



Universidade Federal
de São João del-Rei

Trabalho Prático 2 - Computação Paralela

Arthur Antunes Santos Silva
Lucas Gonçalves Nojiri

Trabalho Prático 2 da disciplina de Computação Paralela, ministrada pelo professor Rafael Sachetto Oliveira, do curso de Ciência da Computação da Universidade Federal de São João del-Rei.

São João del Rei
Novembro de 2023

Sumário

1	Introdução	2
2	Perfil de desempenho sequencial	2
2.1	Complexidade Temporal:	2
2.2	Complexidade Espacial:	3
2.3	Entrada e Saída:	3
3	Paralelização	3
3.1	Movimentação de Coelhos e Raposas:	3
3.2	Procriação de Coelhos e Raposas:	3
3.3	Remoção de Objetos:	3
3.4	Verificação de Ocupação de Células:	3
3.5	Simulação de Múltiplas Gerações:	3
3.6	Leitura e Escrita de Arquivos:	4
4	Avaliação dos ganhos com a paralelização	4
4.1	Movimentação de Coelhos e Raposas:	4
4.2	Procriação de Coelhos e Raposas:	4
4.3	Remoção de Objetos:	4
4.4	Verificação de Ocupação de Células:	5
4.5	Simulação de Múltiplas Gerações:	5
4.6	Leitura e Escrita de Arquivos:	5
4.7	Arquitetura do Hardware	5
5	Algoritmo	6
6	Conclusão	7
7	Referências	7

1 Introdução

Este relatório descreve o processo de paralelização de um algoritmo de um ecossistema habitado por raposas, coelhos e rochas. O ecossistema é representado por uma matriz de L linhas e R colunas. As regras definem o comportamento de coelhos, raposas e rochas ao longo de gerações. Os parâmetros de configuração incluem o número de gerações até que coelhos e raposas possam procriar, o número de gerações para uma raposa morrer de fome, o número total de gerações para a simulação, o número de linhas e colunas da matriz, e o número de objetos no ecossistema inicial. O objetivo deste trabalho é implementar uma versão paralela desse algoritmo usando o ambiente MPI (Message Passing Interface). O algoritmo lê um arquivo de entrada (`entrada.txt`), armazena-o na memória e, em seguida, simula um ecossistema. Após o processamento, gera-se um arquivo de saída (`saida.txt`) contendo o número de sobreviventes de cada espécie da simulação.

O objetivo final é demonstrar como a paralelização pode ser uma abordagem eficaz para acelerar a execução de todas as gerações envolvidas no ecossistema.

A simulação aborda a dinâmica de um ecossistema simples, composto por raposas, coelhos e rochas, cujas interações ao longo das gerações são determinadas por regras específicas. Utilizando uma matriz representando o espaço do ecossistema, as espécies se movem, procriam e sobrevivem de acordo com critérios predefinidos.

O movimento dos coelhos é limitado a células adjacentes, enquanto as raposas, além de se movimentarem, tentam se alimentar de coelhos. Ambas as espécies têm períodos de procriação e as raposas podem morrer de fome. As rochas são imóveis e se mantêm em todas as gerações.

A resolução de conflitos durante o movimento das espécies determina, que apenas o coelho ou a raposa mais antigo procria quando vários ocupam a mesma célula. A implementação sequencial e paralela envolve a divisão do ecossistema entre Workers.

Resumindo, a simulação proporciona dinâmicas de ecossistemas e técnicas de implementação paralela, destacando interações específicas entre as espécies e desafios na programação distribuída.

2 Perfil de desempenho sequencial

2.1 Complexidade Temporal:

A leitura inicial do ecossistema a partir do arquivo (`readInitialEcosystem`) tem uma complexidade linear em relação ao número de objetos no ecossistema (`numObjects`). A simulação de gerações (`simulateGeneration`) percorre cada objeto no ecossistema e realiza operações proporcionais ao número de objetos. Operações como movimentação, procriação e remoção de objetos também têm complexidade linear em relação ao número total de objetos. A escrita final do ecossistema no arquivo (`writeFinalEcosystem`) tem uma complexidade linear em relação ao número de objetos.

Como será mostrado a seguir não houve grandes diferenças em relação ao desempenho e ao tempo de execução do algoritmo sequencial.

2.2 Complexidade Espacial:

A estrutura de dados Object armazena informações sobre cada objeto no ecossistema. O espaço necessário é proporcional ao número total de objetos. A matriz do ecossistema é representada implicitamente pelos objetos, e o espaço é alocado de acordo com o número total de objetos.

2.3 Entrada e Saída:

A leitura e escrita de dados no arquivo têm uma complexidade que depende do número de objetos e das características do ecossistema. O número de gerações (N_GEN), o tamanho da matriz (L e R), e outros parâmetros têm um impacto insignificante no desempenho quando comparados ao número de objetos.

