



EP 2

MAC0422

Sistemas Operacionais

Anderson Andrei da Silva (8944025),
Bruno Boaventura Scholl (9793586),
Victor Seiji Hariki (9793694)

Simulador : Corrida por pontos





Modularização

- `typedef.h`
 - `struct Velodrome{};`
 - `struct Rider{};`
- `rider.c`
 - `ride();`
 - `coordinator();`
 - `step();`
- `velodrome.c`
 - `create_velodrome();`
 - `destroy_velodrome();`
- `ep2.c`
 - `main();`



Thread de ciclista

- Thread independente
- Faz o papel do ciclista

Estrutura:

- int id;
- pthread_t rider_t;
- Velodrome velodrome;
- int speed;
- bool broken;
- int score;
- int total_dist;
- int lane;
- int step_time;
- uint turn;
- int *overtake;



Esse é um ciclista

```
1.  set initial speed
2.  wait start
3.  if lap completed:
4.      choose new speed
5.      check if scores
6.      if lap is multiple of 15:
7.          decide if breaks
8.  go forward, left or right:
9.      if is a rider in front:
10.          wait rider in front, left or right do its turn
11.          checks max speed possible and reduce if needed
12.  notify global barrier // Also local barrier
13.  wait global barrier
14.
```



Gerenciador Velódromo

Gerencia:

- Ciclistas
- Pista
- Corrida

Estrutura:

- Length of velodrome, in meters
 - `int length;`
- Total number of riders
 - `uint rider_cnt;`
- Number of active riders
 - `uint a_rider_cnt;`
- Turn count
 - `uint lap_cnt;`
- This is the track. It stores rider ids (or -1)
 - `int **pista;`
- This is a vector of riders
 - `struct Rider *riders;`
- Coordinator thread
 - `pthread_t coordinator_t;`
- This is the barrier for riders to wait for start
 - `pthread_barrier_t start_barrier;`



Gerenciador Velódromo

Gerencia:

- Ciclistas
- Pista
- Corrida


Estrutura:

- Semaphore for writing in track array
 - `sem_t velodrome_sem;`
- How much time passes in one barrier round
 - `int round_time;`
- Array of arrive at barrier flags
 - `sem_t *arrive;`
- Flag to pass barrier
 - `sem_t *continue_flag;`
- Remaining riders
 - `int *placings;`
- Stack of placings by round
 - `int *s_indexes;`
 - `uint **placings_v;`
- Random generator semaphore
 - `sem_t rand_sem;`



Esse é um velódromo

1. Allocate the struct Velodrome
2. Start the semaphores (blocked)
3. Allocate the track
4. Allocate the stack of scores
5. Create the riders
6. Create the placings
7. Start the riders
8. Start the global barrier `//pthread_barrier_init`
9. Create the threads `//pthread_create`
10. Create the coordinator `//sem_t`
11. Wait the threads `//pthread_barrier_wait`
- 12.



Barreiras de sincronização

- Largada
 - Sincroniza a largada.
- Global
 - Sincroniza o disparo das threads a cada ciclo.



Semáforos

- Controla casos que existem mais de um ciclista em uma faixa da pista.



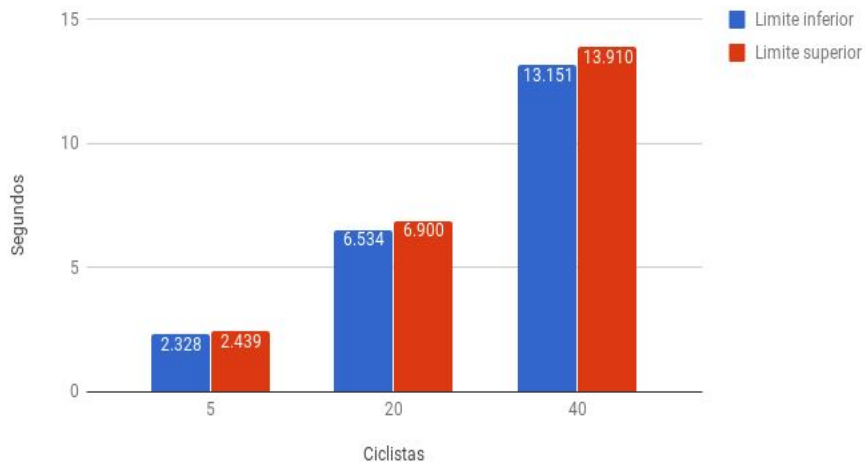
Ambiente de testes

- Executamos com 5, 20 e 40 ciclistas em cada velódromo
- Os velódromos foram os seguintes:
 - 250m e 300 voltas
 - 400m e 200 voltas
 - 600m e 120 voltas

