



# Zero and Shark

**Gary Benson**  
**Senior Software Engineer**

# What is Zero?

- Interpreter-only port of HotSpot
- Uses (almost) no assembler
- Can trivially be built on any (Linux) system
- Known to work on
  - Alpha
  - ARM
  - IA-64
  - MIPS
  - PowerPC
  - x86-64
  - zSeries

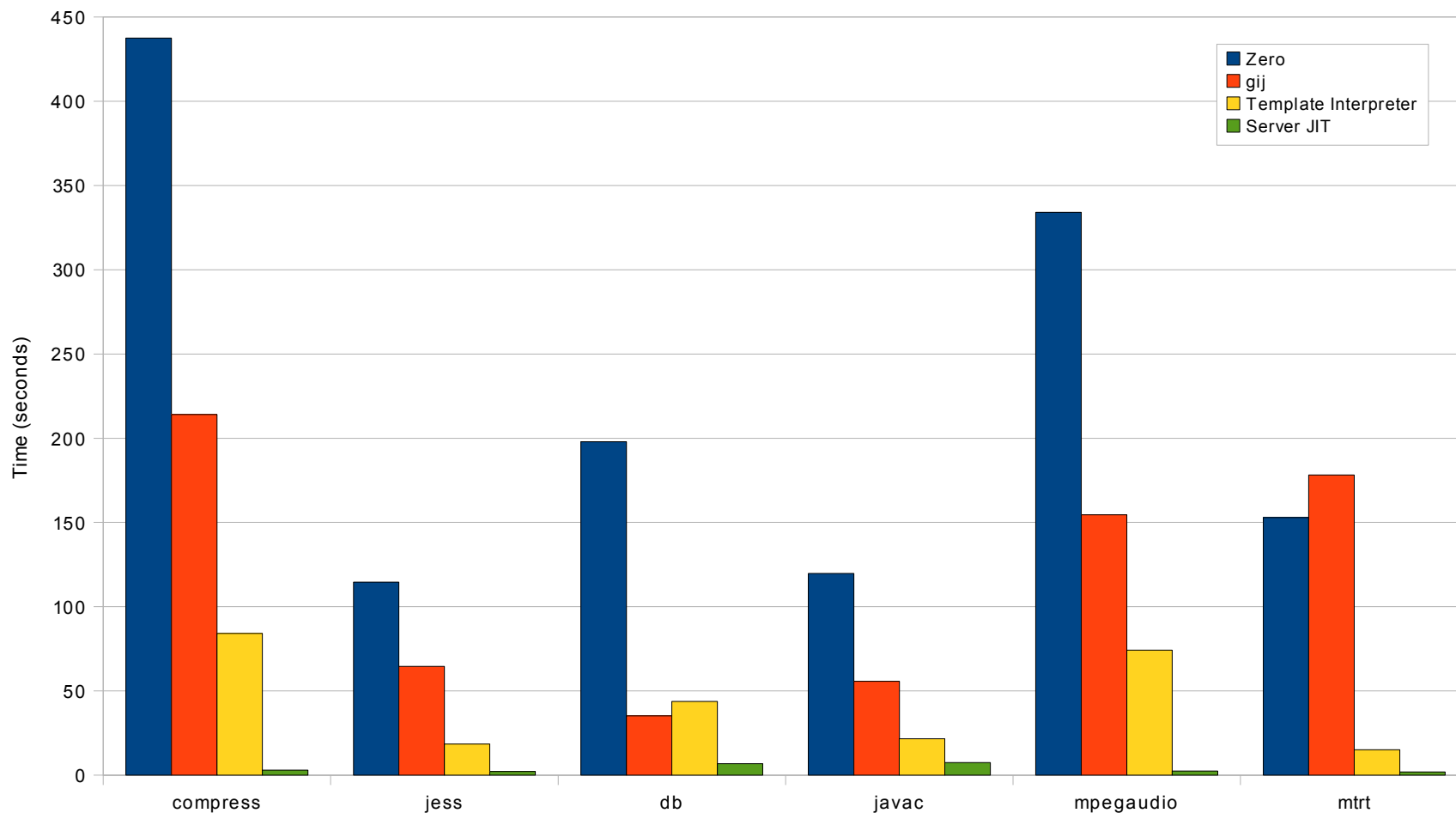
# How does it work?

