Ajudando o decendente do rei Henrique VIII da Inglaterra usando a GPU

MAC412 - Organização de Computadores

Entrega Segunda, dia 10/11/2008

Introdução

Como vocês todos sabem, o rei Henrique VIII da Inglaterra fazia muito sucesso com as mulheres (ele teve seis esposas) e sempre teve problemas para conciliá-las. Hoje em dia, seu tetra-tetra-...-tetra[1] neto quer seguir os passos do tetra-tetra-...-tetra vô. Porém, ele sabe os problemas que pode ter se suas rainhas se encontrarem ou mesmo se elas se virem. Para facilitar sua vida, ele está construindo um castelo térreo onde os quartos estão dispostos de forma quadriculada e ele pretende dar um quarto para cada uma de suas futuras esposas.

O arquiteto que ele contratou para projetar o castelo era muito esperto então ele ajudou bastante. O projeto que ele elaborou é de um castelo quadrado com pré-instalações quadriculadas. Nesse sistema, ele garante que os quartos podem ser instalados dinamicamente (basta levantar o quarto com uma grua e colocá-lo em um dos quadrados). Os quartos foram feitos de forma a dar a impressão para as rainhas que elas estão vendo tudo que está ao redor delas. Para isso, existem janelas que permitem que as rainhas vejam todos os quartos que estão na mesma linha e coluna de seu quarto assim como aqueles que estão nas diagonais que passam por seu quarto como na Figura 1.

O tetra-tetra-...-tetra neto do rei Henrique, que passarei a chamar de Riquinho, sabe que é besteira ele ficar tentando organizar os quartos para que as rainhas não se vejam por si só (já que ele precisa casar e isso toma tempo). Como ele não sabe programar, Riquinho descobriu que alguém na família do "homem de ouro" ficou devendo um pão a 500 anos atrás para

[1] Ele morreu em 1547. Façam a conta de quantos `tetra`s deveriam ter. Considerem que o intervalo entre cada geração é de 30 anos.

a família real e exige que o "homem de ouro" lhe entregue um programa que descubra como ele pode colocar os quartos no castelo de forma que suas rainhas não se vejam. O problema é que ele quer rodar esse programa em seu computador de ultíssima geração que possui uma placa de vídeo NVidia no qual ele joga xadrez. Como o programa contra o qual ele joga consome muito processamento, ele exige que o programa use a placa de vídeo que ele tem e nunca usa.

Como queremos ajudar a família do "homem de ouro", pedimos que você com a ajuda de três de seus colegas, elabore o programa para que Riquinho perdoe a dívida agora estimada em um bilhão de libras esterlinas. Como achamos ligeiramente injusto que você resolva esse problema sem compensação, decidimos dar alguns incentivos para que você o faça. Em primeiro lugar, o resultado de seu trabalho valerá uma nota que contará na sua média da matéria. Além disso, queremos entregar o melhor possível para Riquinho para evitar futuros problemas e para incentivá-los, decidimos que a equipe que conseguir fazer o programa que roda mais rápido terá 10 de média na matéria toda.

Instruções de desenvolvimento

Para desenvolver o programa, vocês deverão usar a linguagem CUDA elaborada pela NVidia. A solução de vocês será testada em um MacBook Pro com uma GeForce 8600 GT com 256 MB de VRAM em modo emulado para a correção. Obviamente, qualquer solução cuja implementação meramente devolver imprimir as soluções sem descobrí-las em tempo de execução (*hard-coded*) ganhará 0 de nota e estará fora da competição.

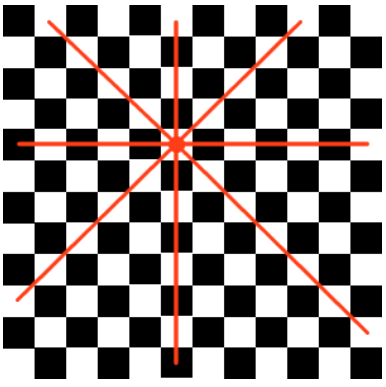

Figura 1: A visão de uma rainha no castelo

Entrada

O programa deverá receber um argumento na linha de comando correspondente ao número `n` de esposas que Riquinho precisa acomodar em seu castelo.

Saída

A saída esperada do programa é uma linha por possível posicionamento dos quartos. Os posicionamentos devem ser listados de forma a estarem em ordem lexicográfica de acordo com a descrição a seguir.

Para cada posicionamento, vocês devem imprimir a posição dos `n` quartos separados por ', ' e indo da esquerda para a direita e de cima para baixo. Cada quarto deve ser descrito com a linha sendo uma letra maiúscula (a primeira linha é 'A') e a coluna sendo um número na base 10.

Nenhuma mensagem pode conter acentos.

Exemplo

```
>nrainhas 1
A0
>nrainhas 2
Sem solucoes para o tamanho 2.
>nrainhas 3
Sem solucoes para o tamanho 3.
>nrainhas 4
A1, B3, C0, D2
A2, B0, C3, D1
```

0.1 Escolha da melhor solução

Para escolher a melhor solução, 5 finalistas serão selecionados de acordo com o seguinte critério. Para ser um possível finalista, o programa deve produzir a resposta certa em, no máximo, 110% do tempo necessário para a solução que usa a CPU que eu elaborei. Dessas soluções, as 5 mais rápidas para tamanhos até 16 serão selecionadas. Essas 5 soluções serão rodadas em modo real (sem emulação, direto na placa de vídeo) para tamanhos de até 26. Aquela que conseguir os melhores resultados, vencerá a competição.

Formato de entrega

Você deve entregar sua solução usando o sistema PACA disponível em:

`http://paca.ime.usp.br/`

1. Código fonte do programa

   Você e sua equipe deverão empacotar todos os arquivos necessários para que eu compile e rode sua solução. Tipicamente isso incluirá os seguintes arquivos:

   `Makefile`, `nrainhas.cu`, `nrainhas_kernel.cu`

   Se vocês incluírem mais arquivos, tenham certeza de adicioná-los ao pacote. Por favor, NÃO incluam os binários gerados ou arquivos temporários (como arquivos .o). Eu devo conseguir rodar `make` para compilar seu programa. O programa será extraído na pasta *projects* que a instalação do CUDA gera (ele já estará instalado).

2. Um relatório do trabalho em PDF

   Existem duas soluções principais para esse problema, uma iterativa e outra recursiva. Gostaria de ver um pseudo-código no relatório da solução que vocês não implementaram. Também quero saber se vocês elaboraram uma versão que usa a CPU antes de fazer aquela para GPU. Se sim, quero saber o que vocês sentiram que é a maior diferença entre uma solução e a outra. Se não, quero saber se vocês acham que teria sido mais fácil fazer dessa forma ou não e porquê.

   Por fim, quero uma avaliação de vocês sobre as vantagens e desvantagens de usar uma GPU para elaborar programas e o que vocês acharam da experiência.

A entrega deve consistir de um único arquivo em formato tar.gz contendo todos esses arquivos. O nome desse arquivo deve ser o seguinte formato:

Nome-Integrante-1-**Nome-Integrante-2**-**Nome-Integrante-3**-**Nome-Integrante-4**-ep2.tar.gz

e ele deve se extrair em uma pasta com o mesmo nome (sem a extensão) com todos esses arquivos. Tomem cuidado para entregar esse arquivo apenas uma vez para cada grupo. Apenas um membro de cada equipe deve entregar o arquivo. Caso contrário, pegarei o arquivo entregue mais tarde (e descontarei nota se for entregue atrasado).

Para cada item fora do formato de entrega pedido, o grupo perderá um ponto na nota final.

Divirtam-se!