OMR

A JVM's Journey Into Polyglot Runtimes

Charlie Gracie
February 8, 2016

Who am I?

Charlie Gracie
charlie_gracie@ca.ibm.com
@crgracie

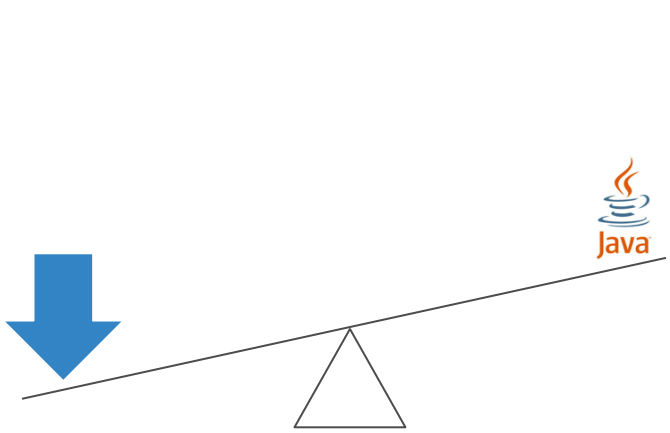
- Garbage Collection Architect for IBM J9 JVM
- Current focus on developing technologies for the OMR project

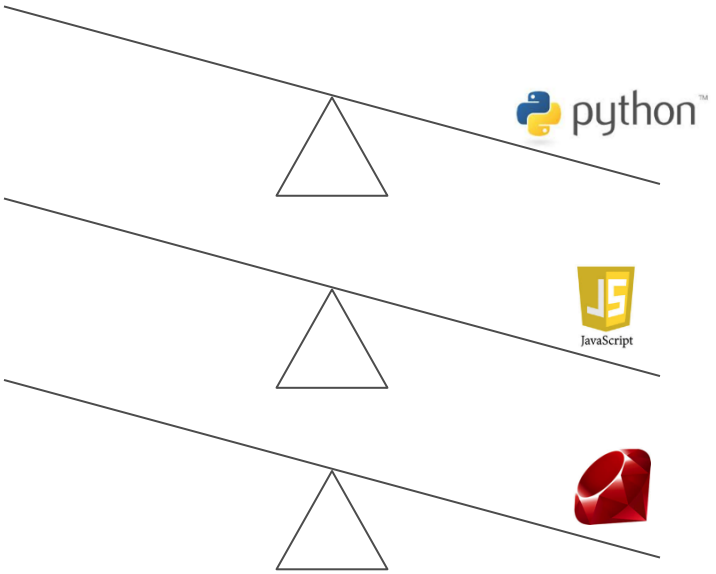
Java and the JVM

It’s a Polyglot World

No Shared Technology

- Effort in one runtime has no leverage in other runtimes





- …looking very costly to support many language runtimes

Languages on The JVM

- Leverage the investment in JVM
  - Cross platform support
  - High performance runtime
  - Production quality
  - Tooling / monitoring support
  - Interoperability with Java

Languages on The JVM

- Works great for new languages like Scala and Groovy
- Existing languages have vibrant communities
  - Not all languages map nicely to Java semantics
  - We decided to experiment with a different approach that would allow new and existing language communities to leverage JVM capabilities

OMR

An open source toolkit for language runtime technologies.

Open Source Toolkit for Building Runtimes

- Eclipse OMR project proposal
  - https://goo.gl/ZTBoeu
- Will be open sourced under EPL
- Derived from the source code of IBM's production runtimes

Open Source Toolkit for Building Runtimes

- Implements language-agnostic parts of a managed runtime
- Bootstraps development of new runtimes

