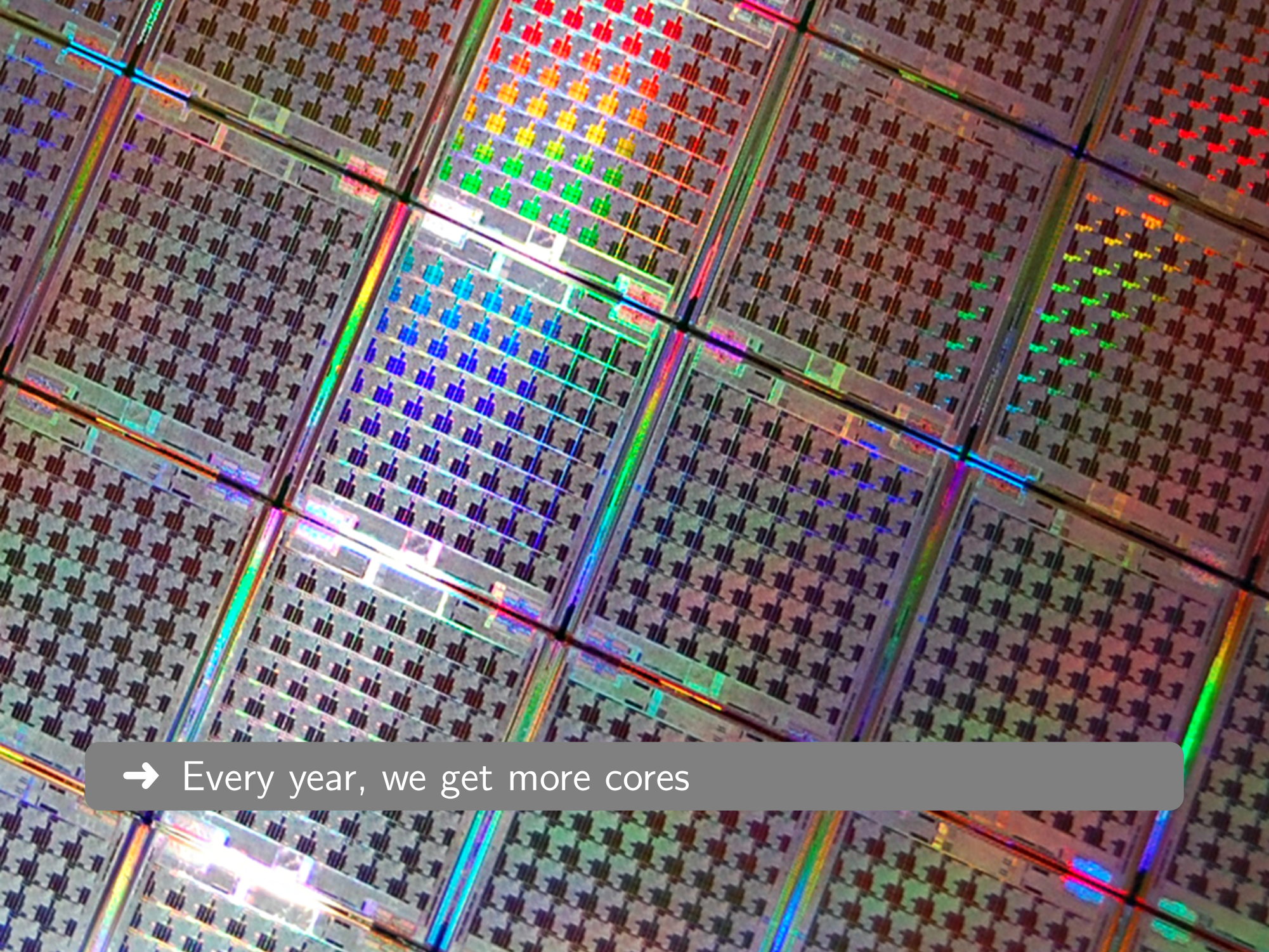
Speculating on top of an unmodified Java VM

