The Tasks with Effects Model for Safe Concurrency

Stephen T. Heumann, Vikram S. Adve, Shengjie Wang

University of Illinois

presented by Jost Joller

University of Illinois

Parallel Computing Research Center at the University of Illinois at Urbana-Champaign.

Sponsored by Intel Corporation

Prof. Vikram Adve

- Parallel Computing
- Compilers
- Computer Security
- Operating Systems
- Programming Languages

Tasks with Effects (TWE)

- Objects are associated with regions
- Effects are write or read operations on regions
- Running tasks have exclusive access to regions

Deterministic Parallel Java

```
class Image {
    region Top, Bottom;
    int[] topHalf in Top;
    int[] bottomHalf in Bottom;
    void increaseContrastTop() writes Top { // write topHalf }
    void increaseContrastBottom() writes Bottom { // write bottomHalf }
    void increaseContrast() writes Top, Bottom {
        cobegin {
            this.increaseContrastTop();
            this.increaseContrastBottom();
        }
    }
}
```

- Data race freedom
- Atomicity
- Deadlock freedom
- Determinism

Lack of flexibility :-(

Tasks With Effects Java (simplified)

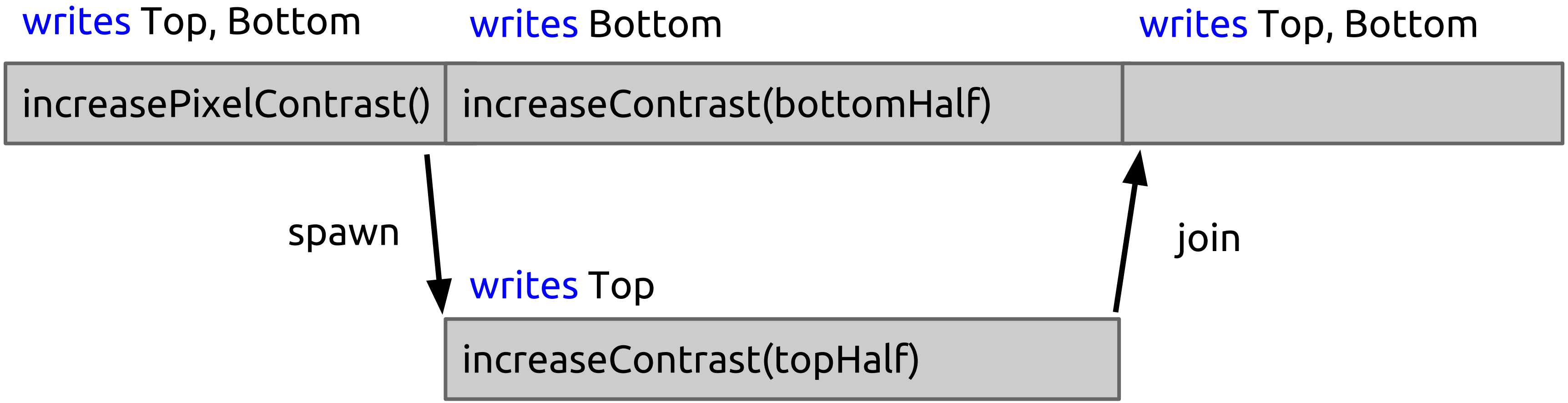
```
abstract class Task<type TRet, TArg, effect E>
{
    // code to be run when task is executed
    public abstract TRet run(TArg arg) effect E;
    // start task
    public SpawnedTaskFuture<TRet> spawn(TArg arg);
}

class SpawnedTaskFuture<type TRet, effect E>
{
    // await completion of task and get return value
    public TRet join();
}
```

Tasks With Effects Java

```
class Image {
    region Top, Bottom;
    int[] topHalf in Top;
    int[] bottomHalf in Bottom;
    public void increasePixelContrast() writes Top, Bottom {
        SpawnedTaskFuture<Void, writes Top> f = increaseContrast(topHalf).spawn(null);
        increaseContrast(bottomHalf).run(null);
        f.join();
    }
    private Task<Void, Void, writes R> increaseContrast(final int[] in R pixels) pure {
        return new Task<Void,Void, writes R>(){
            public Void run(Void _){ // modify pixels }
}   }   }
```

Effect Transfer in TWEJava



writes Top, Bottom

increasePixelContrast()

writes Bottom

increaseContrast(bottomHalf)

writes Top, Bottom

spawn

writes Top

increaseContrast(topHalf)

join

Flexibility? Not really...