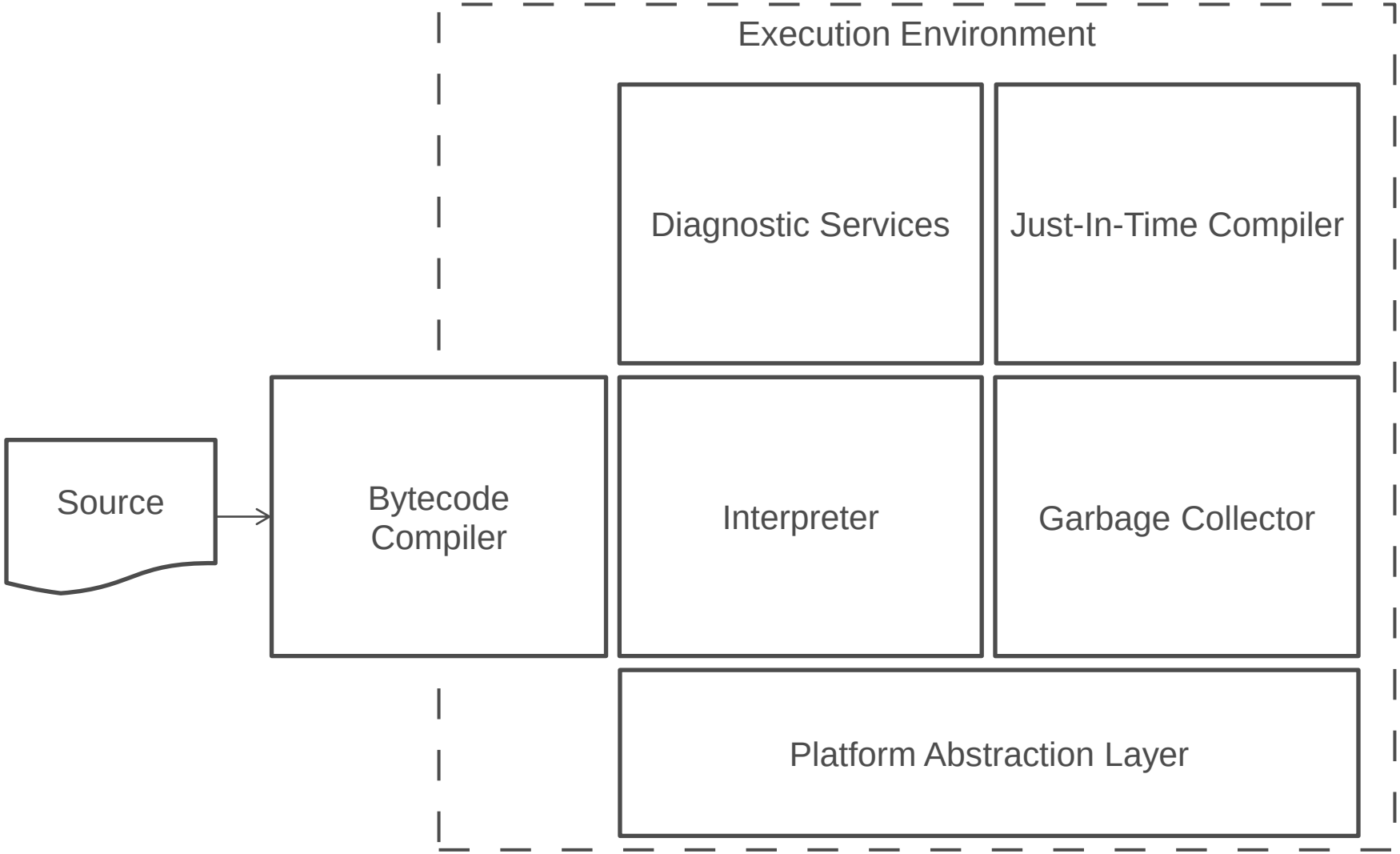
Open Source Toolkit for Building Runtimes

- Allows incremental enablement of advanced functionality
- Easily leverage new hardware features
  - GPU
  - FPGA

