Intel's Open Runtime Platform

Ondřej Lhoták
October 10, 2001
308-762

What is ORP?

- "A platform for bytecode system research"
- Free research JVM from Intel
  - Designed for playing around with JIT compilers and garbage collection
  - Has been the basis of Intel's recent JVM papers

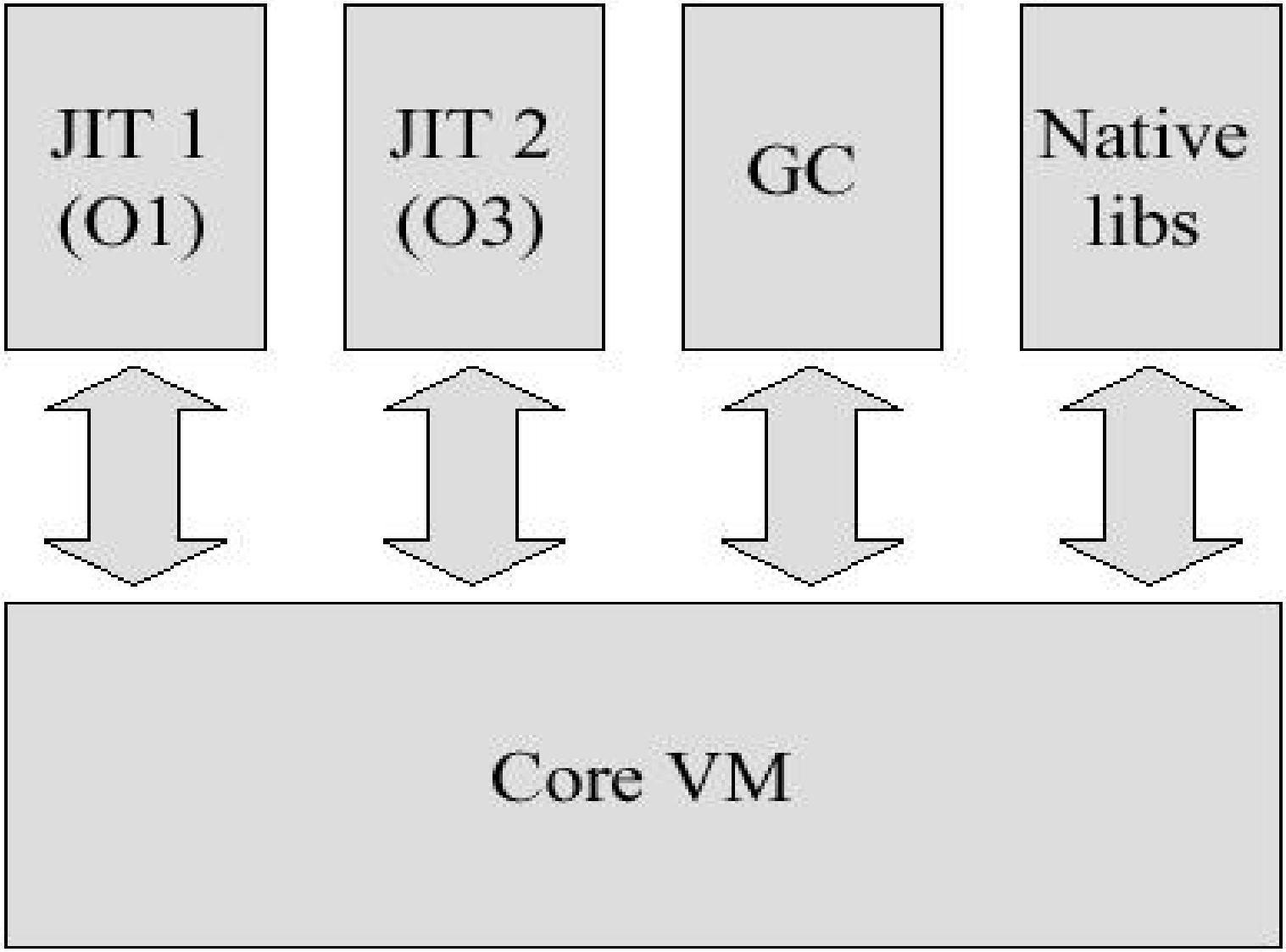
Why use ORP?