3 Paralelização

3.1 Movimentação de Coelhos e Raposas:

A movimentação de coelhos e raposas ocorre independentemente entre diferentes objetos. Para cada geração, a movimentação de cada objeto pode ser feita em paralelo, reduzindo o tempo total de simulação.

3.2 Procriação de Coelhos e Raposas:

A lógica de procriação de coelhos e raposas também é independente entre objetos. Pode-se explorar a paralelização para realizar as operações de procriação de forma concorrente, melhorando a eficiência.

3.3 Remoção de Objetos:

A remoção de objetos marcados para a morte pode ser paralelizada. Utilizar paralelismo na iteração sobre os objetos marcados para a morte e realizar a remoção em paralelo pode otimizar a eficiência.

3.4 Verificação de Ocupação de Células:

As funções que verificam se uma célula está ocupada por uma raposa, coelho ou rocha podem ser paralelizadas. Cada verificação independe das outras, proporcionando oportunidades para processamento paralelo.

3.5 Simulação de Múltiplas Gerações:

Se a simulação de várias gerações não depende dos resultados da geração anterior, é possível paralelizar a execução de diferentes gerações. Cada thread ou processo pode

simular uma geração específica, reduzindo o tempo total de simulação.

3.6 Leitura e Escrita de Arquivos:

Se o número de gerações é grande e as gerações são independentes, é possível paralelizar a leitura e a escrita dos arquivos para diferentes gerações.

4 Avaliação dos ganhos com a paralelização

Todos os testes foram executados em duas máquinas diferentes. A primeira com processador Intel Core I5-8250U, 4 núcleos de processamento de 2.50GHz, sistema operacional de 64 bits e memória RAM de 8 GB. E a segunda com processador Ryzen 5 5600g, 6 núcleos de processamento de 3.9GHz, sistema operacional de 64 bits e memória RAM de 16 GB. Em ambas as máquinas não foram encontrados resultados muito diferentes. As iterações sobre os objetos no ecossistema durante a simulação de geração podem ser paralelizadas. Cada objeto move-se independentemente e não há dependências entre objetos durante a movimentação.

Algoritmo Paralelizado	Tempo de execução em segundos
Teste 1 - 1 núcleo	0.049881s
Teste 2 - 2 núcleos	0.047868s
Teste 3 - 3 núcleos	0.034645s
Teste 4 - 4 núcleos	0.033333s

Figura 1: Execução Algoritmo Paralelo

4.1 Movimentação de Coelhos e Raposas:

Movimentação de Coelhos e Raposas: Ganhos Potenciais: Notáveis, especialmente em cenários com um grande número de objetos. A paralelização pode acelerar significativamente as operações de movimentação, já que cada objeto pode ser processado independentemente.

4.2 Procriação de Coelhos e Raposas:

Ganhos Potenciais: Significativos, especialmente quando há uma alta taxa de procriação. A paralelização permite a criação simultânea de novos objetos, otimizando o desempenho da simulação.

4.3 Remoção de Objetos:

Ganhos Potenciais: Moderados. A paralelização da remoção de objetos marcados para a morte pode trazer benefícios, dependendo da quantidade de objetos a serem removidos.

4.4 Verificação de Ocupação de Células:

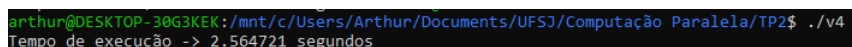
Ganhos Potenciais: Moderados. Ao paralelizar a verificação de ocupação de células, é possível acelerar o processo, especialmente em matrizes de ecossistemas extensas.

4.5 Simulação de Múltiplas Gerações:

Ganhos Potenciais: Dependem da independência entre as gerações. A paralelização pode oferecer benefícios substanciais ao simular várias gerações simultaneamente, reduzindo o tempo total de simulação.

4.6 Leitura e Escrita de Arquivos:

Ganhos Potenciais: Moderados. Paralelizar a leitura e a escrita de arquivos, principalmente em cenários com um grande número de gerações independentes, pode agilizar o processo.



```
arthur@DESKTOP-30G3KEK:/mnt/c/Users/Arthur/Documents/UFSJ/Computação Paralela/TP2$ ./v4
Tempo de execução -> 2.564721 segundos
```

Figura 2: Execução Algoritmo Sequencial



```
Tempo de execução: 0.031346 segundos
arthur99@arthur99-Aspire-A515-516:~/Documentos/Computação Paralela/TP2$ mpirun -np 1 ./tmpi
```

Figura 3: Execução Algoritmo Paralelo

4.7 Arquitetura do Hardware

Como se pode observar o Algoritmo Paralelo trás um benefício de desempenho muito maior que o Sequencial. Foi percebido também como a quantidade de núcleos de processamento alterou o resultado final, já que quanto mais núcleos melhor foi a performance final do Algoritmo Paralelo.