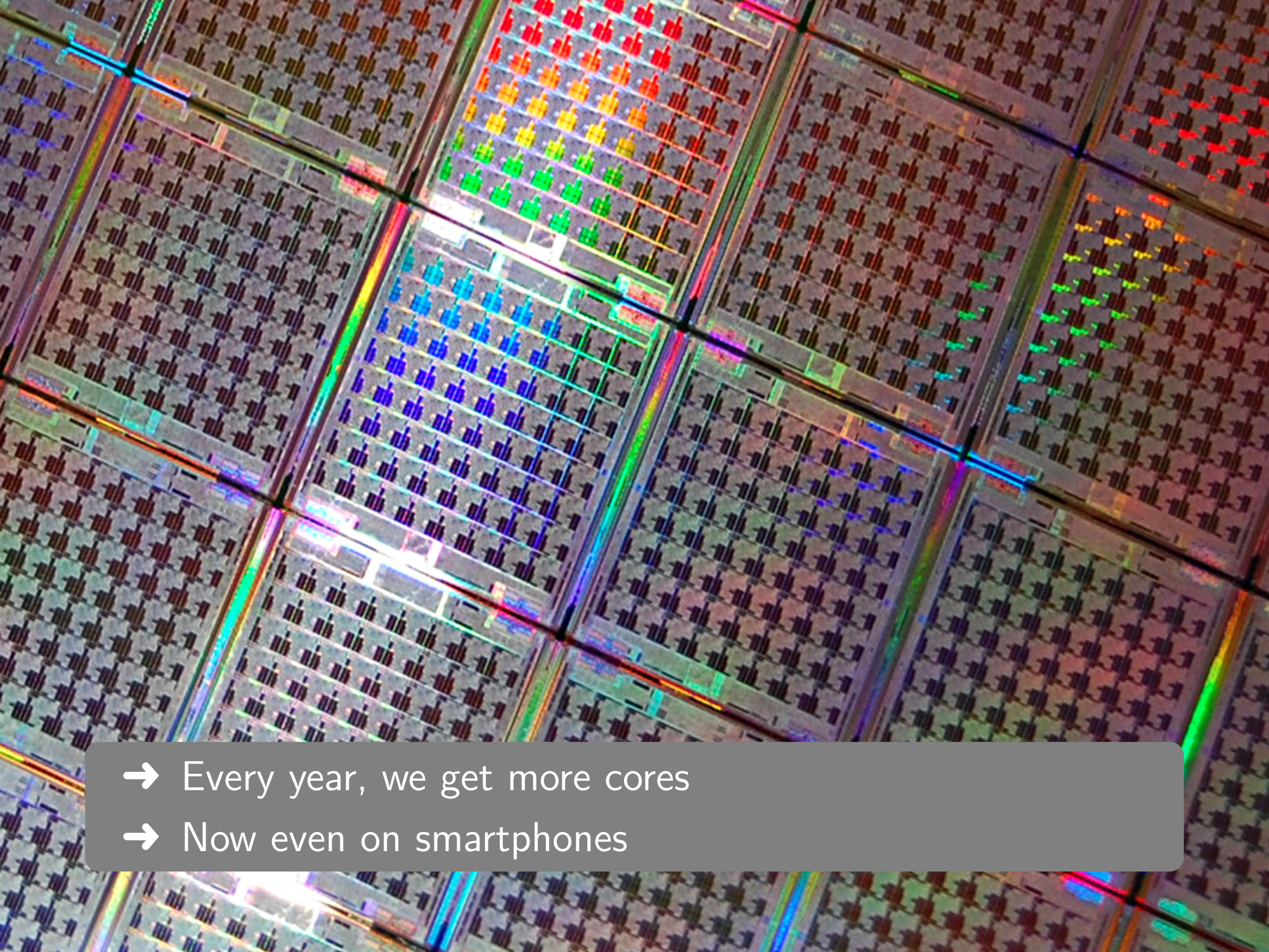
Ivo Anjo João Cachopo

ESW
INESC-ID Lisboa/Instituto Superior Técnico

October 2011



➜ Every year, we get more cores



- ➜ Every year, we get more cores
- ➜ Now even on smartphones

- Many applications still not parallel

- Many applications still not parallel
- Concurrent programming still very hard

- Many applications still not parallel
- Concurrent programming still very hard
  - ➜ Automatic parallelization?

