ORACLE

Project Panama: Native Interconnect

John Rose

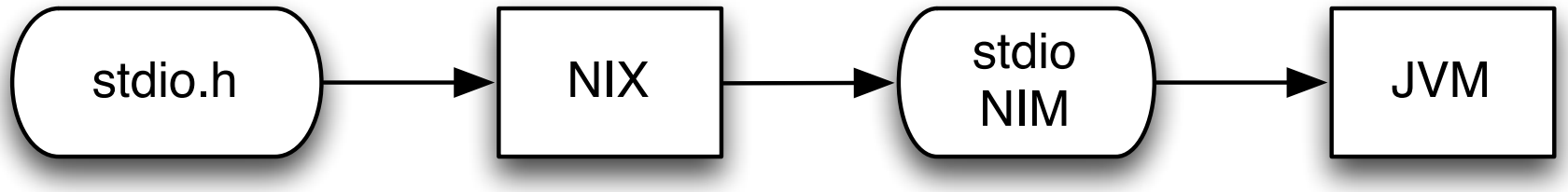
Big Goals

- Let Java coders use libraries as easily as C coders
- Simple incorporation of C APIs
- Provision for hand adjustment of safety invariants
- High performance

Q&A:

- For structured data we don't own (hardware, etc.)?
  - Yes. Foreign data definitions, foreign allocation rules
  - Also "foreign" hardware ops — e.g., on 128-bit vectors
- GC interactions? Minimal. JVM still owns its objects.
  - Non goal: Java object with fixed address
  - (java to C obj)

API Capture



(Note: This pipeline does not exist yet.)

API Capture

- native interface extractor
  - C(++) header file compiler
  - SWIG is old tech
  - libclang is more promising: parse event generator (SAX-like)
- Important: inter-language “hands-free” mappings
  - no hand-written adapters or stubs, in either C or Java
- Needed: Format for “native interconnect metadata”
  - (class or JAR file or JDK 9 “module”, cf. .NET ass’y)
- Similar moves for other API sources
  - Fortran (BLAS), Python, Ruby, Scala, ...

Native data access

- Should use native metadata, not hand-rolled.
- As efficient as “Unsafe”

- also, should be interface-based
  - This allows accesses to be be “woven” with special rules
  - Allows gdb-style remote debugging, if needed (SA++ !)

Native function access

- Same points as for native data access
- No hand-rolled adapters (use JNR)
- Compiler integration for performance
- Respect JVM safety invariants

- Safety first. But allow for a “hard hat only” area.

Native interconnect metadata API

- Describe foreign data and functions to Java
- Usable by interpreter, compiler, linker, etc.
- Works in progress: JNR, David Chase's data access

- Open question: Is this built on JAR? XML? TBD?

Requirement: Compilability

- Native data access readily reduces to “unsafe.getInt”, etc.
- Native function calls readily reduce to FFI primitives.
  - Some “wrapping” may still be needed, but minimized.

- Q&A: What does this really mean?
  - Clear separation of metadata from payload data.
  - Metadata available at compile (JIT) time for optimization.
  - Explicit JNI brackets will allow JNI-coarsening optimizations!
- Trustability is always important (trust levels are OK)

Discussions outside OpenJDK

- JNR examples
  - https://github.com/jnr/jnr-ffi-examples/tree/master/getpid/.../src/main/java/getpid/Getpid.java
- jvm-ffi Google group
  - http://groups.google.com/forum/#!topic/jvm-ffi/8drjHY7yX0I
- JavaOne '13 talk by Ryan Sciampacone & Tom Enebo
  - https://oracleus.activeevents.com/2013/connect/sessionDetail.ww?SESSION_ID=4767
- IBM Packed Objects
  - http://www.slideshare.net/mmitran/ibm-java-packed-objects-mmit-20121120
- Old Scheme/C++ work showing "bilingual" code
  - https://blogs.oracle.com/jrose/resource/esh/RoseMullerIX.html

Postscript: Why a new project?

- We need a safe container for developed IP
  - ...this is the function of the OpenJDK Contributor Agreement
- Now is the time to do this. By end of year.
  - Mark R.: “It’s past time for it.”
- What’s “Panama”?
  - (a) a place name, so not taboo
  - (b) a canal, connecting oceans
  - (c) an isthmus, connecting continents