Dino2
The Evolution of the VA Smalltalk Virtual Machine

John O’Keefe
Chief Technical Officer
Instantiations, Inc.
Dino2

• Why am I giving the presentation instead of a real VM guy?
Dino2

- Because the real VM guy is busy!!
  - Seth and Kate’s daughter Adelyn, born June 19
Dino2
Agenda

• Driving forces
• VAST VM history
• Do we need a new VM?
• Challenges
• How we did it
• Results
• Demo
• Still to do
• Q&A
Dino2
Driving Forces

• 64-bit support required
  • Dramatically expands available memory space
  • Interface with 64-bit DLLs/SOs

• Simplify maintenance and enhancement of the VAST VM
  • Enables use of modern tool chains
  • Replaces current proprietary modeling language with C
Dino2
VAST VM History

- Extremely stable - basically unchanged in 25+ years
- Developed using proprietary Smalltalk VM Modeling Language
  - Maximize efficiency on constrained hardware
  - Cross-platform model
  - No dependency on C compiler

Smalltalk Model Code → Portable Assembler → x86, Power PC, SPARC
Dino2
Do We Need a NEW VM?

• Smalltalk Modeling Language
  • Obscure – hard to learn/extend
  • Obfuscates the algorithms

• Portable Assembler
  • Does not take advantage of new machine architectures

• Generated machine code
  • Non-standard calling conventions
  • Standard debuggers don’t work
  • Hard to map performance tools result back to model

• JIT
  • Must be hand-built to match machine architecture
VMprCharacterTestBit

self

systemPrimitive: 'VMprCharacterTestBit'
receiverClass: Character

body: [ | receiver byteArray bit addr |
receiver := registerModel allocateDataRegister.
byteArray := registerModel allocateAddressRegister.
bit := registerModel allocateDataRegister.
byteArray gets: (self parm: 1 of: 1).
((self isImmediate: byteArray) || (self isBytes: byteArray) not) do: [
    self failAsmPrimitiveViaCache: PrimErrInvalidClass arg: 1].
receiver gets: (self receiverForParms: 1).
self convertToCharacter: receiver.
bit gets: receiver.
bit &= 7.
receiver shiftRightLogical: 3.
(receiver greaterThanOrEqualUnsigned: (byteArray at: (constant field: 'size' of: K8ObjectHeader))) do: [
    self failAsmPrimitiveViaCache: PrimErrInvalidSize arg: 1].
registerModel region: [
    addr := registerModel allocateAddressRegister asBytePointer.
    addr gets: (constant addressOfLabel: (label global data named: 'K8SetBits')).
    bit loadUnsigned: (addr indexedBy: bit).
receiver loadUnsigned: ((byteArray asBytePointer at: constant objectHeaderSize) index: receiver).
and setFlags source: bit dest: receiver.
condition zero do: [receiver gets: false] else: [receiver gets: true].
self return: receiver parms: 1]
EsPrimitive(VMprCharacterTestBit)
{
    U_16 value;
    EsObject byteArray;
    U_8 bit;

    byteArray = EsPrimitiveArgument(1, 1);
    if (!EsIsBytes(byteArray))
        EsPrimitiveFailed(EsPrimErrInvalidClass, 1);
    value = EsCharacterToU16(EsPrimitiveReceiver(1));
    bit = (U_8)(value & 7);   /* 0 to 7  bit number within byte */
    value = (value >> 3) + 1; /* 1 to (MAX_CHARACTER_VALUE/8)+1  byte number within table */
    if (value > (byteArray
                  >size))
        EsPrimitiveFailed(EsPrimErrInvalidSize, 1);
    EsPrimitiveSucceed((((EsByteAt(byteArray, value)) & (1<<bit)) ? EsTrue : EsFalse), 1);
}
Dino2
Challenges

- Minimal existing test cases
  - If the basic image tests run, the VM is OK
- ‘VM in C’ performance
  - 32-bit x86 VM loses an available register (-)
  - C compilers produce far superior code; example: instruction reordering (+)
  - Many benchmarks (both micro and macro) ported and developed
- Tool chain convergence
- Image conversion
- Impedance mis-match
  - “Jump where ever I want to”, stack and register mgmt
Dino2
How We Did It

- Moved to cmake and gcc based tool chain
  - Use ‘register intrinsics’ for performance
  - Nightly build and test
- Minimal assembler
  - Low-level arithmetic, exception handling, OLE support
- Incremental changes
  - Shim code developed to cross old/new boundary
    - VM always works
  - 64-bit ‘clean’ changes as we go
  - Detour from plan: Interpreter was done all in one piece
Dino2
How We Did It

• Example: Garbage Collector
  • 3 major components: Scavenger, Mark-Compact, Allocator
  • Components converted one-at-a-time
  • Millions of lines of trace output produced to verify everything worked the same
  • Incremental changes means we always had a working VM to test the changes
Dino2
How We Did It

- Just in time image conversion (64-bit VM)
  - 32-bit images and image components (ICs) converted on first use
  - Image can be saved in 64-bit format
  - 32-bit ICs loadable from 64-bit image
Dino2
How We Did It

There’s no magic in software, just hard work with a result that may appear to be magic!

