Sudoku Solution Validator

Bao Nguyen, Cuong Dang, Thanh Bui, Thanh Nguyen[1]

[1]CS Department
Ho Chi Minh City University Information of Technology (UIT)

Tổng quan

I. Giới thiệu

Trò chơi Sudoku là gì ?

Sudoku là trò chơi xếp số trong bảng 9×9, nơi người chơi điền các con số từ 1 đến 9 sao cho mỗi hàng, mỗi cột và mỗi vùng 3×3 không có số trùng nhau.

- Quy tắc: Mỗi hàng, cột và vùng 3×3 phải chứa đầy đủ các số từ 1 đến 9.
- Mục tiêu: Điền các số còn thiếu để hoàn thành bảng.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
2			4	1	9			5
3				8			7	9

Mục tiêu

- Mục tiêu của dự án này là thiết kế một chương trình đa tiểu trình (luồng) để xác định xem giải pháp cho một bài toán Sudoku có hợp lệ hay không ?
- So sánh hiệu suất thực tế giữa các phương pháp (đơn luồng, đa luồng), phân tích ưu nhược điểm của mỗi cách tiếp cận.

II. Tiếp cận

Truyền tham số cho mỗi tiểu trình

Tiểu trình cha sẽ tạo các tiểu trình con và truyền cho mỗi tiểu trình vị trí cần kiểm tra trong bảng Sudoku. Cách dễ nhất là sử dụng một cấu trúc dữ liệu để truyền các tham số cho tiểu trình.

Truyền tham số cho mỗi tiểu trình

Tiểu trình cha sẽ tạo các tiểu trình con và truyền cho mỗi tiểu trình vị trí cần kiểm tra trong bảng Sudoku. Cách dễ nhất là sử dụng một cấu trúc dữ liệu để truyền các tham số cho tiểu trình.

Ví dụ, một cấu trúc để truyền dòng và cột mà tiểu trình cần bắt đầu kiểm tra có thể như sau:

```
/* structure for passing data to threads */
typedef struct
{
    int row;
    int column;
} parameters;
```

Trả kết quả về cho tiểu trình cha

Mỗi tiểu trình con được giao nhiệm vụ kiểm tra tính hợp lệ của một vùng trong bảng Sudoku. Sau khi hoàn thành, các tiểu trình con phải trả kết quả cho tiểu trình cha. Một cách đơn giản là tạo một mảng số nguyên có thể truy cập bởi tất cả các tiểu trình. Chỉ số thứ i trong mảng này tương ứng với tiểu trình thứ i.

Nếu tiểu trình thiết lập giá trị của chỉ số đó là 1, nghĩa là vùng Sudoku của nó hợp lệ. Nếu giá trị là 0, nghĩa là không hợp lệ. Khi tất cả các luồng hoàn thành, tiểu trình cha kiểm tra mảng kết quả để xác định tính hợp lệ của bảng Sudoku.

III. Thiết kế thuật toán đa tiểu trình

Thực hiện đa tiểu trình với 3 luồng

Trong trường hợp sử dụng 3 luồng, công việc được chia thành ba nhóm lớn, mỗi nhóm đảm nhiệm một phần quan trọng của bài toán. Phương pháp này vẫn tăng tốc quá trình kiểm tra, đồng thời giảm bớt chi phí tạo và quản lý nhiều tiểu trình hơn.

Thực hiện đa tiểu trình với 3 luồng

Trong trường hợp sử dụng 3 luồng, công việc được chia thành ba nhóm lớn, mỗi nhóm đảm nhiệm một phần quan trọng của bài toán. Phương pháp này vẫn tăng tốc quá trình kiểm tra, đồng thời giảm bớt chi phí tạo và quản lý nhiều tiểu trình hơn.

Với 3 luồng, chúng ta có thể chia công việc như sau:

- **Luồng 1**: Kiểm tra rằng tất cả các dòng của bảng Sudoku chứa các chữ số từ 1 đến 9.
- **Luồng 2**: Kiểm tra rằng tất cả các cột của bảng Sudoku chứa các chữ số từ 1 đến 9.
- **Luồng 3**: Kiểm tra rằng tất cả các vùng con 3×3 chứa các chữ số từ 1 đến 9.

