Multi Threaded **Testing with AOP is Easy** And it Finds Bugs!

Shmuel Ur (ur@il.ibm.com)
Shady Copty (shady@il.ibm.com)
Software and Verification Technologies

Outline

- Noise making as a testing technique
- The ConTest testing tool
- AspectJ
- AOP based noise maker
- Results
- Other applications
- Conclusions

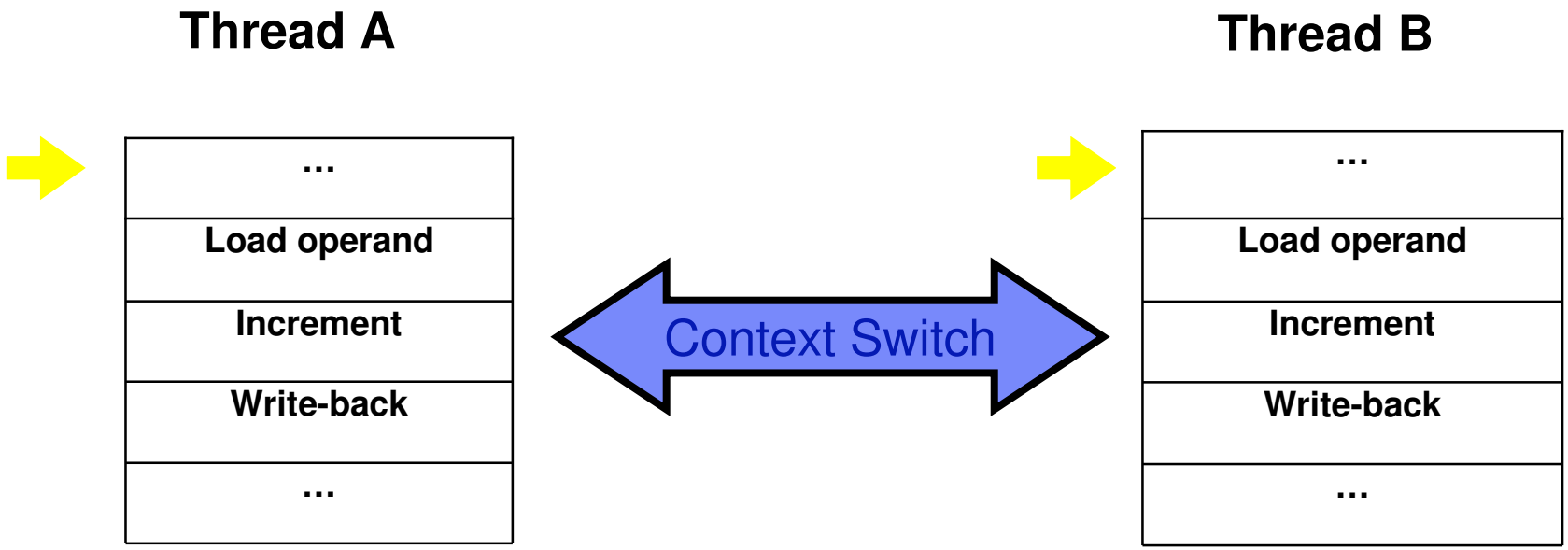
A Simple Java Example

- Function increase()

```
public void
increase()
{
        counter++;
}
```

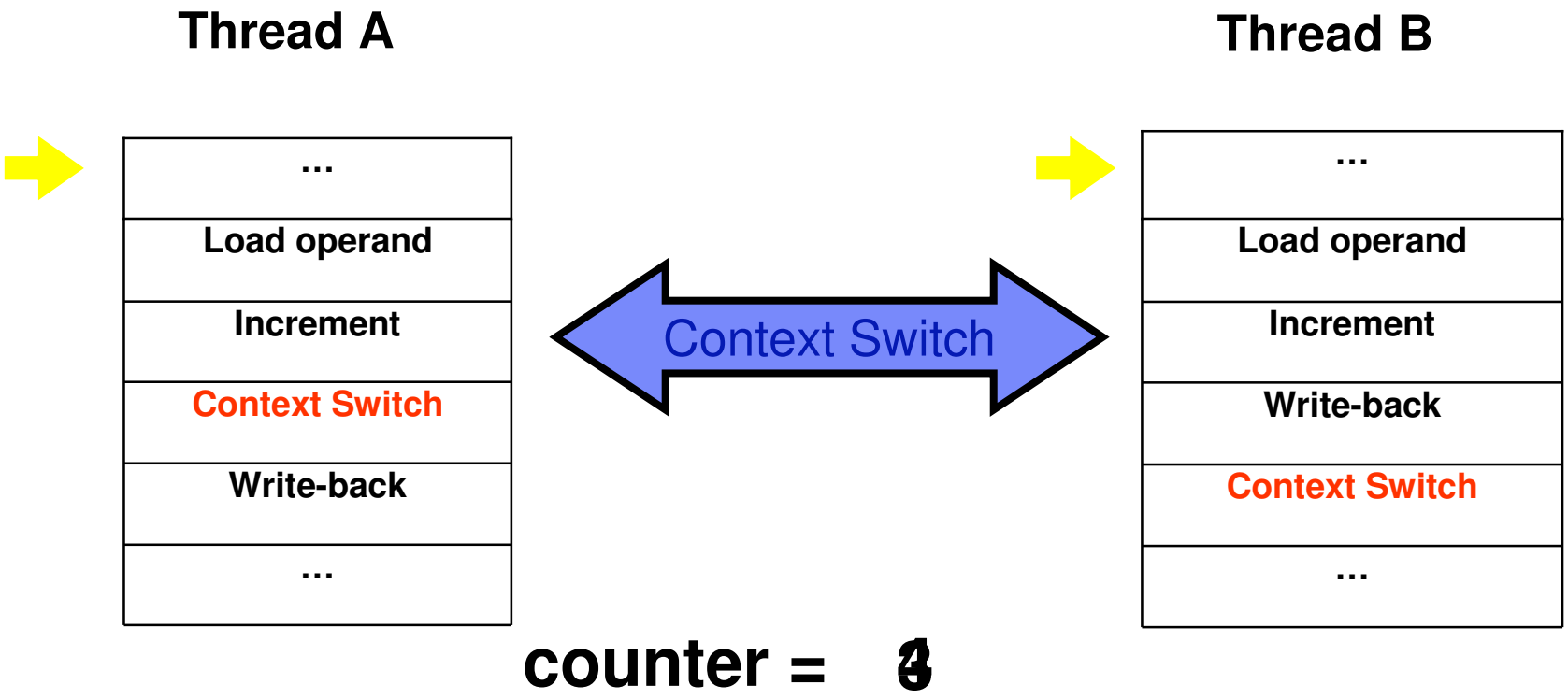
- Although written as a single “increment” operation, the “++” operator is actually mapped into three JVM instructions [load operand, increment, write-back]

A Simple Java Example - Continued



counter = 1

A Simple Java Example – Noise Makers



ConTest Overview

- **ConTest is composed of the following components**
  - An instrumentation engine that
    - Creates hooks for the irritator and for coverage printing
