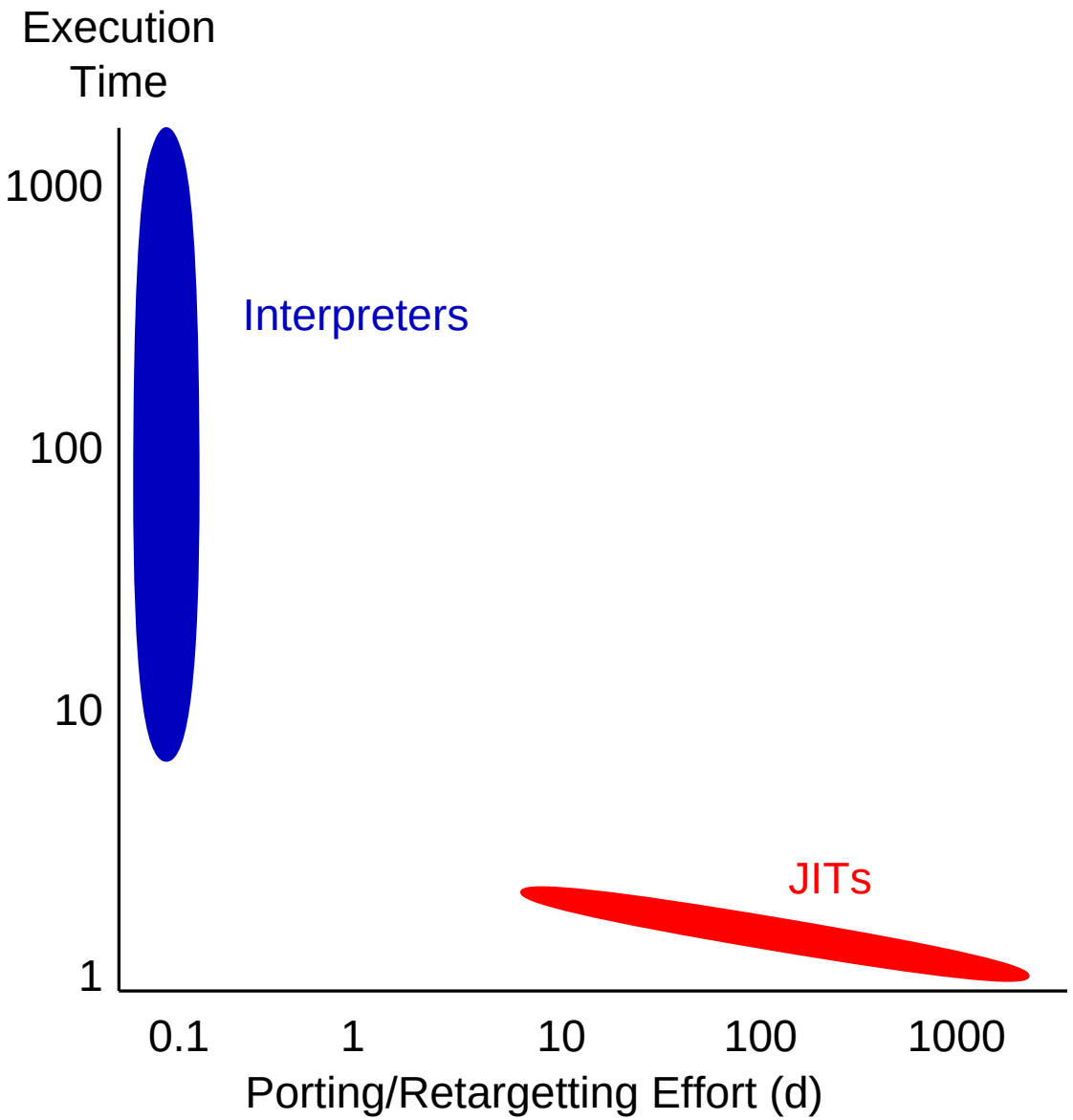
Superinstructions and Replication in the Cacao JVM interpreter