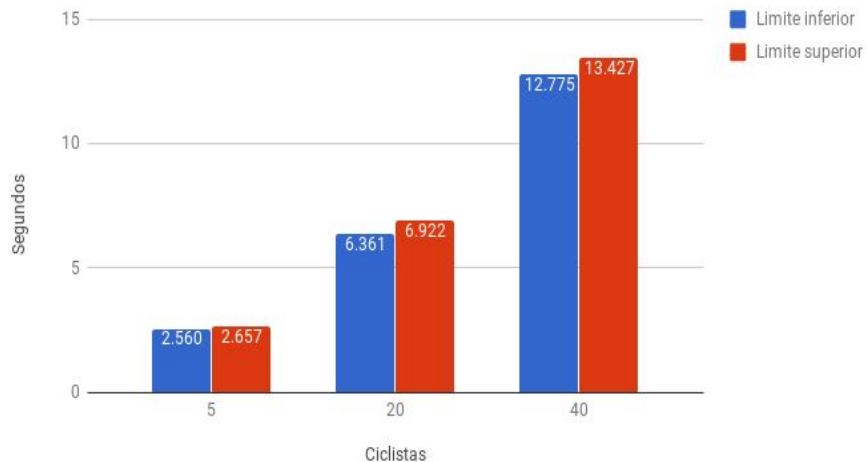
Tempo de execução

Não incluímos o intervalo de confiança pois este é facilmente derivado dos limites superior e inferior

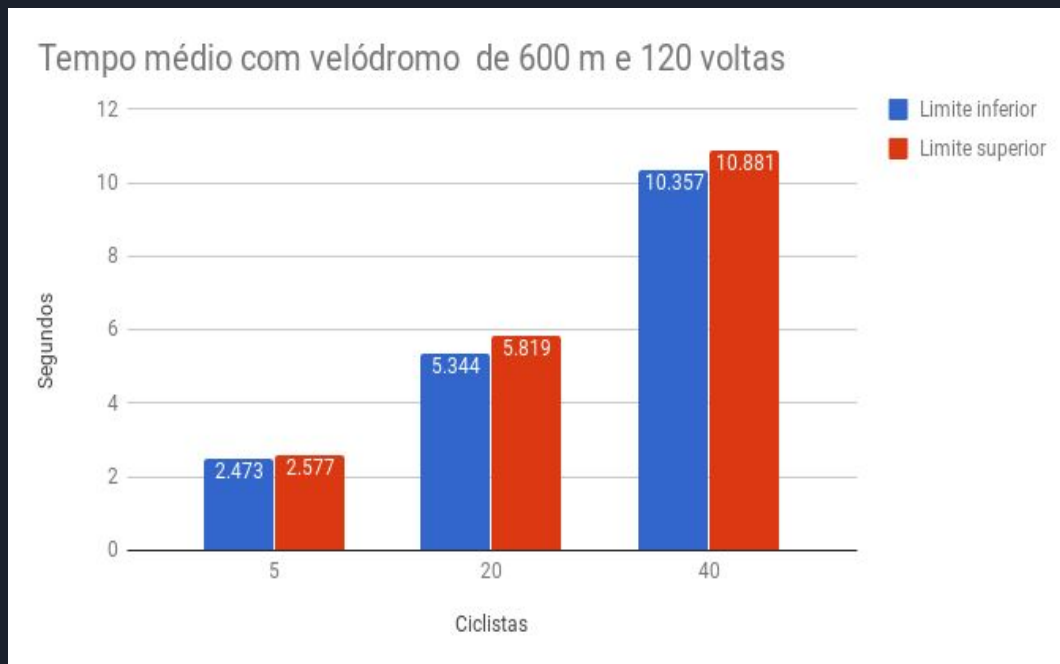
Tempo médio com velódromo de 250 m e 300 voltas