Thực hiện đa tiểu trình với 11 luồng

Trong trường hợp sử dụng 11 luồng, mỗi luồng sẽ đảm nhận một phần công việc độc lập. Số lượng luồng này được chia thành các nhóm nhỏ, mỗi nhóm sẽ xử lý một phần của bài toán, ví dụ như kiểm tra các vùng khác nhau của bảng Sudoku. Phương pháp này giúp tăng tốc quá trình giải quyết bài toán khi chia nhỏ công việc ra.

Thực hiện đa tiểu trình với 11 luồng

Trong trường hợp sử dụng 11 luồng, mỗi luồng sẽ đảm nhận một phần công việc độc lập. Số lượng luồng này được chia thành các nhóm nhỏ, mỗi nhóm sẽ xử lý một phần của bài toán, ví dụ như kiểm tra các vùng khác nhau của bảng Sudoku. Phương pháp này giúp tăng tốc quá trình giải quyết bài toán khi chia nhỏ công việc ra.

Ví dụ, chúng ta có thể chia bảng Sudoku thành các vùng con để mỗi luồng xử lý một vùng, chẳng hạn như sau:

- Một luồng để kiểm tra rằng mỗi cột chứa các chữ số từ 1 đến 9.
- Một luồng để kiểm tra rằng mỗi dòng chứa các chữ số từ 1 đến 9.
- Chín luồng để kiểm tra rằng mỗi vùng con 3×3 chứa các chữ số từ 1 đến 9.

Thực hiện đa tiểu trình với 27 luồng

Trong trường hợp sử dụng 27 luồng, việc chia công việc sẽ được chi tiết hơn nữa. Mỗi luồng xử lý một phần nhỏ của công việc, giúp tăng cường sự song song và tối ưu hóa hiệu suất.

Thực hiện đa tiểu trình với 27 luồng

Trong trường hợp sử dụng 27 luồng, việc chia công việc sẽ được chi tiết hơn nữa. Mỗi luồng xử lý một phần nhỏ của công việc, giúp tăng cường sự song song và tối ưu hóa hiệu suất.

Với 27 luồng, chúng ta có thể chia bảng Sudoku thành các vùng nhỏ hơn nữa, ví dụ:

- Chín luồng để kiểm tra rằng mỗi cột chứa các chữ số từ 1 đến 9.
- Chín luồng để kiểm tra rằng mỗi dòng chứa các chữ số từ 1 đến 9.
- Chín luồng để kiểm tra rằng mỗi vùng con 3×3 chứa các chữ số từ 1 đến 9.

IV. So sánh hiệu suất

So sánh thời gian thực thi

Ta thực thi trên hai giải pháp Sudoku : **Hợp lệ** và **Không hợp lệ**.

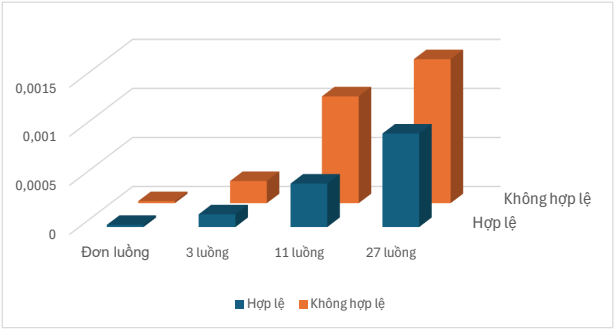
So sánh thời gian thực thi

Ta thực thi trên hai giải pháp Sudoku : **Hợp lệ** và **Không hợp lệ**.

	Hợp lệ	**Không hợp lệ**
Đơn luồng	0,000022	0,000021
3 luồng	0,00013	0,000226
11 luồng	0,000443	0,001089
27 luồng	0,000953	0,001468

Bảng 1: Bảng thời gian thực thi giữa các cách tiếp cận.

So sánh thời gian thực thi



Tại sao lại có kết quả như vậy ?

1. **Overhead quản lý luồng tăng khi số luồng tăng**
   - Với ít luồng, hệ điều hành tốn ít tài nguyên để tạo, quản lý và đồng bộ hóa các luồng.
   - Với nhiều luồng, hệ điều hành phải:
     - Cấp phát bộ nhớ cho mỗi luồng.
     - Theo dõi trạng thái và đồng bộ hóa luồng.
     - Chuyển đổi ngữ cảnh giữa các luồng, làm giảm hiệu suất.