- SableVM, Kaffe are free, but do not have aggressive JIT compilers
- Jalapeño, HotSpot, IBM JDK have aggressive JIT compilers, but are not free
- ORP is best of both worlds
  - BSD license, snapshots about monthly

Requirements

- OS: Linux or Windows
- CPU: x86
  - IA-64 support being added
- Uses GNU Classpath
- Microsoft CLR (.net) support being added

Overall Structure



Compilers

- O1 JIT Compiler
  - Fast, few optimizations
  - Bytecode → native code
- O3 JIT Compiler
  - Aggressive optimizations for hot methods
  - Bytecode → IR → native code

O1 JIT Compiler

- "Lazy Code Selection"
  - Use compile-time "mimic stack" to get rid of Java operand stack
  - Use three scratch registers
  - Fold operations into complex x86 instructions (use addressing modes)
- Cheap CSE
  - Repeated bytecode sequences

O1 JIT Compiler

- Method-level register allocation
  - Simple, linear-time allocator
- Array bounds check elimination
- Exception handler moved to end of method

O3 JIT Compiler

- Local CSE
- Copy and type propagation
- Constant folding
- Dead code elimination
- Bounds check elimination
- Loop invariant hoisting
- Inlining

O3 JIT Compiler

- Floating point stack optimization
- Register allocation
- Tail recursion elimination

Optimizations Not Done

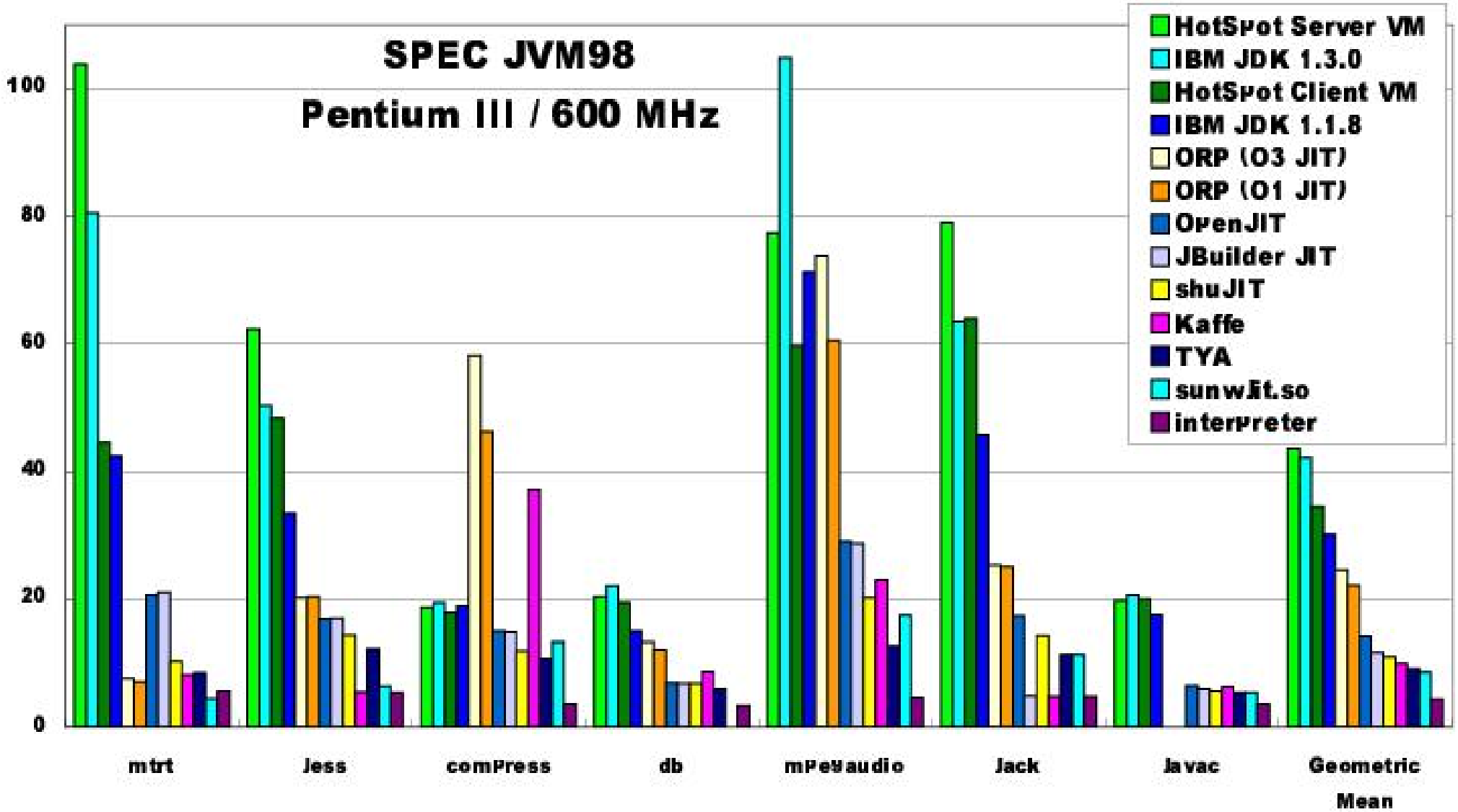
- Instruction scheduling
  - Leave it to hardware
- CSE
  - Too much register pressure