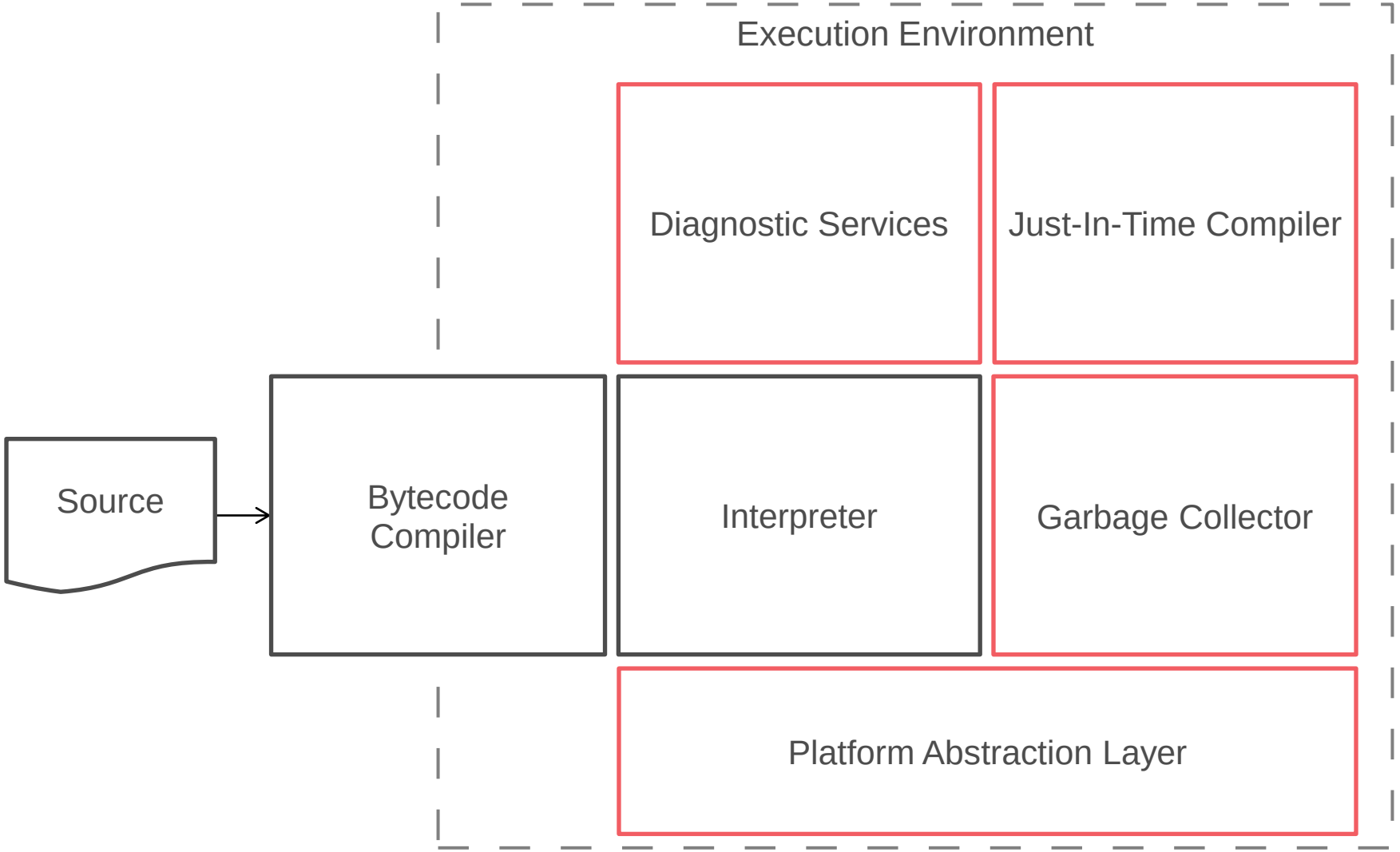
Open Source Toolkit for Building Runtimes

- Shipped as part of IBM SDK for Java 8
- Development of IBM SDK for Java 9 consuming OMR daily
- Proof-of-concept integration with Ruby MRI, CPython, SOM++

