Speculative Parallelization on the JVM

Ivo Anjo

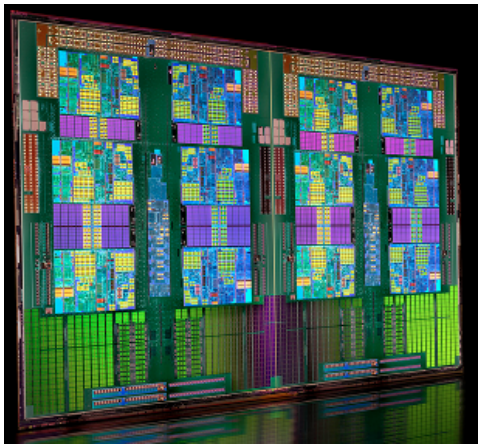
ESW
INESC-ID Lisboa

June 2010

- You can get a lot of processing cores these days

- You can get a lot of processing cores these days (Azul Vega!)

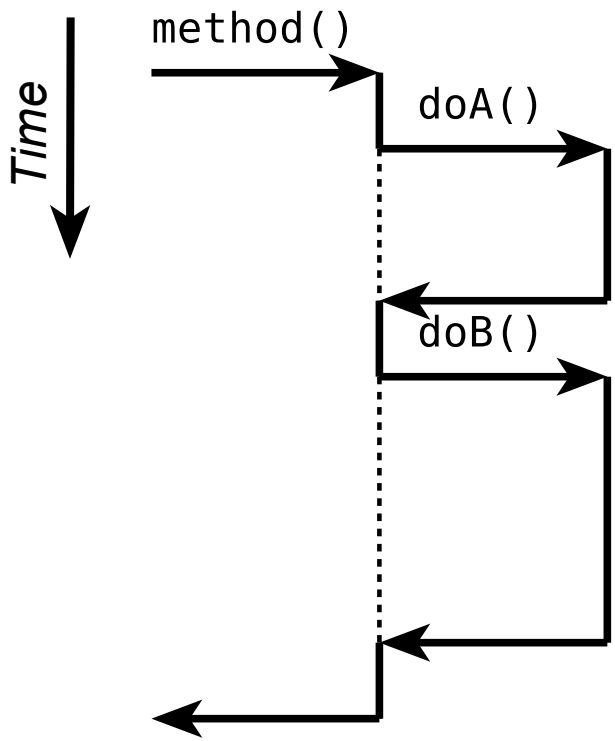
- You can get a lot of processing cores these days (Azul Vega!)
- ...even on a cheap x86 CPU:



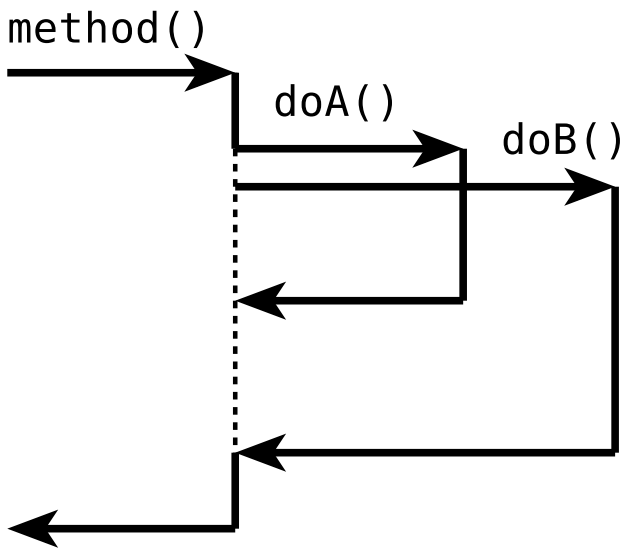
SUNNYVALE, Calif. — 3/29/2010 AMD announces [...] the world's first 8- and **12-core** *x86 processor for the high-volume 2P and value 4P server market.*

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

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



Sequential execution



Parallel execution

Problem

- Can `doA()` and `doB()` be run in parallel?

