDA. Um deles é o algoritmo original, os outros dois são uma forma diferente de codificar a ideia apresentada por [1]. Além disso, os seguintes algoritmos originais precisaram ser codificados em CUDA por questões de compatibilidade:

- Algoritmo 11: calcula a parcimônia de uma sub-árvore.
- Algoritmo 12: calcula a parcimônia de uma árvore filogenética.
- Algoritmo 13: verifica se uma determinada árvore é viável.

Para o segundo momento, iremos utilizar a GPGPU e a linguagem CUDA para aceleram o processo de gerar as árvores filogenéticas, fazer os cálculos de matriz de distância (que podem ser utilizadas para cálculos estatísticos que determinam a relação filogenética entre as espécies), construir métodos randomização das árvores – que irá permitir realizar a permutação completa das árvores com base nas árvores filogenéticas parciais e nas melhores informações disponíveis.

Ao concluir estas etapas podemos averiguar se o uso de GPGPU traz benefícios, ou seja, ganhos de performance (speedup) nas construção de árvores filogenéticas. Os resultados obtidos deste processamento podem ser utilizados por biólogos para realizar análises da estrutura filogenética de comunidades e comparação entre características de espécies

3. Resultados e Discussão

Os algoritmos que constroem a árvore filogenética quando a entrada do problema é uma filogenia perfeita foram implementados e testados com os mesmos dados de [1], os resultados foram os mesmos.

Durante os estudos conduzidos sobre os algoritmos paralelos propostos por [1], constatamos que o algoritmo da forma como foi proposto não seria o ideal para se implementar em uma GPU. Isso é devido a uma estrutura rígida imposta pela organização das threads na GPU, esta estrutura é chamada de blocos.

Como não foi possível implementar a rotina que gera a entrada para o módulo paralelo, a implementação em GPU não pode ser comparada com a implementação original, pois este módulo é extremamente dependente do primeiro.

4. Conclusões

Durante a pesquisa percebemos que o algoritmo paralelo proposto e implementado em um cluster de computadores utilizado MPI por [1], não apresentava uma relação de 1-processo – 1-thread quando implementado em GPU usando CUDA. A conclusão a que chegamos é que mesmo se a “heurística_serial” tivesse sido implementada com sucesso, alterações seriam necessárias nos algoritmos paralelos propostos afim de utilizar o máximo potencial das placas gráficas, ou algo perto disso.

5. Referências.

[1] VIANA, G.V.R. **Técnicas para construção de árvores filogenéticas.** Fortaleza: UFCE, 2007.

[2] PINTEIRO, L. C., VIANA, G.V.R. Técnicas Algorítmicas para Construção de Árvores Filogenéticas. Ver. Cient. Fac. Lour. Filho. V.4. n1. 2005.

[3] AMORIM, D. S., Fundamento de Sistemática Filogenética. Holos. 2002.

[4] NVIDIA. CUDA Manual – Version

[5] NVIDIA. CUDA Manual – Version 4.0. Disponível em <http://developer.download.nvidia.com/compute/cuda/4_0_rc2/toolkit/docs/CUDA_Toolkit_Reference_Manual.pdf> Acesso em: 14/06/2011.

[6] NVIDIA. CUDA Programming Guide – Version 4.0. Disponível em http://developer.download.nvidia.com/compute/cuda/4_0_rc2/toolkit/docs/CUDA_Toolkit_Reference_Manual.pdf> Acesso em: 14/06/2011.

[7] NVIDIA. CUDA c, Best Practices – Version 4.0. Disponível em: http://developer.download.nvidia.com/compute/cuda/4_0_rc2/toolkit/docs/CUDA_C_Best_Practices_Guide.pdf> Acesso em: 14/06/2011.

[8] KIRK, B. D.; HWU, W .W. **Programando para Processadores Paralelos: uma abordagem prática à programação de GPU**. São Paulo: Ed. Campus, 2011.