- Template interpreter:
  - Bytecodes implemented in assembly language
- C++ interpreter:
  - Bytecodes implemented in C++
  - Thin assembly layer

# How to remove assembly?

- Atomic operations and memory access ordering
  - Use GCC atomic intrinsics
- Native function calls (for JNI)
  - Use `libffi`
- Interfacing with stack
  - Manage our own

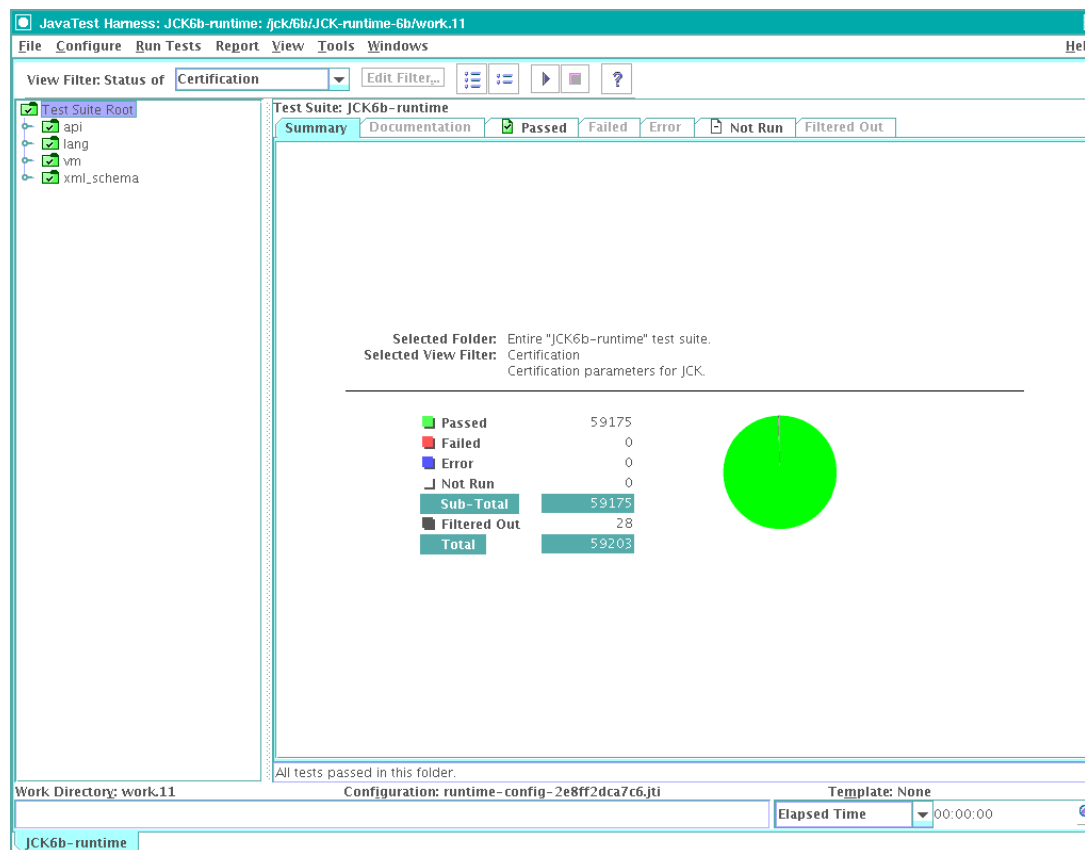
# Performance



These times were generated using the SPECjvm98 benchmark, but they were not produced in compliance with the SPECjvm98 run rules and therefore are not comparable with a SPECjvm98 metric.

# Compatibility

The latest OpenJDK packages included in Fedora 10 for 32- and 64-bit PowerPC have passed the Java SE 6 TCK and are compatible with the Java SE 6 platform.



JavaTest Harness: JCK6b-runtime: /jck/6b/JCK-runtime-6b/work.11

File Configure Run Tests Report View Tools Windows Help

View Filter: Status of Certification Edit Filter...

Test Suite: JCK6b-runtime

Summary Documentation Passed Failed Error Not Run Filtered Out

Selected Folder: Entire "JCK6b-runtime" test suite.  
Selected View Filter: Certification  
Certification parameters for JCK.

Passed	59175
Failed	0
Error	0
Not Run	0
Sub-Total	59175
Filtered Out	28
Total	59203

All tests passed in this folder.