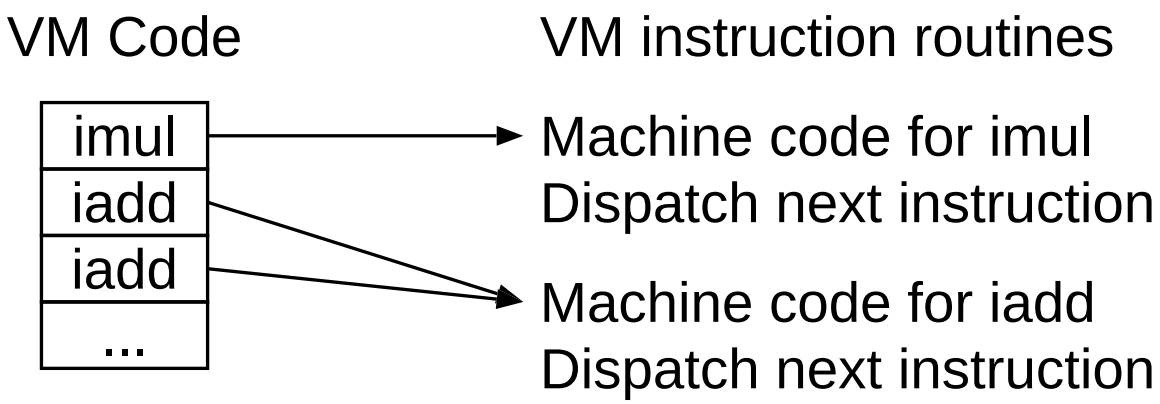
M. Anton Ertl Christian Thalinger Andreas Krall
TU Wien

Why interpreters?



Architecture	Mono	Cacao
Alpha	interp.	JIT
AMD64	JIT	JIT
ARM	JIT	JIT
HP-PA	interp.	
IA32	JIT	JIT
IA64	JIT	
MIPS		JIT
MIPS64		JIT
PowerPC	JIT	JIT
PowerPC64		interp.
s390	JIT	
s390x	JIT	
SPARC		
SPARC64		

Threaded Code

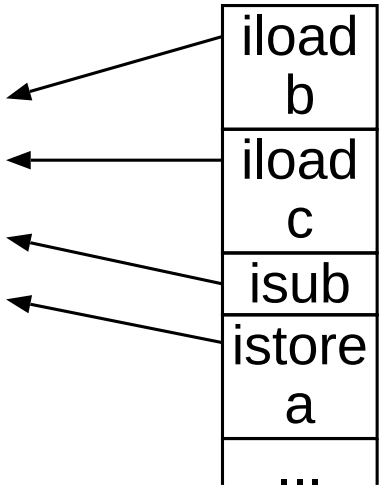


Dynamic Superinstructions

data segment
dyn. superinst code

Machine code for iload
Machine code for iload
Machine code for isub
Machine code for istore
...
Dispatch next

