Relatório - MAC0422 - EP2

Matheus Barbosa Silva e Gustavo Santos Morais

Instituto de Matemática e Estatística

05 de Novembro de 2020

Sumário

Sumário

Detalhes da implementação - Definições básicas

- **Pista**: a pista é uma matriz nx10, onde cada posição possui dois campos: um para guardar um ponteiro para um possível ciclista que poderá ocupá-la, e também um semáforo mutex, para controlar a mudança de conteúdo dessa posição;
- **Ciclista**: cada ciclista possui um ID próprio e guarda o ID de sua thread correspondente. Além disso, cada ciclista também possui suas informações pessoais, como a volta atual, instante, velocidade, posição etc.
  - **Thread**: cada ciclista executa uma Thread que é responsável por sincronizar o ciclista com a pista, atualizar a posição do ciclista e administrar eliminações ou classificações.
- **Volta**: cada volta começa no momento em que algum ciclista a inicia, e termina apenas quando o ciclista na última colocação termina esta mesma volta;

Detalhes da implementação - Colocações

- **Vetor de Colocações**: cada volta tem seu vetor de colocações correspondente. Os IDs dos ciclistas eliminados ou quebrados são armazenados ao fim do vetor, enquanto os IDs dos demais são classificados na mesma ordem em que chegam a partir do início do vetor;
- **Matriz de colocações**: uma matriz de colocações é alocada dinamicamente, de modo que cada linha desta matriz corresponda ao vetor de colocações de uma volta (e cada coluna é alocada à medida que as voltas avançam);
- *Obs.:* O acesso a cada vetor de colocações é controlado por um mutex, de modo a não permitir que as colocações de uma mesma volta sejam alteradas simultaneamente.

Detalhes da implementação - Eventos

- **Início de Voltas**: para cada início de volta, aloca-se dinamicamente um vetor de colocações com o tamanho necessário para armazenar a mesma quantidade de IDs que na volta anterior, decrescida de um (no caso de eliminações) e/ou dos possíveis ciclistas quebrados;
- **Quebra de Ciclistas**: em inícios de voltas múltiplas de 6 onde há mais de 5 ciclistas ainda competindo, cada ciclista realiza um sorteio com 5% de chance de quebrar e ser, em seguida, eliminado da competição. Cada ciclista quebrado preenche sua posição de eliminado nas voltas posteriores já iniciadas.

Detalhes da implementação - Eventos

- **Largada**: os ciclistas são posicionados, inicialmente, de forma intercalada com 5 ciclistas em cada metro da pista (ora à esquerda e ora à direita, sempre lado a lado);
- **Adição à Fila de Eliminações**: um ciclista é adicionado à fila de eliminações e será eliminado na próxima volta par quando este foi o penúltimo colocado de uma volta, mas o último colocado dessa mesma volta quebra antes de concluí-la;
- **Velocidade de 90Km/h**: no momento da eliminação do terceiro colocado, há 10% de chance de que **um** dos ciclistas (aleatoriamente sorteado) passe a andar a 90Km/h até o fim da corrida. É feita uma requisição à Thread controladora (através de uma flag) para que as mudanças de velocidade e intervalo sejam feitas de modo sincronizado entre os ciclistas.

Detalhes de implementação - Sincronização

- **Barreiras**: para sincronizar todos os elementos da corrida, de tal forma que não haja leitura ou escrita em momentos indesejados (como interação de ciclistas em instantes diferentes, por exemplo), usaram-se barreiras. Assim, todos os ciclistas aguardam os demais em alguns pontos do programa, fazendo com que as condições da pista e dos ciclistas estejam no mesmo ritmo após a liberação da barreira. Quando ocorre alguma eliminação, essas barreiras são destruidas e criadas novamente, para esperar menos ciclistas no próximo instante.
- No total, foram utilizadas 3 barreiras para sincronizar o início de um instante (após os deslocamentos dos ciclistas), sorteio de velocidades, e atualização das posições dos ciclistas.

