

Exercício de Programação 2 (EP2) - Programação Paralela

1. Introdução

Dado um conjunto A com n números (distintos), o problema de seleção (Selection Problem) é o de encontrar o elemento $x \in A$ tal que x é maior que i-1 elementos de A. Existem diversos algoritmos para resolver o problema e neste trabalho vamos considerar o Randomized-Select (discutido em detalhes em "Introduction to Algorithms. by Cormen, Leiserson, Rivest and Stein") apresentado no Algoritmo abaixo.

Algorithm 1 Randomized select.

```
1: procedure RANDOMIZED-SELECT(A, p, r, i)
      if p == r then
2:
          return A[p]
3:
      end if
4:
      q = RANDOMIZED - PARTITION(A, p, r)
5:
      k = q - p + 1
6:
      if i == k then
7:
          return A[q]
8:
      else if i < k then
9:
          RANDOMIZED - SELECT(A, p, q - 1, i)
10:
      else
11:
12:
          RANDOMIZED - SELECT(A, q + 1, r, i - k)
      end if
13:
14: end procedure
   procedure RANDOMIZED-PARTITION(A, p, r)
      i = RANDOM(p, r)
16:
      exchange A[r] with a[i]
17:
      return PARTITION(A, p, r)
18:
19: end procedure
  procedure PARTITION(A, p, r)
      x = A[r]
21:
      i = p - 1
22:
      for j = ptor - 1 do
23:
          if A[j] \le x then
24:
             i = i + 1
25:
             exchange A[i] with A[j]
26:
          end if
27:
      end for
28:
      exchange A[i+1] with a[j]
29:
30:
      return i+1
31: end procedure
```

2. Enunciado

Deve-se implementar um algoritmo usando C/C++ e paralelizar o mesmo utilizando Pthreads. Você deve utilizar um pool de threads para reaproveitá-las em caso de assinalamento dinâmico de tarefas. Você deverá



avaliar diversas granularidades de tarefas e utilizar uma fila para comunicação entre a thread master e as trabalhadoras.

2.1. Entrada do Programa

O programa deverá receber quatro parâmetros como entrada: um inteiro n que representa o tamanho do conjunto A, i, type e p: quantidade the threads. O vetor A deve ser inicializados com número aleatórios.

2.2. Saída do programa

O programa deverá imprimir o ith número em A, o tempo de execução em segundos com precisão de seis casas decimais e os valores em A, conforme parâmetro de entrada type:

- "time": Uma linha contendo o tempo de execução do programa.
- "all": A primeira linha contendo a lista de elementos em A, a segunda contendo o ith valor e a terceira com o tempo de execução.

3. Entregas e avaliações

O código deverá ser entrege no moodle da disciplina juntamente com o relatório.

3.1. Código

- O código deve conter todo o fonte do seu programa (C ou C++) juntamente com instruções claras sobre sua compilação e execução (Makefile). Você deverá entregar o código sequencial e o paralelo.
- O ambiente de testes será Linux GCC.
- Programas com erros de compilação receberão nota 0.
- Programas que não cumprirem os requisitos (resultados incorretos, erros durante a execução...) descritos na seção anterior receberão nota 0.
- Programas sem boa documentação no código terão a sua nota reduzida.
- Programas desorganizados (nomes de variáveis/funções ruins, falta de identação,...) terão a sua nota reduzida.

3.2. Relatório

Juntamente com o seu código você deverá entregar um relatório (máximo de 5 páginas) que contenha:

- Uma explicação detalhada de como o particionamento das tarefas foi feito entre os processos na sua implementação.
- Você deverá avaliar diversas granularidade das tarefas executadas, similarmente ao que é feito the chunkSize em OpenMP. Tabelas e gráficos do speedup, eficiência e tempo variando os valores de N e P deve ser apresentados. Não esqueça de incluir a especificação da máquina utilizada para a execução recomendo o uso da máquina n1-standard-8 no google cloud. Você deverá utilizar p até o valor de 8.
- Interpretação dos valores mostrados na tabela anterior. Descreva e interprete a variação do speedup e da eficiência em relação à n e p. Explique os resultados obtidos levando em consideração as características da máquina utilizada.
- Análise da escalabilidade da sua implementação. Ela é escalável? Apresenta escalabilidade forte ou fraça?
- Análise do balanceamento de carga. Todos os processos recebem a mesma quantidade de trabalho?
 Todos acabam a execução aproximadamente ao mesmo tempo?

Nesses experimentos sugiro utilizar a execução com o parâmetro time, para evitar impacto do I/O no tempo.



3.3. Desempenho

• Os desempenhos tanto da versão sequencial (otimize seu código) quanto da versão paralela do seu algoritmo serão avaliados.