    - Generates coverage models
  - The instrumentation is done at the byte code level
  - An irritator that randomly, or using heuristics, generates new interleaving on-the-fly
  - Seed replay component
  - Coverage component
  - Debugging aids

AspectJ by Example

```
class IDManager{
    public int NewID() { return ++_counter; }
    private int _counter;
}
```

```
public aspect Example {
   pointcut IssueID():
       call (int IDManager.NewID());

   after(): IssueID() {
      System.out.println("Just gave a new ID");
   }
}
```

A more specific Aspect

```
class IDManager{
    public int NewID() { return ++_counter; }
    private int _counter;
}
```

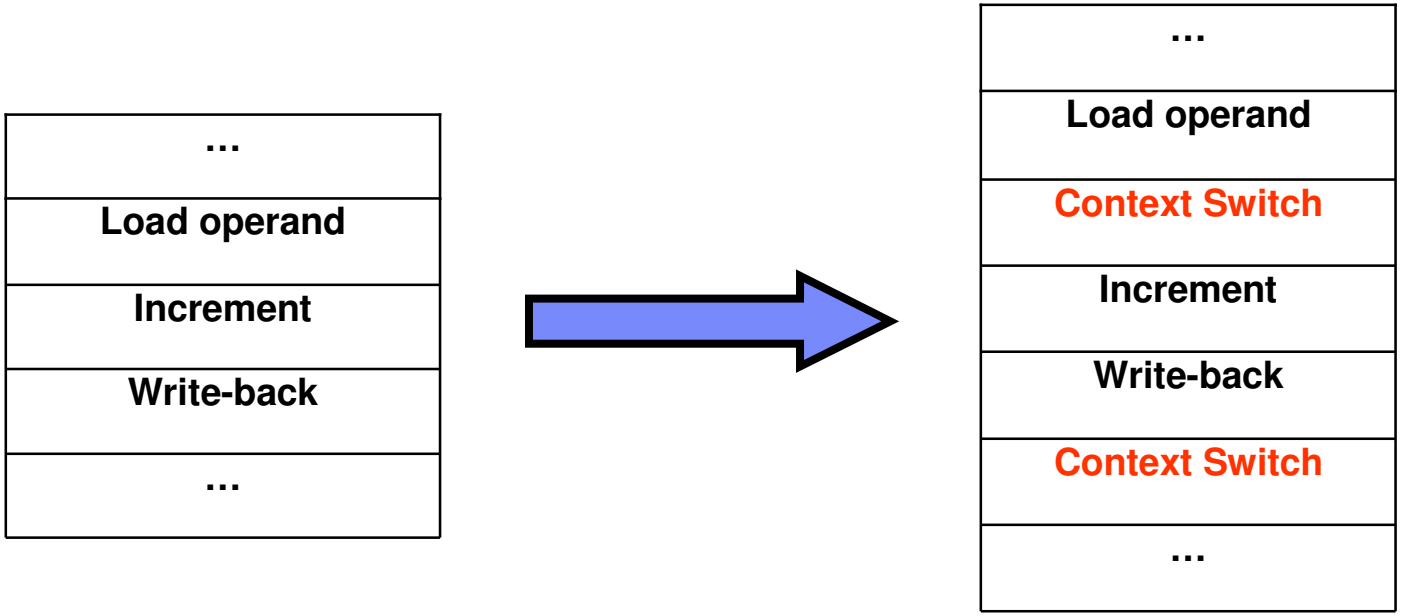
```
public aspect Example2 {
   pointcut ReadCounter():
      get(int IDManager._counter);

   after(): ReadCounter() {
      System.out.println("read the counter");
   }
}
```

