MAC422 - EP2

Bruno Ferrero & Rodrigo Alves Souza

Problema

- Simular `n` ciclistas dando `v` voltas em uma pista `P` de tamanho `d` e 10 raias;
- A cada 10 voltas os 4 primeiros ciclistas da volta devem pontuar;
- Ganha quem somar mais pontos.

Solução: ideia geral ...

1. Criar `n` threads;
2. Usar o `main()` como um cronômetro que incrementa uma variável global (relógio) a cada *timestep* arbitrário (`usleep(timestep)`);
3. Sincronizar as `n` threads a cada `k * timestep` (60/20ms) intervalo de tempo;
4. Durante a sincronização registra-se as posições dos ciclistas em `P`;
5. Usar uma matriz `A[#voltas][#ciclistas]` global e registrar para cada volta a classificação de cada ciclista naquela volta;

Os ciclistas e a criação das `n` threads

```c
typedef struct param_ciclistas {
    int id;
    int posicao;
    int v;
    int d;
    int rank;
    int clock;
    int pontos;
    int quebrou;
} parametros_ciclistas;
```

```c
/*dispara os ciclistas (threads)*/
for (i = 0; i < n; i++) {
    arg[i].id = i+1;
    arg[i].posicao = 0;
    arg[i].v = v;
    arg[i].d = d;
    arg[i].rank = -1;
    arg[i].clock = -1;
    arg[i].pontos = 0;
    arg[i].quebrou = 0;
    if ( pthread_create( &threads[i], NULL, ciclista, (void*)&arg[i]) ) {
        fprintf(stderr,"Erro ao criar thread.");
        abort ();
    }
```

O cronômetro `main()` e o `relógio_global`

```
/* SIMULADOR: "cronometro" */
while (1) {
    usleep (timestep);
    relogio_global += 1;
    /*TODOS OS CICLISTAS TERMINARAM A CORRIDA*/
    if (finished == n)
        break;
}
```

A barreira e a sincronização

Solução para não perder os waits da barreira:

Uma thread só termina (`return NULL`) quando todas tiverem terminado suas corridas. Isso foi feito para evitar perder os `n waits` da barreira criada;

```
/* Inicializa a barreira */
pthread_barrier_init (&barreira_ciclo, NULL, n);


while(1) {
    pthread_barrier_wait(&barreira_ciclo);
    if (finished == n) return NULL;
}
```

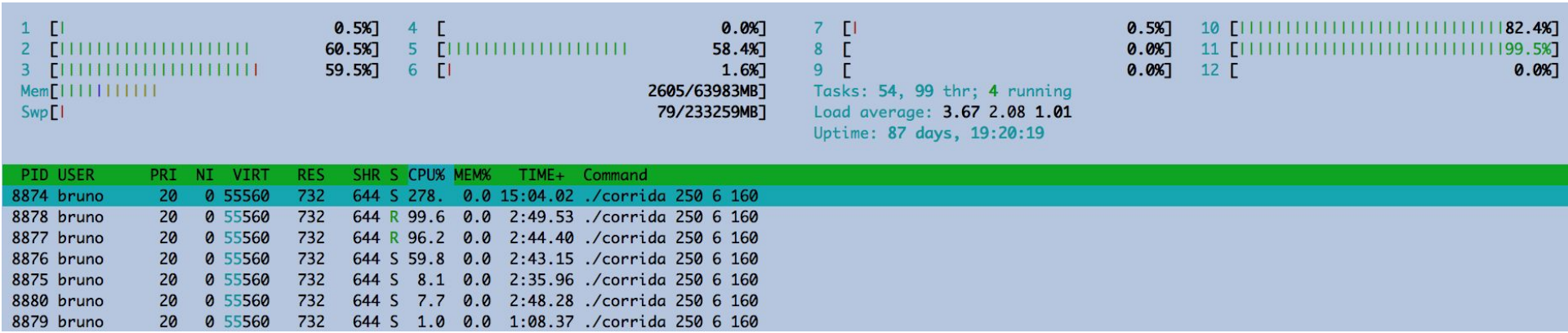
Escrevendo as posições com `mutex`

Ciclistas entrando na pista

```
/* COLOCA OS CICLISTAS NA PISTA */
pthread_mutex_lock ( &mutex1 );
for (j = 0; j < tamanho_pista && empty == 0; j++)
    for (i = 0; i < LARGURA && empty == 0; i++) {
        if (PISTA[i][j] == -1) {
            raia = i;
            posicao = j;
            empty = 1;
        }
    }
/* Primeiro slot livre */
PISTA[raia][posicao] = id;
/*printf ("pista[%d][%d] = %d\n",raia,posicao,PISTA[raia][posicao]);*/
pthread_mutex_unlock ( &mutex1 );
pthread_barrier_wait(&barreira_ciclo);
```

Execução

$ ./corrida 250 6 160 -> 6 threads e o processo pai (7 PIDS)