Problem

- Can `doA()` and `doB()` be run in parallel?
  - Shared state

Problem

- Can `doA()` and `doB()` be run in parallel?

Our Approach: Speculative Parallelization

Problem

- Can `doA()` and `doB()` be run in parallel?

Our Approach: Speculative Parallelization

- Maybe, let's try and we'll see how it goes

Speculative Parallelization

- Parallelization system does not have to prove a parallelization valid

Speculative Parallelization

- Parallelization system does not have to prove a parallelization valid
- Maps speculative executions to memory transactions

Speculative Parallelization

- Parallelization system does not have to prove a parallelization valid
- Maps speculative executions to memory transactions
- Most TLS approaches need hardware support for transactions

Speculative Parallelization

- Parallelization system does not have to prove a parallelization valid
- Maps speculative executions to memory transactions
- Most TLS approaches need hardware support for transactions
  - Limits

Speculative Parallelization

- Parallelization system does not have to prove a parallelization valid
- Maps speculative executions to memory transactions
- Most TLS approaches need hardware support for transactions
  - Limits
  - Hardware support just isn't there

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc
- TLS system for Java/JVM

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

- TLS system for Java/JVM
- Works at the JVM bytecode level

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

- TLS system for Java/JVM
- Works at the JVM bytecode level
- Used the JVSTM

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

- TLS system for Java/JVM
- Works at the JVM bytecode level
- Used the JVSTM
  - Now “jvstm-rulam”

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

- TLS system for Java/JVM
- Works at the JVM bytecode level
- Used the JVSTM
  - Now "jvstm-rulam"
- Also done in Java

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

- TLS system for Java/JVM
- Works at the JVM bytecode level
- Used the JVSTM
  - Now "jvstm-rulam"
- Also done in Java

- Two big components:

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

- TLS system for Java/JVM
- Works at the JVM bytecode level
- Used the JVSTM
  - Now “jvstm-rulam”
- Also done in Java

- Two big components:
  - Transactification

JaSPEx

Java Speculative Parallel Executor

- Developed during my MSc

- TLS system for Java/JVM
- Works at the JVM bytecode level
- Used the JVSTM
  - Now "jvstm-rulam"
- Also done in Java

- Two big components:
  - Transactification
  - Speculation

Transactification

- No support for memory transactions on the JVM

Transactification

- No support for memory transactions on the JVM
- JVSTM is just a Java library

Transactification

- No support for memory transactions on the JVM
- JVSTM is just a Java library

- Application needs to be modified to use the JVSTM

Transactification

- No support for memory transactions on the JVM
- JVSTM is just a Java library

- Application needs to be modified to use the JVSTM
  ➜ "Transactification"

Transactification Problems

- `native` methods

Transactification Problems

- `native` methods
- Cannot load modified versions of `java.*` classes on the Sun JVM

Transactification Problems

- `native` methods
- Cannot load modified versions of `java.*` classes on the Sun JVM
- Arrays, reflection, ...

Transactification Problems

- `native` methods
- Cannot load modified versions of `java.*` classes on the Sun JVM
- Arrays, reflection, ...

➜ We need to detect and handle nontransactional operations

Speculation

- Speculative execution of methods

Speculation

- Speculative execution of methods
- When a method is invoked, some of the methods it invokes may be run speculatively