Tempo médio com velódromo de 400 m e 200 voltas

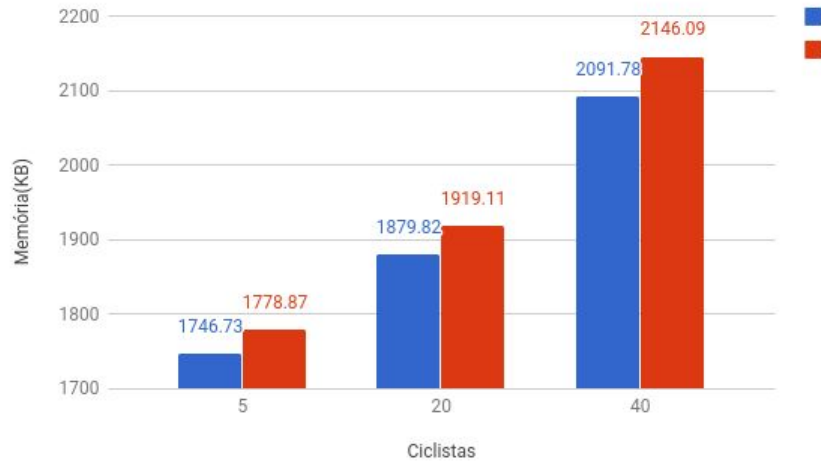


Tempo de execução

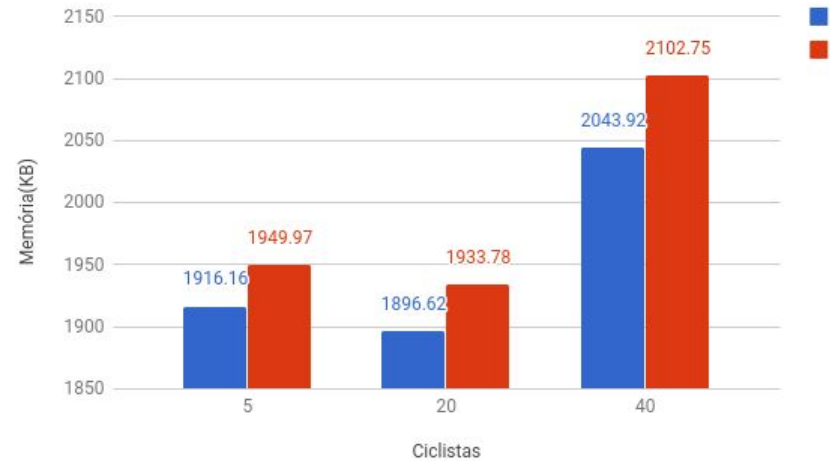


Uso de memória

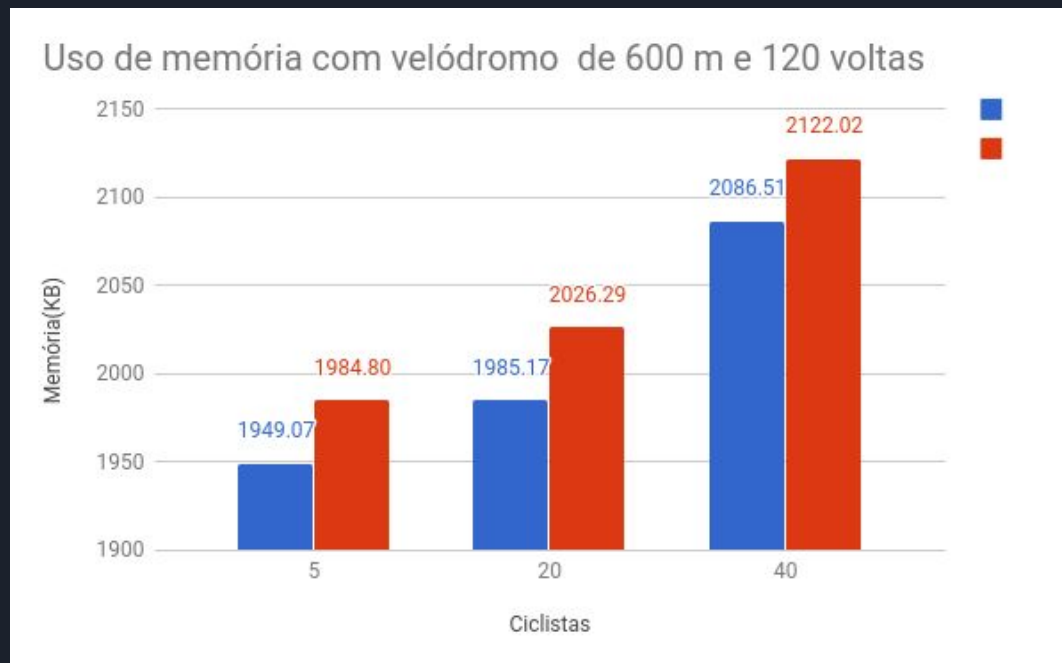
Uso de memória com velódromo de 250 m e 300 voltas



Uso de memória com velódromo de 400 m e 200 voltas



Uso de Memória





Conclusões a partir dos dados

- A quantidade de ciclistas (threads) é a principal influência de tempo e memória;
- Dados inconclusivos quanto à influência do número de voltas e comprimento da pista.

Fim