Triggering Recompilation

- Counters at method entry and back-edges
- Recompile either immediately when counter hits zero, or with separate thread that watches counters

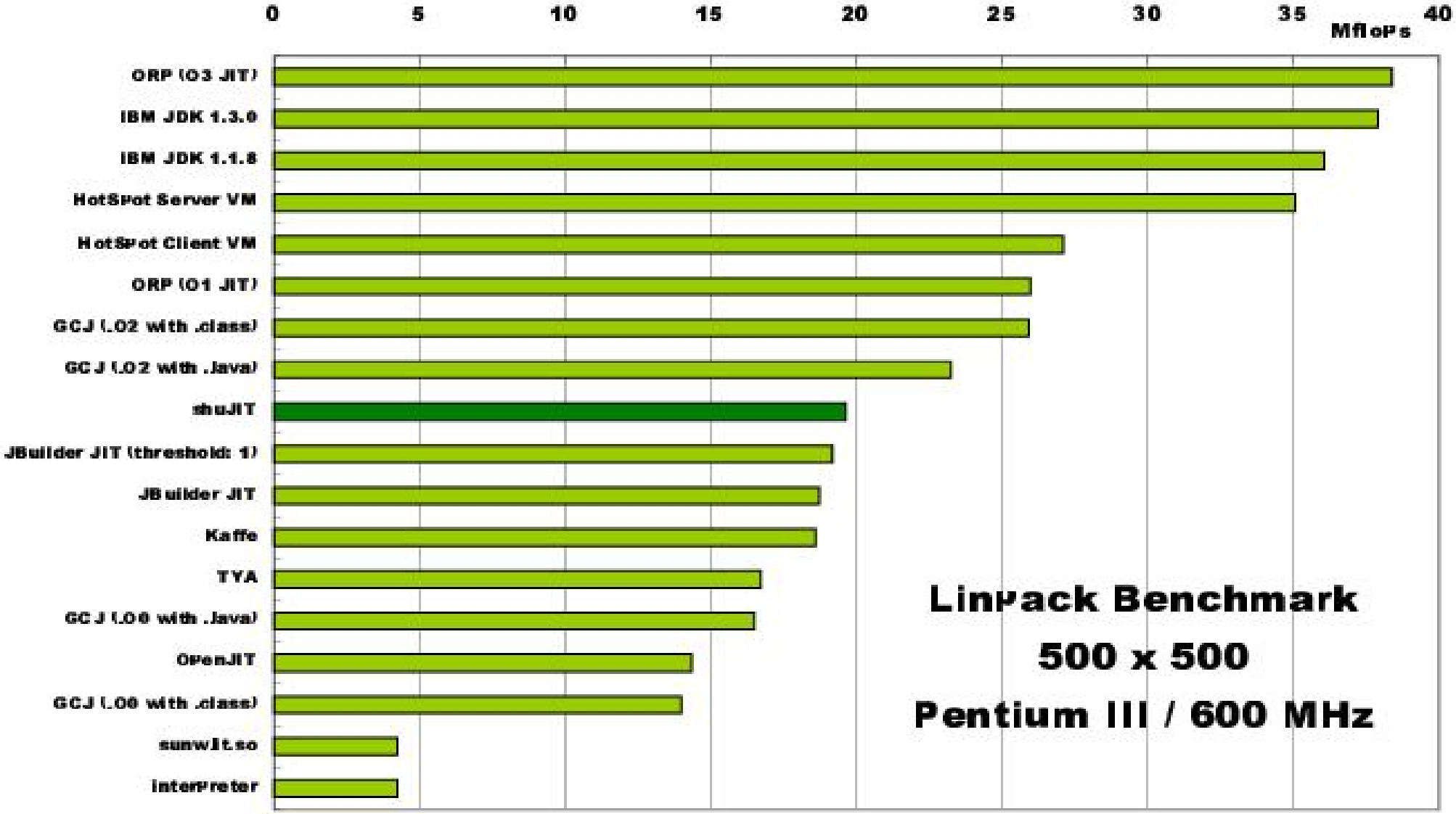
How fast is ORP?