data segment
threaded VM Code

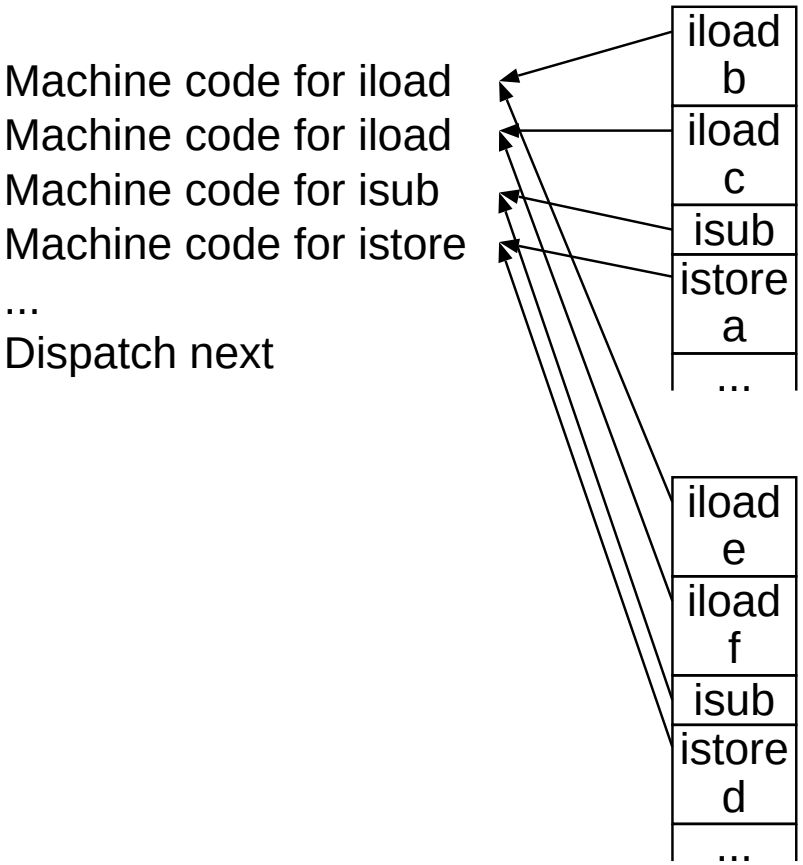


code segment
VM routine template

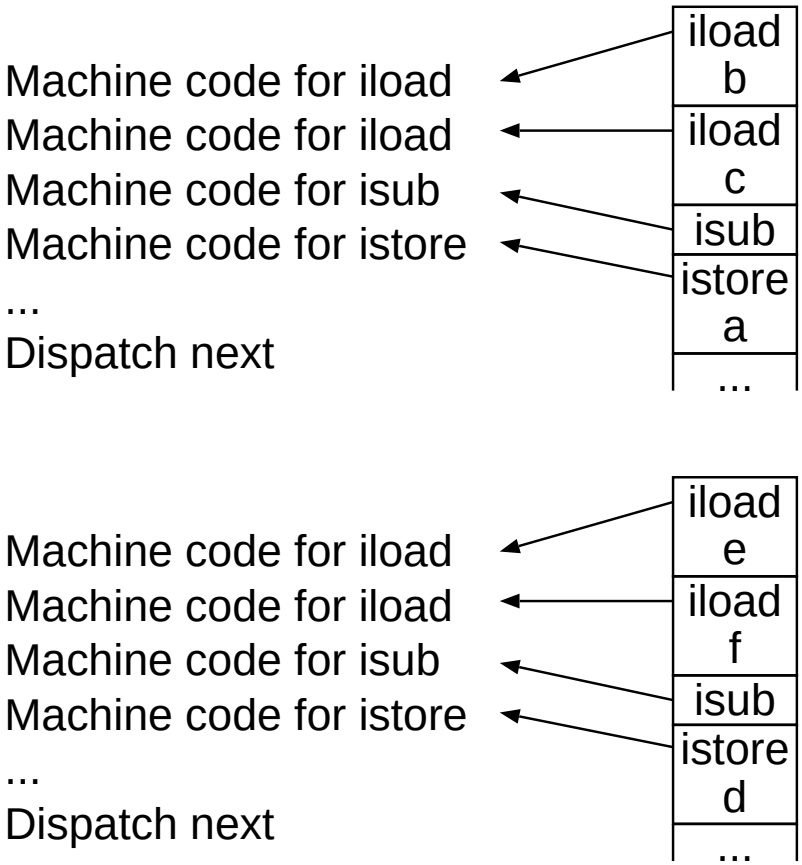
Machine code for isub
Dispatch next
Machine code for iload
Dispatch next
Machine code for istore
Dispatch next

