BÀI THỰC HÀNHMÔN HỌC: HỆ PHÂN TÁNCHƯƠNG 6: ĐỒNG BỘ HÓA

1. Triển khai đồng bộ các luồng trong một chương trình đa luồng sử dụng ngôn ngữ Java.

|  |
| --- |
| Câu hỏi 1: Chạy chương trình trên vài lần. Bạn nhận thấy điều gì? Giải thích! |

*Trả lời:*

Kết quả của chương trình không giống nhau sau mỗi lần chạy và không bằng 3000.

Giải thích: do các luồng không được đồng bộ khi sử dụng hàm override: hàm run()

|  |
| --- |
| Câu hỏi 2: Thay đổi đoạn mã trong chương trình chạy (phương thức *main*), thay đổi kiểu của 3 thực thể worker1-3 thành *ThreadedWorkerWithSync.* Bạn nhận thấy sự thay đổi gì ở đầu ra khi chạy chương trình đó khi so sánh với câu hỏi 1? Giải thích! |

*Trả lời:*

Đầu ra luôn bằng 3000. Do các luồng lúc này đã được đồng bộ (áp dụng *synchronized* ở biến rExp)

|  |
| --- |
| Câu hỏi 3: Thay đổi đoạn code của chương trình chạy chính bằng cách thay thế kiểu của 3 thực thể worker1-3 thành *ThreadedWorkerWithLock.* Có khác nhau gì so với đầu ra của câu hỏi 1. Giải thích! |

*Trả lời:*

Đầu ra luồn bằng 3000. Do các luồng lúc này cũng đã được đồng bộ hóa.

**2.** Lập trình song song với đoạn găng (ngôn ngữ C).

|  |
| --- |
| Câu hỏi 4: Hoàn thiện file trên (điền vào phần **YOUR-CODE-HERE**) với một vòng lặp để tăng biến *shared* lên một đơn vị trong vòng 5 giây. (gợi ý: hàm time(NULL) sẽ trả về giá trị thời gian của hệ thống với đơn vị là giây). |

*Trả lời:*

while (start != end) {

shared++;

}

|  |
| --- |
| Câu hỏi 5: Bây giờ hãy tăng giá trị số luồng và giá trị của số lần giao dịch  NUM\_TRANS sau mỗi lần chạy chương trình cho đến khi nào bạn thấy sự khác nhau giữa giá trị Balance (giá trị còn lại trong tài khoản) và  *INIT\_BALANCE+credits-debits.* Giải thích tại sao lại có sự khác nhau đó. |

*Trả lời:*

* Sau lần chạy chương trình với tham số luồng là 200 và NUM\_TRANS = 1000 thì có sự sai khác giữa giá trị Balance và INIT\_BALANCE+credits-debits.
* Lí do có sự khác nhau đó là do có nhiều luồng cùng lúc thực thi hàm transactions can thiệp vào biến credits và debits. Giá trị đó bị sửa đổi bởi nhiều luồng cùng lúc nên có thể kết quả của luồng này lại được sử dụng trong luồng khác, gây ra tính toán sai lệch.

|  |
| --- |
| Câu hỏi 6: Hãy build và chạy chương trình này. Chạy lặp đi lặp lại đến bao giờ bạn thấy sự khác nhau giữa 2 giá trị *Shared* và *Expect*. Phân tích mã nguồn để hiểu vấn đề. |

*Trả lời:*

* Chương trình chạy cho ra giá trị Shared và Expect giống nhau cho đến khi chạy vài lần chương trình có số luồng tăng lên giá trị khoảng 400.
* Chương trình vẫn có sự sai khác hai giá trị Shared và Expect: mặc dù đã sử dụng biến khóa lock quy định khi giá trị lock = 0 luồng mới được thực thi, tuy nhiên khi biến lock = 0 vẫn có thể có nhiều luồng cùng lúc được thực hiện. Như vậy hiện tượng biến credits và debits bị nhiều luồng can thiệp cùng lúc vẫn xảy ra.

|  |
| --- |
| Câu hỏi 7: Bây giờ hãy thay đổi đoạn code của file without-lock.c bằng cách triển khai cơ chế mutex lock như trên (bạn có thể tạo file mới và đặt tên khác đi như *mutex-lock-banking.c*). Chạy chương trình nhiều lần và đánh giá đầu ra. Nó có cải thiện gì hơn so với naive-lock? |

*Trả lời:*

* Chạy nhiều lần chương trình mutex-lock-banking với 400 vẫn cho ra kết quả hai giá trị Balance và INIT\_BALANCE+credits-debits bằng nhau.
* So với naive-lock, mutex-lock có khả năng xử lý chính xác với số luồng lớn hơn 500 cùng lúc mà vẫn ổn định. Sự sai khác xảy ra nhanh hơn khi NUM\_TRANS được tăng lên theo số nhân.

|  |
| --- |
| Câu hỏi 8: so sánh và đo đạt thời gian để chứng minh là Fine Locking sẽ nhanh hơn Coarse Locking. |

*Trả lời:*

* So với Coarse Locking, Fine Locking có thời gian thực hiện chương trình nhanh hơn.
* Tuy nhiên với phép tính can thiệp biến credits và debits đơn giản (credits = credits + v) thì Coarse Locking cho kết quả nhanh hơn Fine Locking.

|  |
| --- |
| Câu hỏi 9: chạy chương trình trên và bạn nhận thấy điều gì? Giải thích thông qua việc phân tích mã nguồn. |

*Trả lời:*

* Chương trình deadlock-test không chạy được do cả 2 đoạn găng can thiệp biến a và b đều không thể vào được vì chờ đợi lẫn nhau.