Execução: saída

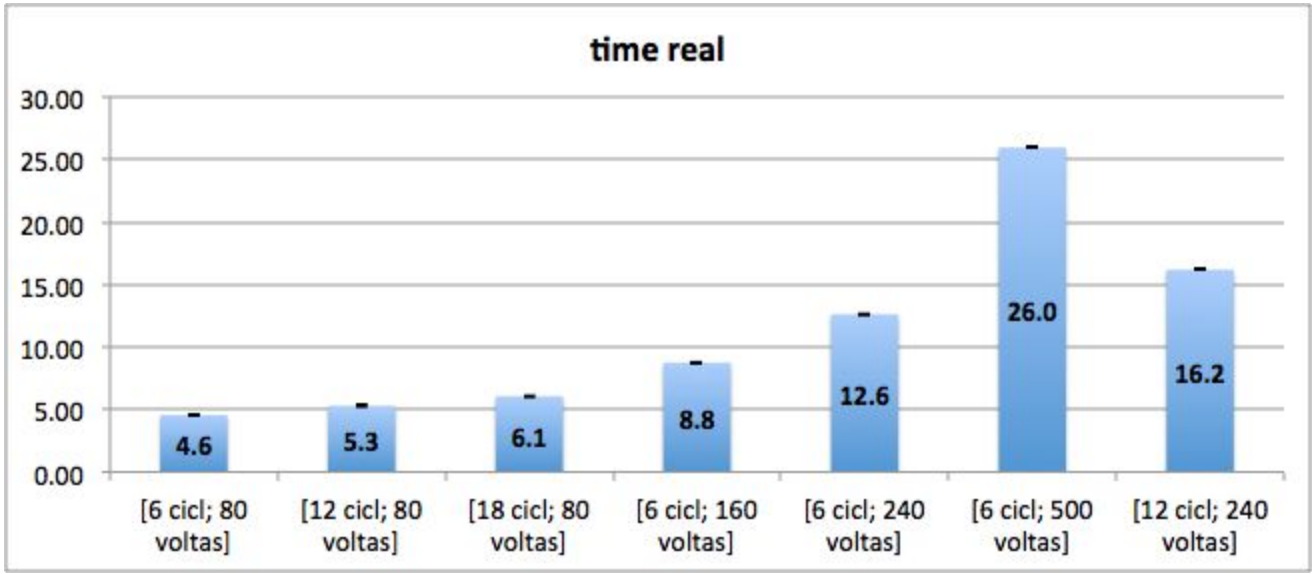
```
Volta 150: Id:  1 [ 55] | Id:  6 [ 58] | Id:  2 [ 28] | Id:  4 [  7] | Id:  3 [  4] | Id:  5 [ 13] |
Volta 151: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 152: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 153: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 154: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 155: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 156: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 157: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 158: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 159: Id:  6 | Id:  1 | Id:  2 | Id:  4 | Id:  3 | Id:  5 |
Volta 160: Id:  1 [ 60] | Id:  6 [ 61] | Id:  2 [ 30] | Id:  4 [  8] | Id:  3 [  4] | Id:  5 [ 13] |
Ciclista:  1 | Pontos:  60 | Terminou em:    3471780
Ciclista:  2 | Pontos:  30 | Terminou em:    3496740
Ciclista:  3 | Pontos:   4 | Terminou em:    3516740
Ciclista:  4 | Pontos:   8 | Terminou em:    3506760
Ciclista:  5 | Pontos:  13 | Terminou em:    1322880 | Quebrou na volta 60
Ciclista:  6 | Pontos:  61 | Terminou em:    3471780
bruno@polar:~/disciplinas/mac422/ep/ep2/mc422_02$
```

Resultados

- 30 execuções utilizando o `perf`
  - `$ perf stat -r 30 ./corrida 250 6 80 >> logs/log_06c_080v.txt 2>&1`