Replication

No Replication



Replication



\+ Increases BTB prediction accuracy

\+ Simpler

– Increases code size

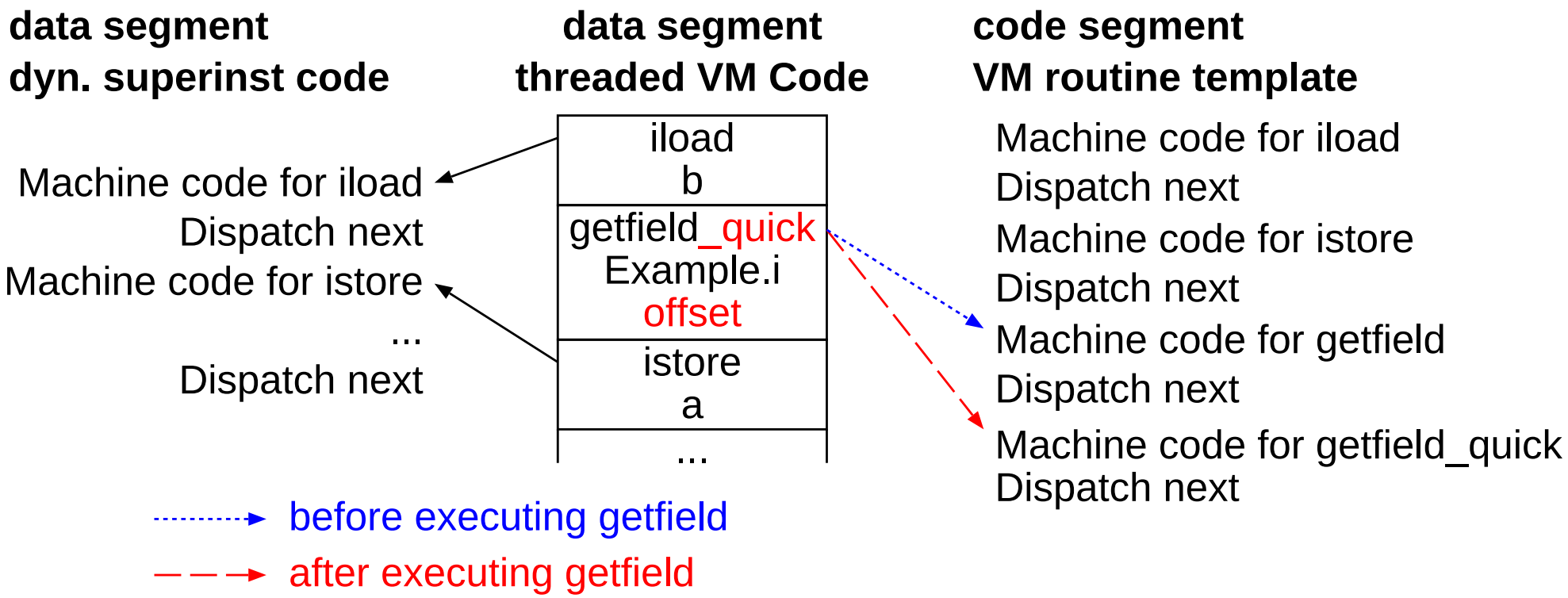
JVM and .NET problems

- Quickening
- Potential exception-throwing instructions
- How much benefit?

