TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO MÔN : HỆ ĐIỀU HÀNH

ĐỀ TÀI

SUDOKU SOLUTION VALIDATOR

GV hướng dẫn: KS. THÂN THẾ TÙNG

Nhóm sinh viên thực hiện:

Họ và tên	*MSSV*	*Mã lớp*
Nguyễn Thiên Bảo	23520127	KHTN2023
Đặng Quốc Cường	23520192	KHTN2023
Bùi Ngọc Thiên Thanh	23521436	KHTN2023
Nguyễn Trọng Tất Thành	23521455	KHTN2023

Hồ Chí Minh, 2024

Mục lục

Lời cảm ơn

Trong suốt quá trình thực hiện báo cáo này, chúng em đã nhận được sự hỗ trợ quý báu và những ý kiến đóng góp chân thành từ quý thầy cô và các bạn.

Nhóm chúng em xin gửi lời cảm ơn sâu sắc đến quý thầy cô bộ môn Hệ điều hành, những người đã tận tâm hướng dẫn, chia sẻ kiến thức và kinh nghiệm quý báu giúp chúng em hoàn thành báo cáo.

Chúng em cũng xin chân thành cảm ơn các bạn trong lớp, những người đã không ngần ngại chia sẻ ý kiến và góp ý hữu ích, giúp nhóm chúng em cải thiện và hoàn thiện nội dung báo cáo này.

Chúng em hy vọng sẽ tiếp tục nhận được sự chỉ bảo và góp ý từ quý thầy cô, để không ngừng nâng cao kỹ năng, kiến thức, cũng như ý thức học tập và làm việc, phục vụ tốt hơn cho công việc trong tương lai.

Xin chân thành cảm ơn!

1 Giới thiệu:

1.1 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.

1.2 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.

2 Phương pháp triển khai

2.1 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.

