inesc id lisboa

Recursos Humanos

RELATÓRIO DE ACTIVIDADES DE BOLSA DE INVESTIGAÇÃO

(A preencher pelo Bolseiro)

RELATÓRIO INTERCALAR ☐ RELATÓRIO FINAL ☒

Identificação do bolseiro

Nome completo: Marcelo de Jesus Pardal Vicente

Identificação da bolsa

Tipo de bolsa: Bolsa Integração na Investigação Referência: BII_2008

Período: De: 2008 - 10 - 31 a: 2009 - 10 - 31

Nome do projecto: Desenvolvimento e Análise de Algoritmos Distribuídos para Resolução de Problemas Dinâmicos Lineares

Área de trabalho: Engenharia Electrótecnica e de Computadores

Orientador científico: Prof. Luís Miguel Silveira

Actividades desenvolvidas

(Descreva sucintamente as principais actividades desenvolvidas e resultados obtidos, usando como referência o Plano de Trabalhos aprovado para o período).

Este trabalho teve como principal finalidade o desenvolvimento de algoritmos para a simulação de *powergrids* usando a placa gráfica como meio de processamento. Podendo as *powergrids* ser decompostas em sistemas de equações lineares, o trabalho desenvolvido baseia-se em usar métodos directos ou iterativos para a resolução dos mesmos. Estes métodos foram implementados usando a arquitectura CUDA (***C**ompute **U**nified **D**evice **A**rchitecture*), desenvolvida pela nVidia, integrada com C/C++ como ferramenta de programação para estabelecer a ligação com a placa gráfica.

Este trabalho foi dividido em várias fases, sendo as seguintes:

- Estudo sobre *powergrids*;
- Pesquisa e iniciação à arquitectura CUDA;
- Pesquisa sobre métodos iterativos para a resolução sistemas lineares;
- Implementação do gerador de *powergrids* e da estrutura de dados usados;
- Implementação do método de Jacobi;
- Implementação do método do Gradiente Conjugado;
- Implementação do método Multigrid (a ser concluído).

Nota: Todos os métodos aqui descritos foram implementados em GPU e CPU, usando precisão simples e dupla.

Esta bolsa foi iniciada em grupo, sendo o segundo elemento o João Teixeira Pinto. O estudo e pesquisa inicial foram feitos em conjunto e seguidamente foi feita a separação entre métodos iterativos e métodos directos. O último método está neste momento a ser implementado em conjunto.

(Continua na página 3)

(continuar em folhas adicionais, se necessário)

Desvios em relação ao planeado e respectiva justificação

(Deverão ser mencionadas as circunstâncias que tenham influenciado positiva ou negativamente o cumprimento do plano de trabalhos)

Neste momento só falta concluir a implementação do método Multigrid que já foi iniciada.

Como nunca se tinha trabalhado com arquitecturas de multiprocessamento a adaptação demorou algum tempo. No inicio da bolsa, eu e o meu colega perdemos algum tempo durante a pesquisa sobre CUDA para testar alguns programas e adaptarmo-nos à sua estrutura o que ocupou um pouco mais tempo do que era previsto.

Também durante a implementação dos primeiros dois métodos, muito tempo foi gasto a fazer *debugging*. Acontece que quando iniciamos a programação em CUDA, a nVidia ainda não tinha lançado nenhum *debugger* sendo necessário usar o "método das tentativas" para encontrar e corrigir os erros. Muito recentemente foi lançado a ferramenta *cuda-gdb* que é o *debugger* de CUDA e que permitiu aumentar a produtividade e reduzir o tempo nesse aspecto.

Publicações e trabalhos elaborados no âmbito da bolsa

(Caso se trate de uma bolsa para obtenção de grau ou diploma académico, juntar a este relatório uma cópia do respectivo trabalho final)

Neste momento todo o trabalho desenvolvido e resultados comparativos estão a ser colocados online.

Muito brevemente esta informação estará disponível na Wiki do Algos criada para fim de partilha e distribuição de informações sobre o CUDA. O endereço é o seguinte:

http://algos.inesc-id.pt/wikicuda/index.php/Main_Page

Serão disponibilizados todos os algoritmos e programas desenvolvidos, ou seja:

- Factorização de Cholesky
- Método de Jacobi; (Implementado por Marcelo Vicente)
- Método do Gradiente Conjugado. (Implementado por Marcelo Vicente)
- Gerador de powergrids em formato binário e texto; (Adaptado por Marcelo Vicente)
- Biblioteca CSparse (conversão para CUDA).