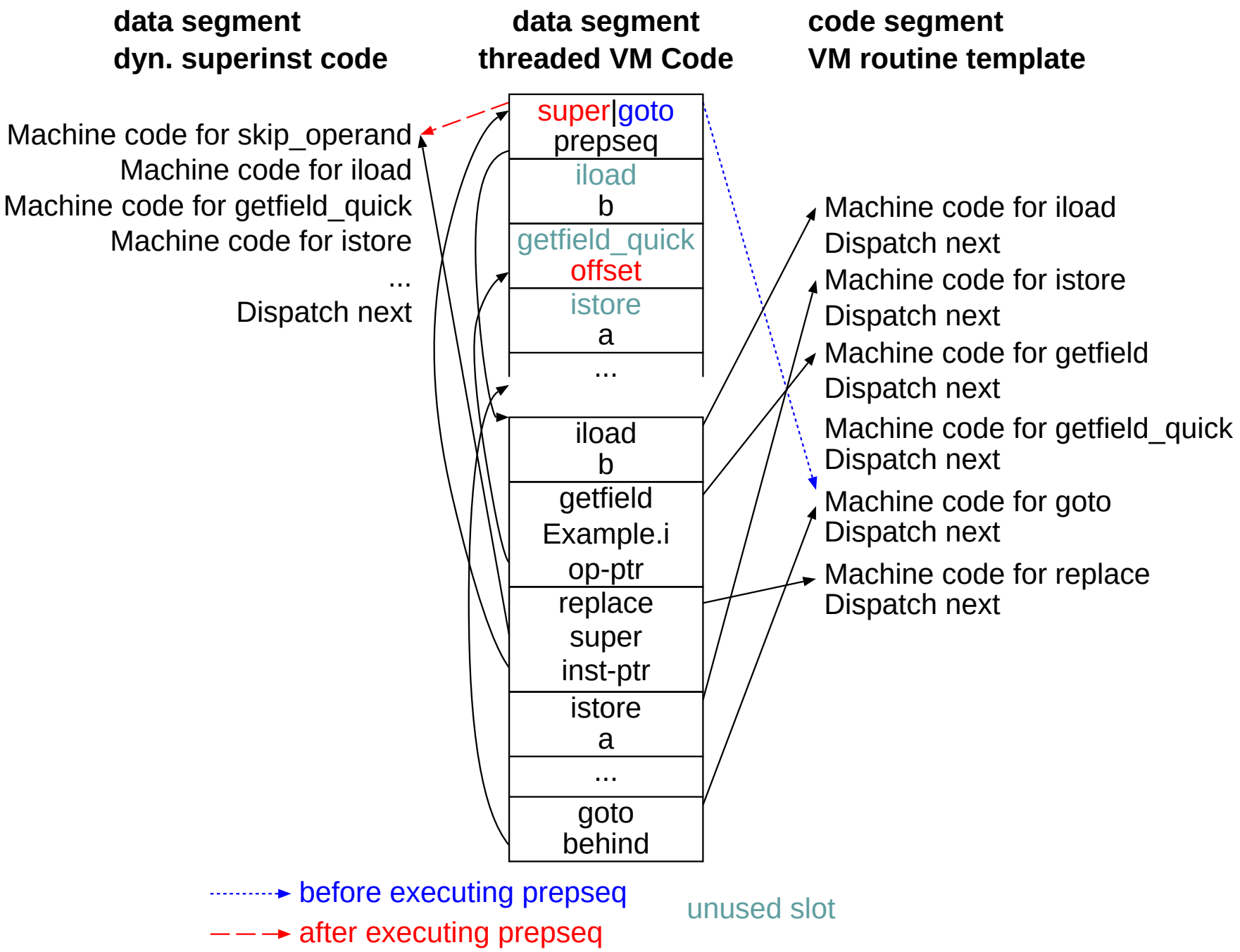
Quickable Instructions

ACONST
ARRAYCHECKCAST
CHECKCAST
GETFIELD_CELL
GETFIELD_INT
GETFIELD_LONG
GETSTATIC_CELL
GETSTATIC_INT
GETSTATIC_LONG
INSTANCEOF
INVOKEINTERFACE
INVOKESPECIAL
INVOKESTATIC
INVOKEVIRTUAL
MULTIANEWARRAY
NATIVECALL
PUTFIELD_CELL
PUTFIELD_INT
PUTFIELD_LONG
PUTSTATIC_CELL
PUTSTATIC_INT
PUTSTATIC_LONG