- ORP 20001208, graph from http://www.shudo.net/jit/perf/



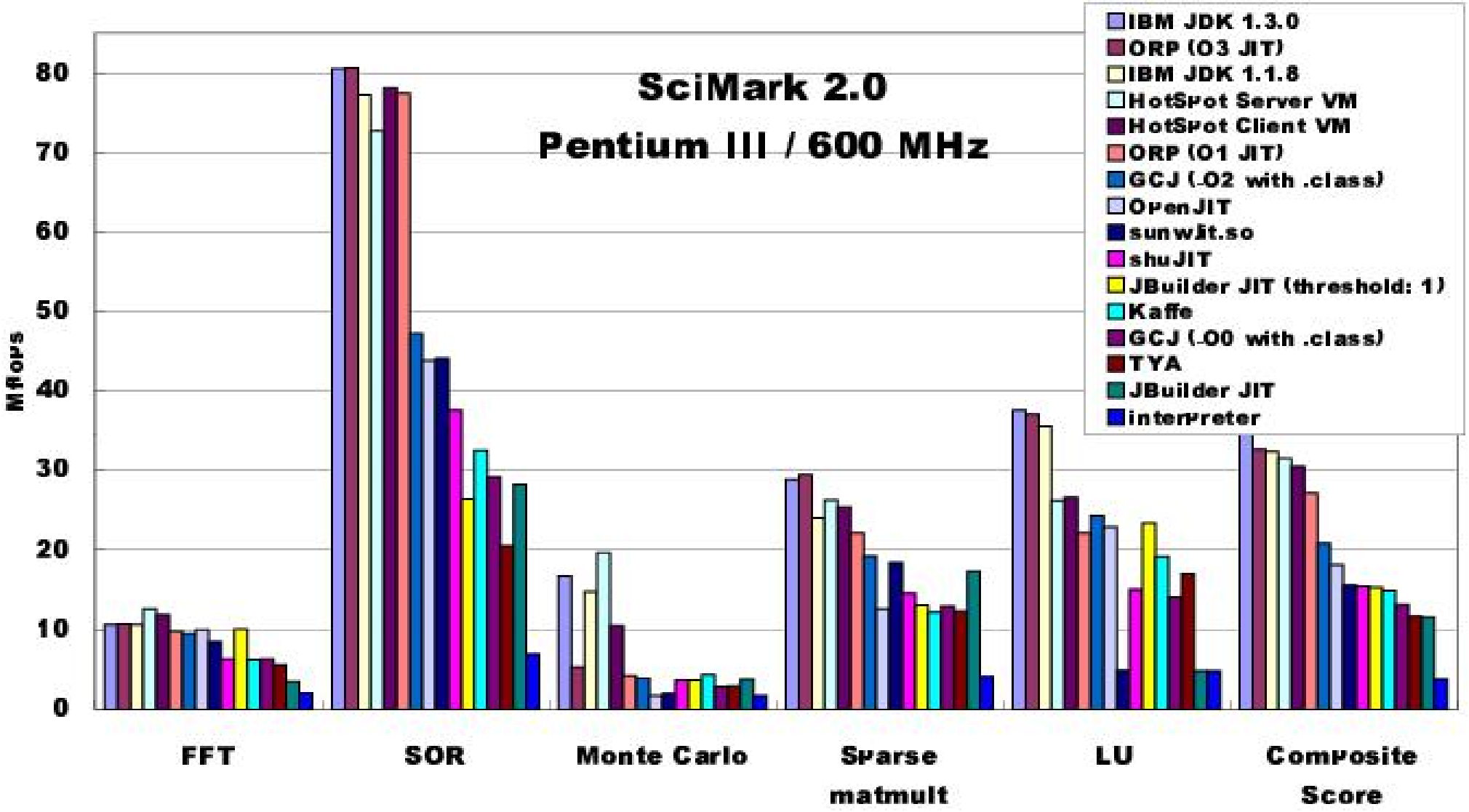
How fast is ORP?

- ORP 20001208, graph from http://www.shudo.net/jit/perf/



How fast is ORP?

- ORP 20001208, graph from http://www.shudo.net/jit/perf/



Garbage Collectors

- Includes several different algorithms and combinations of them
- "Easy to add more"...
- Almost every point is GC safe-point (see PLDI 99 paper)

Why not use ORP?

- "Research" code ⇒ quite messy, hard to compile
- Very limited documentation available with the source

References

- http://orp.sourceforge.net
- http://groups.yahoo.com/group/orp
- http://prdownloads.sourceforge.net/orp/orp-javagrande2001.pdf
- *Fast, Effective Code Generation in a Just-In-Time Java Compiler*, Adl-Tabatabai et al., PLDI 1998 (describes O1 compiler)
- *Practicing JUDO: Java Under Dynamic Optimizations*, Cierniak, Lueh and Stichnoth, PLDI 2000 (describes O3 compiler)
- *Support for Garbage Collection at Every Instruction in a Java Compiler*, Stichnoth, Lueh and Cierniak, PLDI 1999
- *Sapphire: Copying GC Without Stopping the World*, Hudson and Moss, Java Grande 2001