

Problema: One-Dimensional Stencil

Fonte: <https://dzone.com/articles/java-phasers-made-simple>

Dado um vetor com N elementos, o valor de cada elemento é calculado de acordo com a função:

$$A_n = \begin{cases} k & \text{if } n=0 \text{ or } n=N-1 \\ (A_{n-1} + A_{n+1})/2 & \text{in any other case} \end{cases}$$

O valor do primeiro e último elemento são fixos.

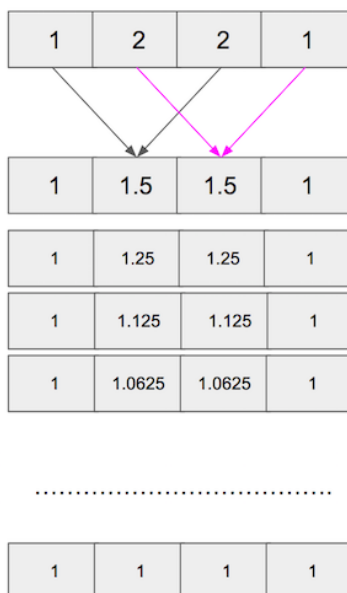
O valor dos outros elementos é a média dos vizinhos.

Processo recursivo: se um valor de um vizinho muda, cada elemento precisa recalcular seu próprio valor.

Após poucas iterações, o valor de todos elementos convergem e as iterações recursivas são finalizadas.

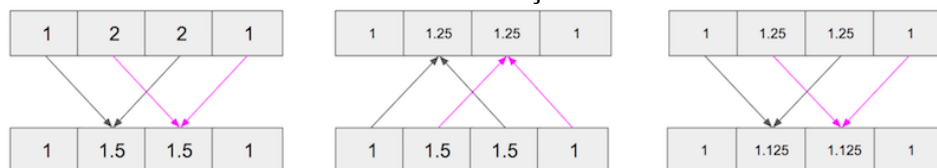
Exemplo:

Considere um vetor de 4 elementos:



Detalhes de implementação:

Usar dois vetores: vetor com valores no início da iteração e outro vetor com os resultados no final:



Implemente uma solução sequencial para o problema.

Implementações Paralelas

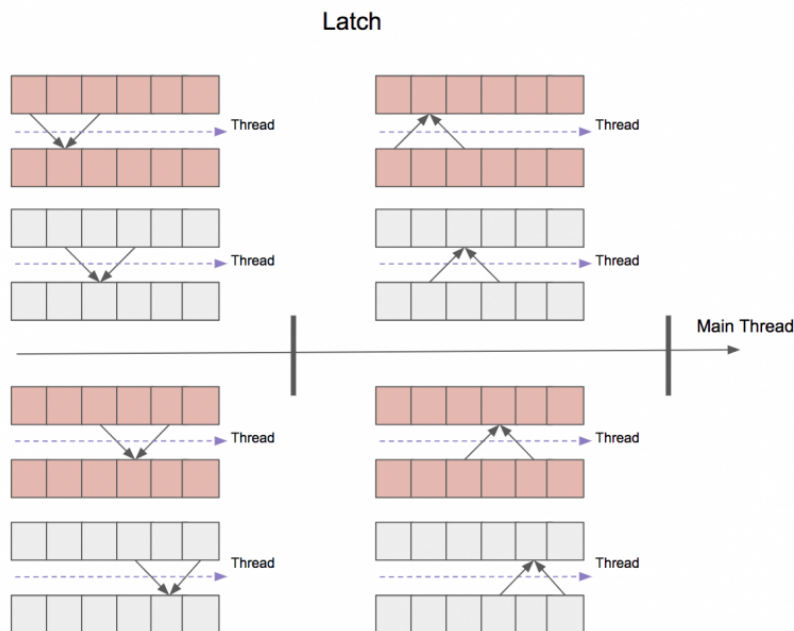
Latch

- Criar um novo latch para cada iteração com um contador igual N-2.
- Criar N-2 tarefas para serem executadas em paralelo por threads. Uma thread principal espera todas as tarefas. Cada tarefa calcula o valor de 1 elemento do vetor.
- Uma tarefa termina e avisa a thread principal que finalizou.

- Quando todas as tarefas finalizam, a thread principal retoma sua atividade e inicia uma nova iteração.

Observar que:

- Uma vez que uma tarefa é finalizada, libera uma thread para executar outra tarefa. Sugestão de manter um thread pool pequeno.
- O latch é uma barreira para a execução da thread principal.

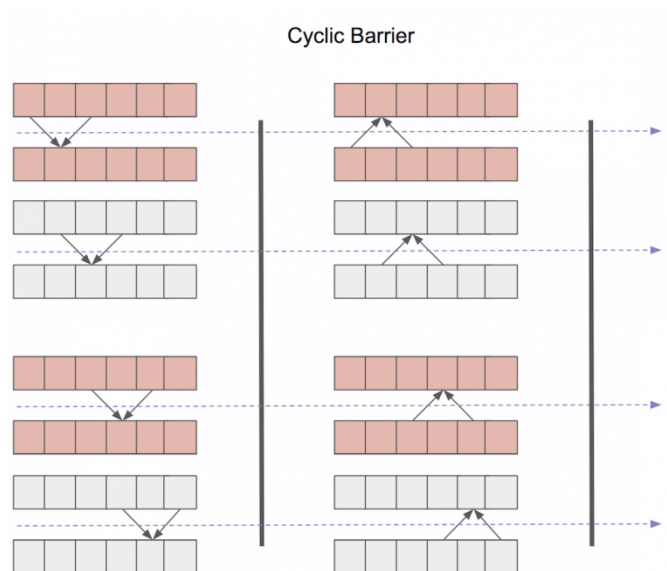


Barreiras cíclicas

- Criar uma barreira cíclica com $N-2$ partes.
- Criar $N-2$ tarefas para serem executadas em paralelo por threads.
- No final de cada iteração, cada thread espera na barreira até todas outras alcançarem.
- Arrays são invertidos e nova iteração inicia.

Observar que:

- Nenhuma thread é liberada depois de cada iteração, então o thread pool deve ter no mínimo o número de tarefas $(N-2)$.



Phasers

- Cada tarefa precisa esperar somente pelas tarefas adjacentes para computar o valor de seus vizinhos.
- Cada thread compartilha uma barreira com as adjacentes.