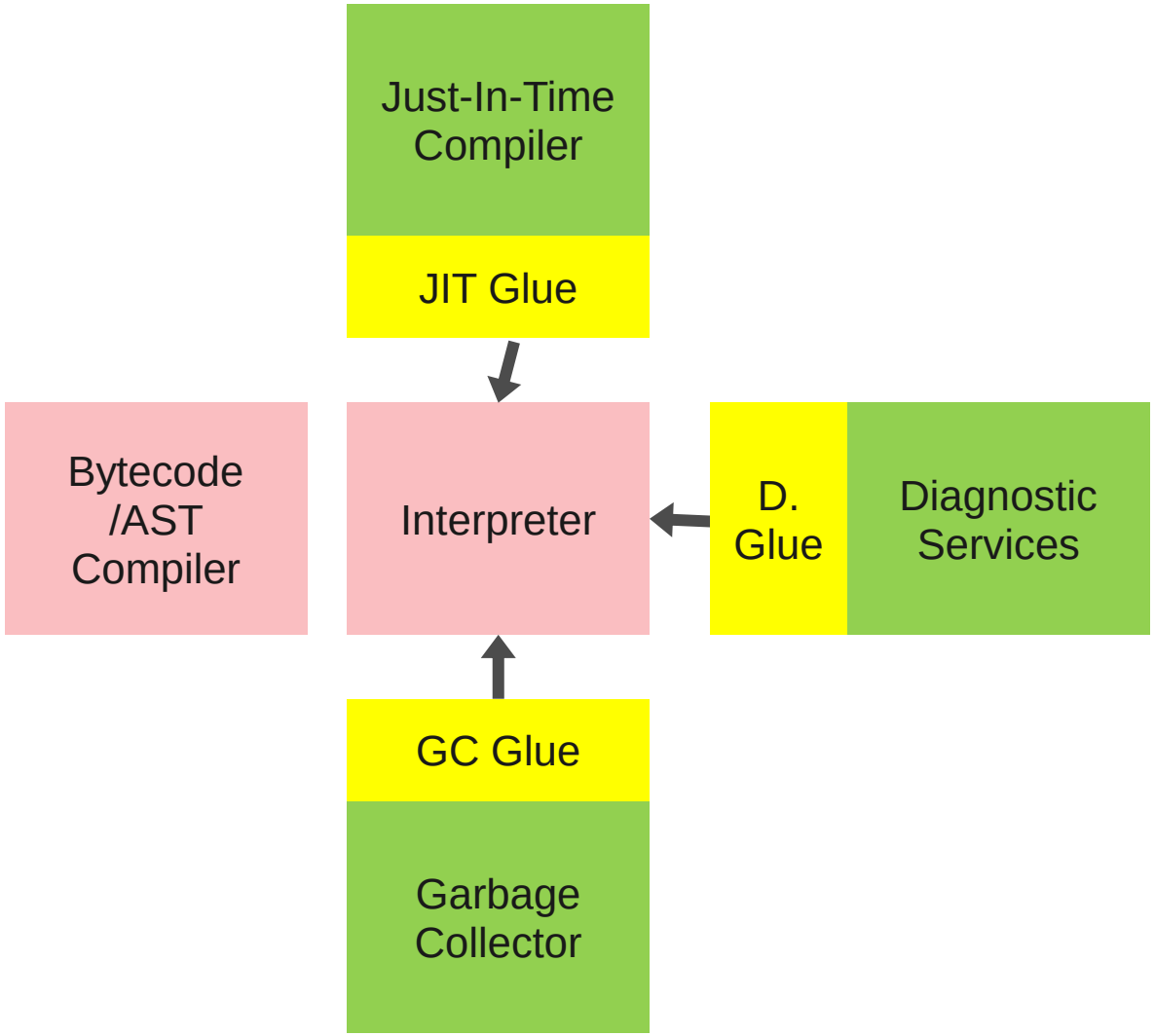
Language Runtime Components



Language-Agnostic Components



Gluing it Together



Ruby + OMR Preview

Ruby + OMR Preview

- GitHub location for the preview
  - https://goo.gl/P3yXuy
- Available as Docker images

Ruby + OMR Preview

- Based on Ruby 2.2.3
  - Working to merge with the master branch
- Successfully passes “make test”
- Runs rails apps

Ruby + OMR Preview

- Integrated OMR components include
  - Platform Abstraction Layer
  - Garbage Collection
  - Just In Time Compilation
  - Diagnostic Tooling

Garbage Collection

Our Strategy:

- Be 100% compatible

Our Strategy:

- Be 100% compatible
- Decrease pause times

Our Strategy:

