Thuật toán song song - Thuật toán phân tán

Hồ Ngọc Luật (23520900)
Nguyễn Trần Quang Minh (23520943)

CS112.P11.KHTN
University of Information Technology

Ngày 22 tháng 11 năm 2024

Mục lục

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

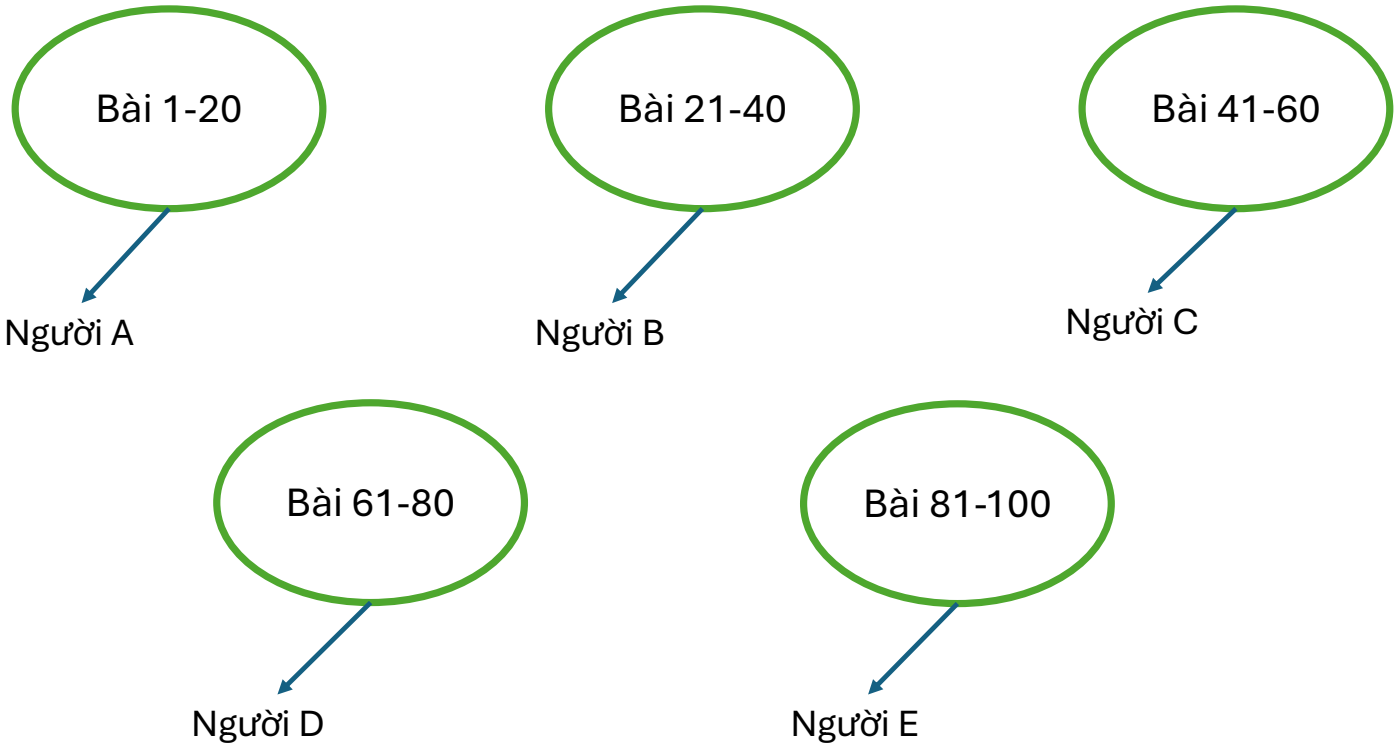
Định nghĩa thuật toán song song

Thuật toán song song là các thuật toán được thiết kế đặc biệt để tận dụng khả năng xử lý song song. Đây là thuật toán chia nhỏ một bài toán thành các phần độc lập và thực hiện chúng đồng thời nhằm tăng tốc độ giải quyết bài toán.

Ví dụ: Giả sử bạn được thầy giao làm 100 câu truy vấn sql, nhưng bạn thấy những câu truy vấn này quá dễ và lặp lại, làm phí thời gian. Giải pháp?