Thread-Level Speculation (TLS)

- Automatic

Thread-Level Speculation (TLS)

- Automatic
- Optimistic

Thread-Level Speculation (TLS)

- Automatic
- Optimistic
  - ➜ Uses some kind of memory transactions

```
void method() {
    doA();
    doB();
    doC();
}
```

Can `doA()`, `doB()` and `doC()` be executed in parallel?

TLS on the Java platform

- Modify the VM to perform TLS

TLS on the Java platform

- Modify the VM to perform TLS
- Our idea: What if we left the VM **untouched**?

TLS on the Java platform

- Modify the VM to perform TLS
- Our idea: What if we left the VM **untouched**?
  - Propose missing features as TLS-independent Java APIs

(Not) Diving into the VM

Take advantage of Java concurrency features

- Memory model

(Not) Diving into the VM

Take advantage of Java concurrency features

- Memory model
  - Programming without locks

(Not) Diving into the VM

Take advantage of Java concurrency features

- Memory model
  - Programming without locks
- Concurrency utility classes

(Not) Diving into the VM

Take advantage of Java concurrency features

- Memory model
  - Programming without locks
- Concurrency utility classes
  - Collections

(Not) Diving into the VM

Take advantage of Java concurrency features

- Memory model
  - Programming without locks
- Concurrency utility classes
  - Collections
  - Atomic operations

(Not) Diving into the VM

Take advantage of Java concurrency features

- Memory model
  - Programming without locks
- Concurrency utility classes
  - Collections
  - Atomic operations
  - Fork/Join framework

(Not) Diving into the VM

- State-of-the-art garbage collection

(Not) Diving into the VM

- State-of-the-art garbage collection
  - Oracle Java Hotspot G1 GC

(Not) Diving into the VM

- State-of-the-art garbage collection
  - Oracle Java Hotspot G1 GC
  - Azul Pauseless GC

(Not) Diving into the VM

- State-of-the-art garbage collection
  - Oracle Java Hotspot G1 GC
  - Azul Pauseless GC
- Take advantage of JIT Optimizations

(Not) Diving into the VM

- State-of-the-art garbage collection
  - Oracle Java Hotspot G1 GC
  - Azul Pauseless GC
- Take advantage of JIT Optimizations
  - Do not need to be rewritten to support TLS

(Not) Diving into the VM

- State-of-the-art garbage collection
  - Oracle Java Hotspot G1 GC
  - Azul Pauseless GC
- Take advantage of JIT Optimizations
  - Do not need to be rewritten to support TLS
- Research VM vs Production

(Not) Diving into the VM

The big disadvantage: **overheads**!

- Spawn speculation

(Not) Diving into the VM

The big disadvantage: **overheads**!

- Spawn speculation
- Transaction setup

(Not) Diving into the VM

The big disadvantage: **overheads**!

- Spawn speculation
- Transaction setup
- Transactional execution

(Not) Diving into the VM

The big disadvantage: **overheads**!

- Spawn speculation
- Transaction setup
- Transactional execution

- Speculations will need to be longer-running

(Not) Diving into the VM

The big disadvantage: **overheads**!

- Spawn speculation
- Transaction setup
- Transactional execution

- Speculations will need to be longer-running
  - Will need to support nested speculation

Challenges

Transactification

- Need to validate speculative reads
- Need to buffer speculative writes

Transactification

- Need to validate speculative reads
- Need to buffer speculative writes

- Rely on Software Transactional Memory (STM)

Transactification

- Need to validate speculative reads
- Need to buffer speculative writes

- Rely on Software Transactional Memory (STM)
  - ➜ Transactify code, intercepting field/array accesses

Transactification

Can we improve on current STMs?

- Fixed commit ordering

Transactification

Can we improve on current STMs?

- Fixed commit ordering
- Relax isolation

Transactification

Can we improve on current STMs?