- Be 100% compatible
- Decrease pause times
- Increase allocation speed

Our Strategy:

- Be 100% compatible
- Decrease pause times
- Increase allocation speed
- Improve object locality

100% Compatible

- Do not break C extensions
- Pass all tests that ship with MRI
- Test 3rd party applications

Decrease Pause Times

- Complete the following in parallel
  - Root Marking
  - Object tracing
  - Object finalization
  - Sweep

Increase Allocation Speed

- Use TLH mechanism from OMR
  - Threads reserve blocks of heap memory for exclusive allocation
  - Threads bump allocate out of these blocks

```
alloc(rbthread_t thread, int size)
        if (thread->tlhAlloc < thread->tlhTop - size)
                object = thread->tlhAlloc
                thread->tlhAlloc = object + size
        else
                object = OMR_GC_Allocate(thread, size);
```


Improve Object Locality

- Create new OMRBuffer object type
- Allocate OMRBuffers on heap
- Data is regularly adjacent to object in heap
- OMRBuffers are automatically reclaimed during collection

Decrease Pause Times

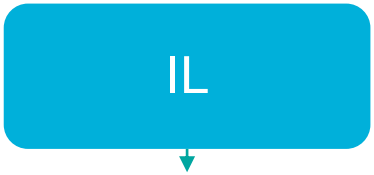
- OMRBuffers do not require calls to obj_free
- Significant reduction in pause times

Future Work

- Improve heap fragmentation
- Add support for concurrent marking

Just In Time Compilation

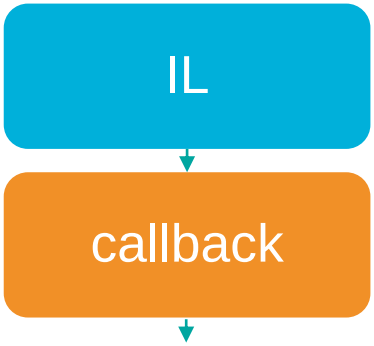
Our Strategy:



- Mimic interpreter for maximum compatibility.
- Implement simple opcodes directly in IL

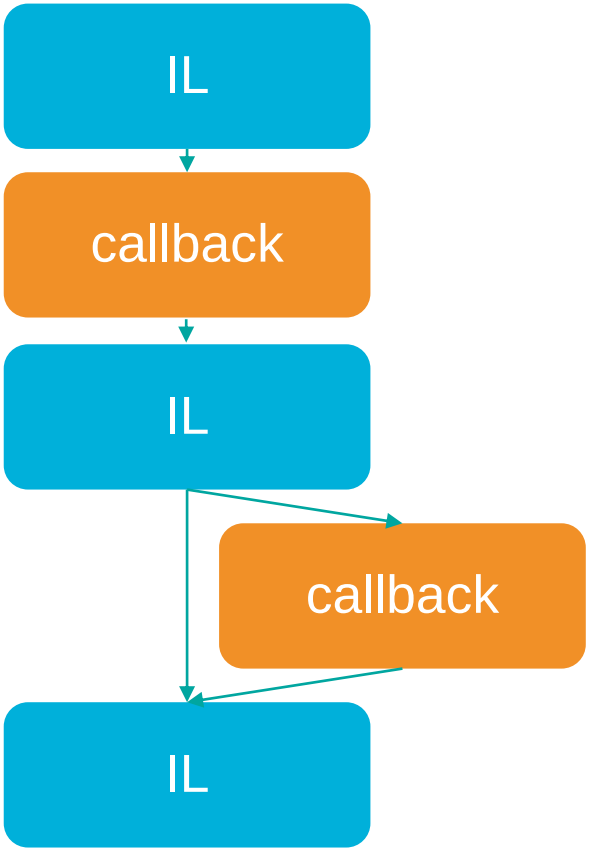
Our Strategy:

- Build callbacks to VM for complex opcodes.
  - Automatically generate callbacks from the instruction definition file.



