Universidade Estadual de Mato Grosso do Sul - UEMS
Curso de Ciência da Computação
Disciplina de Algoritmos Paralelos e Distribuídos

Professor: Dr. Rubens Barbosa Filho

Data: 24/05/2022
Ano: 4
Turma: U

Trabalho: Implementação de uma simulação do modelo da formiga de Langton

DESCRIÇÃO DO TRABALHO:
A formiga de Langton é uma máquina de Turing bidimensional com um conjunto muito simples de regras, mas um complexo comportamento emergente. Foi inventado por Chris Langton em 1986 e é executado em uma rede quadrada de células brancas e pretas.

Esta simulação não exige a interação com o usuário durante sua execução. Esta aplicação caracteriza-se por um grau de iteratividade muito grande, sendo desta forma um problema que demanda uma grande quantidade de processamento.

Parte I
Formiga de Langton Tradicional

Em sua forma mais simples, a formiga de Langton funciona da segunte forma:

- Comece uma formiga voltada para o norte (”para cima”) no centro de uma grade bidimensional com todas as células coloridas em branco.
- Ande vários passos. Em cada passo:
  1. A formiga muda a cor da célula em que se encontra ($branco \longrightarrow preto$ e $preto \longrightarrow branco$);
  2. A formiga avança uma célula (na direção para a qual estava voltada);
  3. A formiga gira no lugar de acordo com as seguintes regras:
     (a) Se a célula em que a formiga acabou de entrar for branca, ela vira 90 graus para a direita.
     (b) Se a célula em que a formiga acabou de entrar for preta, ela vira 90 graus para a esquerda. Curiosamente, embora as regras que governam a evolução da formiga sejam muito simples, elas podem levar a um comportamento emergente complicado.

Parte II

Formiga de Langton Paralela

Embora a Formiga de Langton seja interessante por si só, examinaremos uma versão ligeiramente estendida neste trabalho. Em vez de apenas preto e branco, permitiremos que as células da matriz sejam coloridas arbitrariamente (usaremos números para representar a cor de cada célula, em vez de nomes como ”preto”e ”branco”).

Também variaremos as regras pelas quais a formiga decide em que direção virar. Usaremos uma seqüência de caracteres ”E”e ”D”para representar as regras pelas quais a formiga deve girar. O sistema simples acima pode ser simplificado pela string ”DE”; se a formiga observar a cor 0, ela irá virar para a direita (o caractere de índice 0 na string é ”D”), e se ela observar a cor 1, ela irá virar para a esquerda (o caractere de índice 1 na string é ”E”). No entanto, observe que não estamos limitados a apenas duas cores e, portanto, podemos ter regras mais complicadas como ”DEDD”, que diz à formiga para virar à direita quando vê as cores 0, 2 ou 3, e à esquerda quando vê a cor 1.

Neste trabalho teremos uma matriz de (2000 x 2000). Esta matriz quadrada será dividida entre quatro máquinas. E cada máquina será responsável por comandar a caminhada de uma formiga.

Lembre-se que durante a caminhada uma formiga pode invadir o quadrante de outra formiga e, neste caso, teremos duas formigas caminhando em uma mesma submatriz. A forma de controle, fica a cargo da implementação do programador. Uma sugestão é: A máquina que teve a formiga caminhando para outro quadrante continua controlando a formiga mesmo esta estando em uma submatriz que não seja a sua.

Cada máquina terá a sua combinação própria de cores. Esta combinação fica a cargo do programador.

Para este trabalho utilizaremos um paralelismo de dados e, assim sendo, você deverá se preocupar principalmente em realizar a divisão da matriz em 4 partes.

Parte III

Regras do Jogo

As Regras são baseadas na noção de vizinhança de cada célula. Células adjacentes, incluindo as diagonais, são denominadas vizinhas. A simulação é baseada em passos de tempo discretos. Ele é executado em uma grade quadricular. Para representar o mundo em que se passa a simulação, os lados opostos da grade não serão conectados. Se uma formiga atingir a borda da matriz, ela não deve atravessá-la.

