EP2- Simulador de corrida por eliminação

MAC0422 - Sistemas Operacionais

Beatriz Viana Costa - 13673214

SUMÁRIO

01 Decisões do projeto

02 Ambiente e execução dos testes

03 Resultados

04 Análise dos resultados

Decisões de projeto – Bibliotecas

- Para o gerenciamento das `threads` foi utilizada a biblioteca `pthread.h` ;
- Já para o gerenciamento as barreiras de sincronização, foi utilizada a tanto a biblioteca `pthread.h` quanto a `semaphore.h` ;
- Todas as regiões de leitura e escrita de variáveis compartilhadas, como o velódromo, são protegidas por `mutex`;
- O modelo de barreira de sincronização utilizado foi o de `Coordinator` e `Workers`;
- Como a simulação da corrida não precisa levar o mesmo tempo que uma corrida verdadeira levaria, contamos apenas as **iterações** realizadas pelo simulador, e calculamos quanto tempo levaria uma corrida real.

Decisões de projeto - Velódromo

- O vetor que representa o velódromo foi estruturado da seguinte forma:
  - A linha de largada dos ciclistas é a coluna `d-1`, já a linha de chegada é a coluna `0`;
  - Na largada, os ciclistas são organizados da linha de índice `10` até a de índice `6`;

- Uma volta é contabilizada assim que pelo menos um ciclista dê uma volta inteira no velódromo.

Decisões de projeto – Saídas

- Na saída por volta, indicamos qual é o `ID` do ciclista e a sua posição `x` no velódromo (`x` está no intervalo de `0` a `d-1`);
- Caso o ciclista esteja em uma volta anterior à volta atual, ou seja, está retardatário, é mostrado um `[RET]` ao lado da sua posição `x`;
- No momento em que um ciclista quebra, é mostrado na tela seu `ID` e também a volta em que o mesmo se encontrava.
- No relatório final da simulação, é indicado o placar, com o `ID` do ciclista, sua posição e o instante em ms em que cruzou a linha de chegada (levando em consideração o tempo real);
- O mesmo é feito para os ciclistas quebrados, exceto que ao invés de ser indicada a posição, é indicada a volta que o ciclista quebrou.

Ambiente e execução dos testes

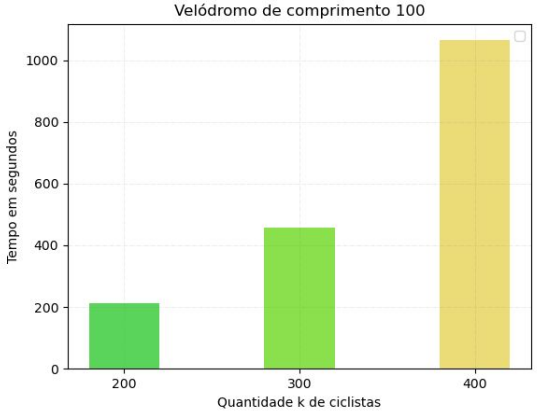
Ambiente

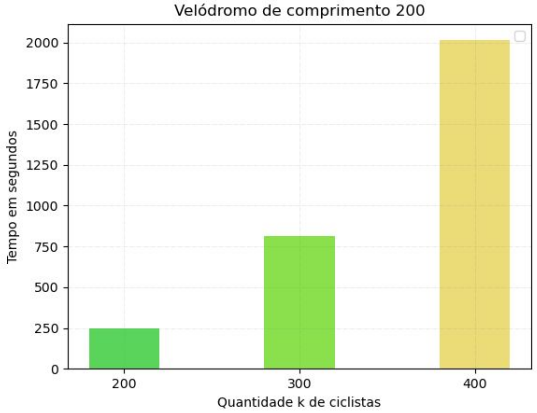
- Sistema operacional: Manjaro Linux 23.0.4 Uranos
- Modelo: 12th Gen Intel(R) Core(TM) i7-1255U
- CPU(S): 12

