Trần Tấn Phát
2274802010644

Lab1

Bài làm

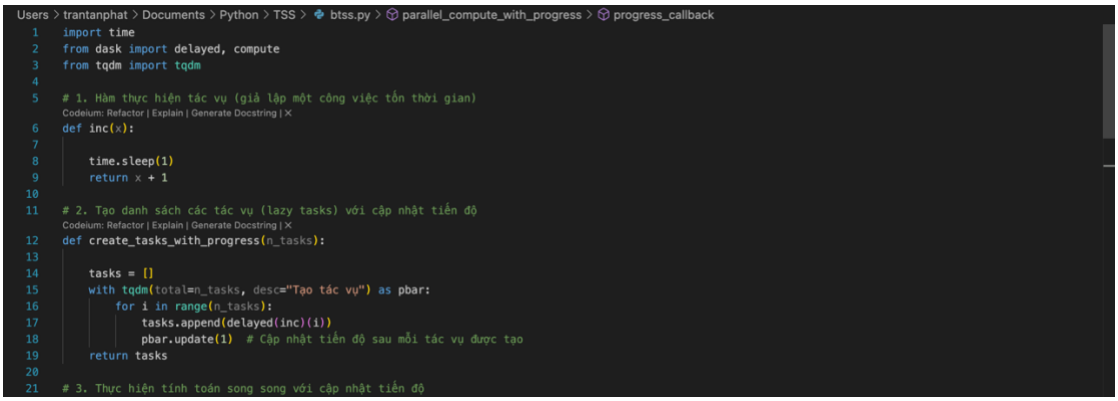
a) Đoạn code tuần tự dưới đây được viết bởi một sinh viên IT năm 2. Hãy song song hóa nó sử dụng Dask

```python
import time

def inc(x):
    """
    Stupid function that basically just wait before doing an addition.
    """
    time.sleep(1)  # simulate artificially some computing intensive computation
    return x + 1
```

```python
from tqdm import tqdm  # for nice progress bar display

ls = []
for i in tqdm(range(60)):
    ls.append(inc(i))
ls[:5]
```

b) Viết đoạn code tính tổng các phần tử có trong 1 mảng gồm 1 triệu phần tử bằng tuần tự và song song. So sánh tốc độ tính toán của 2 cách.
Gợi ý: Sử dụng *numpy random* để tạo mảng, sử dụng *time* để tính thời gian thực thi.

Câu a:



```python
Codeium: Refactor | Explain | Generate Docstring | ×
def parallel_compute_with_progress(tasks):

    with tqdm(total=len(tasks), desc="Tiến độ tính toán") as pbar:
        Codeium: Refactor | Explain | Generate Docstring | ×
        def progress_callback(future):

            pbar.update(1)

        # Tính toán song song sử dụng Dask
        results = compute(*tasks, scheduler="threads", traverse=False, callback=progress_callback)
    return results

# 4. Chương trình chính
if __name__ == "__main__":
    print("Bắt đầu chương trình...")

    # Số lượng tác vụ
    n_tasks = 60

    # Bắt đầu thời gian thực thi
    start = time.time()
```

```python
        results = compute(*tasks, scheduler="threads", traverse=False, callback=progress_callback)
    return results

# 4. Chương trình chính
if __name__ == "__main__":
    print("Bắt đầu chương trình...")

    # Số lượng tác vụ
    n_tasks = 60

    # Bắt đầu thời gian thực thi
    start = time.time()

    # Tạo danh sách các tác vụ với tiến độ
    tasks = create_tasks_with_progress(n_tasks)
    results = parallel_compute_with_progress(tasks)
    end = time.time()

    print("Kết quả 5 phần tử đầu tiên:", results[:5])
    print(f"Thời gian thực thi: {end - start:.2f} giây")

```

```
(base) trantanphat@Macintosh-5 ~ % /Users/trantanphat/anaconda3/bin/python /Users/trantanphat/Documents/Python/TSS/btss.py
Bắt đầu chương trình...
Tạo tác vụ: 100%|████████████████████████████████████████████████████████████| 60/60 [00:00<00:00, 33332.22it/s]
Tiến độ tính toán:   0%|                                                          | 0/60 [00:08<?, ?it/s]
Kết quả 5 phần tử đầu tiên: (1, 2, 3, 4, 5)
Thời gian thực thi: 8.04 giây
(base) trantanphat@Macintosh-5 ~ %
```

Câu b:

Users > trantanphat > Documents > Python > TSS > bt2.py > ...

```python
import numpy as np
import time
import dask.array as da

# Tạo mảng NumPy gồm 1,000,000 phần tử ngẫu nhiên
array = np.random.rand(1_000_000)

# 1. Tính tổng các phần tử bằng phương pháp tuần tự (sử dụng vòng lặp for)
start_sequential = time.time()  # Bắt đầu đo thời gian tuần tự
total_sequential = 0
for value in array:
    total_sequential += value  # Cộng dồn từng phần tử
end_sequential = time.time()  # Kết thúc đo thời gian tuần tự

# 2. Tính tổng các phần tử bằng phương pháp song song sử dụng Dask (vòng lặp trên từng chunk)
start_parallel = time.time()  # Bắt đầu đo thời gian song song
dask_array = da.from_array(array, chunks=10_000)  # Chia mảng NumPy thành các chunk nhỏ
chunks = dask_array.to_delayed()  # Chuyển mỗi chunk thành một đối tượng "delayed"
chunk_sums = [chunk.sum() for chunk in chunks]  # Tính tổng từng chunk
total_parallel = sum([chunk.compute() for chunk in chunk_sums])  # Tính tổng các chunk
end_parallel = time.time()  # Kết thúc đo thời gian song song

# Tính thời gian thực thi cho từng phương pháp
```

```python
chunk_sums = [chunk.sum() for chunk in chunks]  # Tính tổng từng chunk
total_parallel = sum([chunk.compute() for chunk in chunk_sums])  # Tính tổng các chunk
end_parallel = time.time()  # Kết thúc đo thời gian song song

# Tính thời gian thực thi cho từng phương pháp
time_sequential = end_sequential - start_sequential
time_parallel = end_parallel - start_parallel

# Hiển thị kết quả và so sánh
print("Tổng (tuần tự):", total_sequential)
print("Thời gian (tuần tự): {:.5f} giây".format(time_sequential))
print("Tổng (song song):", total_parallel)
print("Thời gian (song song): {:.5f} giây".format(time_parallel))

# So sánh và xác định phương pháp nhanh hơn
if time_sequential < time_parallel:
    print("Phương pháp nhanh hơn: Tuần tự")
    print("Thời gian nhanh hơn: {:.5f} giây".format(time_parallel - time_sequential))
else:
    print("Phương pháp nhanh hơn: Song song")
    print("Thời gian nhanh hơn: {:.5f} giây".format(time_sequential - time_parallel))
```

```
(base) trantanphat@Macintosh-5 ~ % /Users/trantanphat/anaconda3/bin/python /Users/trantanphat/Documents/Python/TSS/bt2.py
Tổng (tuần tự): 499453.7868979383
Thời gian (tuần tự): 0.07030 giây
Tổng (song song): 499453.7868979388
Thời gian (song song): 0.01958 giây
Phương pháp nhanh hơn: Song song
Thời gian nhanh hơn: 0.05073 giây
(base) trantanphat@Macintosh-5 ~ %
```