ORACLE®

Intrinsic Methods in HotSpot VM

Krystal Mo
Member of Technical Staff
HotSpot JVM Compiler Team

Agenda

- Background
- Intrinsic Methods in HotSpot VM
- Intrinsic Methods added in TaobaoJDK
- Experiment: Implement Your Own Intrinsic
- Further Experiment

Agenda

- Background
- Intrinsic Methods in HotSpot VM
- Intrinsic Methods added in TaobaoJDK
- Experiment: Implement Your Own Intrinsic
- Further Experiment

ORACLE

What is a Intrinsic Method?

- In compiler theory, an intrinsic function is a function available for use in a given programming language whose implementation is handled specially by the compiler. Typically, it substitutes a sequence of automatically generated instructions for the original function call, similar to an inline function. Unlike an inline function though, the compiler has an intimate knowledge of the intrinsic function and can therefore better integrate it and optimize it for the situation. This is also called builtin function in many languages.
- (via Wikipedia)

Intrinsic Functions in Native Compilers

Examples

- Microsoft Visual C/C++ Compiler
  - __debugbreak()
    - inserts an “int 3” instruction for breaking into debugger
- GCC
  - __builtin_ia32_pause()
    - inserts a “pause” instruction and necessary compiler barriers to prevent the instruction from floating around

Intrinsic Methods in JVMs

- Specific to JVM implementations
  - can’t assume a method is intrinsic across JVMs in general
  - mostly implemented in JVM compilers

Intrinsic Methods in JVMs

- Compatible
  - appear to be no different from normal Java methods on Java source level; all special handling is done within JVM
  - can fallback to normal (non-intrinsic) version on JVMs that don't intrinsify them

Intrinsic Methods in JVMs

- Reliable
  - are good “anchor” points for JVM optimizations for common code patterns
  - e.g. java.lang.System.arraycopy()
    - easier to recognize than matching an explicit array-copying loop

Intrinsic Methods in JVMs

- Extend Java semantics
  - the same way native methods do, but usually more performant
    - reliably inlined
    - no JNI overhead
  - e.g. sun.misc.Unsafe.compareAndSwapInt() on x86/AMD64
    - no Java bytecode can express its semantics
    - without intrinsics: implemented in native via JNI (Unsafe_CompareAndSwapInt())
    - with intrinsics: a direct cmpxchg instruction, inlined to caller

Agenda

- Background
- Intrinsic Methods in HotSpot VM
- Intrinsic Methods added in TaobaoJDK
- Experiment: Implement Your Own Intrinsic
- Further Experiment

ORACLE

Intrinsic Methods in HotSpot VM

What to intrinsify?

- Commonly used public APIs in JDK core library
  - to speed up common code patterns
  - so use JDK core library methods whenever you can
    - they may be better optimized!
- Special internal methods
  - to implement special semantics
  - the “secret sauce” to allow more parts of the Java core library be implemented in Java
  - may be exposed indirectly via ease-of-use APIs, e.g. j.u.c.atomic.*

ORACLE

Intrinsic Methods in HotSpot VM

How to intrinsify?

- Declare intrinsic methods in vmSymbols
- Implement intrinsic methods in
  - Interpreter
    - currently only implements intrinsics for
      - some math functions
      - a few MethodHandle internals for bootstrapping
  - Client Compiler (C1)
  - Server Compiler (C2)

Intrinsic Methods in HotSpot VM

Example: java.lang.System.currentTimeMillis() on Linux

- Interpreter
  - not intrinsified
  - java.lang.System.currentTimeMillis() (Java / core library)
    - (-> through normal JNI call path)
    - JVM_CurrentTimeMillis() (C++ / HotSpot VM)
    - os::javaTimeMillis() (C++ / HotSpot VM)
    - gettimeofday() (C / Linux)

Intrinsic Methods in HotSpot VM

Example: java.lang.System.currentTimeMillis() on Linux

- C1
  - intrinsified
  - inlined to caller as a direct call to os::javaTimeMillis()
- C2
  - same as in C1

- (Intrinsification eliminates JNI overhead in compiled code)

Intrinsic Methods in HotSpot VM

Example: sun.misc.Unsafe.compareAndSwapInt() on x86/AMD64

- Interpreter
  - not intrinsified
  - sun.misc.Unsafe.compareAndSwapInt() (Java / core library)
    - (-> through normal JNI call path)
    - Unsafe_CompareAndSwapInt() (C++ / HotSpot VM)
    - Atomic::cmpxchg() (C++ inline asm / HotSpot VM)
    - lock cmpxchgl