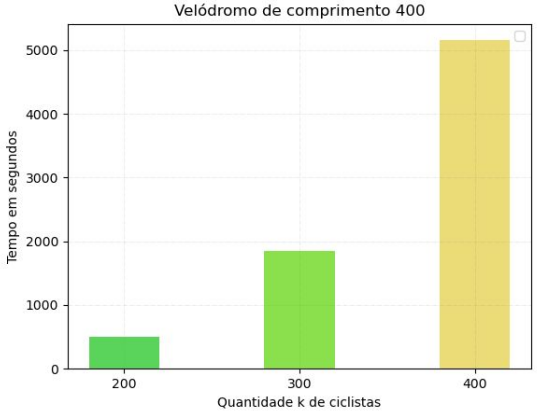
Testes

- Como alguns testes demoram muito para serem executados, foram feitos em média 10 testes para cada entrada;
- Os testes foram realizados combinando os tamanhos de pista e a quantidade de ciclistas, ao todo, 9 combinações.
- Os valores de `d` usados foram:100, 200 e 400. Já os valores de `k` utilizados foram: 200, 300 e 400.
- O comando utilizados para medir o tempo e memória foi `/usr/bin/time -f "%e %M"`.

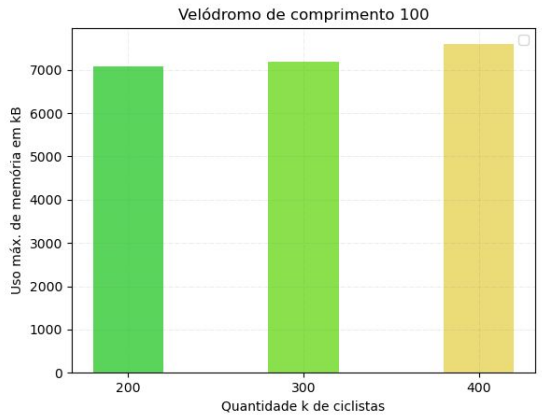
Resultados - Tempo

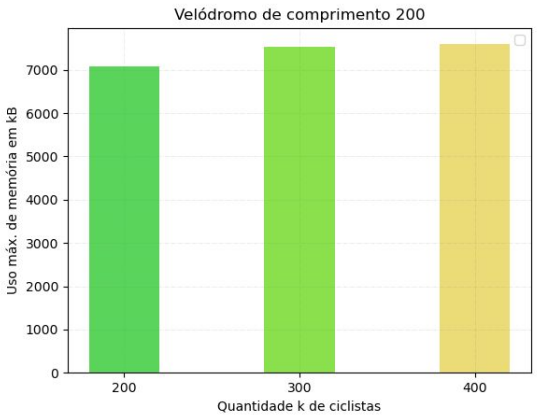


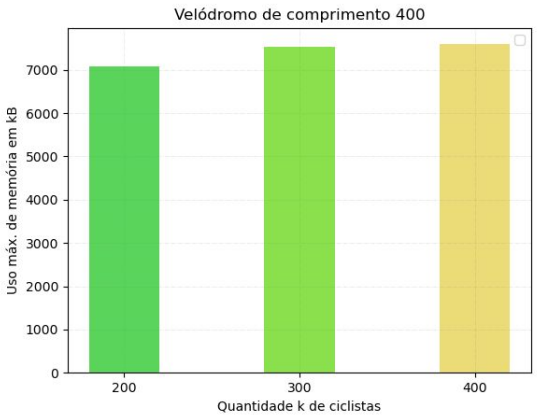




Resultados – Uso de memória







Análise dos resultados - Tempo

- ➔ Os resultados para o tempo ocorreram como esperando, quanto maior o comprimento do velódromo, maior o tempo de simulação;
- ➔ A mesma coisa ocorre quando aumentamos a quantidade de ciclistas na pista, o tempo de execução também aumenta consideravelmente;
- ➔ Os resultados, tanto de tempo, como de uso de memória, foram consistentes;
- ➔ Apesar dos testes não serem muito grandes, uma vez que o tamanho máximo da pista é `2500` e a quantidade máxima de corredores é `5 x (d-1)`, as execuções demoraram bastante;
- ➔ A abordagem utilizada para os `mutex` poderiam ser melhoradas a fim de deixar o programa mais rápido. Ao invés de fazer um semáforo binário para a matriz do velódromo inteira, fazer um semáforo por posição.

Análise dos resultados - Memória

- O uso pico de memória se mostrou constante, o que não era esperado. O mais comum a se pensar seria que os casos em que há mais ciclistas, e por consequência mais `threads` e informações para cada `threads`, o pico de uso de memória se mostraria maior;

- Este fato pode ter ocorrido pelas diferenças na quantidade de `threads` não serem tão distintas de um teste para outro;

- Ou seja, a alocação da matriz gerou uma leve diferença no pico de memória, contudo, pelas quantidades de `threads` a cada teste serem as mesmas, o valor se manteve estável.