Pharo VM performance
Clement Bera
Myself

- Clément Béra
- 2011-2013: Engineer on the Pharo VM
- 2013-2017: PhD student
  - Optimisations of the Pharo VM JIT compiler
Binary tree benchmark

- Interpreter 2005
- Stack 2009
- Cog V1 2010
- Cog V2 2011
- Spur 2014
- Sista future
Plan

- Pharo 5 (stable)
  - First time we out benched most competitors
- Pharo 6 (released next week ???)
- Pharo 7
GC

• Pharo 5
  • New memory manager Spur

• Pharo 6
  • New compactor

• Pharo 7
  • Incremental GC ???
Pharo 5: Spur

- Efficient scavenges
- In most applications, most GC time is now in scavenges
Pharo 6: New compactor

Loading a 200 Mb Moose Model in 250 Mb image

<table>
<thead>
<tr>
<th></th>
<th>February</th>
<th>April</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total time</strong></td>
<td>2 min</td>
<td>1 min</td>
</tr>
<tr>
<td></td>
<td>2 sec</td>
<td>2 sec</td>
</tr>
<tr>
<td><strong>Time in Full GC</strong></td>
<td>1 min</td>
<td>2 sec</td>
</tr>
<tr>
<td><strong>Full GC avg pause</strong></td>
<td>15 sec</td>
<td>0.5 sec</td>
</tr>
<tr>
<td><strong>Time in scavenger</strong></td>
<td>15 sec</td>
<td>15 sec</td>
</tr>
</tbody>
</table>
Pharo 6: New compactor

Loading a 200 Mb Moose Model in 250 Mb image

<table>
<thead>
<tr>
<th></th>
<th>February</th>
<th>April</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total time</strong></td>
<td>2 min</td>
<td>1 min 2 sec</td>
</tr>
<tr>
<td><strong>Time in Full GC</strong></td>
<td>1 min</td>
<td>2 sec</td>
</tr>
<tr>
<td><strong>Full GC avg pause</strong></td>
<td>15 sec</td>
<td>0.5 sec</td>
</tr>
<tr>
<td><strong>Time in scavenge</strong></td>
<td>15 sec</td>
<td>15 sec</td>
</tr>
</tbody>
</table>

<- GC tuning gets it down to 5 sec
Pharo 7: Incremental GC ??

• Full GC pauses: ~500 ms at ~500Mb

• Java default GC at 200ms soft real time

• Solution
  
  • Incremental marking

  • Incremental compaction
Code execution

- Pharo 5:
  - Spur got 1.8x

- Pharo 6:
  - Polishing and micro-optimisations

- Pharo 7:
  - Sista gets 1.5x-5x
Pharo 5: Spur 1.8x

- Class table speeds-up look-up caches
- New immediate objects
- 22 bits hash
Pharo 6

- Register allocation improvements
- Two path compilation
  - Frameless code for setter-like methods
Sista: Pharo 7?

- Program introspection
  - Speculate on types based on previous runs

- Optimize frequently used code
- Deoptimize and reoptimize code incorrectly speculated
Goals

- Program readability
- Performance
Program readability

array do: #yourself.

array do: [ :elem | elem yourself ].

1 to: array size do: [ :i | (array at: i) yourself ].
Program readability

array do: #yourself.

array do: [:elem | elem yourself ].

1 to: array size do: [:i | (array at: i) yourself ].
Getting stable

• Support most development workflow

• Support image recompilation

• Integration has started
In-image design

- In-image design
- Smalltalk image
- Virtual machine
- CompiledCode (persisted across start-ups)
- Cogit
  - CompiledCode to native code
  - Machine-specific optimisations
- Baseline JIT
- Optimising JIT
- Scorch
  - CompiledCode to CompiledCode
  - Smalltalk-specific optimisations
- native functions
  - (discarded on shut-down)
Missing

- IDE support
- Debugger
- Methods to show

- Stability, testing
Are you interested?

• Incremental GC?
• VM performance?
• VM features?

• Come and talk to us!
We are looking for...

- Use-cases showing what to improve
- Large real-world benchmarks
- Contributors
- Investment
Conclusion

• Pharo 5: Fastest VM

• Pharo 6: Polishing

• Pharo 7: Going further