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ví dụ



Hình: Minh họa phân chia công việc, giúp tiết kiệm thời gian

$\Longrightarrow$ **Chia ra thực hiện, tăng hiệu suất, tốc độ thực thi**

multiprocessing và concurrent.future

```python
from multiprocessing import Pool
import time

def SLEEP(x):
    time.sleep(x)
    print(f"Task completed after {x} seconds.")

if __name__ == "__main__":
    start = time.time()
    with Pool() as pool:
        pool.map(SLEEP, [1, 3, 2, 5, 4])
    end = time.time()
    print("Total time with multiprocessing: ", end - start)
    start = time.time()
    for i in [1, 3, 2, 5, 4]:
        SLEEP(i)
    end = time.time()
    print("Total time w/o multiprocessing: ", end - start)
```

```python
from concurrent.futures import ThreadPoolExecutor
import time

def SLEEP(x):
    time.sleep(x)
    print(f"Task completed after {x} seconds.")

if __name__ == "__main__":
    start = time.time()
    with ThreadPoolExecutor(max_workers=5) as executor:
        for x in [3, 1, 4, 2, 5]:
            executor.submit(SLEEP, x)
    end = time.time()
    print("Total time with ThreadPoolExecutor: ", end - start)
    start = time.time()
    for i in [3, 1, 4, 2, 5]:
        SLEEP(i)
    end = time.time()
    print("Total time w/o ThreadPoolExecutor: ", end - start)
```

```
Task completed after 1 seconds.
Task completed after 2 seconds.
Task completed after 3 seconds.
Task completed after 4 seconds.
Task completed after 5 seconds.
Total time with multiprocessing:  5.216216087341309
Task completed after 1 seconds.
Task completed after 3 seconds.
Task completed after 2 seconds.
Task completed after 5 seconds.
Task completed after 4 seconds.
Total time w/o multiprocessing:  15.002712726593018
```

```
Task completed after 1 seconds.
Task completed after 2 seconds.
Task completed after 3 seconds.
Task completed after 4 seconds.
Task completed after 5 seconds.
Total time with ThreadPoolExecutor:  5.002339601516724
Task completed after 3 seconds.
Task completed after 1 seconds.
Task completed after 4 seconds.
Task completed after 2 seconds.
Task completed after 5 seconds.
Total time w/o ThreadPoolExecutor:  15.003512144088745
```

Hình: Sử dụng thư viện multiprocessing và concurrent để thực hiện chạy song song

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

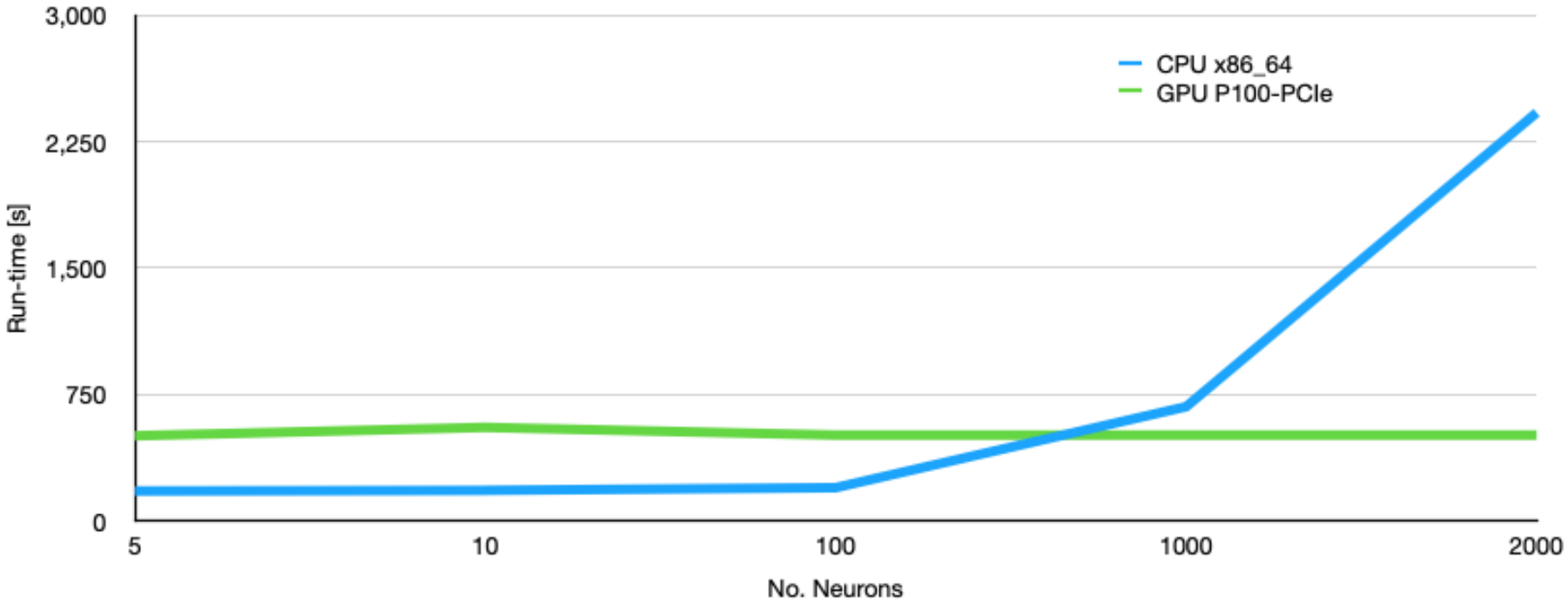
Huấn luyện AI sử dụng GPU bằng pytorch

Khi ta có một mô hình chưa được huấn luyện và 1 bộ dữ liệu, ta có thể sử dụng dòng lệnh sau để sử dụng GPU cho việc huấn luyện mô hình:

