Universidade de São Paulo - ICMC - BCC
SSC0903 - Computação de Alto Desempenho (2022/2)
Segundo Trabalho Prático (TB2) - Resolução em Grupo

Turma: A
Grupo: 4
Nomes dos integrantes deste grupo que resolveram o trabalho:
Dikson Ferreira dos Santos
Eduardo Rodrigues Amaral
Fabio Dias da Cunha
Laís Saloum Deghaide

Resultados das execuções

A tabela abaixo contém os resultados dos experimentos realizados. Cada linha possui os seguintes dados: número de vértices do grafo (N), números de *cores* usados pelo algoritmo paralelo (C), speedup, eficiência, tempo de resposta sequencial sem considerar E/S e o tempo de resposta paralelo sem considerar E/S. Tais experimentos foram gerados a partir de um script para realizar testes em Python.

Nota-se que o algoritmo paralelo possui tempos de resposta menores que o sequencial em grande parte dos experimentos. Tal melhoria pode ser explicada pela divisão do trabalho entre os processos.
Para N de tamanhos pequenos (até 7) o algoritmo sequencial possui tempos de resposta menores que o paralelo. Tal ineficiência pode ser explicada devido ao grande custo para gerar processos e realizar comunicações entre eles.
Para N de 8 a 13, o algoritmo paralelo supera o sequencial, sendo até 3.71x mais rápido. A eficiência é maior para quantidades baixas de cores. A eficiência é próxima de 1 para grande parte das execuções, o que mostra que o gargalo da comunicação e da geração de processor perde relevância rapidamente com o aumento de N.

O grupo gostaria de testar para N maiores, porém o tempo de resposta escala muito rapidamente tornando impraticável realizar os experimentos para N maiores.

N	C	Speedup	Efficiency	Sequential Time	Parallel Time
2	2	0.11	0.06	0.000001s	0.000009s
2	3	0.08	0.03	0.000001s	0.000012s
2	4	0.07	0.02	0.000001s	0.000015s
3	2	0.04	0.02	0.000001s	0.000024s
3	3	0.02	0.01	0.000001s	0.000043s
3	4	0.03	0.01	0.000001s	0.000030s
4	2	0.04	0.02	0.000001s	0.000027s

4	3	0.04	0.01	0.000001s	0.000028s
4	4	0.03	0.01	0.000001s	0.000038s
5	2	0.08	0.04	0.000002s	0.000026s
5	3	0.06	0.02	0.000002s	0.000032s
5	4	0.04	0.01	0.000002s	0.000053s
6	2	0.23	0.12	0.000007s	0.000030s
6	3	0.13	0.04	0.000007s	0.000053s
6	4	0.05	0.01	0.000007s	0.000130s
7	2	0.82	0.41	0.000036s	0.000044s
7	3	0.18	0.06	0.000036s	0.000195s
7	4	0.25	0.06	0.000036s	0.000143s
8	2	2.17	01.09	0.000380s	0.000175s
8	3	2.35	0.78	0.000380s	0.000162s
8	4	1.23	0.31	0.000380s	0.000309s
9	2	1.90	0.95	0.002192s	0.001151s
9	3	1.71	0.57	0.002192s	0.001284s
9	4	1.95	0.49	0.002192s	0.001122s
10	2	1.74	0.87	0.021140s	0.012155s
10	3	2.85	0.95	0.021140s	0.007417s
10	4	2.37	0.59	0.021140s	0.008926s
11	2	1.99	0.99	0.225786s	0.113714s
11	3	2.41	0.80	0.225786s	0.093751s
11	4	2.38	0.59	0.225786s	0.094888s
12	2	1.83	0.91	2.634061s	1.440069s
12	3	2.14	0.71	2.634061s	1.231970s
12	4	02.07	0.52	2.634061s	1.275115s
13	2	1.98	0.99	33.316750s	16.839196s
13	3	2.89	0.96	33.316750s	11.539435s
13	4	3.71	0.93	33.316750s	8.991809s

Script *Python* para realização dos testes:

```
import subprocess
import os
import json
import re

runs = []
```

```
def run(N):

   runs = []

   os.system(f"../pvcSeq {N} > outSeq")
   with open("outSeq", 'r') as f:
       output = f.read()
       timeSeq = re.search(
                r'Tempo de resposta sem considerar E\/S, em segundos:
(\d+\.\d+)s', output).group(1)
       timeSeq = float(timeSeq)

   os.system('sed -i "$ d" outSeq')

   for c in range(2, 5):
       os.system(f"mpirun -np {c} ../pvcPar {N} > outPar")
       with open("outPar", 'r') as f:
           output = f.read()
           timePar = re.search(
                  r'Tempo de resposta sem considerar E\/S, em segundos:
(\d+\.\d+)s', output).group(1)
           timePar = float(timePar)

       os.system('sed -i "$ d" outPar')

       diffCode = subprocess.call(["diff", "outSeq", "outPar"])

       runs.append({
           "N": N,
           "C": c,
           "timeSeq": timeSeq,
           "timePar": timePar,
           "error": diffCode != 0
       })

   os.unlink("outSeq")
   os.unlink("outPar")

   return runs


runs = []
```

```
for N in range(2, 14):
   runs += run(N)
```