Sumário

Execução dos Experimentos - Memória

- **Experimentos com a memória:** Para testar a relação entre o uso de memória do simulador e os valores de cada parâmetro, fez-se 30 execuções usando cada um dos seguintes valores:
  - **Parâmetro d** (comprimento da pista): 250 (denominado pequeno), 2.500 (médio) e 25.000 (grande). Para que os efeitos da mudança de d fossem comparáveis, fixou-se o valor de n como 20 em todas as execuções do experimento.
  - **Parâmetro n** (quantidade de ciclistas): 5 (denominado pequeno), 50 (médio) e 500 (grande). Para que os efeitos da mudança de n fossem comparáveis, fixou-se o valor de d como 1000 em todas as execuções do experimento.
- **Comandos:**

  sudo /usr/bin/time -f “%M” -o arquivo.txt -a ./ep2 d 20
  sudo /usr/bin/time -f “%M” -o arquivo.txt -a ./ep2 1000 n

Execução dos Experimentos - Tempo

- **Experimentos com o tempo:** Para testar a relação entre o tempo de execução do simulador e os valores de cada parâmetro, fez-se 30 execuções usando cada um dos seguintes valores:
  - **Parâmetro d** (comprimento da pista): 250 (denominado pequeno), 2.500 (médio) e 25.000 (grande). Para que os efeitos da mudança de d fossem comparáveis, fixou-se o valor de n como 20 em todas as execuções do experimento.
  - **Parâmetro n** (quantidade de ciclistas): 5 (denominado pequeno), 50 (médio) e 500 (grande). Para que os efeitos da mudança de n fossem comparáveis, fixou-se o valor de d como 1000 em todas as execuções do experimento.
- **Comandos:**

  sudo /usr/bin/time -f “%e” -o arquivo.txt -a ./ep2 d 20
  sudo /usr/bin/time -f “%e” -o arquivo.txt -a ./ep2 1000 n

Uso de Memória

- Em ambos os gráficos a seguir, usa-se um intervalo de confiança de 95%.