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
X, y = X.to(device), y.to(device)
```

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Huấn luyện AI sử dụng GPU bằng pytorch



Hình: Đồ thị biểu diễn tốc độ tính toán khi huấn luyện AI bằng CPU và GPU.
Hình được lấy từ: Machine Learning on GPU

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

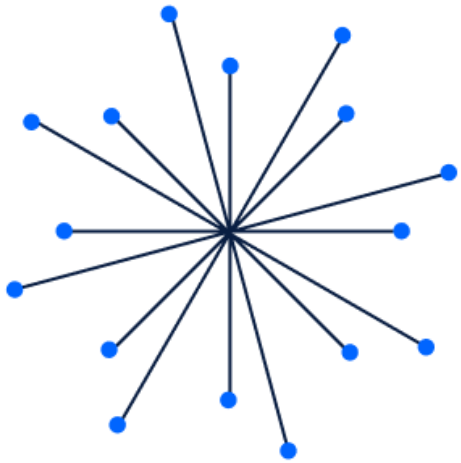
Ứng dụng

- Trí tuệ nhân tạo (AI): Ứng dụng trong các mô hình học sâu (deep learning), giúp tăng tốc quá trình huấn luyện mô hình trên GPU hoặc nhiều máy tính.
- Mô phỏng và tính toán khoa học: Các mô phỏng vật lý, thiên văn, sinh học,... đòi hỏi việc xử lí dữ liệu lớn và phức tạp, thường sử dụng thuật toán song song để tăng tốc độ tính toán.
- Xử lý hình ảnh và đồ họa máy tính

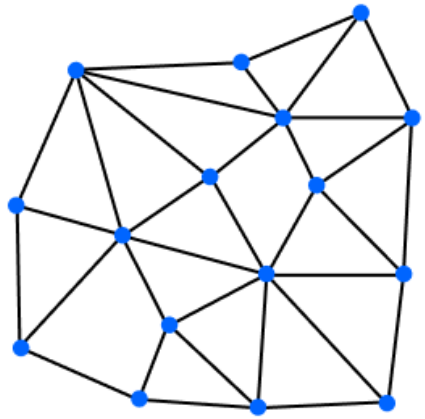
Nhược điểm

- Không phải lúc nào cũng áp dụng được, đặc biệt là những bài toán có tính tuần tự hay phụ thuộc nhau.
- Tốn tài nguyên cho việc truyền thông tin. Đôi khi việc giao tiếp tốn quá nhiều thời gian dẫn đến thuật toán hoạt động chậm hơn cả khi thực hiện tuần tự $\longrightarrow$ **parallel slowdown**.

Thuật toán phân tán



Centralized



Distributed

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Định nghĩa thuật toán phân tán

Thuật toán phân tán là loại thuật toán được thiết kế để xử lý dữ liệu và thực hiện tính toán trên một hệ thống bao gồm nhiều máy tính hoặc nút (nodes) kết nối với nhau qua mạng còn được gọi là hệ thống phân tán. Thay vì tập trung mọi tài nguyên xử lý trên một máy đơn lẻ, thuật toán phân tán phân chia công việc cho các nút khác nhau để cùng tham gia xử lý, cho phép tận dụng tài nguyên của toàn bộ hệ thống phân tán.

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Các cấp bậc của xử lý phân tán

1. Lưu trữ phân tán, xử lý tập trung.
2. Lưu trữ tập trung, xử lý phân tán.
3. Lưu trữ phân tán, xử lý phân tán.

Ứng dụng

1. **Lưu trữ đám mây:** phân phối và lưu trữ dữ liệu trên hàng nghìn máy chủ, đảm bảo dữ liệu được sao lưu và sẵng sàn truy cập ngay cả khi một số máy chủ gặp sự cố.
2. **Blockchain:** dựa vào các thuật toán phân tán để duy trì sổ cái bằng các thuật toán đồng thuận như Proof of Work (PoW), Proof of Stake (PoS) và Raft.
3. **Trí tuệ nhân tạo phân tán:** Distributed Stochastic Gradient Descent (SGD), Federated learning.

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Thuật toán phân tán phổ biến

1. **Communication Algorithms**: Message Passing, Publish-Subscribe, Group Communication
2. **Synchronization Algorithms**: Distributed Locks, Semaphores, Distributed Clocks
3. **Consensus Algorithms**: Paxos, Raft, BFT
4. **Replication Algorithms**
5. **Distributed Query Processing Algorithms**

Thuật toán phân tán phổ biến

6. **Load Balancing Algorithms**: Round Robin, Least Connection, IP Hash, Weighted Round Robin, Least Response Time
7. **Distributed Data Structures and Algorithms**
8. **Failure Detection and Failure Recovery Algorithms** Heartbeat-Based Detection, Neighbor Monitoring, Quorum-Based Detection
9. **Security Algorithms for a Distributed Environment**: Cryptography, Authentication and Authorization, Access Control, Secure Communication Protocols, Intrusion Detection and Prevention, Key Management

Thuật toán Ford-Bellman trong hệ thống phân tán

Bài toán: Trong một hệ thống phân tán, các nút được kết nối với nhau qua các cạnh và các cạnh sẽ có trọng số, tìm đường đi ngắn nhất từ một đỉnh bất kỳ đến các đỉnh khác

Giải thuật: Sử dụng thuật toán Ford-Bellman.

- Bởi vì là trong hệ thống phân tán, không có nút nào biết được toàn bộ các cạnh hay nút trong hệ thống.
- Các nút biết được trọng số nối với các nút hàng xóm.
- Các nút cùng lúc hoạt động theo từng vòng (round).

Tham khảo: video tham khảo về thuật toán Ford-Bellman trong hệ thống phân tán: youtube

Thuật toán Ford-Bellman trong hệ thống phân tán

Thực hiện:

- Ban đầu, mỗi nút sẽ có một giá trị *dist*. Giá trị *dist* của nút xuất phát sẽ bằng 0, còn các nút còn lại bằng ∞.
- Tại mỗi vòng, toàn bộ các nút gửi giá trị *dist* của nó cho các nút hàng xóm.
- Nếu một nút bất kì nhận được một giá trị *dist* được gửi đến, nó sẽ so sánh với giá trị *dist* của chính nó:

$$dist_i = min(dist_i, dist_j + weight_{ij})$$

- Thuật toán sẽ dừng lại tại một vòng nào đó mà không có bất cứ giá trị *dist* nào được cập nhật, hoặc khi tới một vòng đủ lớn.

Thuật toán Ford-Bellman trong hệ thống phân tán