Work Directory: work.11 Configuration: runtime-config-2e8ff2dca7c6.jti Template: None Elapsed Time 00:00:00

JCK6b-runtime

# What is Shark?

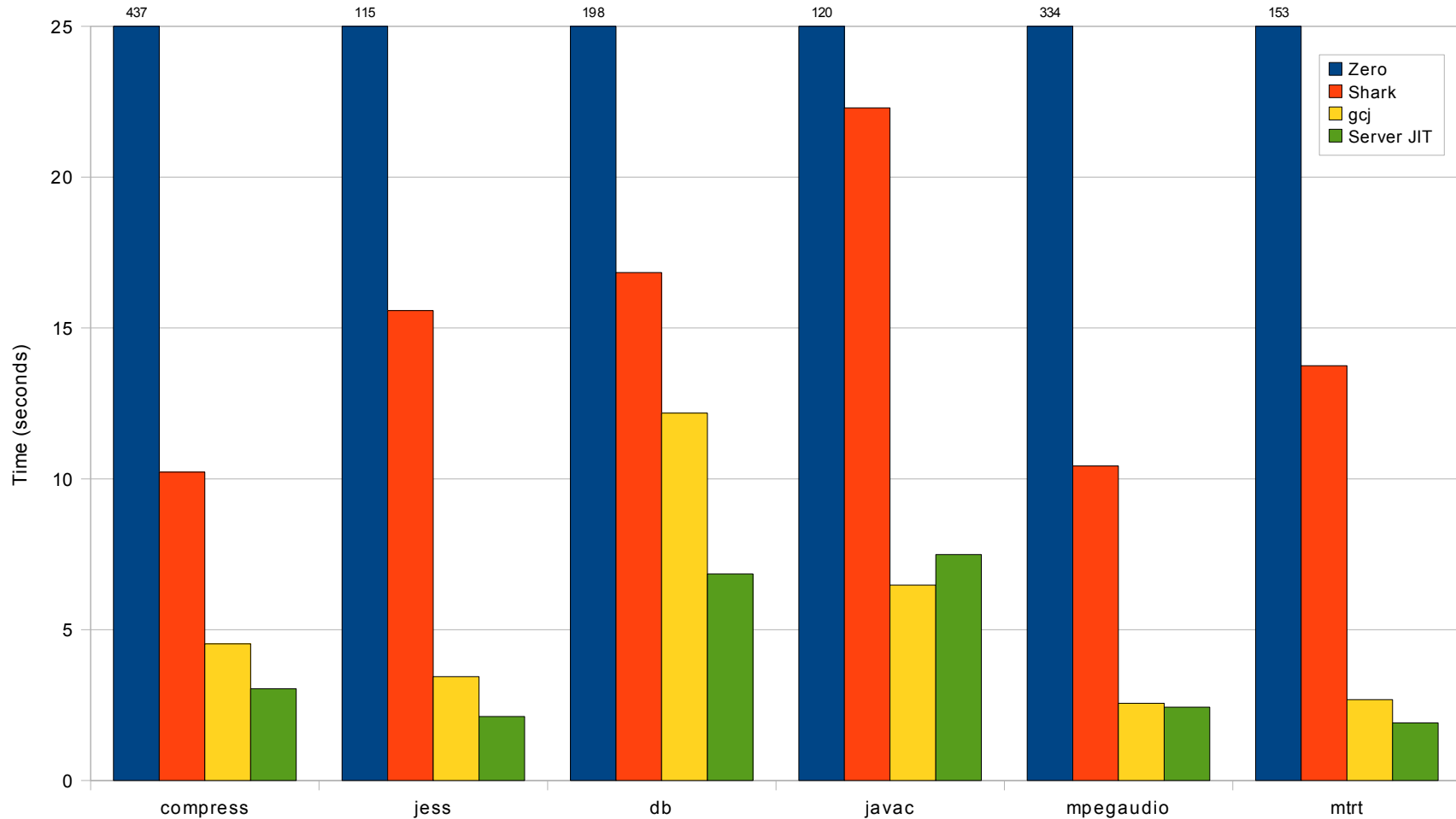
- Shark is a just-in-time (JIT) compiler for Zero
- Uses LLVM compiler infrastructure
- Available on Zero architectures with an LLVM JIT:
  - Alpha
  - ARM
  - PowerPC
  - x86-64

# How does it work?

- Uses the same interface as HotSpot's client and server compilers
  - Same options: `-Xcomp`, `-Xmixed`, `-XX:CompileThreshold...`
  - Tiered compilation should be possible
- Three passes:
  - Server compiler's typeflow pass produces annotated bytecode
  - Shark converts annotated bytecode to LLVM IR
  - LLVM converts LLVM IR into native code



# Performance



These times were generated using the SPECjvm98 benchmark, but they were not produced in compliance with the SPECjvm98 run rules and therefore are not comparable with a SPECjvm98 metric.

# Future work

- Stabilization
- Optimization
  - Profiling
  - Method and field lookups
  - LLVM's optimizers
  - Warmup

# Questions

- *<http://openjdk.java.net/projects/zero/>*
- *<http://icedtea.classpath.org/wiki/ZeroSharkFaq>*
- *<http://gbenson.net/>*
  
- Gary Benson
- *[gbenson@redhat.com](mailto:gbenson@redhat.com)*