• The image has to change -- because 64-bitness shows through
  • Foreign Function Interfaces (FFI) aka PlatformFunctions
  • Memory mapping objects (OSObjects)
• Goal is to minimize changes in user code
  • So most of the changes are in VAST framework code
Dino2
How We Did It

- Elastic PlatformFunctions
  - Holds template for making FFI call
  - Parameter sizes and offsets were fixed
  - Changed parameter sizes and offsets from fixed to relative
Dino2
How We Did It

- Elastic OSStructures
  - Accessors *were* based on fixed size and structure offsets
  - Changed accessors from absolute to relative offset
  - Compute fixed offsets on image startup
Dino2
How We Did It

• Elastic OSStructure Example (C)

```c
#ifdef _WIN32
#include <pshpack1.h>
#endif
typedef struct NMHDR
{
    HWND hwndFrom;
    UINT_PTR idFrom;
    UINT code;        // NM_ code
};
typedef struct TVKEYDOWN {
    NMHDR hdr;
    WORD wVKey;
    UINT flags;
};
#ifdef _WIN32
#include <poppack.h>
#endif
```
Dino2
How We Did It

- Elastic OSStructure Example (Smalltalk)

"Define NMHDR Struct"  
OSNmhdr members: #(#hwndFrom #idFrom #code) types: #(pointer pointer uint32).

"Define TVKEYDOWN Struct - Pack1 if 32-bit"  
OSTvKeydown members: #(hdr wVKey flags) types: #(OSNmhdr uint16 uint32).  
System is64BitVM ifFalse: [OSTvKeydown updateAlignmentType: AlignNone]. "Pack on byte boundary"

OSTvKeydown>>#flags
"Answer the member: UINT flags.  
   32/64-bit compatible"
^ self uint32At: #flags
Dino2
How We Did It

- Additional Benefits of Elastic OSStructures
  - Custom Packing for data structures
    - OSStructure members: #(a b) types: #(int8 int8) alignment: Align2 "pack2"
  - Custom Padding
  - Embedded OSStructures
    - OSStructureA members #(a) types: #(int8)
    - OSStructureB members #(a b) types: #(int8 OSStructureA)
  - Nested Anonymous Structures/Unions
    - OSStructure members: #(a (b c) ) types: #(int32 ((int32 int32)) )
      - struct { int a; struct { int b; int c; } }
    - OSStructure members: #(a (b c) ) types: #(int32 (int32 double) )
      - struct { int a; union { int b; double c; } }
Dino2
How We Did It

• Additional Benefits of Elastic OSStructures
  • Pointer Types
    • OSStructure members #(a b) types: #(pointer int32) "4 bytes on 32-bit, 8 bytes on 64-bit"
    • OSStructure members #(a b) types: #('uint8*' int32) "Also a pointer with additional type info"
  • Arrays
    • OSStructure members #(a b) types: #('int8[10]' int32) "Array types are supported"
    • OSVariableStructure members: #(a b) types: #(int8 pointer[]) "Flexible array types supported"
Dino2
How We Did It

• Additional Benefits of Elastic OSStructures
  • Dependency Analyzer
    • Don't need to define OSStructures in order of their dependencies
    • Invalid Circular dependencies will be detected
  • Extensible Base Types
    • You can add your own types, either globally or method override
    • We do a method override for TCHAR for future Unicode support
      • Currently a char8, but may later be a char32. Existing definitions using TCHAR are now future proofed for this change
Dino2
Results

- 64-bit VM is just a recompile
- No separate 32-to-64 bit image converter
- Interpreter benchmarks are > 80% of current VM
  - Before algorithm tuning
  - Before C tuning
- User code is largely unaware of change
Dino2

Demo
Dino2
Still To Do

- 80% done means more work to do
  - Performance tuning (algorithms and C)
  - JIT
  - 64-bit Packager
  - Improved garbage collector
  - Installation and setup
  - UNIX
Dino2
When can we have it?

• Windows - 3 delivery dates
  • Alpha
    • 1Q2016
    • Early customer involvement program; entry by invitation
  • Beta
    • 2Q2016
    • Open registration
  • Production
    • V9.0 on normal product delivery schedule
• UNIX later
Contact us

• General information
  • info@instantiations.com

• Sales
  • sales@instantiations.com

• Support
  • support@instantiations.com

• Me
  • john_okeefe@instantiations.com
Thank you for your attention

Questions?