- Fixed commit ordering
- Relax isolation
- Concurrent nesting

Transactification

Can we improve on current STMs?

- Fixed commit ordering
- Relax isolation
- Concurrent nesting
- Non-conservative/dynamic transactification

Non-Transactional Operations

Operations that change state outside STM control

Non-Transactional Operations

Operations that change state outside STM control

Strategies

- Static identification of `native` calls

Non-Transactional Operations

Operations that change state outside STM control

Strategies

- Static identification of `native` calls
- Runtime detection and prevention

Non-Transactional Operations

Lifting restrictions

- Extend transactional system

Non-Transactional Operations

Lifting restrictions

- Extend transactional system
  - Transactional IO

Non-Transactional Operations

Lifting restrictions

- Extend transactional system
  - Transactional IO
- Identify harmless `native` calls

Stack Manipulation

Saving/Reading ➜ Continuations

Stack Manipulation

Saving/Reading ➜ Continuations

- Bytecode modification

Stack Manipulation

Saving/Reading ➜ Continuations

- Bytecode modification
- OpenJDK patches

Threads & Scheduling

- No control over scheduling

Threads & Scheduling

- No control over scheduling
- Thread pools

Threads & Scheduling

- No control over scheduling
- Thread pools
  - Issues with waiting

Threads & Scheduling

- No control over scheduling
- Thread pools
  - Issues with waiting
  - Can we design a better pool?

Threads & Scheduling

- No control over scheduling
- Thread pools
  - Issues with waiting
  - Can we design a better pool?
- Control Issues

Threads & Scheduling

- No control over scheduling
- Thread pools
  - Issues with waiting
  - Can we design a better pool?
- Control Issues
  - Loops caused by inconsistent reads

Threads & Scheduling

- No control over scheduling
- Thread pools
  - Issues with waiting
  - Can we design a better pool?
- Control Issues
  - Loops caused by inconsistent reads
    - Cannot kill threads

Threads & Scheduling

- No control over scheduling
- Thread pools
  - Issues with waiting
  - Can we design a better pool?
- Control Issues
  - Loops caused by inconsistent reads
    - Cannot kill threads
  - User code can catch any exception

Threads & Scheduling

- No control over scheduling
- Thread pools
  - Issues with waiting
  - Can we design a better pool?
- Control Issues
  - Loops caused by inconsistent reads
    - Cannot kill threads
  - User code can catch any exception
- Green threads would be helpful