Tasks With Effects Java (complete)

```
abstract class Task<type TRet, TArg, effect E> {
        // code to be run when task is executed
        public abstract TRet run(TArg arg) effect E;
        // execute a task at some point in the future without effect transfer
        public final TaskFuture<TRet> executeLater(TArg arg);
        // spawn a subtask of the current task, with effect transfer
        public final SpawnedTaskFuture<TRet, effect E> spawn(TArg arg);
}
class TaskFuture<type TReturn> {
        // await completion and get return value without effect transfer
        public TReturn getValue();
        // check if task is done without blocking
        public boolean isDone();
}
class SpawnedTaskFuture<type TReturn, effect E> extends TaskFuture<TReturn>{
        // await completion and get return value with effect transfer
        public TReturn join();
}
```

Parallel Control Flow with TWEJava

```
class Scientist {
    region Lab, Auditorium;
    Work research in Lab;
    Work teaching in Auditorium;

    public void doJob() writes Lab, Auditorium {
        TaskFuture researching = new Task<ResearchPaper, Work, writes Lab>() {

            public ResearchPaper run(Work research) { research.justDoIt();
                                                      return new ResearchPaper(research); }

        }.spawn(research);

        while ( !researching.isDone() ) {

            new Task<Void, Work, writes Auditorium>() {

                public Void run(Work teaching) { teaching.justDoIt(); return null; }

            }.spawn(teaching).join();
        }

        publish(writing.join());
}   }
```

Flexibility!

Security properties

- Data race freedom

  Exclusive access to regions
- ~~Atomicity~~

  Can break if a task does create new tasks or waits for other tasks.
- ~~Deadlock freedom~~

  Can happen since there are locks on regions.
- ~~Determinism~~

  Only limited control over task scheduling and termination.

@Deterministic

- Can be used to enforce determinism
- Only allows **spawn()** and **join()**

Limited to Fork-Join parallelism!

Regions are a burden

```
class Zoo {
    region Water, Jungle, Desert;
    Animal fish in Water;
    Animal monkey in Jungle;
    Animal tiger in Jungle;
    Animal camel in Desert;

    private void feed(Animal animal) effect E { // feed animal };

    public void feedAnimals(){
        // parallelizable (more or less)
        feed(fish); feed(monkey); feed(tiger); feed(camel);
    }
}
```

What is a smart way of defining regions?

Regions are a burden

```
class Zoo {
    region Water;
    Animal fish in Water;
    Animal monkey in Water;
    Animal tiger in Water;
    Animal camel in Water;

    private void feed(Animal animal) effect E { // feed animal };

    public void feedAnimals() {
        // not parallelizable :-(
        feed(fish); feed(monkey); feed(tiger); feed(camel);
    }
}
```

Multiple objects in same region hinders parallelization!

Regions are a burden

```
class Zoo {
        region Fish, Monkey, Tiger, Camel;
        Animal fish in Fish;
        Animal monkey in Monkey;
        Animal tiger in Tiger;
        Animal camel in Camel;

        private void feed(Animal animal) effect E { // feed animal };

        public void feedAnimals() {
                // parallelizable :-)
                feed(fish); feed(monkey); feed(tiger); feed(camel);
        }
}
```

In practice: Just put every object in its own region.