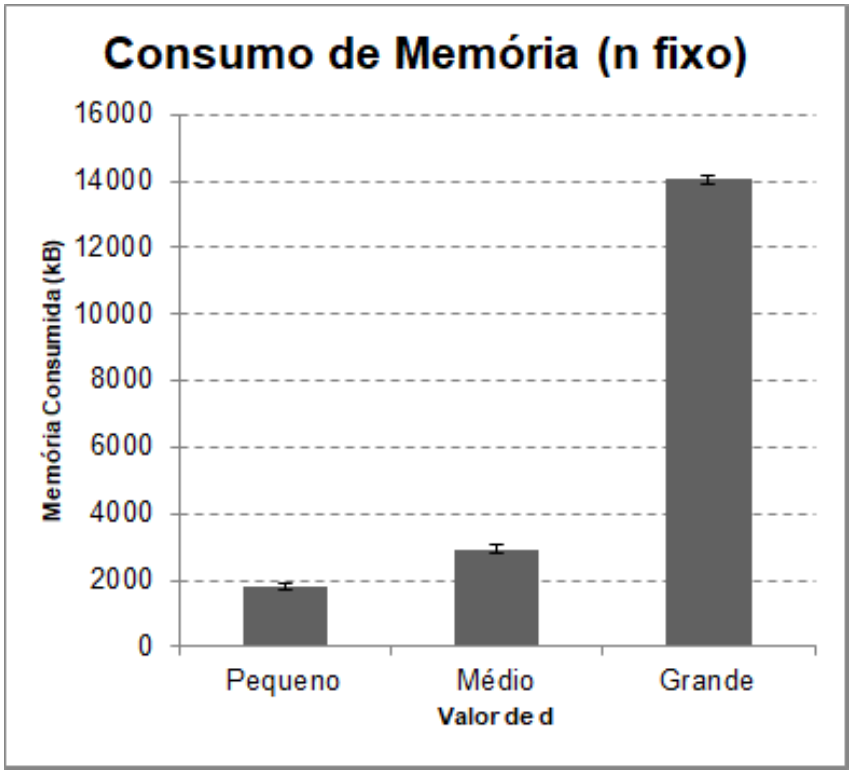

Figura 1: Gráfico do Uso de Memória para o parâmetro 'n' fixo

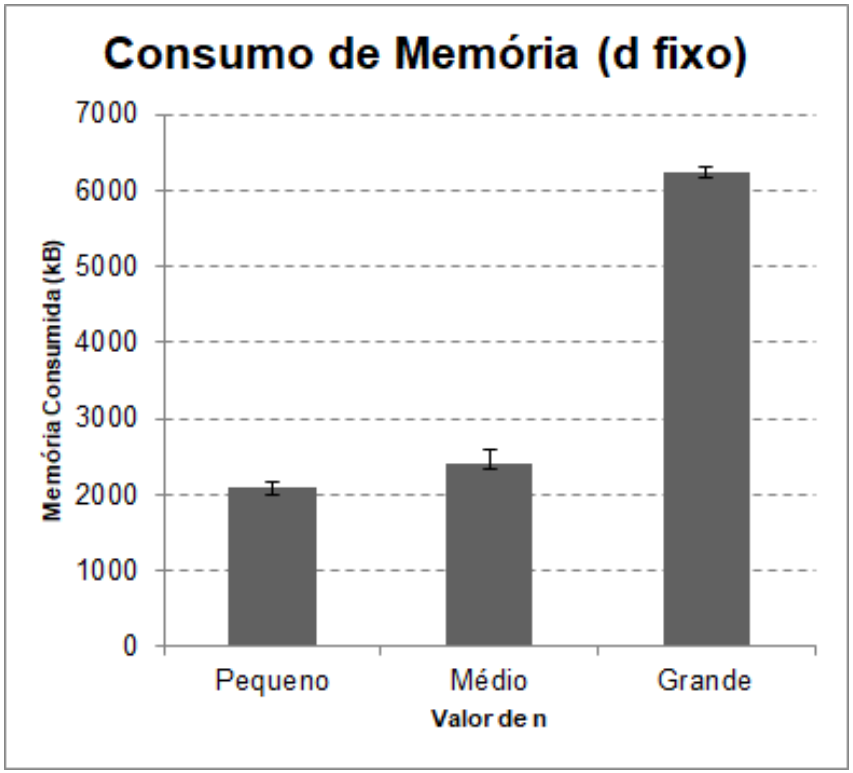

Figura 2: Gráfico do Uso de Memória para o parâmetro 'd' fixo

Uso de Memória - Conclusões

- Assim como esperado para ambos os casos, o consumo de memória cresce (em média) à medida que os parâmetros d e n crescem;
- Como esperado, nota-se que o uso de memória aumenta mais expressivamente com o aumento dos valores para o parâmetro d (dado que cada metro adicional da pista demanda a alocação de 10 posições na matriz);
- Como esperado, o consumo de memória varia – dado que, dentre outros motivos, os tamanhos dos vetores de colocações são alterados a depender da quantidade de ciclistas quebrados.

Tempo de Execução

- Em ambos os gráficos a seguir, usa-se um intervalo de confiança de 95%.