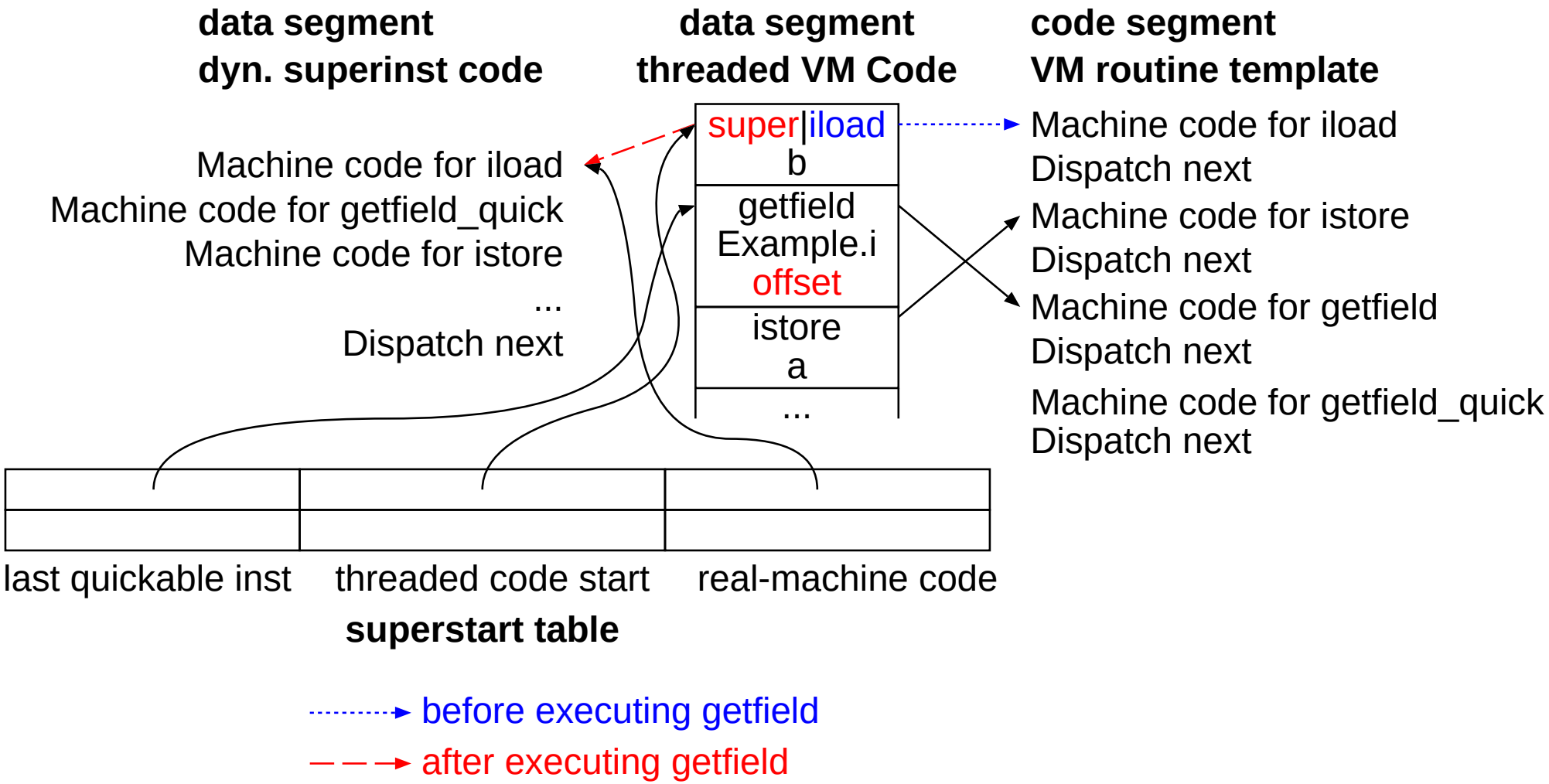
Simple Solution



SableVM's Sophisticated Solution



Cacao's Sophisticated Solution



Potential Exception-Throwing Instructions

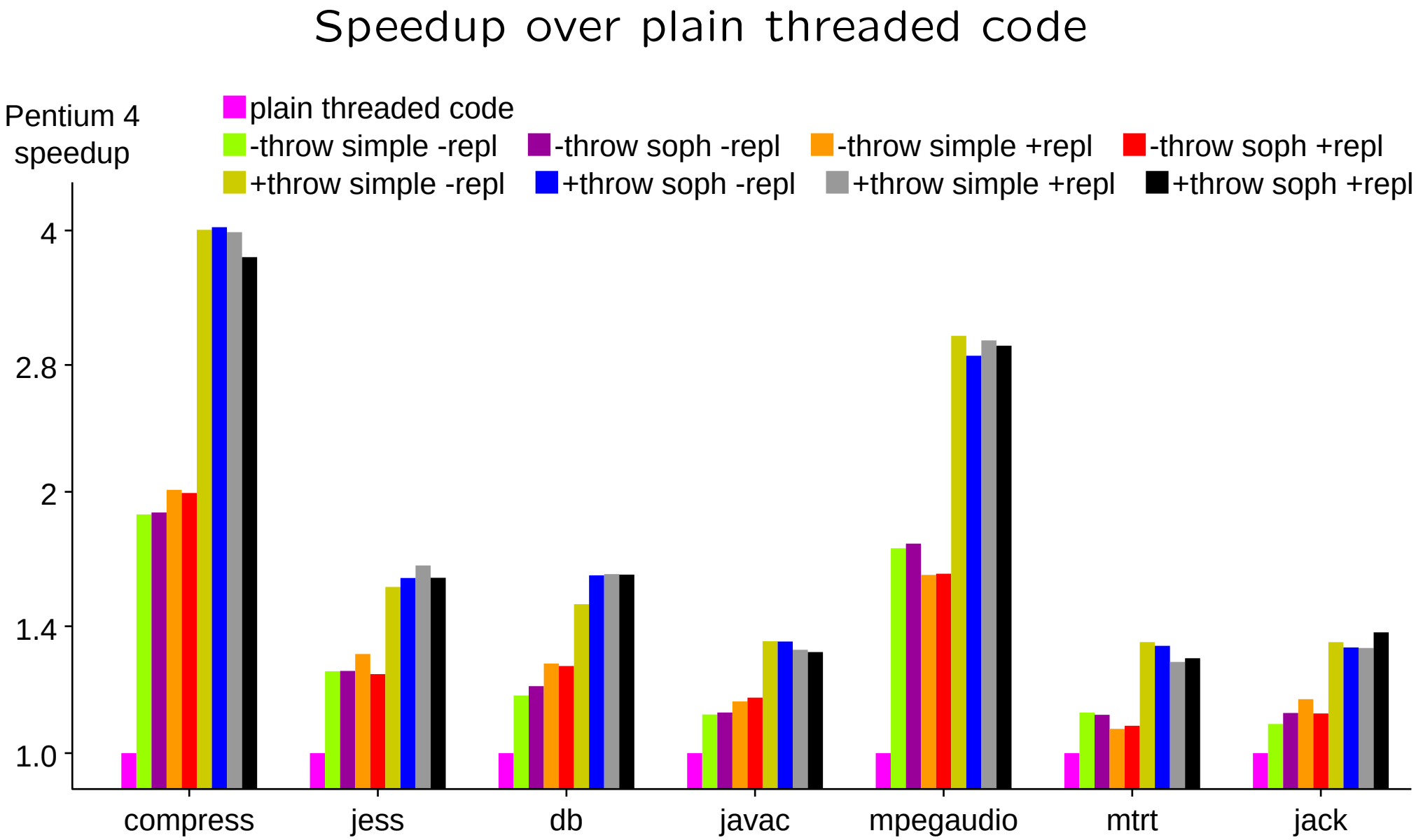
```
IALOAD          GETFIELD_CELL
LALOAD          GETFIELD_INT
AALOAD          GETFIELD_LONG
BALOAD          PUTFIELD_CELL
CALOAD          PUTFIELD_INT
SALOAD          PUTFIELD_LONG
IASTORE         INVOKEVIRTUAL
LASTORE         INVOKESPECIAL
BASTORE         INVOKEINTERFACE
CASTORE         ARRAYLENGTH
IDIV            CHECKNULL
IREM
```

