Code Optimization
Introduction

• Working at Soops since 1995
• Customers include
  - Research (Economical)
  - Exchanges (Power, Gas)
  - Insurance
• Projects in
  - VisualWorks
  - GemStone
  - VisualAge

Code Optimization
Adriaan van Os
ESUG 2006
Demanding projects

• **Data-intensive**
  - Rule based data warehouse like application built with VisualWorks and GemStone.

• **Calculation-intensive**
  - Decentralized coupling of electricity markets done with VisualWorks
Prologue
Controversial

- Hard to predict what piece of code might become a problem
- Often there won’t be a problem
- Optimizations will break some (Smallllint) rules
- Optimized systems can become harder to extend
Context

- Optimizations are often not reusable
- Any change might outdate them
  - VM
  - Image
  - Platform
- Demand for Unit Tests
Strategy Against Performance Problems

• Concentrate on the design first
• Result too slow?
  - Analyze it
    • Tools
  - Solve it
    • Tips
    • Tricks
  - Test it
    • Tests
Tools
Tools

• For analyses
  – Time millisecondsToRun: []
  - (Multi)TimeProfiler (VW)
  - (Multi)AllocationProfiler (VW)

• For inspiration
  - A few Smalllint rules
  - RBBYTECodeTool (VW)
  - RBDecompiledTool (VW)
VW RB Integration
Time millisecondsToRun: []

- Run multiple times
  - 10000 times
- Beware of large integers
- Beware of allocation/garbage collector
- Sometimes it's still hard to get consistent results 😞
TimeProfiler profile: [1000000 timesRepeat:[123456789 printString size]]
TimeProfiler profile: [1000000 timesRepeat:[123456789 printString size]]
Tips & Tricks
Selectors

• **Special selectors**
  - Specialized opcode will be generated if syntactic requirements are met
  - Can’t be overwritten
  - See DefineOpcodePool class
  - The set of selectors can be modified

• **Optimized selectors**
  - Transformed if syntactic requirements are met
  - Selectors will be inlined
  - Can’t be overwritten
  - See MessageNode class
  - You can add your own transformations

• **Primitives**
  - Write your own
Selector Transformations

transformIfNil

"MacroSelectors at: #ifNil: put: #transformIfNil"

^((self testLiteralBlock: 0 at: 1)
   and: [self receiver hasEffect not])
ifTrue:
   [ConditionalNode new
      sourcePosition: self sourcePosition;
      condition: (MessageNode new
         receiver: (LiteralNode new value: nil) selector: #== arguments: (Array with: self receiver))
      trueBlock: arguments first falseBlock: (BlockNode new body: self receiver) from: self]
ifFalse: [nil]
Special and Optimized Selector Examples

• Use and: instead of && even if you have the argument ready
• The VW compiler warns you for aBoolean and: aBlock ...
• ...but aBoolean and: [aBlock value] doesn’t seem to be faster
• It’s true for, eg. timesRepeat:
• Compiler inlines self do: in SequenceableCollection, but not outside
Inlining

- Get rid of those message sends
- Inline cases the compiler don’t know about
Blocks

- Keep them clean
  - Declare variables in innermost scope
  - Avoid assigning values to outer scope variables
  - Avoid return (^) inside the block
- Some clean blocks can be inlined
  - No instance of BlockClosure is created
  - Share context with sender
Numbers

- Avoid coercion
- Avoid LargeIntegers
- Avoid Fractions
  - Avoid sending / and // with Integers
  - Finding gcd’s is very expensive
- Use Doubles
- Higher generality first
  - 10.0 * 10
Collections and Iterations

- Avoid intermediates and repeated iterations
  - Use `aDict keysDo:`, not `aDict keys do:`
    • Avoid keys anyway
  - Implement
    • `select:do:`
    • `select:collect:`
    • `collect:select:`
  - Use `modify:`, not `collect:` if possible

- Pregrow collections
Collections and Inlining

• Enable inlining
  – anArray do: [:each|
    • 1 to: anArray size do: [:idx|]
      – Not (1 to: anArray size) do: [:idx|]
  – anOrderedCollection firstIndex to: lastIndex do: [:|idx|]
    • Used to be faster than
      1 to: anOrderedCollection size do: [:idx|]
    • Slower in VW 7.4.1
Collection Types

- For faster lookups, use
  - Set/Dictionary
  - IdentitySet/IdentityDictionary
    - Beware of maximum identity hash
    - Implement your own hash algorithm
  - RBSmallDictionary
    - For very small collections
- For faster iterations, use
  - Array
- For faster growing, use
  - OrderedCollection
- Hybrid
  - OrderedDictionary
Conditionals

- If you have a lot of conditionals you maybe have too few classes
- Common cases first
Caching

- Don’t do anything twice
- Make cache lookups fast
- Keep cache management simple
GemStone

- Just a dialect(?)
  - All previous slides apply
- Objects on disk
  - Makes usage of identity more preferable
- Objects on other side of the wire
  - Reducing round trips is a major design issue
  - Minimize copying (replicating)
- Shared Objects
  - Garbage collection is harder
- Use the specialized collection types
Tests
testToDoOrderedCollection

| oc |

oc := (1 to: 10000) asOrderedCollection.
self slower: [oc do: [:each | ]] than: [1 to: oc size do: [:idx | ]]
Test Method

slow: aSlowBlock than: aFastBlock
| slowCount slowTime fastCount fastTime faktor |
slowCount := fastCount := 1.

ObjectMemory garbageCollect.

[(slowTime := TimemillisecondsToRun: [slowCount timesRepeat: [aSlowBlock value]]) < 200] whileTrue: [slowCount := slowCount * 2].

ObjectMemory garbageCollect.

[(fastTime := Time millisecondsToRun: [fastCount timesRepeat: [aFastBlock value]]) < 200] whileTrue: [fastCount := fastCount * 2].

faktor := slowTime / slowCount / (fastTime / fastCount).

self assert: faktor > 1
description: aSlowBlock method DecompiledSource , ' is slower than ' , aFastBlock method DecompiledSource
Conclusions
Conclusions

• Concentrate on design first
• Don’t try to predict problems
• Easy wins with
  - Inlining Blocks
  - Caching
  - Choosing the right collection types
• Test your tricks
• Test again with new VM, image, hardware, etc
References

- Efficient Smalltalk
  - Travis Griggs, Smalltalk Solutions 2006
- VisualWorks Implementations Limits (PDF)
  - Cincom
- VisualWork Optimization (PDF)
  - Bernard Horan, Laura Hill, Mario Wolezko
- The Hitch Hiker's Guide to the Smalltalk Compiler
  - Vassili Bykov
- Niet zo, maar zo
  - Adriaan van Os
Questions?

Thanks

adriaan@soops.nl