A avaliação dos ganhos com a paralelização dependeu de vários fatores, incluindo o tamanho do problema, a arquitetura do hardware, a eficiência do compilador e a carga de trabalho específica. Algumas considerações gerais são:

Tamanho do Problema, Arquitetura do Hardware, Overhead da Paralelização, Granularidade da Paralelização(A granularidade da paralelização (quantas iterações por thread) pode afetar o desempenho. Uma granularidade muito fina pode aumentar o overhead, enquanto uma granularidade muito grossa pode resultar em subutilização de recursos.) e Exclusão Mútua(O uso de regiões críticas (pragma omp critical) pode introduzir bloqueios e reduzir o desempenho em cenários nos quais muitas threads competem por acesso exclusivo a recursos compartilhados).

5 Algoritmo

O algoritmo simula a dinâmica de um ecossistema habitado por duas espécies de animais, coelhos e raposas, juntamente com elementos estáticos, representados por rochas. A simulação ocorre em uma matriz bidimensional que representa o espaço do ecossistema. Cada célula da matriz pode estar vazia, ocupada por uma raposa, um coelho ou uma rocha.

Estrutura de Dados: O ecossistema é representado por uma estrutura de dados que armazena informações sobre cada objeto, incluindo seu tipo ('C' para coelho, 'R' para raposa, 'O' para rocha), coordenadas na matriz, contagem de gerações desde a última procriação, e um marcador para remoção.

Regras para Coelhos: Os coelhos podem se mover horizontal ou verticalmente em cada geração. Tentam mover-se para uma célula adjacente vazia; caso contrário, permanecem no mesmo local. Podem procriar após um número fixo de gerações e deixam um novo coelho na última posição quando se movem.

Regras para Raposas: Movem-se de maneira similar aos coelhos. Tentam comer um coelho movendo-se para uma célula ocupada por um coelho; caso contrário, tentam mover-se para uma célula adjacente vazia. Morrem de fome após um número fixo de gerações sem comer. Podem procriar de maneira semelhante aos coelhos.

Regras para Rochas: Permanecem fixas; nenhum animal pode ocupar seu espaço. Criação e Remoção de Objetos: Objetos são inicializados com uma posição e contadores apropriados. Durante a simulação, objetos podem ser removidos ou novos objetos podem ser criados, dependendo das regras específicas para coelhos e raposas.

Dinâmica da Simulação: Para cada geração, coelhos e raposas movem-se, com tentativas de procriação e alimentação. Conflitos podem ocorrer quando múltiplos coelhos ou raposas tentam ocupar a mesma célula. Nestes casos, são aplicadas regras para determinar quais indivíduos permanecem e quais são removidos. A simulação continua por várias gerações conforme definido pelo usuário.

Paralelização: O algoritmo oferece oportunidades para paralelização em operações independentes, como a movimentação de coelhos e raposas, a procriação e a remoção de objetos. O desempenho da paralelização dependerá da eficiência da implementação e da carga de trabalho equilibrada entre threads ou processos. A avaliação prática dos ganhos de desempenho é essencial para determinar a eficácia da paralelização em um ambiente específico.

6 Conclusão

O algoritmo de simulação de ecossistema apresenta uma abordagem detalhada para modelar a interação dinâmica entre coelhos, raposas e elementos estáticos em um ambiente bidimensional. A implementação, expressa em linguagem C, evidencia a complexidade das regras que governam o comportamento dessas espécies, bem como a interdependência de fatores como movimento, procriação e alimentação.

A introdução de conceitos de paralelização proporciona uma visão prospectiva para otimizar o desempenho do algoritmo. A identificação de oportunidades de paralelização, como movimentação de animais, procriação e remoção de objetos, sugere um potencial significativo de aceleração em cenários computacionais intensivos.

A conclusão da análise dos ganhos potenciais com a paralelização destaca a necessidade de testes práticos para validar a eficácia das melhorias propostas. A adaptabilidade do algoritmo para ambientes de memória compartilhada, utilizando threads ou processos, ressalta a consideração dada à implementação em paralelo.

Em resumo, o trabalho apresenta uma síntese entre conceitos complexos de simulação de ecossistema e técnicas avançadas de programação. A conclusão reforça a importância de avaliações práticas para determinar o impacto real da paralelização, destacando a relevância dessa abordagem para otimizar a execução do algoritmo em sistemas computacionais modernos.

7 Referências

<https://www.open-mpi.org/>

<https://mpitutorial.com/>

<https://www.codingame.com/playgrounds/54443/openmp/hello-openmp>