Também irá conter o Método Multigrid quando a sua implementação for terminada.

Serão também disponibilizados os resultados obtidos nas simulações efectuadas e respectivos gráficos para uma melhor visualização e comparação das velocidades de processamento dos diferentes métodos.

(continuar em folhas adicionais, se necessário)

Bolseiro

Assinatura: Marcelo Vaz Data: 2009 - 11 - 23

Orientador Científico

Assinatura: Luís Miguel Silveira Data: 2009 - 11 - 23

(Continuação da página 1)

- **Estudo sobre *powergrids***

Iniciou-se a bolsa começando por fazer uma pesquisa sobre *powergrids*. A maioria da informação neste aspecto foi transmitida pelo Prof. Luís Miguel Silveira que teve o cuidado de nos explicar a motivação e a importância da simulação de *powergrids*.

Uma *powergrid* é uma rede resistiva de baixa impedância que se encontra nos circuitos integrados e tem como finalidade a distribuição da corrente eléctrica por todo o circuito. A sua simulação é muito importante porque pode haver zonas do circuito que necessitem de mais corrente e essa corrente extra pode deixar outras zonas sem ela, sendo necessário adicionar elementos capacitivos à rede.

Estas redes resistivas podem ser representadas como um sistema de equações lineares e logo é possível usar métodos computacionais para os resolver.

Pode-se considerar que uma *powergrid* com N por N resistência pode ser decomposta em N^2 equações, resultando numa matriz de coeficientes com N^2xN^2 elementos. Em termos de espaço de armazenamento pode ser muito superior ao que na realidade dispomos para utilização, mas como a matriz resultante é esparsa é possível usar estruturas próprias para este tipo de matrizes. Para todas as implementações dos métodos iterativos foi usado a estrutura CSparse [1] que é ideal para o problema.

- **Pesquisa e iniciação à arquitectura CUDA**

Depois de ter sido efectuada uma pesquisa sobre o funcionamento de *powergrids* foi iniciada uma pesquisa e iniciação à arquitectura CUDA. Inicialmente foi lido o manual que provou ser útil para muitos aspectos técnicos, mas em termos práticos alguns programas e testes pequenos tiveram que ser implementados com o objectivo de tornar a adaptação mais intuitiva e mais detalhada. Isto demorou mais tempo do que era previsto, mas veio a provar ser útil em termos futuros.

Com isto foi possível perceber como o CUDA funciona internamente na placa gráfica e como programar eficientemente nesta arquitectura para obter o máximo de desempenho possível.

- **Pesquisa sobre métodos iterativos para a resolução sistemas lineares**

Todos os métodos implementados foram propostos pelo Professor. No inicio foi proposto implementar o método de Jacobi [2] como método iterativo e a factorização de Cholesky [3] como método directo, ficando este para o outro elemento do grupo. Depois da implementação do método de Jacobi ficou decidido implementar o método do Gradiente Conjugado [4] e por fim o método do Multigrid [5]. Este último está neste momento a ser acabado de implementar em conjunto como já referido anteriormente.

Em todos os métodos que foram propostos, foi inicialmente feita uma pesquisa mais baseada em pseudocódigo do que no funcionamento do método em si, visto ser a implementação o objectivo. Foi também proposto começar por uma implementação em CPU que depois viria a ser usada para efectuar comparações com resultados temporais obtidos com os do GPU.

A maioria dos pseudocódigos foram encontrados na Wikipedia, com excepção do Multigrid que foi o Professor que facultou um livro com muita informação útil. Também, a estrutura CSparse para CPU, que foi convertida para ser usada em GPU, foi obtida no Web site oficial e desenvolvida por Tim Davis [6].

- **Implementação do gerador de *powergrids* e da estrutura de dados usados**

Antes de começar a programar os métodos foi necessário ter um gerador de sistemas lineares que representassem fisicamente *powergrids* e um modo de as armazenar sem ocupar muito espaço de memória.