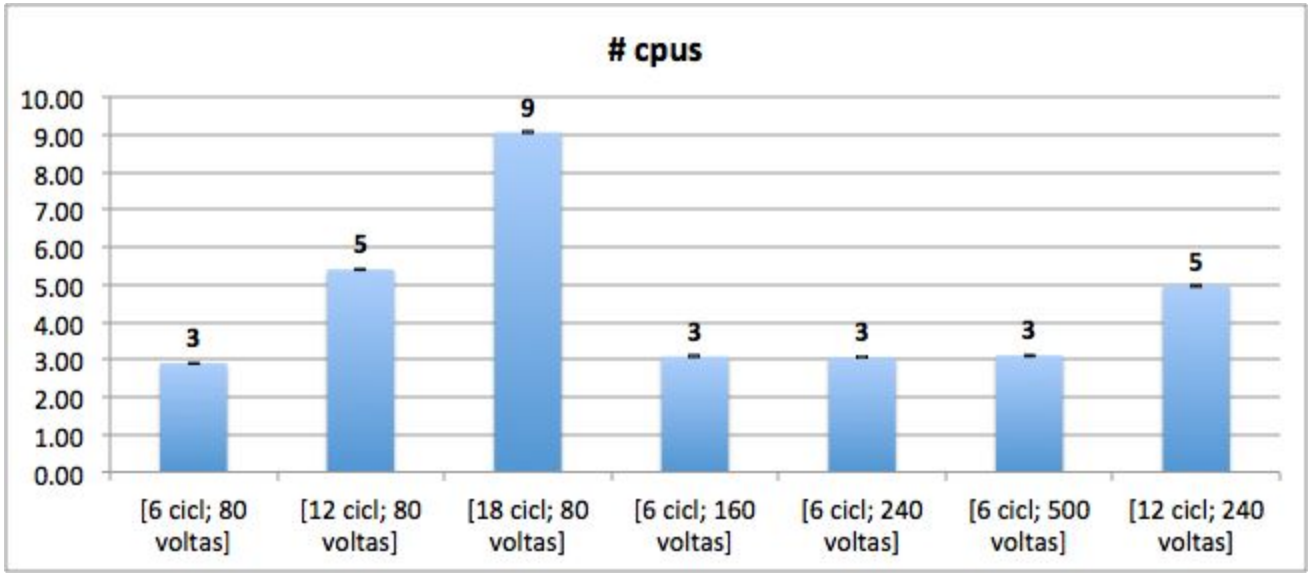
- Os testes considerados foram:
  - 6 ciclistas e 80 voltas; -> poucos ciclistas e poucas voltas
  - 12 ciclistas e 80 voltas; -> medios ciclistas e poucas voltas
  - 18 ciclistas e 80 voltas; -> muitos ciclistas e poucas voltas
  - 6 ciclistas e 160 voltas; -> poucos ciclistas e poucas voltas
  - 6 cicllistas e 240 voltas; -> poucos ciclistas e médias voltas
  - 6 ciclistas e 500 voltas; -> poucos ciclistas e muitas voltas
  - 12 ciclistas e 240 voltas; -> medios ciclistas e medios voltas

Resultados



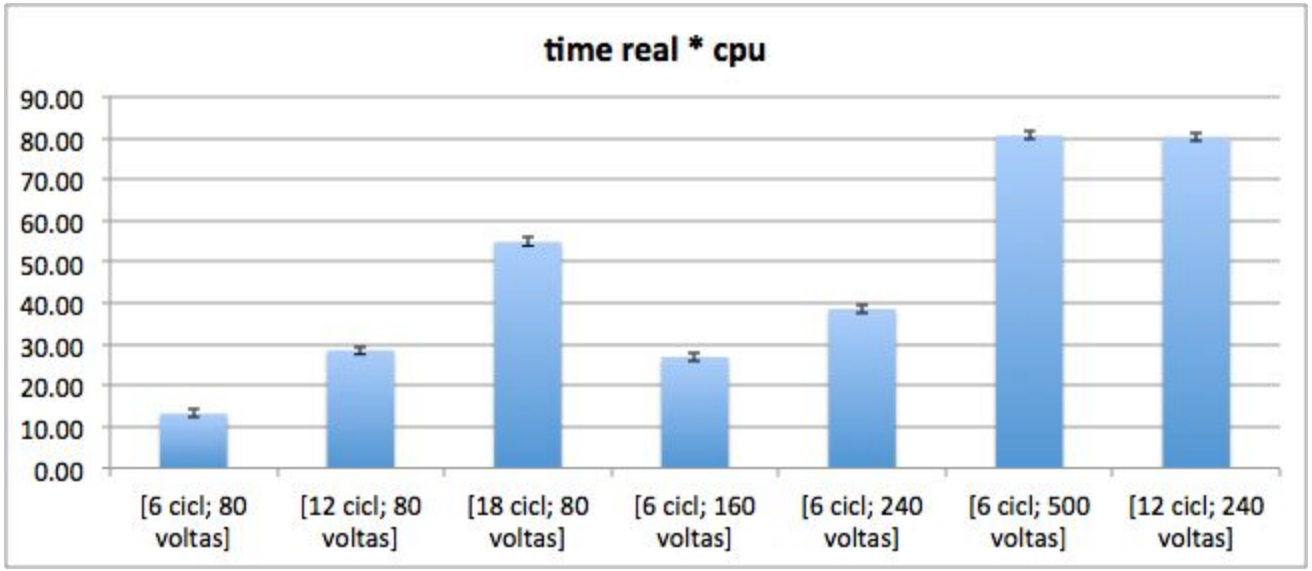
- O # de ciclistas aumenta (mas muito pouco) o tempo de execução;
- O tempo de execução aumenta proporcionalmente (linear) com o aumento da prova (número de voltas).

Resultados



- O # de ciclistas aumenta proporcionalmente o o número de cpus usadas;
- O número de cpus não varia com o tamanho da prova.

Resultados


time real * cpu
90.00
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
[6 cicl; 80 voltas]
[12 cicl; 80 voltas]
[18 cicl; 80 voltas]
[6 cicl; 160 voltas]
[6 cicl; 240 voltas]
[6 cicl; 500 voltas]
[12 cicl; 240 voltas]

Fim