Speculation

- Speculative execution of methods
- When a method is invoked, some of the methods it invokes may be run speculatively

```
int fib(int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

Speculation Problems

- Lots of overhead instrumenting methods so we can do speculation

Speculation Problems

- Lots of overhead instrumenting methods so we can do speculation
- Can only jump intra-method

Speculation Problems

- Lots of overhead instrumenting methods so we can do speculation
- Can only jump intra-method
  - Cannot start execution at a given instruction

Speculation Problems

- Lots of overhead instrumenting methods so we can do speculation
- Can only jump intra-method
  - Cannot start execution at a given instruction
  - Hard to do finer-grain stuff

Speculation Problems

- Lots of overhead instrumenting methods so we can do speculation
- Can only jump intra-method
  - Cannot start execution at a given instruction
  - Hard to do finer-grain stuff
- Currently, cannot reuse threads FJ style, hard to migrate call stacks

jvstm for speculation

What I need:

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

What I don't need:

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

What I don't need:

- atomicity

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

What I don't need:

- atomicity
- versions

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

What I don't need:

- atomicity
- versions
- read-only

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

What I don't need:

- atomicity
- versions
- read-only
- ...

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

What I don't need:

- atomicity
- versions
- read-only
- ...

What would come in handy:

jvstm for speculation

What I need:

- buffer speculative writes (aka write-set)
- remember+validate speculative reads (aka read-set)

What I don't need:

- atomicity
- versions
- read-only
- ...

What would come in handy:

- non-linear nesting :)

jvstm-rulam

- no versions (~~VBoxBody~~)

jvstm-rulam