Os quadrados em um plano são coloridos de acordo com o movimento da formiga e, assim sendo, podemos identifcar o quadrado como sendo a formiga. A formiga pode viajar em qualquer uma das direções cardeais a cada passo que é dado. Em sua forma tradicional, a formiga de Langton é executada apenas com as cores preto e branco. Neste trabalho, nós utilizaremos mais cores.

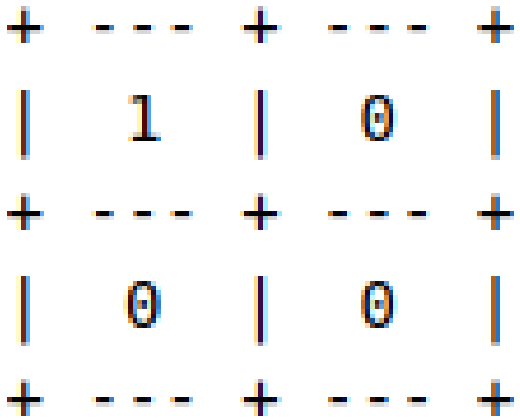
A modelagem das regras é a seguinte:

- **Esta modelagem servirá para as quatro máquinas que utilizaremos nesse trabalho. A diferença é que cada máquina será responsável por um conjunto de cores.**
- Inicialmente todos os quadrados (células) são coloridas zero (0).

- Dê vários passos. Para cada passo:
  - A formiga muda a cor da célula em que está incrementando a cor em 1, mas dando a volta com base no número total de cores no sistema (em um sistema de três cores, por exemplo, 0 vai para 1, 1 vai para 2 e 2 vai para 0, por exemplo).
- Comece uma formiga voltada para o norte ("para cima") no centro de uma grade bidimensional de tamanho *size*. Você pode achar o centro utilizando a expressão (matriz_size / 2, matriz_size / 2).
- A formiga avança uma célula (na direção para a qual estava voltada).
- A formiga gira no lugar com base na cor da célula que acabou de entrar, de acordo com as regras, que são especificadas como uma string da forma explicada acima. A string que usaremos será: **EEDD**. Caso você encontre uma sequência de strings que produza uma saída mais interessante, sinta-se a vontade para trocar. Algumas sugestões: DED, EDDDDDEED, EEDDDEDEDEED, DDEEEDEEEDDD.
- Como a string utilizada possui tamanho 4 (EEDD), você deverá destinar 4 (quatro) cores para a formiga. Lembre-se que cada máquina deverá ter um conjunto de cores com pelo menos duas cores diferentes entre si.

Escreva uma função chamada **exec_langton(regras, tamanho)** que simula a evolução de um sistema descrito acima em uma matriz **NxN** onde o valor de N é 2000, sendo então, 4 milhões de células. Observe que a formiga deve começar na célula central voltada para cima.

Uma sugestão de implementação é usar uma função que retorne uma tupla (contador, matriz), onde contador é o número de passos que o sistema dá até que a formiga saia de sua submatriz de origem para outra submatriz e, matriz é uma lista de listas contendo inteiros, representando a coloração final da matriz. Por exemplo, a matriz colorida como:



É representada pela seguinte lista de listas: [[1,0],[0,0]]. Mas esta é apenas uma sugestão. Você pode utilizar outra solução própria.

Dica 1: Você precisará de uma maneira de representar a posição da formiga, incluindo o ângulo, para saber como ajustar para o movimento da formiga "para frente".

Dica 2: você pode achar mais fácil começar implementando a especificação original da Formiga de Langton, na qual existem apenas duas "cores"possíveis no mundo (1 e 0), e a formiga vira à direita ao entrar em um quadrado de cor 1, e à esquerda ao entrar em um

quadrado de cor 0. Depois de implementar isso, pense em como você pode estendê-lo para o caso mais geral descrito acima.