In Conclusion

- Most things needed for TLS can be done on top of the VM

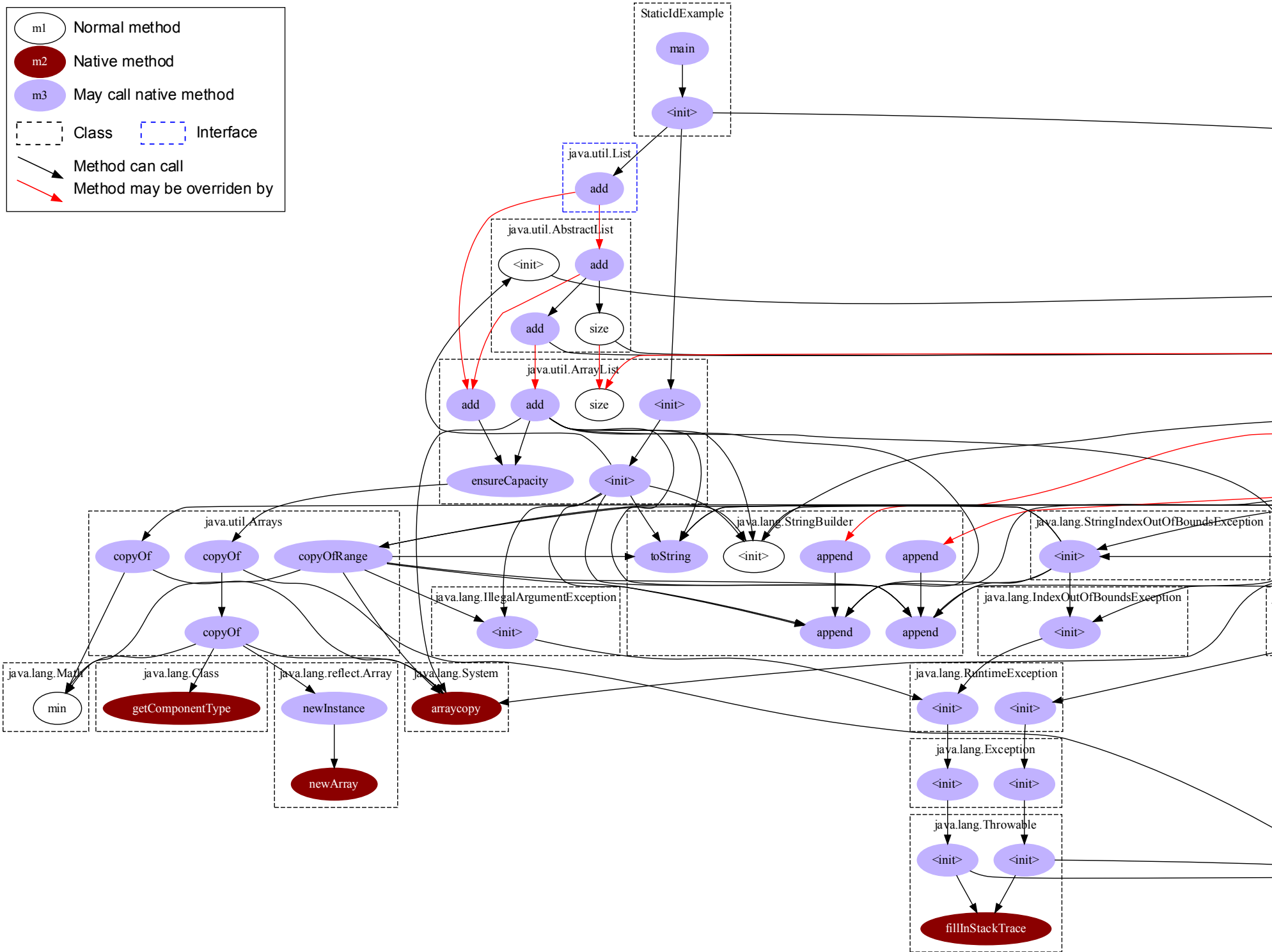
In Conclusion

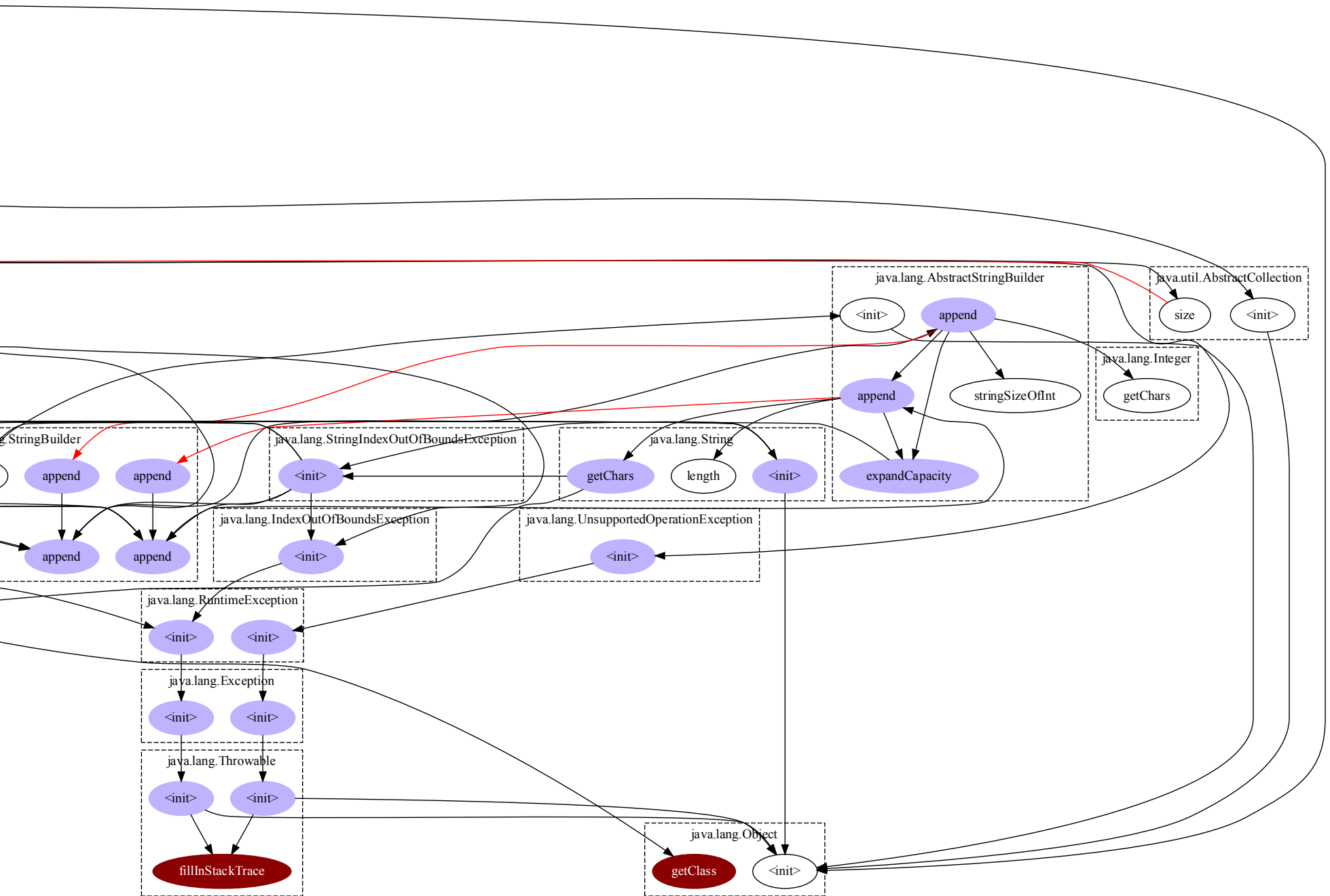
- Most things needed for TLS can be done on top of the VM
- Extra APIs would be helpful, but need not be TLS-specific

In Conclusion

- Most things needed for TLS can be done on top of the VM
- Extra APIs would be helpful, but need not be TLS-specific
  - ➜ Continuations, more control over threads, TM support, ...

Thank you!


m1
Normal method
m2
Native method
m3
May call native method
Class
Interface
Method can call
Method may be overriden by
StaticIdExample
main
<init>
java.util.List
add
java.util.AbstractList
<init>
add
add
size
java.util.ArrayList
add
add
size
<init>
ensureCapacity
<init>
java.util.Arrays
copyOf
copyOf
copyOfRange
copyOf
java.lang.StringBuilder
toString
<init>
append
append
append
append
java.lang.StringIndexOutOfBoundsException
<init>
java.lang.IllegalArgumentException
<init>
java.lang.IndexOutOfBoundsException
<init>
java.lang.Math
min
java.lang.Class
getComponentType
java.lang.reflect.Array
newInstance
newArray
java.lang.System
arraycopy
java.lang.RuntimeException
<init>
<init>
java.lang.Exception
<init>
<init>
java.lang.Throwable
<init>
<init>
fillInStackTrace


java.lang.AbstractStringBuilder
<init>
append
append
stringSizeOfInt
expandCapacity
java.util.AbstractCollection
size
<init>
java.lang.Integer
getChars
.StringBuilder
append
append
append
append
java.lang.StringIndexOutOfBoundsException
<init>
java.lang.String
getChars
length
<init>
java.lang.IndexOutOfBoundsException
<init>
java.lang.UnsupportedOperationException
<init>
java.lang.RuntimeException
<init>
<init>
java.lang.Exception
<init>
<init>
java.lang.Throwable
<init>
<init>
fillInStackTrace
java.lang.Object
getClass
<init>