Intrinsic Methods in HotSpot VM

Example: sun.misc.Unsafe.compareAndSwapInt() on x86/AMD64

- C1
  - intrinsified
  - inlined into caller as a plain “lock cmpxchgl” instruction
- C2
  - same as in C1

- (Intrinsification easily leverages special hardware instructions)

Intrinsic Methods in HotSpot VM

Example: java.lang.Math.log() on x86/AMD64

- Interpreter
  - intrinsified
  - java.lang.Math.log() (Java / core library)
    - (-> through special interpreter method entry)
    - “flog” x87 instruction

Intrinsic Methods in HotSpot VM

Example: java.lang.Math.log() on x86/AMD64

- C1 and C2
  - intrinsified
  - inlined into caller as “flog” x87 instruction

- (Intrinsification ignores Java-level implementation)
  - java.lang.Math.log() is implemented in pure Java in JDK core library
  - HotSpot VM ignores that implementation on platforms where hardware floating point instructions are available

Intrinsic Methods in HotSpot VM

Other intrinsic methods of interest (on AMD64)

- java.lang.Thread.currentThread()
  - mov reg, [r15 + java_thread_offset]
- java.lang.String.indexOf()/compareTo()/equals()
  - use STTNI instructions in SSE4.2
- com.sun.crypto.provider.AESCrypt.encryptBlock()/decryptBlock()
  - use AES instructions

Intrinsic Methods in HotSpot VM

Find out if you’re using intrinsics in compiled code

- -XX:+PrintCompilation -XX:+PrintInlining
  - (prepend -XX:+UnlockDiagnosticVMOptions when running on product VM)

Intrinsic Methods in HotSpot VM

Find out if you’re using intrinsics in compiled code: example

```
public class Foo {
  private static Object bar() {
    return Thread.currentThread();
  }

  public static void main(String[] args) {
    while (true) { bar(); }
  }
}
```

Intrinsic Methods in HotSpot VM

Find out if you’re using intrinsics in compiled code: example

```
$ java -XX:+UnlockDiagnosticVMOptions -XX:CompileCommand='exclude,Foo,main' \
  -XX:+PrintCompilation -XX:+PrintInlining Foo
CompilerOracle: exclude Foo.main
     50    1     n       java.lang.Thread::currentThread (0 bytes)   (static)
### Excluding compile: static Foo::main
     50    2             Foo::bar (4 bytes)
                            @ 0   java.lang.Thread::currentThread (0 bytes)   (intrinsic)
```

Intrinsic Methods in HotSpot VM

Find out what intrinsic compiled into

- -XX:+PrintAssembly
  - (prepend -XX:+UnlockDiagnosticVMOptions when running on product VM)
  - requires hsdis disassembler plugin

Intrinsic Methods in HotSpot VM

Find out what intrinsic compiled into: example

```
$ java -XX:+UnlockDiagnosticVMOptions -XX:CompileCommand='exclude,Foo,main' \
  -XX:+PrintAssembly Foo
...
  # {method} 'bar' '()Ljava/lang/Object;' in 'Foo'
  #           [sp+0x20]  (sp of caller)
  0x00007f91cd061bc0: sub    $0x18,%rsp
  0x00007f91cd061bc7: mov    %rbp,0x10(%rsp)    ;*synchronization entry
                                                ; - Foo::bar@-1 (line 3)
  0x00007f91cd061bcc: mov    0x1b0(%r15),%rax   ;*invokestatic currentThread
                                                ; - Foo::bar@0 (line 3)
  0x00007f91cd061bd3: add    $0x10,%rsp
  0x00007f91cd061bd7: pop    %rbp
  0x00007f91cd061bd8: test   %eax,0xb7ed422(%rip)        # 0x00007f91d884f000
                                                ;   {poll_return}
  0x00007f91cd061bde: retq
  0x00007f91cd061bdf: hlt
```

Agenda

- Background
- Intrinsic Methods in HotSpot VM
- Intrinsic Methods added in TaobaoJDK
- Experiment: Implement Your Own Intrinsic
- Further Experiment

Instrinsic Methods Added in TaobaoJDK

Why make your own custom intrinsic method?

