VM Rumble

JamVM

Dr. Robert Lougher

Changes In JamVM 1.5.2

Big code clean-up

Performance

- Inlining Interpreter Improvements

Memory Usage

- New Object layout

Reliability

- Race-conditions

1.5.2 is faster, uses less-memory and is more reliable than 1.5.1

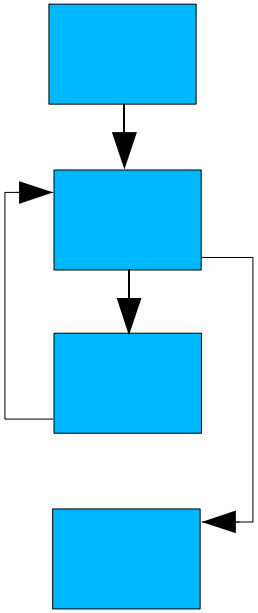
Inlining Interpreter (Code-copying JIT)

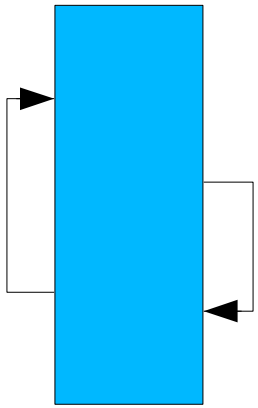
Inlining across basic block boundaries

Improves code for fall-through control flow

multiple entry and exit points

symbolic resolution





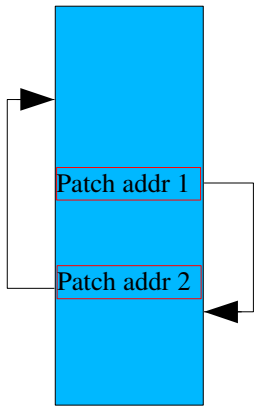
Inlining Interpreter (Part Two)

Branches in inlined sequences now patched with native jumps

- **Control-flow within a single inlined sequence**
- **Control-flow between non-shared sequences**

  e.g. replace

```
movq (%rbp), %rcx      →     jmp 0x...
jmp  *%rcx
```



Inlining Interpreter (Part Three)

Simple basic block profiling

- Execution count and threshold
- History allows bigger basic block ranges to be inlined
- Substantially reduces size of code-cache

Performance improvement

- PowerPC : up to 50% (typical 20%)
- i386/AMD64 : up to 25% (typical 15%)
- ARM : up to 50%

Object Layout

Previous Object Layout

- **On 64-bit machines, object layout was wasteful**
  - Padding on fields < 64-bits
- **On 32-bit machines 64-bit quantities not guaranteed to be 64 bit aligned**

New Layout

- **On 64-bit machines uses approx 10 – 15% less heap**
- **Maintains 64-bit alignment**
  - if possible without adding padding

Alignment fixes

- **Ensure operand stack is 64-bit aligned**
- **Ensure static variable data is 64-bit aligned**
- **Ensure long/double array data 64-bit aligned**

Reflection

Now uses new VM interface (GNU Classpath 0.98)

- **VMConstructor**
- **VMMethod**
- **VMField**

Improved Performance

- **Offset extra cost of VM interface**
  - invocation
  - field access

Miscellaneous Changes

Solaris/OpenSolaris Port

- **i386 and AMD64**

sun.misc.Unsafe

- **thread park and unpark implemented**
  - replaces inefficient “empty” implementation
- **Object methods fixed on 64-bit platforms**
  - e.g. compareAndSwapObject

JNI

- **Re-implemented global references handling**
  - memory leak
  - improved performance
- **Bug fixes**

Misc Changes (Part Two)

Garbage Collection

- “Asynchronous GC” no longer enabled by default

Exception Handling

- Fix frame skipping when filling in stack trace
- Main thread now uses the thread's `unCaughtExceptionHandler`

Threading

- Use thread local storage if available (__thread)

Misc

- Compatibility command-line options
- Minimum heap increased to 16MB
- GCC 4.3 strict-aliasing fixes

The Future

Garbage Collector

- **Already implements Soft/Weak/Phantom references, Compaction, Class unloading, “JIT” code unloading ...**
- **Improvements**
  - Generational
  - Fully-accurate
  - Faster

OpenJDK

LLVM ???

IR optimisation ???