Software Transactional Memory II

Lecture 19

Alma Rahat

CS-210: Concurrency

20 April 2021

What did we do in the last session?

- Fixing Single Lane Bridge
- Data Parallelism
- Software Transactional Memory
- Bank account example

In this lecture...

Learning outcomes.

1. To apply data parallelism using the multiverse library for Software Transactional Memory.

Outline.

1. Examples: bank account and array element swapping.

Software Transactional Memory
Data Parallelism

Steps to follow.

1. Import the libraries.
2. Use TxnObject and associated method calls.
3. Wrap the sequence of actions that are needed to be atomicised.

Example: Simple Bank Account

```java
import org.multiverse.api.references.*;
import org.multiverse.api.StmUtils;
public class Account {
    private TxnDouble balance;
    Account(double initBalance){
        balance = StmUtils.newTxnDouble(initBalance);
    }
    public double getBalance(){
        return balance.atomicGet();
    }
    public void addBalance(double amount){
        balance.incrementAndGet(amount);
    }
    public void subtractBalance(double amount){
        balance.incrementAndGet(-amount);
    }
    public void transfer(Account to, double amount){
        subtractBalance(amount);
        to.addBalance(amount);
    }
    public void atomicTransfer(Account to, double amount)
        StmUtils.atomic(()->{ // atomic wrapper
           transfer(to, amount);
        });
    }
}
```

```java
public class AccountNonSTM {
    private double balance;
    AccountNonSTM(double initBalance){
        balance = initBalance;
    }
    public double getBalance(){
        return balance;
    }
    public void addBalance(double amount){
        balance = balance + amount;
    }
    public void subtractBalance(double amount){
        balance = balance - amount;
    }
    public void transfer(Account to, double amount){
        subtractBalance(amount);
        to.addBalance(amount);
    }
}
```

Example: Simple Bank Account

Live demonstration: code is on github.

Example: Array Swapping

Swansea University
Prifysgol Abertawe

Scenario.

We have a simple `BinaryArray` class, and we want to create a program where it is possible to concurrently swap elements between *two* instances of `BinaryArray`.

In task parallelism, we would have to think about locking first instance and then second instance for this, which may lead to deadlock. So, we will just make sure that if a `Thread` instance cannot get both, then simply let's go of the first array.

Example: Array Swapping

Swansea University
Prifysgol Abertawe

We will first import the library.

```
import org.multiverse.api.StmUtils;
import org.multiverse.api.references.*;
```

Example: Array Swapping

```
public class BinaryArray {
    private TxnBoolean[] array;
    private TxnRef<Date> lastModified;
    private int length;
    BinaryArray(int length){
        this.length = length;
        this.array = new TxnBoolean[length];
        for(int i=0; i<length; i++)
            array[i] = StmUtils.newTxnBoolean(false);
        this.lastModified = StmUtils.newTxnRef(new Date());
    }
    public int getLength(){
        return length;
    }
```

- We need a `TxnObject` for this purpose; there are a range of possibilities available in the library.
- To create new instances, we would use a method from `StmUtils`.

Example: Array Swapping

```
private boolean getStateAtIndex(int Id){
    return array[Id].get();
}
private boolean setStateAtIndex(int Id, boolean value){
    return array[Id].set(value);
}
public boolean atomicGetStateAtIndex(int Id){
    return array[Id].atomicGet();
}
public boolean atomicSetStateAtIndex(int Id, boolean value){
    return array[Id].atomicSet(value);
}
```

- Internally we can use standard `get` and `set` methods from within the class instance.
- Externally this throws errors, so we will only use the `atomic` versions of these.

Example: Array Swapping

```
public void atomicSwap(BinaryArray destination, int originId, int destId){
    StmUtils.atomic(() -> swap(destination, originId, destId));
}
private void swap(BinaryArray destination, int originId, int destId){
    boolean currentOrig = getStateAtIndex(originId);
    boolean currentDest = destination.getStateAtIndex(destId);
    setStateAtIndex(originId, currentDest);
    destination.setStateAtIndex(destId, currentOrig);
}
```

- `atomicSwap` is the method that will be called by `Thread` instances.
- The structure we will follow is: `StmUtils.atomic(() -> function(parameters))`.
- `swap` method just contains the set of steps that we need to perform atomically.

Example: Array Swapping

```
public class Swapper implements Runnable{
    private BinaryArray one, two;
    private int length;
    private String name;
    Swapper(String name, BinaryArray one, BinaryArray two){
        this.name = name;
        this.one = one;
        this.two = two;
        this.length = one.getLength();
    }
```

- The `Swapper` models our threads that will independently swap things around.
- It has a couple of `BinaryArrays`, on which it performs the swap operations.

Example: Array Swapping

```
@Override
public void run() {
    Random random = new Random();
    while(true){
        int rand0 = random.nextInt(length);
        int randD = random.nextInt(length);
        one.atomicSwap(two, rand0, randD);
        System.out.println(name + " swapped");
        try {
            Thread.sleep(100);
        } catch (InterruptedException ex) {
            System.out.println(name + " was interrupted!");
            break;
        }
    }
}
```

- This is the `run` method in the `Swapper`: arbitrarily swaps values between two arrays.

Example: Array Swapping

```
public static void main(String[] args) throws InterruptedException {
    BinaryArray a = new BinaryArray(10);
    BinaryArray b = new BinaryArray(10);
    a.atomicSetStateAtIndex(0, true);
    boolean n = a.atomicGetStateAtIndex(0);
    System.out.println("array a[0] current: " + n);
    n = b.atomicGetStateAtIndex(0);
    System.out.println("array b[0] current: " + n);
    a.atomicSwap(b, 0, 0);
    n = a.atomicGetStateAtIndex(0);
    System.out.println("array a[0] current: " + n);
    n = b.atomicGetStateAtIndex(0);
    System.out.println("array b[0] current: " + n);
```

- Firstly, we show standard calls to the methods in `BinaryArray` in the `main` method.

Example: Array Swapping

```
// threads
Swapper s1 = new Swapper("s1", a, b);
Thread t1 = new Thread(s1);
Swapper s2 = new Swapper("s2", b, a);
Thread t2 = new Thread(s2);
t1.start();
t2.start();
Thread.sleep(2000);
t1.interrupt();
t2.interrupt();
t1.join();
t2.join();
```

- We generate a couple of independent threads, and let them perform swapping on the arrays.

Example: Array Swapping

```
s1 swapped
s2 swapped
s1 swapped
s2 swapped
s1 swapped
s2 swapped
s1 swapped
s2 swapped
s1 was interrupted!
s2 was interrupted!
```

- Example output; it just works out of the box.

Any questions?

Summary

STM gives us a different way to approach. however, there is no one size fits all solution: we have to consider the situation and determine what is the best approach for our context.