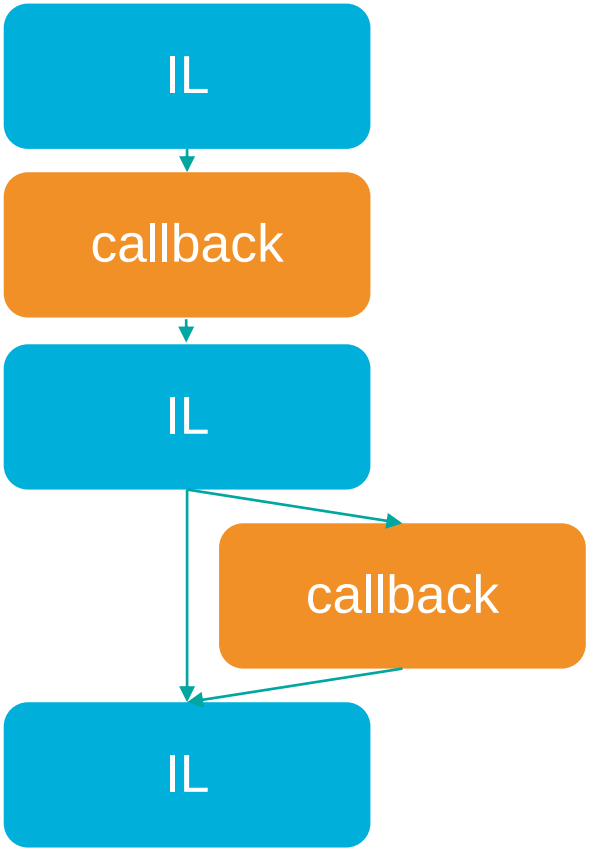
Our Strategy:

- Fast-path particular patterns
  - e.g `trace`



Our Strategy:

- Don’t try to handle everything – let interpreter handle hard cases!



Current Status

- Supports almost all opcodes
- Compile iseq after they are executed N times
- Dispatch to JIT code if the iseq has been compiled

Future Work

- Speculative optimizations powered by decompilation and code-patching
  - Class Hierarchy Based Optimization
  - Guarded inlining
  - Type Specialization
- Recompilation
- Interpreter and JIT Profiling
- Asynchronous Compilation
- More optimization!
  - OMR's Ruby JIT uses only **10 / 100+** optimizations.