- no versions (~~VBoxBody~~)
- no sincronization, locking

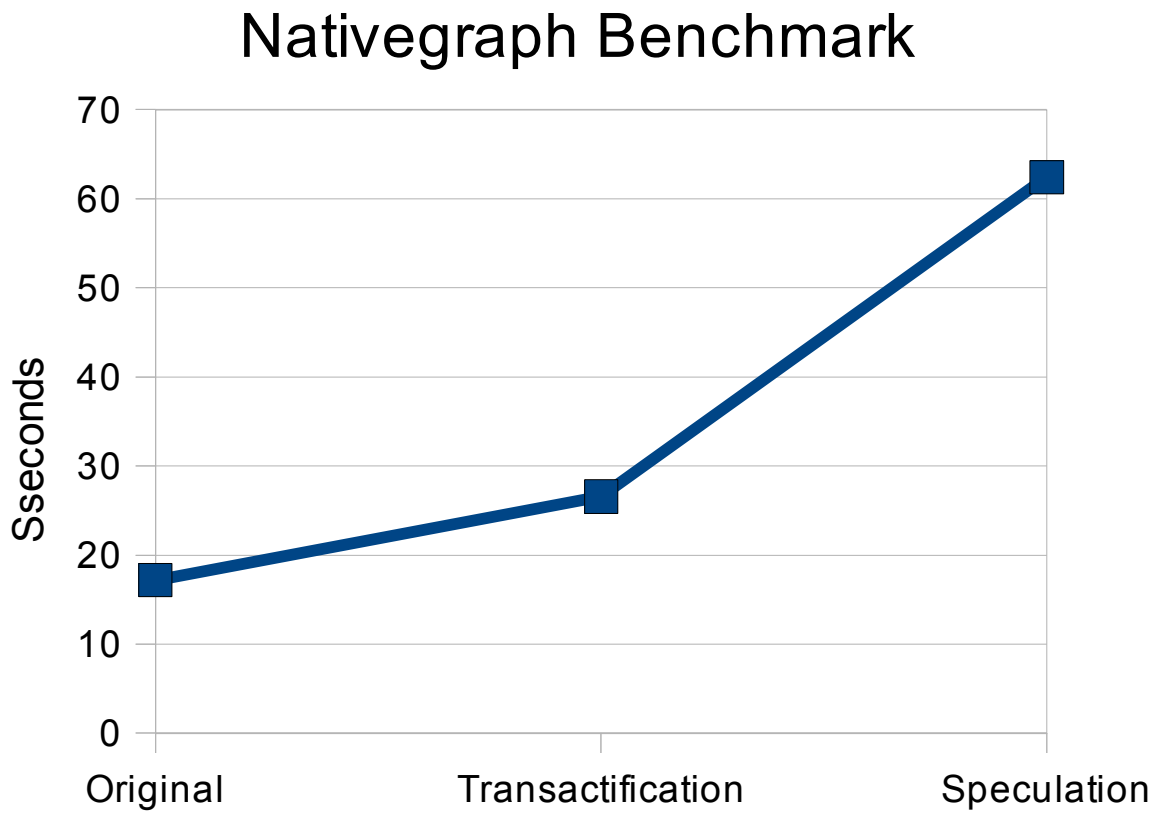
jvstm-rulam

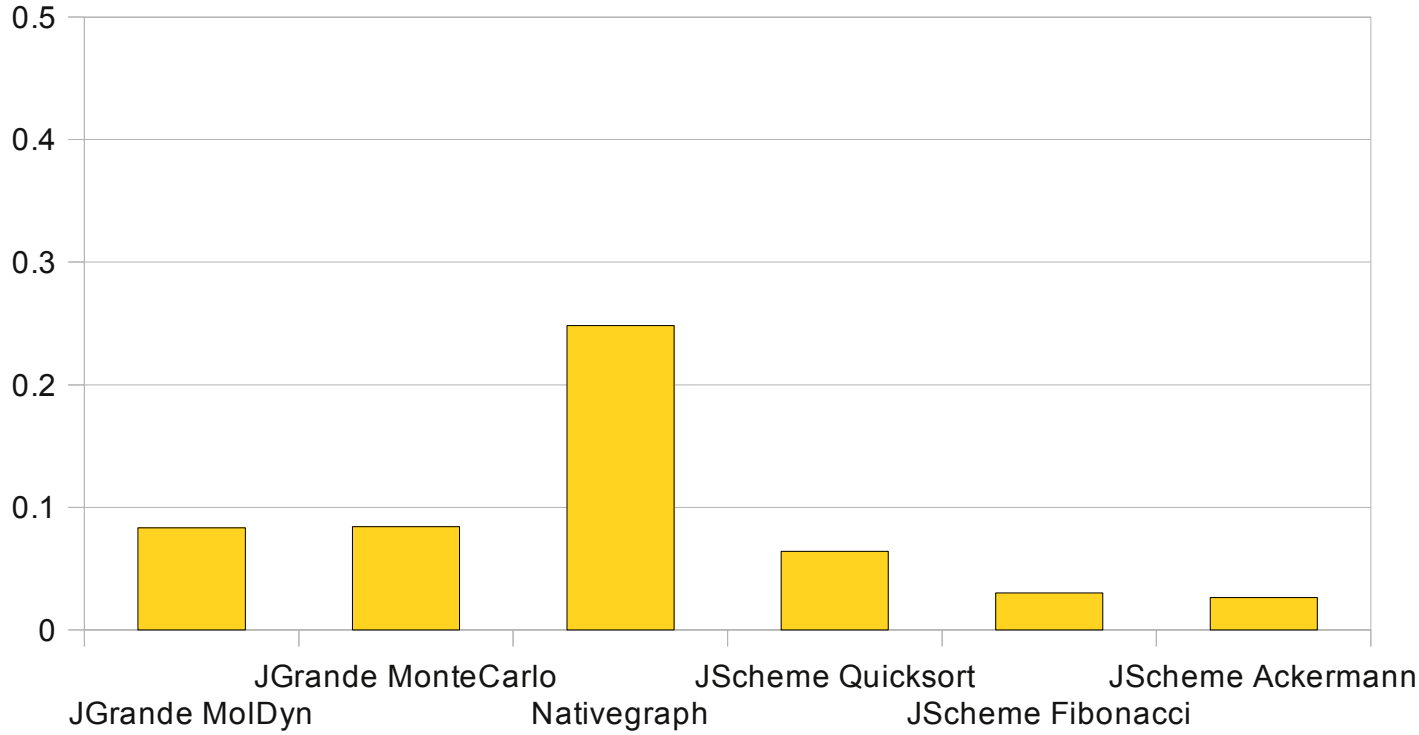
- no versions (~~VBoxBody~~)
- no sincronization, locking
- reads and writes to boxes outside transactions are immediately visible

jvstm-rulam

- no versions (~~VBoxBody~~)
- no sincronization, locking
- reads and writes to boxes outside transactions are immediately visible
- ...

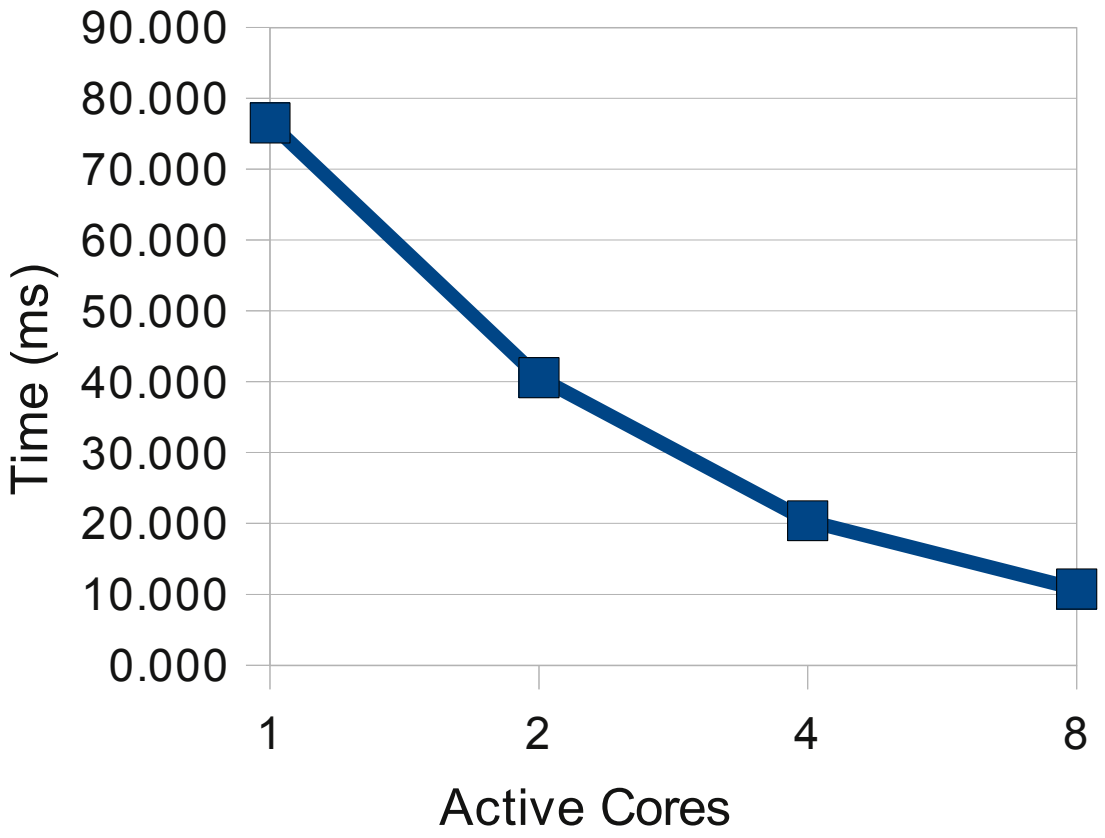
Benchmarks


Nativegraph Benchmark
Sseconds
70
60
50
40
30
20
10
0
Original
Transactification
Speculation



Relative **slowdown** for some benchmarks

➜ Should be a bit better now



Time for calculating `fib(50)` com 1..8 cores

Conclusions/Future Work

- Doing all of this using bytecode was an interesting experiment

Conclusions/Future Work

- Doing all of this using bytecode was an interesting experiment
- We hope to solve some of these issues and obtain real results by adapting a JVM to our use-case

The end

Questions?