RTALK - SMALLTALK ON THE JVM
Roos Instruments, Inc.

HARDWARE AND SOFTWARE FOR IC TEST
Smalltalk Basics

- Message Based
- Everything is an Object
- Simple syntax
- Fast
  - ODA (On Demand Assembler)
- Excellent FFI
- Supports extreme agile programming
testHanoiMove: numberOfDisks from: source to: dest temp: temp
"<modified:sys=GAKRE8CA,time=06/28/11 at 04:39:33 pm>  "
   numberOfDisks == 1 ifTrue: [^self].
   self
   testHanoiMove: numberOfDisks - 1 from: source to: temp temp: dest;
   testHanoiMove: numberOfDisks - 1 from: temp to: dest temp: source
SMALLTALK AT RI

- Since 1989
- Efficiency - 3 to 9x Java
- Low errors - 1/3 Java
- 500K lines of code vs 2.5M
- But we now have Obsolete Platforms
  - OS/2
  - Digitalk
  - Heisenbug
Porting options

- Another Smalltalk
  - Same end game

- Another language
  - Will have to test 2.5M lines of code

- Port at the VM level
  - Difficult until JSR 292
Existing Architecture

APPLICATION CODE

COMPILER

METHODS

VM + OBJECT MEMORY

PRIMITIVES

FFI

OS/2

UI

FILES

NETWORK
Process

- Analyze Existing code usage
- Build Tools
  - Byte Code inspectors
  - Reverse compilers
- Define a translation interface
- Port as is (don't try to improve it)
- Convert Op sys calls to async messages
New Architecture

APPLICATION CODE

COMPILER

METHODS

VM + OBJECT MEMORY

PRIMITIVES

FFI?

Java

Async socket Protocol

UI Browser

FILES Guru

NETWORK Proxies
Architecture Mismatches

- Stack + 2 registers (eax edx)
- Stack space == variable space
- Object Memory (ints stored in pointer)
- Type artifacts
Stack Var Structure

Normal Stack

<table>
<thead>
<tr>
<th>var1...</th>
<th>edx</th>
<th>eax</th>
<th>self</th>
<th>args</th>
</tr>
</thead>
</table>

Remote Context

<table>
<thead>
<tr>
<th>var array</th>
<th>self</th>
<th>args</th>
<th>var1...</th>
</tr>
</thead>
</table>

Block Stack

| self | args | var1... |
Object Structure

ri.RtObject

<table>
<thead>
<tr>
<th>Shape + flags</th>
<th>SIZE</th>
<th>METHOD ARRAY</th>
<th>POINTER</th>
<th>PRIMITIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[methods][methods][]...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>byte[], double[], RtObject[], Object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>long, double</td>
</tr>
</tbody>
</table>
Methods from ST to Java

- Use supplied ST compiler
- Translate to PBC
  - constants serialization
  - byte code conversion
  - fixup dead code, order
- Translate from PBC to Java Class
  - Use ASM
- Build GWT inline cache on demand
Portable Code Example

:CODE,
type=classMethod,
class=RtDictionary,
selector=initialSize,
args=0,vars=0,blocks=0,
pbc=04000000230124022301240240023438240223021E
initialSize
   "Private - Answer the initial number of elements that a new instance of IdentityDictionary contains."
^8
Method Structure

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBC File</td>
<td>JavaClassFile</td>
<td>Method Handle</td>
</tr>
<tr>
<td></td>
<td>class</td>
<td></td>
</tr>
<tr>
<td></td>
<td>selector</td>
<td></td>
</tr>
<tr>
<td></td>
<td>source</td>
<td></td>
</tr>
<tr>
<td></td>
<td>flags</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pragmas</td>
<td></td>
</tr>
</tbody>
</table>
ByteCode Differences

- 25 PBC but only 4 real differences
- Method Invocation
- Primitives
- Blocks and returns
- Constants
GWT as inline cache