Tại sao lại có kết quả như vậy ?

1. **Overhead quản lý luồng tăng khi số luồng tăng**
   - Với ít luồng, hệ điều hành tốn ít tài nguyên để tạo, quản lý và đồng bộ hóa các luồng.
   - Với nhiều luồng, hệ điều hành phải:
     - Cấp phát bộ nhớ cho mỗi luồng.
     - Theo dõi trạng thái và đồng bộ hóa luồng.
     - Chuyển đổi ngữ cảnh giữa các luồng, làm giảm hiệu suất.

2. **Cạnh tranh tài nguyên CPU**
   - **Với ít luồng**: Mỗi luồng có thể tận dụng tốt lõi CPU mà không cần phải chia sẻ nhiều.
   - **Với nhiều luồng**: Số luồng vượt quá số lõi CPU dẫn đến cạnh tranh tài nguyên, làm giảm hiệu suất.

Tại sao lại có kết quả như vậy ?

3. **Chi phí chuyển đổi ngữ cảnh tăng**
   - Với nhiều luồng, việc chuyển đổi ngữ cảnh giữa các luồng trở nên thường xuyên hơn.
   - CPU phải lưu trạng thái của luồng hiện tại và tải trạng thái của luồng tiếp theo.
   - Chi phí này không đóng góp vào công việc thực tế, làm tăng thời gian thực thi tổng thể.

Tại sao lại có kết quả như vậy ?

3. **Chi phí chuyển đổi ngữ cảnh tăng**
   - Với nhiều luồng, việc chuyển đổi ngữ cảnh giữa các luồng trở nên thường xuyên hơn.
   - CPU phải lưu trạng thái của luồng hiện tại và tải trạng thái của luồng tiếp theo.
   - Chi phí này không đóng góp vào công việc thực tế, làm tăng thời gian thực thi tổng thể.

4. **Khối lượng công việc nhỏ trên mỗi luồng**
   - **Ít luồng**: Mỗi luồng có nhiều công việc hơn để xử lý, giúp tối ưu hóa thời gian sử dụng CPU.
   - **Nhiều luồng**: Mỗi luồng chỉ xử lý một phần nhỏ công việc, nhưng chi phí quản lý luồng lại lớn hơn thời gian thực thi thực tế.

Tại sao lại có kết quả như vậy ?

3. **Chi phí chuyển đổi ngữ cảnh tăng**
   - Với nhiều luồng, việc chuyển đổi ngữ cảnh giữa các luồng trở nên thường xuyên hơn.
   - CPU phải lưu trạng thái của luồng hiện tại và tải trạng thái của luồng tiếp theo.
   - Chi phí này không đóng góp vào công việc thực tế, làm tăng thời gian thực thi tổng thể.

4. **Khối lượng công việc nhỏ trên mỗi luồng**
   - **Ít luồng**: Mỗi luồng có nhiều công việc hơn để xử lý, giúp tối ưu hóa thời gian sử dụng CPU.
   - **Nhiều luồng**: Mỗi luồng chỉ xử lý một phần nhỏ công việc, nhưng chi phí quản lý luồng lại lớn hơn thời gian thực thi thực tế.

IV. Kết luận

Kết luận

- Đơn luồng và đa luồng:
  - Với các chương trình đơn giản và ngắn như Sudoku Solution Validator, đơn luồng thường nhanh hơn đa luồng.
  - Nguyên nhân: Chi phí tạo và quản lý luồng (overhead) trong đa luồng làm tăng thời gian thực thi cho các chương trình nhỏ.

Kết luận

- Đơn luồng và đa luồng:
  - Với các chương trình đơn giản và ngắn như Sudoku Solution Validator, đơn luồng thường nhanh hơn đa luồng.
  - Nguyên nhân: Chi phí tạo và quản lý luồng (overhead) trong đa luồng làm tăng thời gian thực thi cho các chương trình nhỏ.
- Lợi ích của đa luồng:
  - Đa luồng mang lại lợi ích đáng kể và cải thiện hiệu suất so với đơn luồng đối với các chương trình phức tạp.
  - Áp dụng tốt cho các bài toán lớn yêu cầu xử lý song song, như xử lý dữ liệu lớn hoặc tính toán khoa học.