TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BTVN NHÓM 10:
CÀI ĐẶT THUẬT TOÁN SONG SONG

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

Nhóm 9

Sinh viên thực hiện:

Họ và tên	MSSV
Bùi Ngọc Thiên Thanh	23521436
Nguyễn Thái Sơn	23521356

Thành phố Hồ Chí Minh, 2024

Mục lục

A. Bài tập 1

1 Giới thiệu

Bài tập yêu cầu xây dựng thuật toán kiểm tra số nguyên tố song song và so sánh thời gian thực hiện giữa phương pháp tuần tự và song song. Các số nguyên đầu vào sẽ được kiểm tra xem có phải là số nguyên tố hay không. Chúng ta sẽ sử dụng Python để cài đặt các thuật toán này.

2 Thuật Toán Kiểm Tra Số Nguyên Tố

Để kiểm tra một số X có phải là số nguyên tố hay không, ta sẽ sử dụng phương pháp kiểm tra bằng cách thử chia X với các số từ 2 đến $\sqrt{X}$. Nếu không tìm thấy ước nào chia hết thì X là số nguyên tố.

2.1 Thuật Toán Kiểm Tra Số Nguyên Tố Tuần Tự

```
import math

def is_prime_sequential(x):
if x <= 1:
    return False
for i in range(2, int(math.sqrt(x)) + 1):
    if x % i == 0:
        return False
return True
```

2.2 Thuật Toán Kiểm Tra Số Nguyên Tố Song Song

Để kiểm tra số nguyên tố song song, ta chia công việc thành nhiều phần và thực hiện đồng thời. Sử dụng module `concurrent.futures` của Python để chạy song song.

```
import math
import concurrent.futures

```

```
    def is_prime_parallel(x):
    if x <= 1:
        return False

    def check_range(start, end):
        for i in range(start, end):
            if x % i == 0:
                return False
        return True

    num_threads = 4
    step = (int(math.sqrt(x)) + 1) // num_threads
    ranges = [(2 + i * step, 2 + (i + 1) * step) for i in
       range(num_threads)]

    with concurrent.futures.ThreadPoolExecutor() as executor:
        futures = [executor.submit(check_range, start, end) for
           start, end in ranges]
        for future in concurrent.futures.as_completed(futures):
            if not future.result():
                return False
    return True
```

3 So Sánh Thời Gian Thực Hiện

Để so sánh thời gian thực hiện giữa phương pháp song song và tuần tự, ta sử dụng module `time` của Python để đo thời gian thực thi.

```
    import time

    def test_sequential(x):
    start_time = time.time()
    result = is_prime_sequential(x)
    end_time = time.time()
    print(f"X = {x} (Sequential) - Result: {result}, Time:
       {end_time - start_time:.5f} seconds")

```

```
    def test_parallel(x):
    start_time = time.time()
    result = is_prime_parallel(x)
    end_time = time.time()
    print(f"X = {x} (Parallel) - Result: {result}, Time:
        {end_time - start_time:.5f} seconds")
```

4 Nhập Dữ Liệu và Kiểm Tra

Chương trình cho phép người dùng nhập vào một số nguyên và kiểm tra xem số đó có phải là số nguyên tố hay không bằng cả hai phương pháp.

```
def get_input():
try:
    x = int(input("Nhap so nguyen X: "))
    return x
except ValueError:
    print("Vui long nhap so nguyen hop le!")
    return None

def main():
x = get_input()
if x is not None:
    # Chay thu ca 2 phuong phap
    test_sequential(x)
    test_parallel(x)

if __name__ == "__main__":
main()
```

Link code: Bài 1

5 Kết Quả

```
PS D:\THANH\HK3\CS112\Assignment\Group 10>
signment\Group 10\BTVN nhóm 10\bai1.py"
X = 1000000000000091 (Sequential) - Result: True, Time: 3.41271 seconds
X = 10000000000000099 (Sequential) - Result: True, Time: 11.11892 seconds
X = 100000000000000049 (Sequential) - Result: True, Time: 54.49449 seconds
X = 1000000000000091 (Parallel) - Result: True, Time: 5.68522 seconds
X = 10000000000000099 (Parallel) - Result: True, Time: 17.23884 seconds
X = 100000000000000049 (Parallel) - Result: True, Time: 36.83287 seconds
```

6 Kết Luận

- Phương pháp song song có thể cải thiện hiệu suất kiểm tra số nguyên tố đối với các số lớn, nhưng hiệu quả còn phụ thuộc vào số lượng lõi xử lý của hệ thống.
- Đối với các số nhỏ hoặc hệ thống có số lõi thấp, phương pháp tuần tự có thể không có sự khác biệt rõ rệt so với song song.