- GWT
- Match
- Fallback
- COLLECT
  - Airity fix
- INSERT
  - Callsite
- SPREAD
- INVOKE
Blocks

- Code plus context
- Code is just another method
- Replaced stack vars with shared array
- Non local return
  - returns to caller of creator
  - use var array to locate return frame
  - throw exceptio
handleMessage:aMessage
"<modified:sys=GAKRE8CA,time=04/26/11 at 07:55:14 am>  "
  "Message is an RtSystemMessage"
| dst hdlrs parameters channel|
parameters := aMessage parameters.
dst := aMessage destination.
channel := (parameters at:1) asUpperCase.
(dst isNil or:[dst = '00000000'])
  ifTrue:[" broadcasts look up by channel "
    " but drop if I am waiting for the response "
    hdlrs := handlers riDetectAll:[ :h | h key = channel].
    hdlrs isEmpty
      ifTrue:[
        Object allSubclasses do:[:c | c monitorsTopic:channel message:aMessage]].
    dst := ".
    handlers do:[ :a | a key = channel ifTrue:[(a value at:2)
         handleSystemMessage:aMessage]] ]
  ifFalse:[ " private so lookup by topic "
    handlers do:[ :a | a key = channel ifTrue:[(a value at:2)
         handleSystemMessage:aMessage]]].
Return Code Example

includes: anObject
  "Answer true if the receiver contains an
element equal to anObject, else answer false."
self do: [:element |
  anObject = element
  ifTrue: [^true].
[^false]
Stack Var Structure

Normal Stack

- var1...
- edx
- eax
- self
- args

Remote Context

- var array
- edx
- eax
- self
- args

Block Stack

- self
- args
- var1...
Primitives

- Along with bytecodes do all the work
- Written in Java with RtObject args
- Supports fallback to Smalltalk code
- Low level (math) and high level (string)
- Largest Java Code effort (1500 lines)
at: anInteger
"Answer the object in the receiver at index position anInteger. If the receiver does not have indexed instance variables, or if anInteger is greater than the number of indexed instance variables, report an error."
<primitive: 60>
^self primitiveFailed

// prim 45
static public RtObject primFLoatExp(RtObject rcvr) {
    // return exponential of the receiver
    double c=rcvr.getDoubleValue();
    return new RtObject(Math.exp(c));
}
Constants/Literals

- In Smalltalk can be any object
- In Java are limited to primitives
- In reality are also limited in ST
  - primitives and arrays of primitives
  - Globals and Class Vars (use prim)
- Use Constant Methodhandle to create
  - name is serialized constant
Callsite management

- Need to invalidate on code changes
- Current approach is to drop all
- Also drop at 30K sites
FUTURE TOPICS

- Coroutines
- Debugger
- FFI
- Performance
allocSharedMem: pBaseAddress name: pszName size: ulSize flags: ulFlags
  <api: '300' struct ulong ulong ulong ulong ulong>
  ^self invalidArgument
Debugger

- Stack var inspection
- Hop step jump
- Instances inspection
- Currently using JVMTI
  - slow
  - Requires C code agent
Hanoi Code Example

testHanoiMove: numberOfDisks from: source to: dest temp: temp
"<modified:sys=GAKRE8CA,time=06/28/11 at 04:39:33 pm>  "
   numberOfDisks == 1 ifTrue: [^self].
   self
       testHanoiMove: numberOfDisks - 1 from: source to: temp temp: dest;
   testHanoiMove: numberOfDisks - 1 from: temp to: dest temp: source

public void testMoveLong(long numberOfDisks, long source, long dest, long temp) {
   if(numberOfDisks == 1L) return;
   testMoveLong(numberOfDisks-1L, source, temp, dest);
   testMoveLong(numberOfDisks-1L, temp, dest, source);
}
Performance for Hanoi 20

- java prims 1.8 ms
- smalltalk 3 ms
- java boxed 4.7 ms
- RtObjects 32 ms
- Rtalk 65 ms