- Make better use of new instructions available on new hardware
  - In Taobao’s case, new instructions on Westmere/Sandy Bridge
- Eliminate JNI overhead when having to invoke hot native methods
  - Instead of calling a native method through normal JNI, implement it as an intrinsic method

(this section is material from Taobao;
TaobaoJDK is a custom version of OpenJDK from Taobao)

Instrinsic Methods Added in TaobaoJDK

A few examples

- TCrc32.xxx
  - crc32c instruction in SSE4.2
- Unsafe.byteArrayCompare()
  - byte array comparison via packed compare instructions
- Unsafe.pause()
  - insert “pause” instruction before backedge of spinlock-like loop
- …

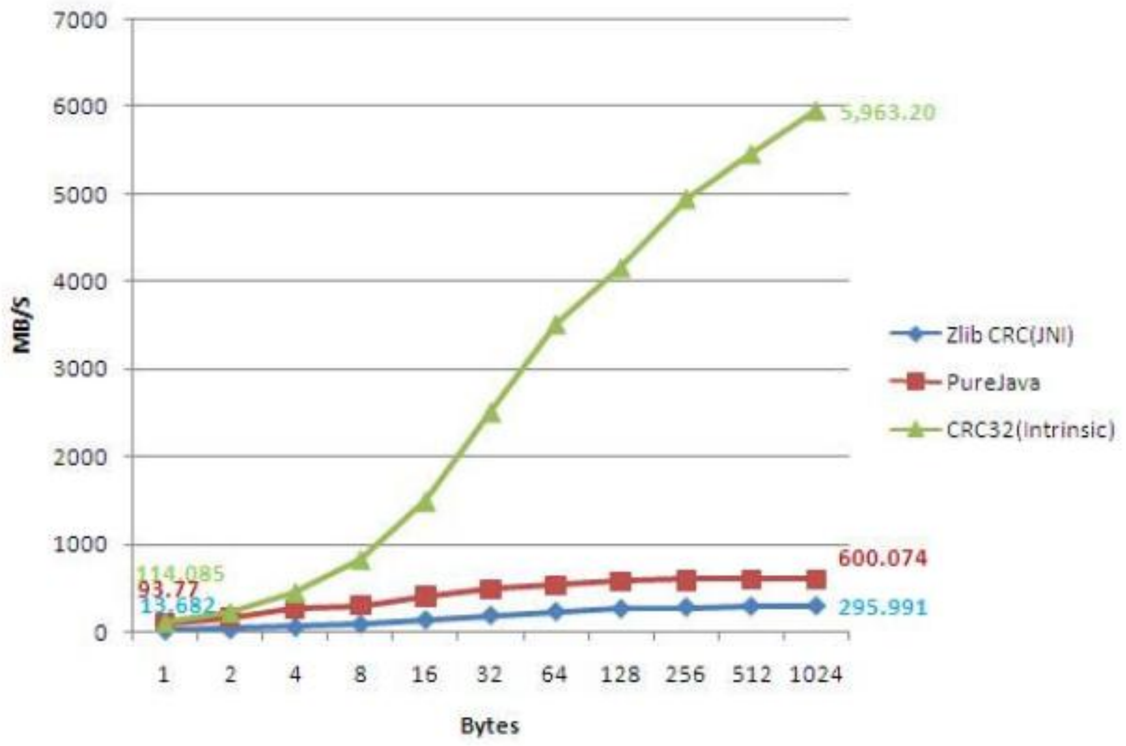
Instrinsic Methods Added in TaobaoJDK

crc32c

- Used in Hadoop
  - throughput in TestDFSIO benchmark increased by 40%-180%

JVM Instrinsics Added in TaobaoJDK

crc32c



Agenda

- Background
- Intrinsic Methods in HotSpot VM
- Intrinsic Methods added in TaobaoJDK
- Experiment: Implement Your Own Intrinsic
- Further Experiment

Implementing an Intrinsic in C1

Example

- Implement java.lang.Class.isInstance() intrinsic in C1
  - https://gist.github.com/rednaxelafx/2830194

- Note: starting point at GraphBuilder::try_inline_intrinsics()

Implementing an Intrinsic in C2

Example

- Implement a simple intrinsic demo in C2
  - https://gist.github.com/rednaxelafx/1986224
- Implement a prototype for java.lang.Math.addExact() intrinsic in C2
  - https://gist.github.com/rednaxelafx/db03ab15ef8b76246b84

- Note: starting point at LibraryCallKit::try_to_inline()

Agenda

- Background
- Intrinsic Methods in HotSpot VM
- Intrinsic Methods added in TaobaoJDK
- Experiment: Implement Your Own Intrinsic
- Further Experiment

Further Experiment

Introducing Graal

- Experiment with implementing your own language-/library-specific intrinsic in HotSpot VM, but don’t want to write C++ or build HotSpot VM?
  - Graal (from Oracle Labs) is the answer to your call!
    - bytecode-to-native compiler implemented in Java, can be plugged into HotSpot VM
    - OpenJDK project page
    - Graal introduction (from JVM Language Summit 2011)

Q & A

ORACLE®