```
# Mỗi nút u thực hiện
initialize:
    if u == source:
        dist[u] = 0
    else:
        dist[u] = ∞
    updated = True

while there are updates:
    if updated:
        # Gửi giá trị dist[u] cho tất cả các nút lân cận
        send dist[u] to all neighboring nodes
        updated = False  # Đánh dấu đã gửi xong

    # Nhận thông tin từ các nút lân cận
    for each message received from neighboring node v:
        if dist[u] > dist[v] + w(u, v):   # Cập nhật giá trị nếu tìm thấy đường đi ngắn hơn
            dist[u] = dist[v] + w(u, v)
            updated = True  # Đánh dấu có thay đổi và sẽ gửi lại
```

Hình: Mã giả cho thuật toán Ford-Bellman trong hệ thống phân tán. Nguồn ảnh: ChatGPT

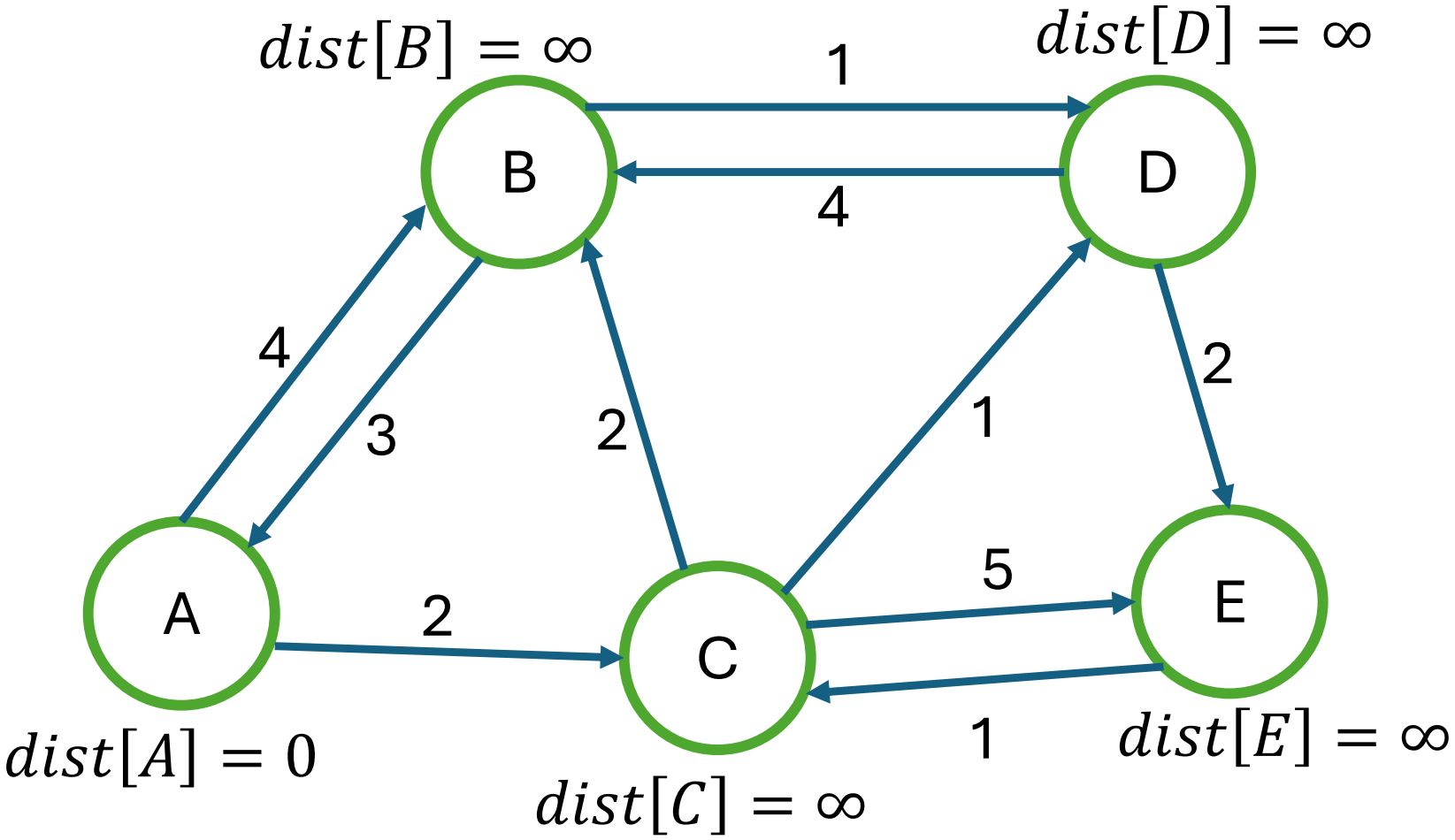
Thuật toán Ford-Bellman trong hệ thống phân tán