Parte IV
Condição de Parada

A execução do trabalho deverá parar após 15000 passos da formiga.

Parte V
Parte Gráfica

Sugestão de configuração: Dentre o total de máquinas utilizadas para resolver o problema, divida as máquinas da seguinte forma:

- Uma máquina mestre (que pode possui uma cópia de toda a matriz);
- Uma máquina responsável por fazer a atualização das telas, ou seja, esta máquina faz a atualização e envia para a máquina mestre; A PARTE GRÁFICA FICA A GOSTO DE CADA ALUNO - A SDL é uma boa opção.
- As demais máquinas trabalham como escravo (ou seja, possuem partes da matriz).

Parte VI
Objetivos

Este trabalho tem como objetivo fazer uma avaliação do algoritmo de aplicação em um ambiente paralelo com foco no paralelismo, limitações e gargalos.

O trabalho deve ser feito para rodar sequencialmente em uma matriz menor que a original (algo em torno de 300x300). Lembre-se que você precisa destes dados para poder calcualar o *speedup*. Ajuste as configurações do problema para rodar em uma máquina.

A parte paralela deve ser executada em 4 máquinas(lab1).

O trabalho deve ser feito em linguagem C e, podem ser utilizadas todas as funções, sejam elas ponto-a-ponto ou coletivas.

Parte VII
O que entregar

1. O código e a documentação devem ser entregues em um arquivo Zip. Dentro deste arquivo Zip devem conter um readme.txt com o nome do autor e o comando para a execução do código e, um arquivo PDF da documentação. Inclua todos os arquivos fontes (.c, .h, makefile, não incluam executáveis ou arquivos objetos), em um único diretório. Um Makefile deve ser fornecido para a compilação do código.

2. O nome do arquivo deve seguir o seguinte padrão: $NomeAluno\_RGM\_NomeArquivo.c$ para arquivo fonte.

3. Submissões onde os programas não sigam as especificações de parâmetro e nome, makefiles não funcionando ou arquivos necessários faltando NÃO SERÃO CORRIGIDOS.Programas que não compilem também não serão corrigidos.

4. Gráfico comparativo de tempo entre a execução sequencial e a execução paralela (para 4 máquinas) para um mesmo tamanho de matriz;

5. Gráfico com o cálculo do *Speedup*.

6. *Log* de execução de cada máquina em separado.

Parte VIII
Documentação

O texto da documentação deve ser breve, de forma que o professor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deverá conter os seguintes itens:

1. O sumário do problema a ser tratado;

2. Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, procedimentos e das decisões de implementação;

3. Decisões de implementação que porventura estejam omissos na especificação;

4. Como foi tratada a comunicação entre as máquinas;

5. Testes, mostrando que o programa está funcionando de acordo com as especificações, seguidos de sua análise;

6. Print screens mostrando o correto funcionamento do simulador e exemplos de testes executados;

7. Conclusão e referências bibliográficas.

Parte IX
Avaliação

A avaliação do trabalho será composta pela execução dos programas desenvolvidos e pela análise da documentação. Todos os alunos deverão apresentar e explicar em laboratório o trabalho.

Os seguintes itens serão avaliados:

1. A qualidade do código (código bem organizado, estruturado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc);

2. Execução correta do código com entrada de testes a serem definidas no momento da avaliação. As entradas de testes irão exercitar a funcionalidade completa do código e, testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto;

3. Conteúdo da documentação que deve conter os itens mencionados anteriormente;

4. Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua portuguesa, qualidade textual e facilidade de compreensão);

5. CASOS DE CÓPIA NÃO SERÃO TOLERADOS;

6. Qualquer elemento que não esteja especificado neste documento, mas que tenha que ser inserido para que o simulador funcione, deve ser descrito na documentação de forma explícita.

Parte X
Data de entrega

Dia 09/08/2024 - segunda feira.

O trabalho deve ser feito individualmente.