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

```
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.

2.2 Phương pháp sử dụng đơn luồng

Phân chia công việc như sau:

1. **Bước 1: Kiểm tra tất cả các hàng của bảng Sudoku.**
   - Duyệt qua từng hàng.
   - Sử dụng một mảng tạm `validarray` để đánh dấu các số từ 1 đến 9.
   - Nếu phát hiện số trùng lặp, kết luận giải pháp không hợp lệ.
2. **Bước 2: Kiểm tra tất cả các cột của bảng Sudoku.**
   - Duyệt qua từng cột với cơ chế tương tự như kiểm tra hàng.
3. **Bước 3: Kiểm tra tất cả các vùng con 3×3.**
   - Duyệt qua từng vùng con, kiểm tra các số từ 1 đến 9 bằng cách sử dụng mảng tạm `validarray`.

2.3 Phương pháp sử dụng 3 luồng

Phân chia công việc như sau:

- **Luồng 1**: Kiểm tra tất cả các hàng (tương tự bước 1 trong phương pháp đơn luồng).
- **Luồng 2**: Kiểm tra tất cả các cột (tương tự bước 2 trong phương pháp đơn luồng).
- **Luồng 3**: Kiểm tra tất cả các vùng con 3×3 (tương tự bước 3 trong phương pháp đơn luồng).

Cách thực hiện:

- Tạo ba luồng song song và giao từng nhiệm vụ kiểm tra cho mỗi luồng.
- Mỗi luồng thực thi độc lập và lưu kết quả vào một mảng `result[3]`:
  - `result[0]` lưu kết quả kiểm tra hàng.
  - `result[1]` lưu kết quả kiểm tra cột.
  - `result[2]` lưu kết quả kiểm tra vùng con.
- Chờ tất cả các luồng hoàn thành và tổng hợp kết quả.

2.4 Phương pháp sử dụng 11 luồng

Phân chia công việc như sau:

- Một luồng để kiểm tra rằng mỗi dòng chứa các chữ số từ 1 đến 9.
- Một luồng để kiểm tra rằng mỗi cột 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.

Cách thực hiện:

- Tạo mảng `result` để lưu kết quả của từng luồng:
  - `result[0]` lưu kết quả kiểm tra hàng.
  - `result[1]` lưu kết quả kiểm tra cột.
  - `result[2..10]` lưu kết quả kiểm tra từng vùng con 3×3.
- Khởi tạo và chạy đồng thời tất cả 11 luồng, mỗi luồng đảm nhận một phần công việc cụ thể.
- Chờ tất cả các luồng hoàn thành và tổng hợp kết quả từ mảng `result`.

2.5 Phương pháp sử dụng 27 luồng

Phân chia công việc như sau:

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

Cách thực hiện:

- Tạo mảng `result` để lưu kết quả của từng luồng:
  - `result[0..8]` lưu kết quả kiểm tra từng hàng.
  - `result[9..17]` lưu kết quả kiểm tra từng cột.
  - `result[18..26]` lưu kết quả kiểm tra từng vùng con 3×3.
- Khởi tạo và chạy đồng thời tất cả 27 luồng, mỗi luồng đảm nhận một phần công việc cụ thể (hàng, cột hoặc vùng con).
- Chờ tất cả các luồng hoàn thành và tổng hợp kết quả từ mảng `result`.

2.6 So sánh các phương pháp

Phương pháp	Số lượng luồng	Ưu điểm	Nhược điểm
Đơn luồng	1	Đơn giản, giảm chi phí quản lý, không có context switching	Hiệu suất thấp trên hệ thống đa lõi, không phù hợp với bài toán lớn
3 luồng	3	Giảm chi phí quản lý so với nhiều luồng hơn	Tính song song hạn chế
11 luồng	11	Phù hợp với CPU có số luồng logic nhỏ hoặc vừa	Song song chưa tối ưu hoàn toàn
27 luồng	27	Tận dụng tối đa khả năng song song	Gây hiện tượng context switching, tăng chi phí quản lý

3 So sánh thời gian thực thi

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

Giải pháp: Sudoku hợp lệ

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

Giải pháp: Sudoku không hợp lệ

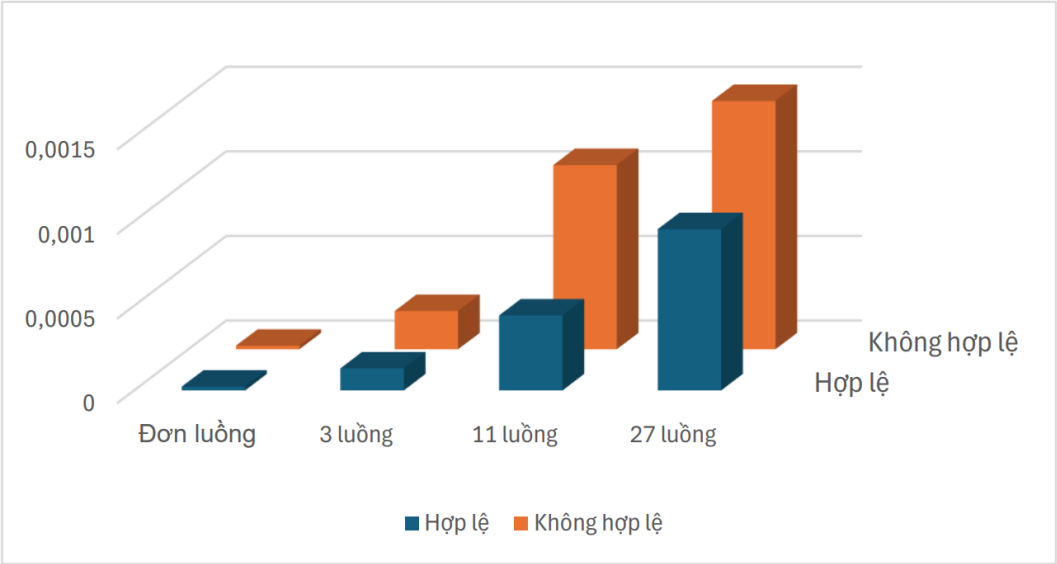
1	2	4	5	3	9	1	8	7
5	1	9	7	2	8	6	3	4
8	3	7	6	1	4	2	9	5
1	4	3	8	6	5	7	2	9
9	5	8	2	4	7	3	6	1
7	6	2	3	9	1	4	5	8
3	7	1	9	5	6	8	4	2
4	9	6	1	8	2	5	7	3
2	8	5	4	7	3	9	1	6

Sau khi thực thi ta được kết quả như sau :

	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 thời gian thực thi giữa các cách tiếp cận.

Từ đó ta có biểu đồ mô phỏng như sau :



So sánh thời gian thực thi giữa các chế độ đơn luồng, 3 luồng, 11 luồng và 27 luồng.

Biểu đồ trên cho thấy rằng trong bài toán như Sudoku Solution Validator, việc thực hiện đa tiểu trình trở nên không hiệu quả. Khi ta thực hiện càng nhiều luồng, thời gian thực thi càng lớn.

4 Giải thích kết quả

Trong quá trình kiểm tra hiệu suất thực thi của chương trình Sudoku Solution Validator, các kết quả cho thấy rằng việc tăng số lượng luồng (threads) không phải lúc nào cũng cải thiện thời gian thực thi. Một vài nguyên nhân của hiện tượng này bao gồm:

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.
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.

5 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.

6 Tài Liệu Tham Khảo

Sách: *Operating System Concepts* - Silberschatz, Galvin, Gagne.

7 Phụ lục

Link github của đồ án : Link