Lưu ý: Trong cả mã giả bên trên và mô phỏng bên dưới, thuật toán sẽ có khác đôi chút với lại mô tả bên trên. Cụ thể là, thay vì toàn bộ các nút gửi giá trị *dist* của nó tại mỗi vòng, chỉ có những nút vừa cập nhật lại *dist* mới gửi đi giá trị *dist*. Điều này giúp giảm thiểu số lần truyền tin không cần thiết.

Độ phức tạp: Độ phức tạp thời gian: $O(n)$, độ phức tạp giao tiếp (tính thêm chi phí giao tiếp): $O(n|E|)$ với n là số nút và $|E|$ là số cạnh.

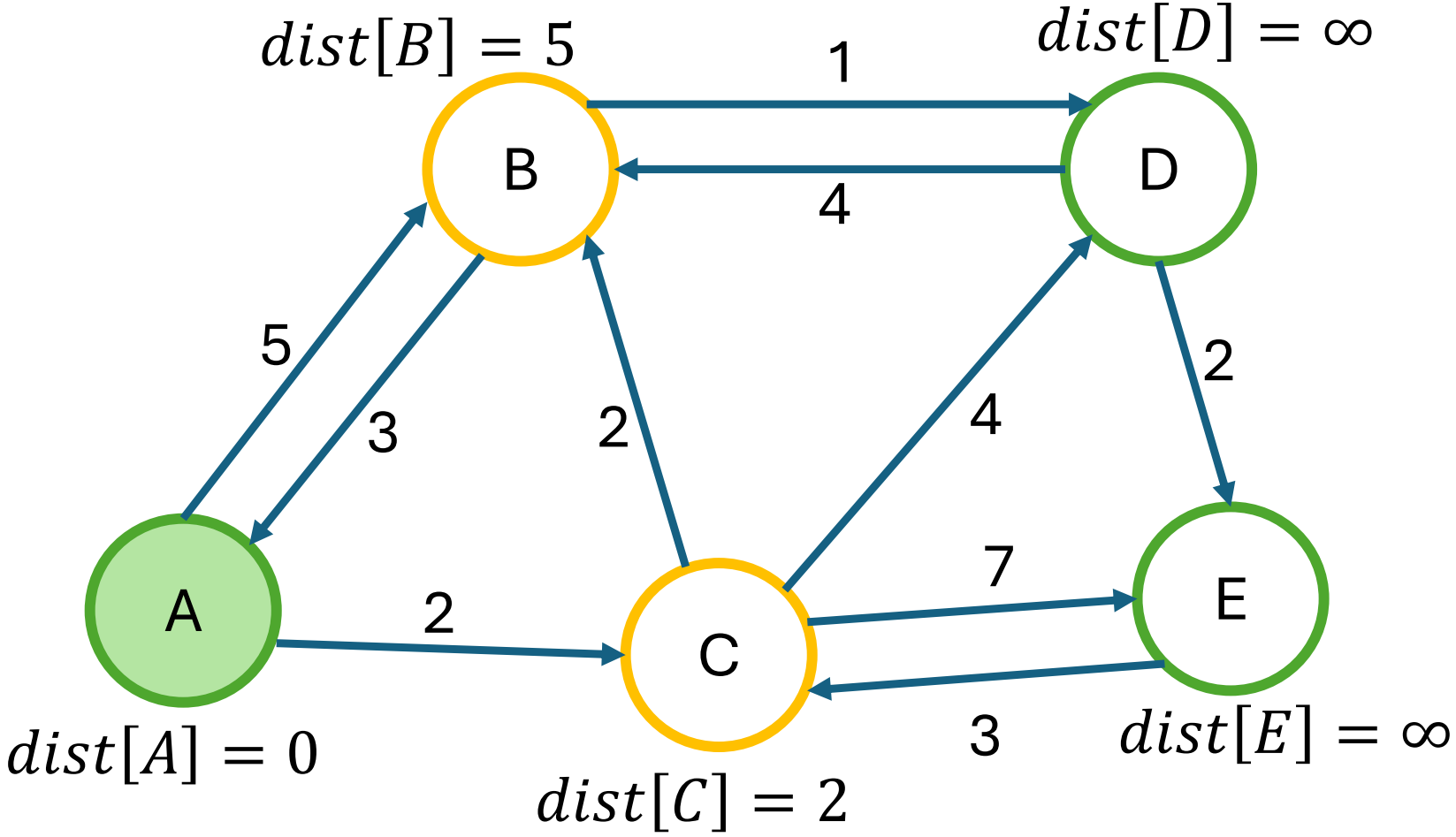
UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Thuật toán Ford-Bellman trong hệ thống phân tán



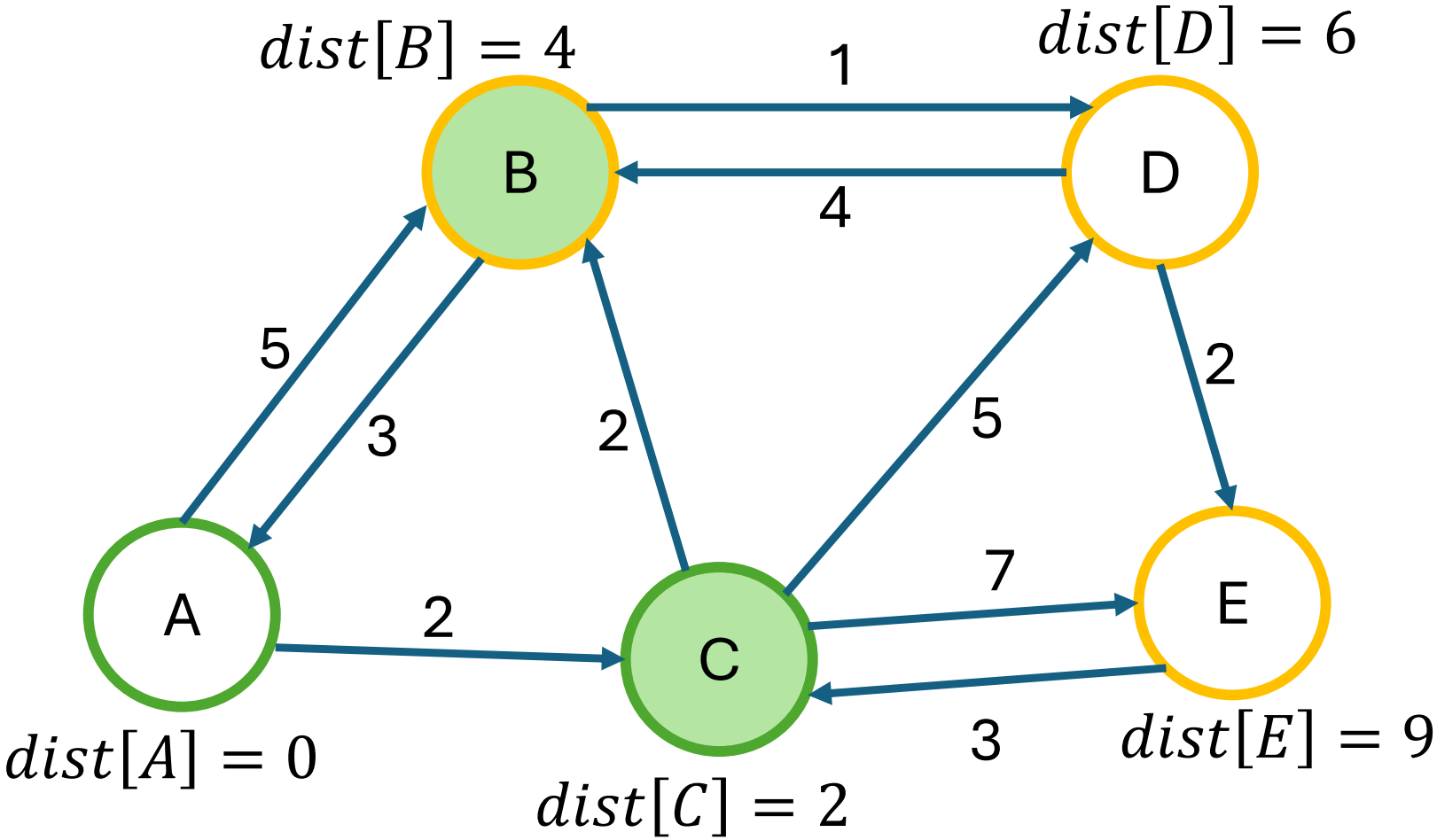
Hình: Đồ thị ban đầu

Thuật toán Ford-Bellman trong hệ thống phân tán



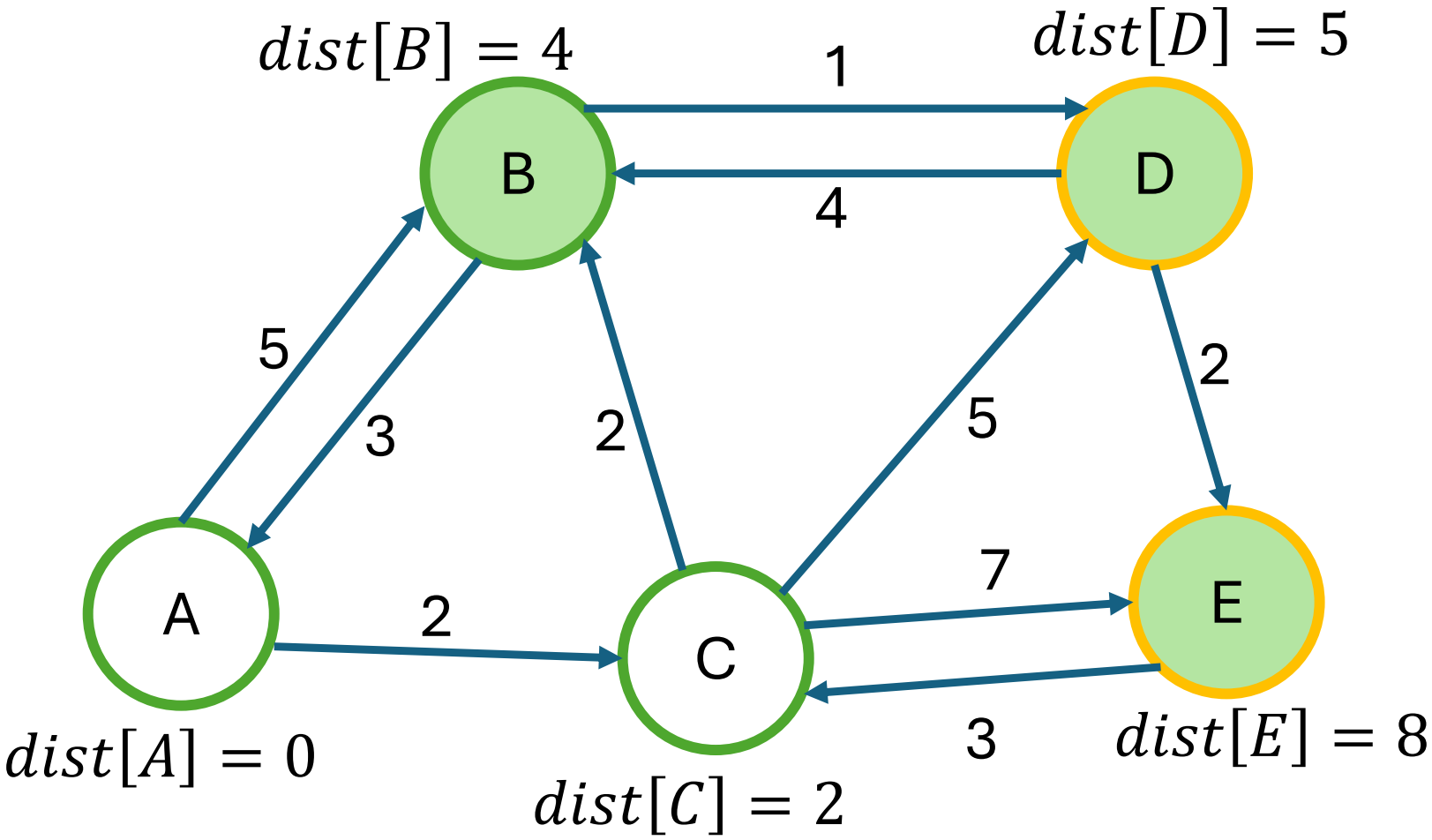