Regions are a burden

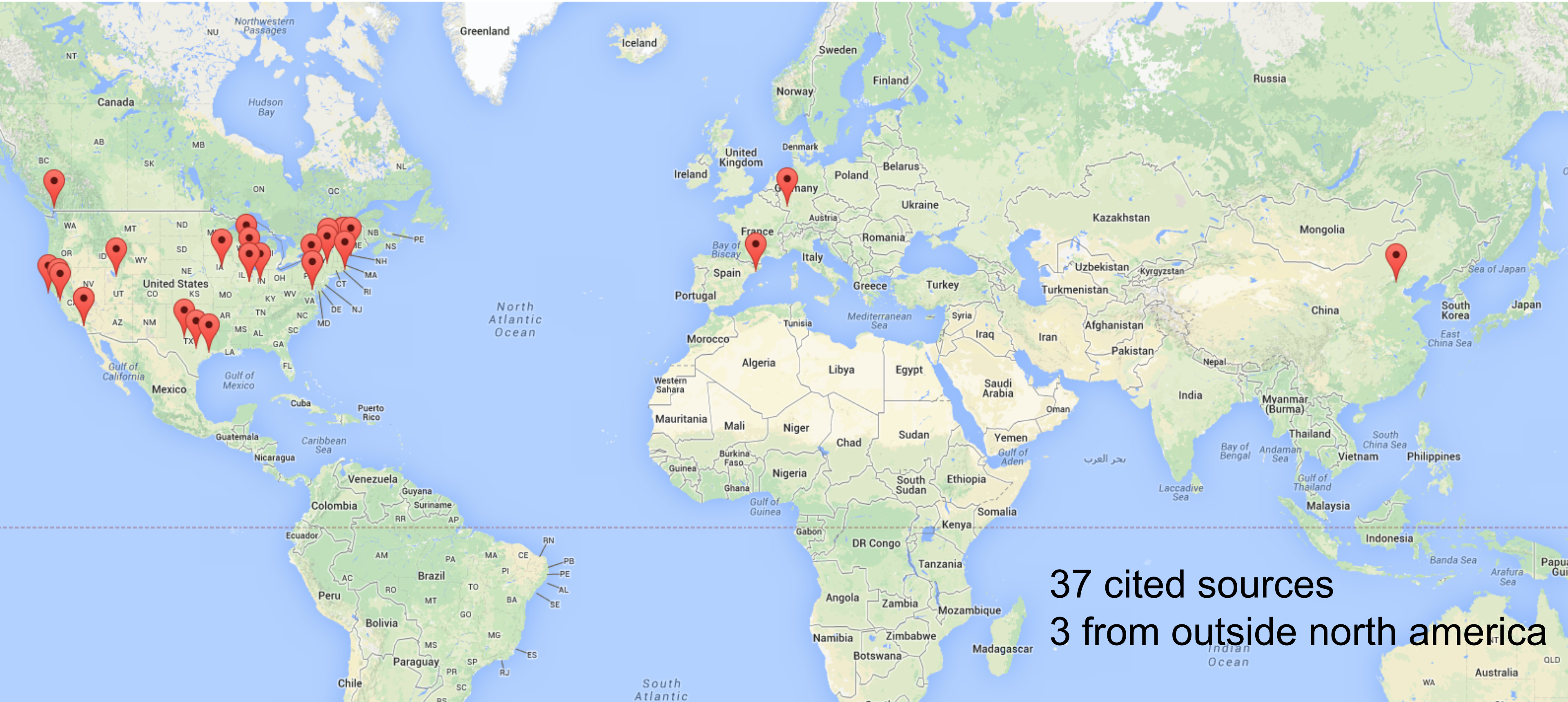
```
class Zoo {
    Animal fish inHisOwnRegion;
    Animal monkey inHisOwnRegion;
    Animal tiger inHisOwnRegion;
    Animal camel inHisOwnRegion;

    private void feed(Animal animal) effect E { // feed animal };

    public void feedAnimals() {
        // parallelizable :-)
        feed(fish); feed(monkey); feed(tiger); feed(camel);
    }
}
```

How about a keyword?

Location of sources



37 cited sources
3 from outside north america

Questions?