Lớp: CS112.P11.CTTN
Nhóm: 14
Sinh viên: Lê Nguyễn Anh Khoa. MSSV: 23520742
Sinh viên: Cáp Kim Hải Anh. MSSV: 23520036

BÀI TẬP
Phương pháp thiết kế thuật toán phân tán

I. Bài 1:

- Đề bài: Xây dựng thuật toán kiểm tra số nguyên tố
- Lời giải:
- Kiểm tra nếu tồn tại ước số khác 1 và chính nó, số đó không phải số nguyên tố. Chỉ cần duyệt đến căn bậc 2 của số đó.
- Đối với thuật toán song song, có thể chia đoạn chạy từ 1 đến $\sqrt{n}$ thành nhiều tiến trình, mỗi tiến trình kiểm tra 1 đoạn.
- Độ phức tạp: $O(\sqrt{n})$ khi chạy bình thường và $O(\sqrt{n}/\text{core})$ với core là số core của CPU khi chạy song song.
- Code:

```
import math, multiprocessing, time
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True
def check_chunk(args):
    n, start, chunk_size = args
    end = min(start + chunk_size, int(math.sqrt(n)) + 1)
    for i in range(start, end):
        if n % i == 0:
            return False
    return True
def parallel_prime_check(n):
    if n < 2:
        return False
    num_cpu = multiprocessing.cpu_count()
    chunk_size = int(math.sqrt(n))//num_cpu
    print(f"Số lượng cores CPU: {num_cpu}")
    chunks = [(n, start, chunk_size) for start in range(2, int(math.sqrt(n)) + 1, chunk_size)]
    with multiprocessing.Pool(processes=num_cpu) as pool:
        results = pool.map(check_chunk, chunks)
    return all(results)
n = int(input("Nhập số cần kiểm tra: "))
start_time = time.time()
result1 = is_prime(n)
normal_time = time.time() - start_time
start_time = time.time()
result2 = parallel_prime_check(n)
parallel_time = time.time() - start_time
```

- Chạy thử các test case:

```
$ /usr/local/bin/python /workspaces/81629306/test.py
Nhập số cần kiểm tra: 1000000000000091
Số lượng cores CPU: 2
Là số nguyên tố

Thời gian chạy:
Cách thông thường: 1.643 giây
Cách song song: 1.452 giây
$
```

```
Nhập số cần kiểm tra: 10000000000000099
Số lượng cores CPU: 2
Là số nguyên tố

Thời gian chạy:
Cách thông thường: 4.999 giây
Cách song song: 5.281 giây
$
```

```
$ /usr/local/bin/python /workspaces/81629306/test.py
Nhập số cần kiểm tra: 10000000000000049
Số lượng cores CPU: 2
Là số nguyên tố

Thời gian chạy:
Cách thông thường: 14.839 giây
Cách song song: 14.751 giây
$
```

- Có thể thấy cách chạy thông thường không nhanh bằng cách chạy song song nhưng khoảng cách thời gian không đáng kể do quá trình thiết lập.

II. Bài 2:

- Đề bài: Xây dựng thuật toán nhân ma trận song song
- Lời giải: đối với tác vụ song song, mỗi tiến trình sẽ nhân từng dòng của ma trận A với ma trận B rồi ghép lại.
- Code:

```python
import random, time, multiprocessing
def generate_matrix(rows, cols):
    return [[random.randint(0, 10) for _ in range(cols)] for _ in range(rows)]
def multiply_matrices_sequential(A, B):
    rows_A, cols_A = len(A), len(A[0])
    rows_B, cols_B = len(B), len(B[0])
    assert cols_A == rows_B, "Số cột của A phải bằng số hàng của B"
    result = [[0] * cols_B for _ in range(rows_A)]
    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):
                result[i][j] += A[i][k] * B[k][j]
    return result
def compute_row(args):
    row, B = args
    cols_B = len(B[0])
    result_row = [0] * cols_B
    for j in range(cols_B):
        for k in range(len(B)):
            result_row[j] += row[k] * B[k][j]
    return result_row
def multiply_matrices_parallel(A, B):
    rows_A = len(A)
    with multiprocessing.Pool(processes=multiprocessing.cpu_count()) as pool:
        result = pool.map(compute_row, [(A[i], B) for i in range(rows_A)])
    return result
if __name__ == "__main__":      You, 1 second ago • Uncommitted changes
    size = 400
    A = generate_matrix(size, size)
    B = generate_matrix(size, size)
    start_time = time.time()
    result_sequential = multiply_matrices_sequential(A, B)
    sequential_time = time.time() - start_time
    start_time = time.time()
    result_parallel = multiply_matrices_parallel(A, B)
    parallel_time = time.time() - start_time
    assert result_sequential == result_parallel, "Kết quả tuần tự và song song không khớp!"
    print(f"Thời gian nhân ma trận tuần tự: {sequential_time:.3f} giây")
    print(f"Thời gian nhân ma trận song song: {parallel_time:.3f} giây")
```

- Chạy thử các test case:

```
$ python test.py
Thời gian nhân ma trận tuần tự: 5.482 giây
Thời gian nhân ma trận song song: 3.976 giây
$
```

```
$ python test.py
Thời gian nhân ma trận tuần tự: 8.971 giây
Thời gian nhân ma trận song song: 4.077 giây
$
```

```
$ python test.py
Thời gian nhân ma trận tuần tự: 8.971 giây
Thời gian nhân ma trận song song: 4.077 giây
$
```

⇨ Trong tất cả trường hợp, code song song chạy nhanh hơn.