A more general Aspect

Your Application

```
class IDManager{

}
public aspect Example4 extends Thread{
   pointcut Noise():
      (get(* *) || set (* *));
   pointcut AccessVariable():
      (get(* *) || set(* *));
   after(): Noise() {
      try {
   after(): AccessVariable() {
      yield();
      System.out.println("accessed something");
      } catch (Exception e) {};
   }
  }
}
```

Woven bytecode

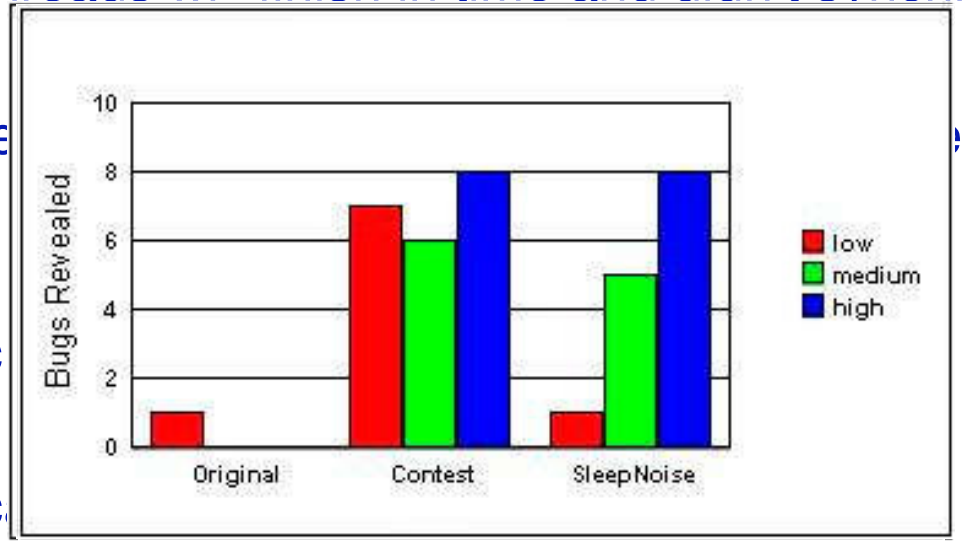


An AOP Testing Tool

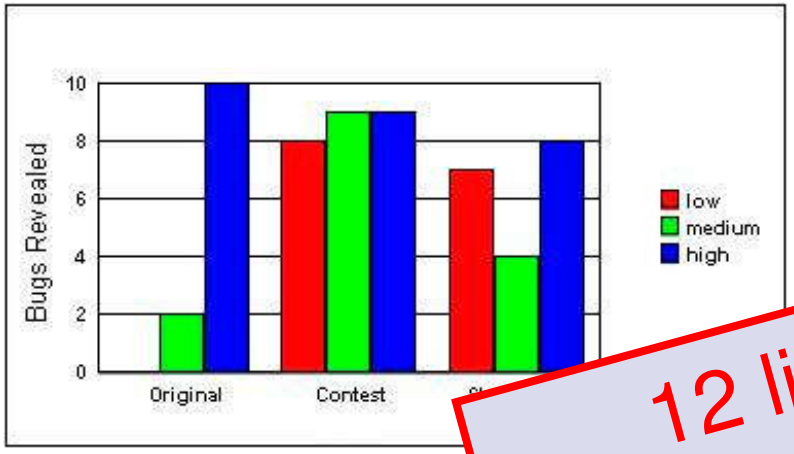
```
public aspect SleepNoise extends Thread{
    private static Random rand = new Random();

    pointcut noiseVictem():
        ((get(* *) || set (* *))&& within(!SleepNoise));

    after(): noiseVictem() {
        try{// noise
        if (rand.nextInt(100) == 1){ // activation
           sleep(rand.nextInt(50)); // type
          }
        } catch (Exception e) {};
    }
}
```