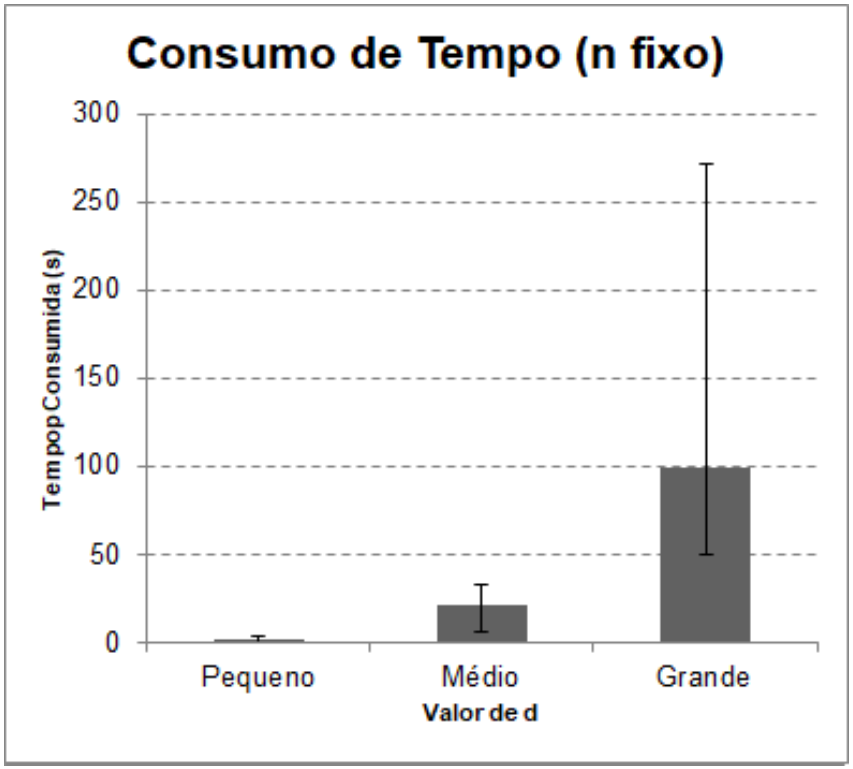

Figura 3: Gráfico do Tempo de Execução para o parâmetro 'n' fixo

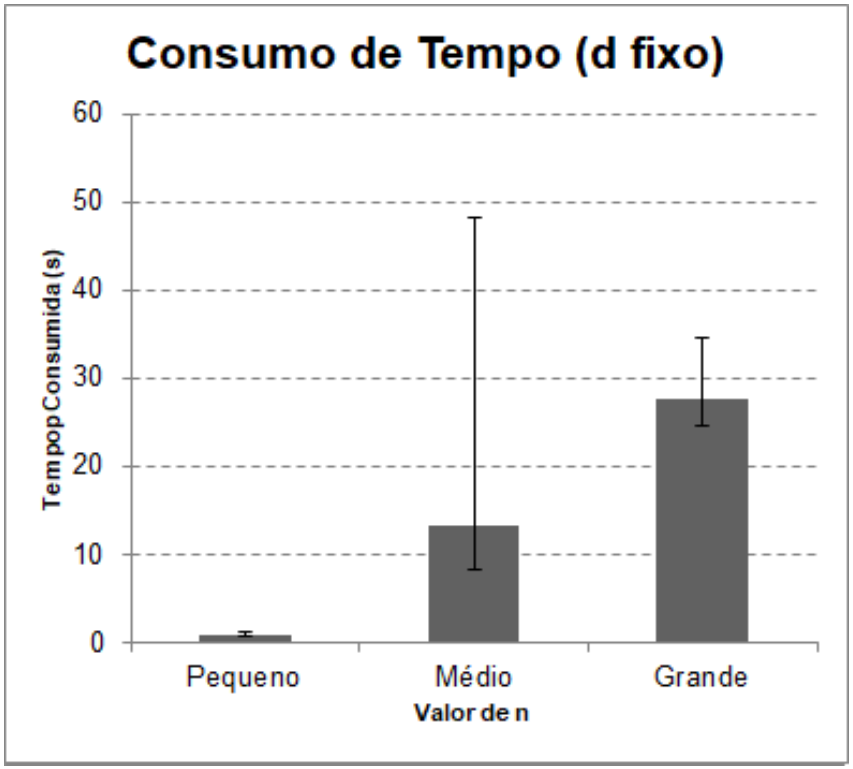

Figura 4: Gráfico do Tempo de Execução para o parâmetro 'd' fixo

Tempo de Execução - Conclusões

- Como esperado, o tempo médio de execução cresce à medida que os parâmetros d e n crescem;
- Assim como esperado para d e n pequenos, há menor variação no consumo de tempo (e cresce conforme d e n aumentam, dado o aumento da quantidade de casos possíveis);
- Como esperado, o aumento do valor de n causa (em média) maior aumento no consumo de tempo do que o aumento de d, já que desta forma, a pista tende a tornar-se mais congestionada e os ciclistas movimentam-se mais lentamente;
- Os limites dos intervalos de confiança variam muito entre os dois casos e não estão de acordo com o esperado. Isso pode ocorrer por conta de limitações de hardware ou controle do SO quanto ao uso de recursos por parte do usuário (o programa pode ter sido suspenso antes de ser finalizado).