T4

schdck & Mostardinha

Entendendo o programa

Um resumo:

- O programa cria uma matriz e infecta uma única pessoa (no centro).
- A pessoa infectada tem uma chance **p** de infectar pessoas ao seu redor.
  - As pessoas ao redor são as na posição acima, abaixo, à esquerda e à direita na matriz.
- Repetimos o passo anterior para cada nova pessoa infectada.
- Continuamos a fazer isso até que nenhuma nova pessoa seja infectada.
- Todas as etapas anteriores são repetidas **n** vezes para obtermos a média.
- Todas as etapas anteriores são repetidas, com **p** variando de 0% a 100%.

Considerações

Com isso, podemos fazer as seguintes afirmações:

- Quanto maior a chance **p**, mais pessoas serão infectadas.
  - Se mais pessoas são infectadas, mais iterações são necessárias para concluir a execução (visto que demora mais tempo para que nenhuma nova pessoa seja infectada).
- Quanto maior o número **n**, maior a duração de cada simulação.
  - Isso acontece pois serão necessários mais testes para obter a média.

Considerações

E com essas afirmações, podemos concluir que:

- O custo para executar uma simulação depende da probabilidade **p**.
  - Quanto maior **p**, mais custosa é a simulação. Logo, o custo de cada execução cresce com o tempo.
  - As primeiras simulações (**p** próximo de 0%) serão mais rápidas que as últimas (**p** próximo de 100%), pois serão necessárias menos execuções até que ninguém seja contaminado.
- O custo para executar um teste é sempre fixo
  - Como todos os **n** testes usados para calcular a média de uma simulação usam a mesma probabilidade **p**, seu custo computacional é muito semelhante entre as execuções.

Programas desenvolvidos

O trabalho utilizou três programas para executar os testes:

- Programa zero: Sem nenhum tipo de paralelismo
- Programa um: Paralelização do tipo ***static*** tanto no laço que varia a probabilidade **p** quanto no laço que executa os **n** testes para obter a média.
- Programa dois: Paralelização do tipo ***guided*** no laço que varia a probabilidade **p** e do tipo ***static*** no laço que executa os **n** testes para obter a média.

Tipo 1: *static static*

- Utilizamos o *static* nas duas situações para exemplificar como um mau planejamento da paralelização pode não explorar o máximo de ganhos ou até prejudicar o rendimento da aplicação.
- O uso do *static* no primeiro laço não é o ideal pois o custo computacional de cada iteração não é o mesmo.
- Como não foi especificado o *chunk size*, a primeira thread fica com menos carga de trabalho que a última (que irá ficar com as iterações mais custosas), desbalanceando a distribuição de trabalho entre as threads.

Tipo 2: *guided static*

- O uso de *guided* para o laço que varia a probabilidade é visto como ideal, visto que ele começa atribuindo um *chunk size* grande para as threas, mas diminui esse tamanho conforme se aproxima do final das iterações. Isso tem efeitos positivos no desempenho, pois:
  - Nas primeiras execuções (que são pouco custosas), usamos *chunks* maiores, evitando o *overhead* de solicitar um novo *chunk* para trabalhar com muita frequência.
  - Nas execuções finais (que são muito custosas), o *chunk* é menor, permitindo que o trabalho seja melhor distribuído entre as threads trabalhadoras.

Testes

Os testes revelaram os resultados esperados. O **Tipo 2** teve um aumento muito expressivo de rendimento (especialmente nos testes maiores), já que explora o máximo das particularidades do programa.

O **Tipo 1** aumenta o rendimento em aproximadamente 50% nos testes maiores. Esse ganho é explicado pela paralelização do laço que executa as **n** tentativas para tirar a média. No laço de probabilidades, as iterações mais demoradas acabam ficando em apenas uma thread, impedindo um aproveitamento ideal.

População	Testes	Prob. Máx.	Paralelização	Execução (ms)	Speedup
100	1000	101	Tipo 0	765267	1,000
			Tipo 1	482275	1,587
			Tipo 2	428550	1,786
50	500	101	Tipo 0	54096	1,000
			Tipo 1	38735	1,397
			Tipo 2	36080	1,499
25	250	101	Tipo 0	4039	1,000
			Tipo 1	3605	1,120
			Tipo 2	3501	1,154