Problem and Solution

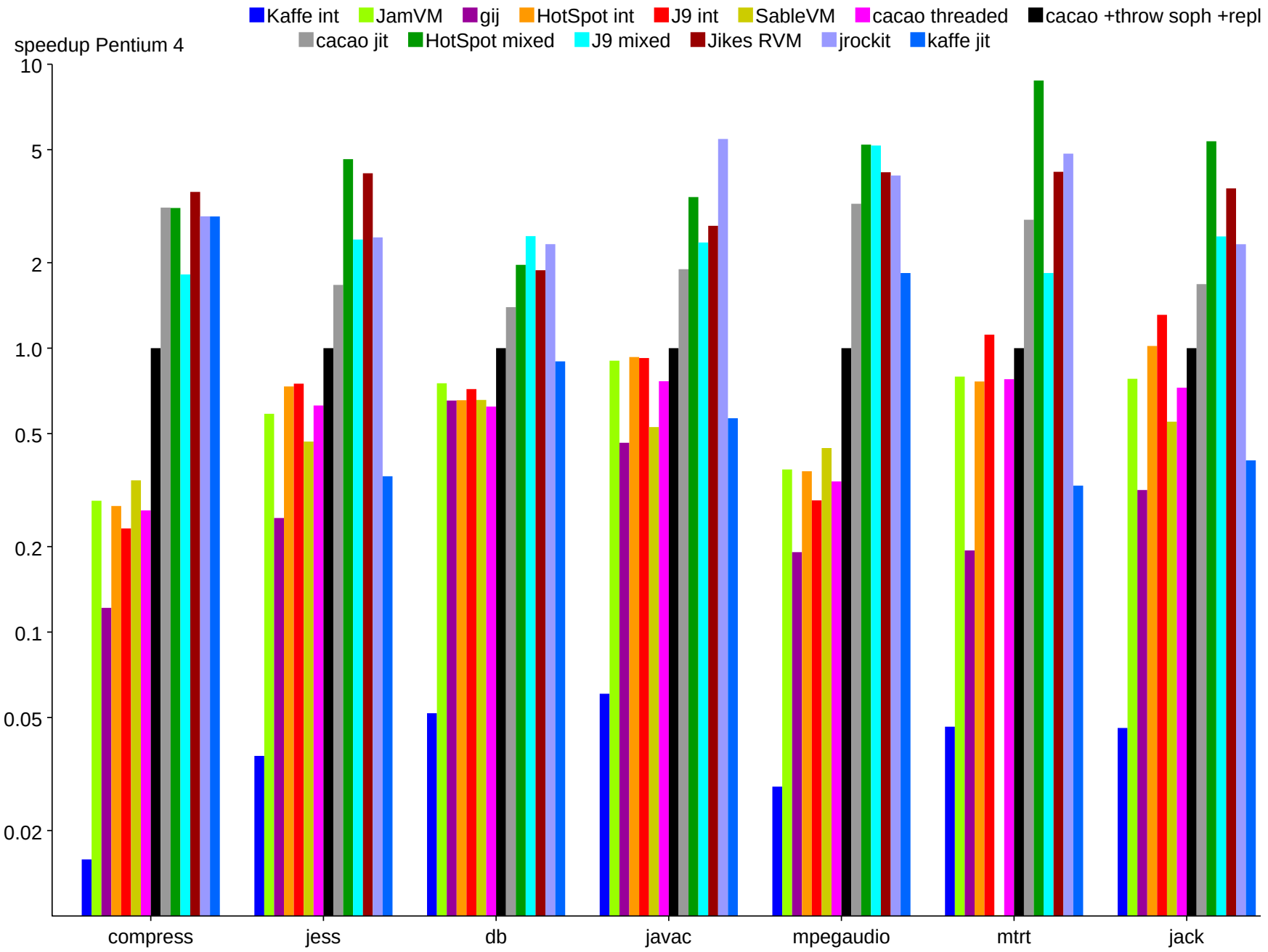
```
getfield_cell:
  mov     (%edi),%eax
  add     $0x8,%edi
  test    %ebp,%ebp
  je      throw

  add     $0x4,%edi
  mov     (%eax,%ebp,1),%ebp
  jmp     *-4(%edi)
```

```
getfield_cell:
  mov     (%edi),%eax
  add     $0x8,%edi
  test    %ebp,%ebp
  jne     no_throw
  jmp     *0x2a0(%esp)
no_throw:
  add     $0x4,%edi
  mov     (%eax,%ebp,1),%ebp
  jmp     *-4(%edi)
```


Speedup over plain threaded code
Pentium 4
speedup
plain threaded code
-throw simple -repl
-throw soph -repl
-throw simple +repl
-throw soph +repl
+throw simple -repl
+throw soph -repl
+throw simple +repl
+throw soph +repl
4
2.8
2
1.4
1.0
compress
jess
db
javac
mpegaudio
mtrt
jack

Speedup of various JVMs over Cacao with Superinstructions



Conclusion

- Superinstructions can provide big speedups
- Replication has little impact
- Quickening:
  New sophisticated solution
  but simple solution performs well in JIT setting
- Relocatability of throwing VM instructions:
  Big performance impact
  Solution: replace relative with indirect jumps