Inicialmente foi pesquisado a forma mais eficiente de guardar matrizes de grande dimensão mas muito esparsas. Por sugestão do Professor, foi sugerido o uso da estrutura de dados CSparse desenvolvida por Tim Davis. Esta estrutura foi testada para um exemplo simples em CPU e mostrou-se ser muito eficaz. Visto estar inicialmente só implementada em CPU houve a necessidade de a adaptar para CUDA. Como o CUDA é uma arquitectura paralela a estrutura só se adaptava para leitura, sendo a escrita impossível em multi-processamento.

Durante a conversão da versão de CPU para CUDA, surgiu um problema com a passagem dos dados em RAM para a placa gráfica que estava a aceder a posições não permitidas em algumas matrizes. Este erro só foi encontrado e corrigido quando já se estava na implementação do método de Jacobi e veio a dar muitos problemas o que fez perder muito tempo.

Em relação ao gerador de *powergrids*, o Professor conseguiu disponibilizar um gerador em MATLAB mas como as matrizes resultantes eram incompatíveis com a estrutura de dados CSparse, foi decidido fazer uma adaptação desse gerador de MATLAB para C usando directamente a estrutura. Isto provou ser muito mais eficiente do que a versão de MATLAB porque foi possível gerar sistemas com o mesmo tamanho 10 vezes mais rápido.

(Continua na página 4)

(Continuação da página 3)

- **Implementação do método de Jacobi**

Como este método foi o primeiro a ser desenvolvido, demorou mais tempo do que o resto. A sua implementação foi relativamente fácil e os resultados foram aceitáveis comparando com os do CPU apesar do tempo de cada resolução ser muito elevado.

Foram desenvolvidas três versões diferentes deste algoritmo usando formas diferentes de processamento do CUDA com o objectivo de testar outros caminhos possíveis e averiguar qual deles o mais eficiente. Em todas as implementações foram usados vários *kernels* com cada um contendo uma função específica e também várias transferências de dados entre a RAM e placa gráfica mas com a vantagem de estar a usar o CPU e o GPU em simultâneo. Depois de se concluir a implementação do método do Gradiente Conjugado veio-se confirmar que esta não era a melhor forma de abordar o problema, porque efectua demasiadas transferências entre a RAM e placa gráfica que diminuem drasticamente o desempenho.

- **Implementação do método do Gradiente Conjugado**

A implementação deste método, como já referido anteriormente, teve uma abordagem diferente do método de Jacobi. Desta vez não foram utilizados diferentes *kernels* mas apenas um único contendo todo o algoritmo e sem usar a troca de dados entre a RAM e placa gráfica. Esta forma de implementação mostrou-se muito mais eficiente que a anterior, como já referido anteriormente.

Tal como aconteceu na implementação anterior, também foram desenvolvidas algumas versões diferentes. Numa delas foi usado memória partilhada entre fios de execução em vez de primitivas de sincronização. Esta abordagem conduziu a um decréscimo temporal pequeno que se pode considerar insignificante para problemas de pequena dimensão.

Comparando a velocidade de resolução de sistemas lineares com este método com a do método de Jacobi, pode-se constatar que este é muito mais rápido. Isto deve-se à forma de implementação que foi usada, que é diferente da do outro método, mas mais ainda ao algoritmo em si que é muito mais eficiente em termos de iterações necessárias e cálculos efectuados para obter o mesmo resultado.

- **Implementação do método Multigrid**

Como já referido anteriormente, a implementação do Multigrid está de momento a decorrer e será disponibilizada online após a sua conclusão.

Bibliografia:

[1]. Página da biblioteca CSparse (Inglês): http://www.cise.ufl.edu/research/sparse/CSparse/
[2]. Método de Jacobi na Wikipedia (Inglês): http://en.wikipedia.org/wiki/Jacobi_method
[3]. Decomposição de Cholesky na Wikipedia (Inglês): http://en.wikipedia.org/wiki/Cholesky_decomposition
[4]. Método do Gradiente Conjugado na Wiki. (Inglês): http://en.wikipedia.org/wiki/Conjugate_gradient_method
[5]. Algoritmo do Multigrid (Inglês): http://www.fou.uib.no/fd/1996/h/413003/node47.html
[6]. Página pessoal do Prof. Tim Davis (Ingês): http://www.cise.ufl.edu/~davis/