Hình: Vòng thứ 1

Thuật toán Ford-Bellman trong hệ thống phân tán



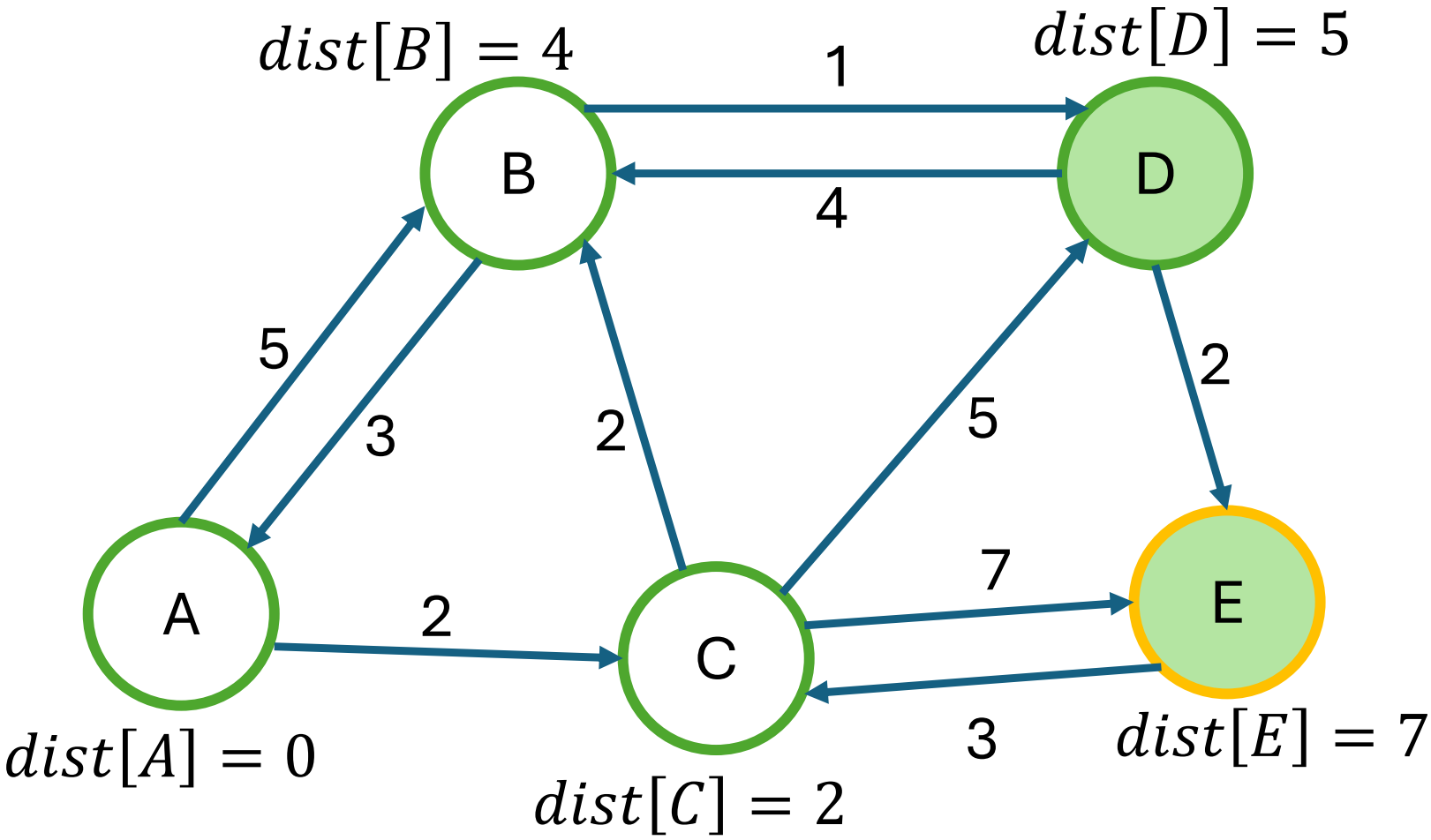
Hình: Vòng thứ 2

Thuật toán Ford-Bellman trong hệ thống phân tán



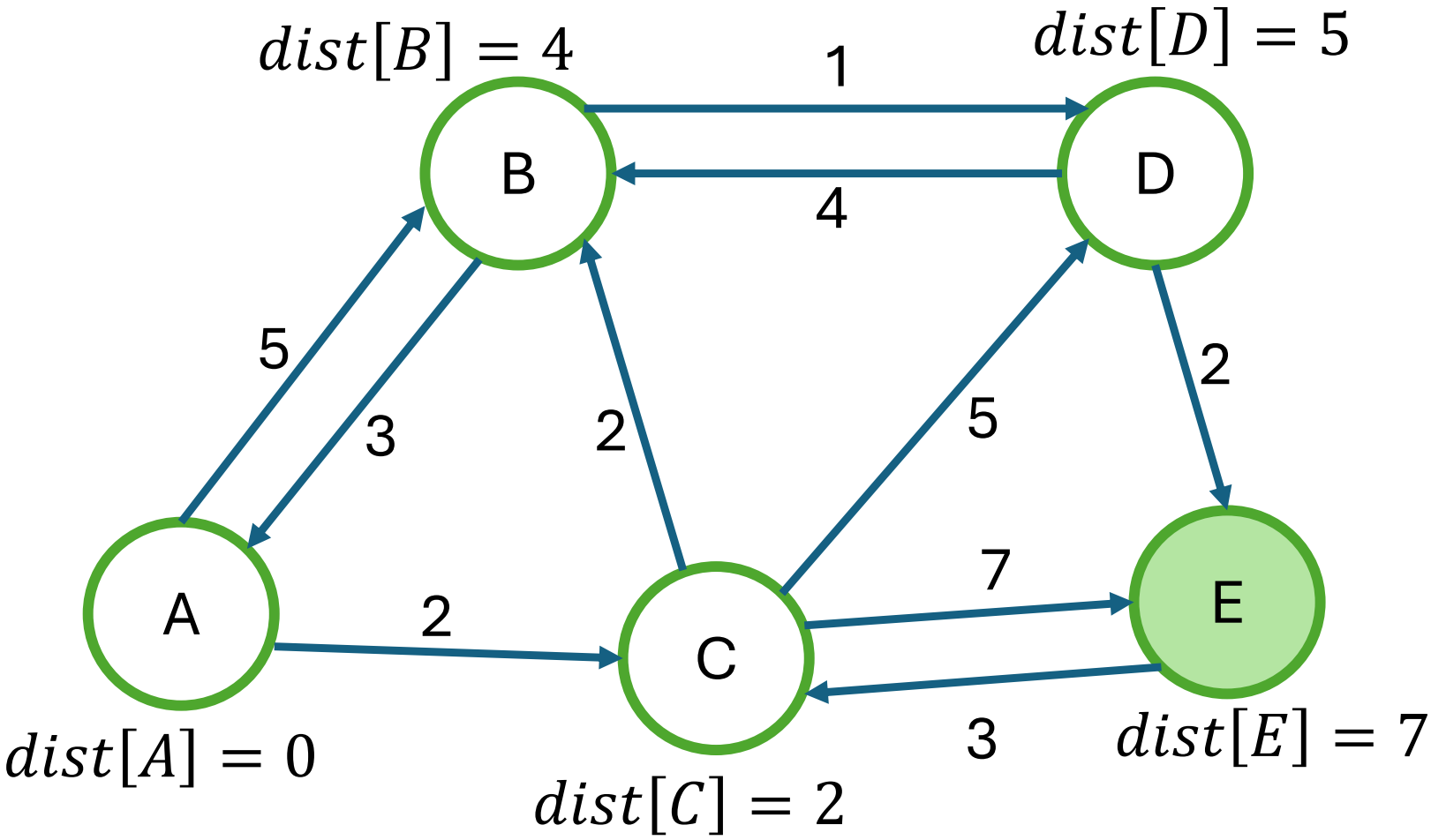
Hình: Vòng thứ 3

Thuật toán Ford-Bellman trong hệ thống phân tán



Hình: Vòng thứ 4

Thuật toán Ford-Bellman trong hệ thống phân tán



Hình: Vòng thứ 5