B. Bài tập 2

1 Giới thiệu

Bài toán này yêu cầu xây dựng thuật toán nhân ma trận song song và tuần tự, so sánh thời gian thực hiện của chúng với các ma trận có kích thước khác nhau. Đoạn mã Python dưới đây thực hiện phép nhân ma trận song song và tuần tự.

2 Thuật toán Nhân Ma Trận

2.1 Mã Python

```python
import random
import time
from multiprocessing import Pool, cpu_count


def generate_matrix(rows, cols):
```

```
7     return [[random.randint(1, 10) for _ in range(cols)] for _
          in range(rows)]
8
9
10 def multiply_matrices_sequential(A, B):
11     rows_A, cols_A = len(A), len(A[0])
12     rows_B, cols_B = len(B), len(B[0])
13
14     C = [[0 for _ in range(cols_B)] for _ in range(rows_A)]
15     for i in range(rows_A):
16         for j in range(cols_B):
17             for k in range(cols_A):
18                 C[i][j] += A[i][k] * B[k][j]
19     return C
20
21
22 def compute_chunk(args):
23     A, B, start_row, end_row = args
24     cols_B = len(B[0])
25     cols_A = len(A[0])
26     C_chunk = []
27     for i in range(start_row, end_row):
28         row = []
29         for j in range(cols_B):
30             value = sum(A[i][k] * B[k][j] for k in range(cols_A))
31             row.append(value)
32         C_chunk.append(row)
33     return C_chunk
34
35 def multiply_matrices_parallel(A, B):
36     rows_A, cols_A = len(A), len(A[0])
37     rows_B, cols_B = len(B), len(B[0])
38
39     num_processes = cpu_count()
40     chunk_size = rows_A // num_processes
41     ranges = [(A, B, i, min(i + chunk_size, rows_A)) for i in
          range(0, rows_A, chunk_size)]
42
```

```
    with Pool(num_processes) as pool:
        results = pool.map(compute_chunk, ranges)

    return [row for chunk in results for row in chunk]

def main():

    test_cases = [10, 30, 50, 100, 300, 500]
    for size in test_cases:
        A = generate_matrix(size, size)
        B = generate_matrix(size, size)

        print(f"Testing with {size} x {size} matrices")


        start_time = time.time()
        result_seq = multiply_matrices_sequential(A, B)
        seq_time = time.time() - start_time
        print(f"Tuan tu: Time = {seq_time:.6f}s")


        start_time = time.time()
        result_par = multiply_matrices_parallel(A, B)
        par_time = time.time() - start_time
        print(f"Song song: Time = {par_time:.6f}s")

        print(f"Toi uu thoi gian: {seq_time / par_time:.2f}x")

if __name__ == "__main__":
    main()
```

Link code: Bài 2

3 Kết Quả

```
PS D:\THANH\HK3\CS112\Assignment\Group 10>
Testing with 10 × 10 matrices
Tuan tu: Time = 0.000000s
Song song: Time = 0.878881s
Toi uu thoi gian: 0.00x
Testing with 30 × 30 matrices
Tuan tu: Time = 0.004330s
Song song: Time = 0.277415s
Toi uu thoi gian: 0.02x
Testing with 50 × 50 matrices
Tuan tu: Time = 0.015000s
Song song: Time = 0.273582s
Toi uu thoi gian: 0.05x
Testing with 100 × 100 matrices
Tuan tu: Time = 0.116676s
Song song: Time = 0.384092s
Toi uu thoi gian: 0.30x
Testing with 300 × 300 matrices
Tuan tu: Time = 4.477990s
Song song: Time = 2.189525s
Toi uu thoi gian: 2.05x
Testing with 500 × 500 matrices
Tuan tu: Time = 18.998738s
Song song: Time = 9.869184s
Toi uu thoi gian: 1.93x
```

3.1 Giải Thích

Trong đoạn mã trên, chúng ta đã thực hiện phép nhân ma trận theo hai cách: tuần tự và song song.

- **Nhân ma trận tuần tự**: Hàm `multiply_matrices_sequential` thực hiện phép nhân ma trận truyền thống bằng cách sử dụng ba vòng lặp để tính toán từng phần tử của ma trận kết quả.

- **Nhân ma trận song song**: Hàm `multiply_matrices_parallel` sử dụng `Pool` từ thư viện `multiprocessing` để chia ma trận A thành các phần nhỏ và phân công cho các tiến trình song song xử lý. Mỗi tiến trình sẽ tính toán một phần của ma trận kết quả, và sau đó kết quả được ghép lại.

3.2 Kết Luận

- Phương pháp song song giúp giảm thời gian tính toán đáng kể, đặc biệt là khi kích thước ma trận lớn. - Việc chia nhỏ công việc và sử dụng nhiều tiến trình giúp tận dụng tối đa khả năng của CPU và cải thiện hiệu suất tính toán.