Experiment

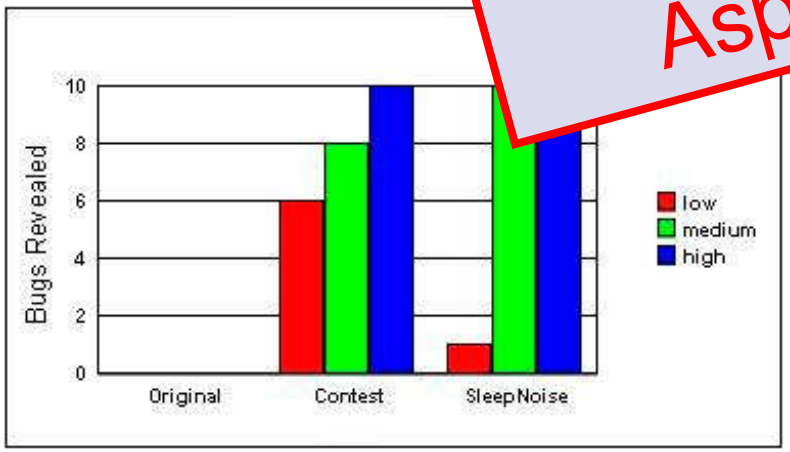
- Several programs with documented bugs
- BubbleSort
  - Assumed threads will finish in time and didn't synchronize
- BubbleSort2
  - Programme… his data structure
- IDManager
  - non-atomic…
- DoubleLock
  - Deadlock c…

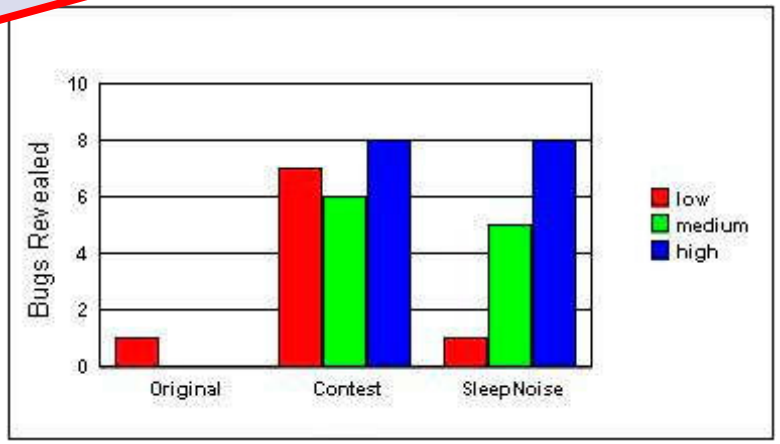


Experiment - results



12 lines of AspectJ code





Contest Instrumentation, AspectJ Falling Short?

- For Testing
  - gets/sets for variables
  - Concurrent related functions (sleep, yield, wait, …)
  - Synchronization blocks
- In addition, for coverage
  - Basic blocks
  - Method calls

Tampering with Timeouts

```
public aspect SleepMutator extends Thread{
   pointcut noiseVictem(long i):
     call(void sleep(long)) && args(i) && within(!SleepMutator);

   private static Random rand = new Random();
   void around(long i): noiseVictem(i) {
   try{
    long newSleepTime = rand.nextInt((int)i*3); // [0,3*sleep]

    if (rand.nextInt(5) == 1)
     proceed(newSleepTime);

    } catch (Exception e) {}
   }
};
```

Tampering with UDP

```
public aspect UDP extends DatagramSocket
    pointcut UDPNoise(DatagramPacket i):
      call(void send(DatagramPacket)) && args(i) && within(!UDP);

    private static Random rand = new Random();
    void around(DatagramPacket i): UDPNoise (i) {
     if (rand.nextInt(5) == 1)
       proceed(i);
       }
};
```

Coverage

- ConTest supports
  - Method coverage
  - Branch coverage
  - Concurrent point coverage
  - Interfered location pairs coverage
  - Synchronization coverage
- 100% coverage?
  - A new feature to AspectJ was added during our work
    - - ShowWeaveInfo

Concluding Remarks

- AOP is suitable for multi threaded testing
  - Finds bugs, extremely easy
- AspectJ as an open source instrumentor
  - If something is missing it is easy to add
- Detailed study of implementing ConTest features in the paper
- Source code is available on our website

Questions?

Bon Appetite