Diagnostic Tooling

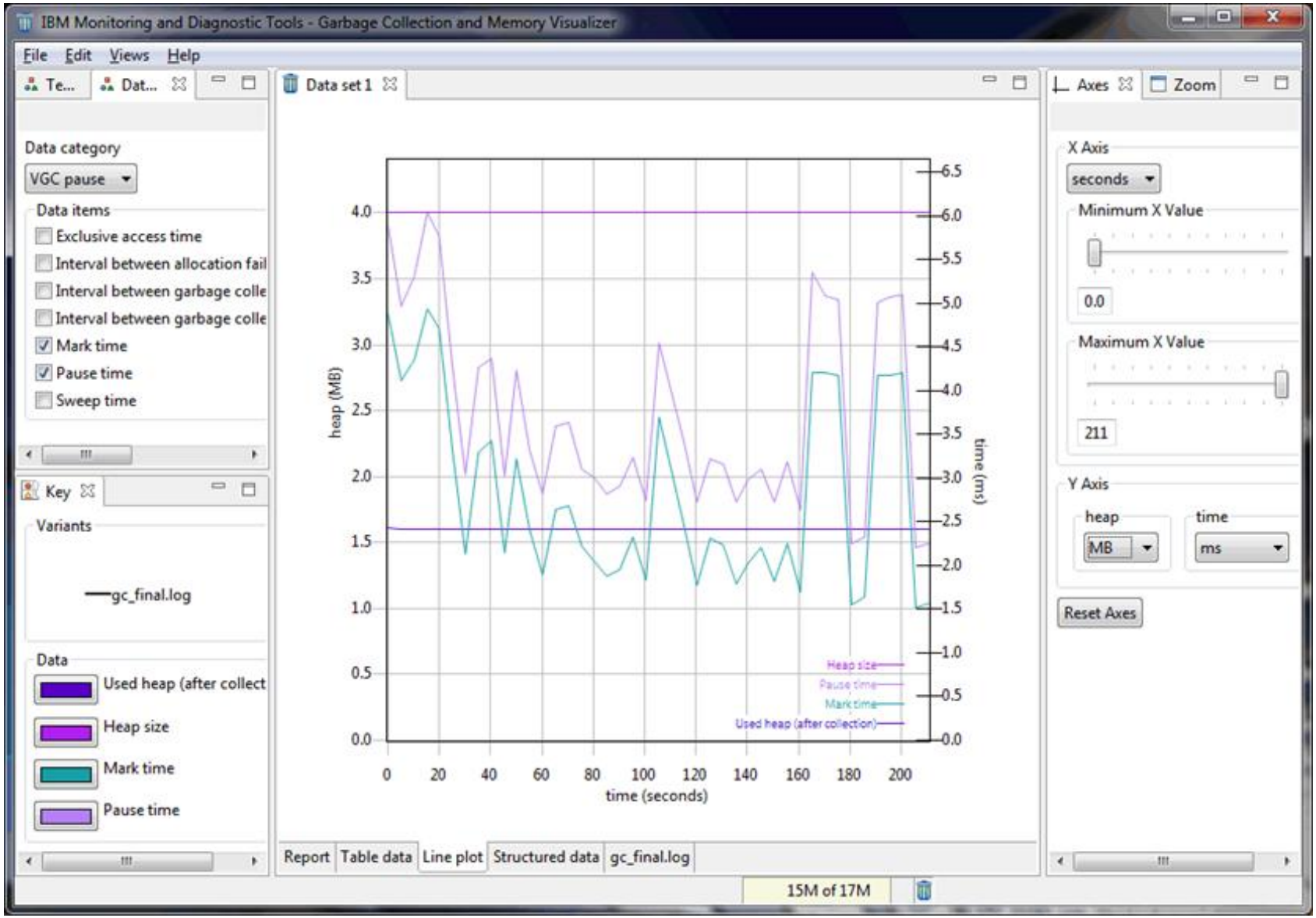
Verbose:gc Output

```
<af-start id="17" threadId="00007F2F3137CFD0" totalBytesRequested="8208" timestamp="2015-12-17T02:16:21.412" intervalms="23.538" />
<cycle-start id="18" type="global" contextid="0" timestamp="2015-12-17T02:16:21.413" intervalms="23.541" />
<gc-start id="19" type="global" contextid="18" timestamp="2015-12-17T02:16:21.413">
  <mem-info id="20" free="201320" total="4194304" percent="4">
    <mem type="tenure" free="201320" total="4194304" percent="4" />
  </mem-info>
</gc-start>
<allocation-stats totalBytes="2009264" >
  <allocated-bytes non-tlh="46328" tlh="1962936" />
  <largest-consumer threadName="OMR_VMThread [" threadId="00007F2F313786E0" bytes="2009264" />
</allocation-stats>
<gc-op id="21" type="mark" timems="3.660" contextid="18" timestamp="2015-12-17T02:16:21.417">
  <trace-info objectcount="27053" scancount="23093" scanbytes="926304" />
</gc-op>
<gc-op id="24" type="sweep" timems="0.232" contextid="18" timestamp="2015-12-17T02:16:21.417" />
<gc-end id="25" type="global" contextid="18" durationms="4.967" usertimems="4.256" systemtimems="0.000" timestamp="2015-12-17T02:16:21.418" activeThreads="1">
  <mem-info id="26" free="1467848" total="4194304" percent="34">
    <mem type="tenure" free="1467848" total="4194304" percent="34" />
  </mem-info>
</gc-end>
<cycle-end id="27" type="global" contextid="18" timestamp="2015-12-17T02:16:21.418" />
<allocation-satisfied id="28" threadId="00007F2F313786E0" bytesRequested="8208" />
<af-end id="29" timestamp="2015-12-17T02:16:21.418" threadId="00007F2F3137CFD0" success="true" />
```

Garbage Collection and Memory Visualizer

- Provides a graphical details on GC events post mortem from verbose:gc logs
- Works with IBM JDK, IBM Node.js and all of our proof-of-concepts
- https://goo.gl/YwNrmI

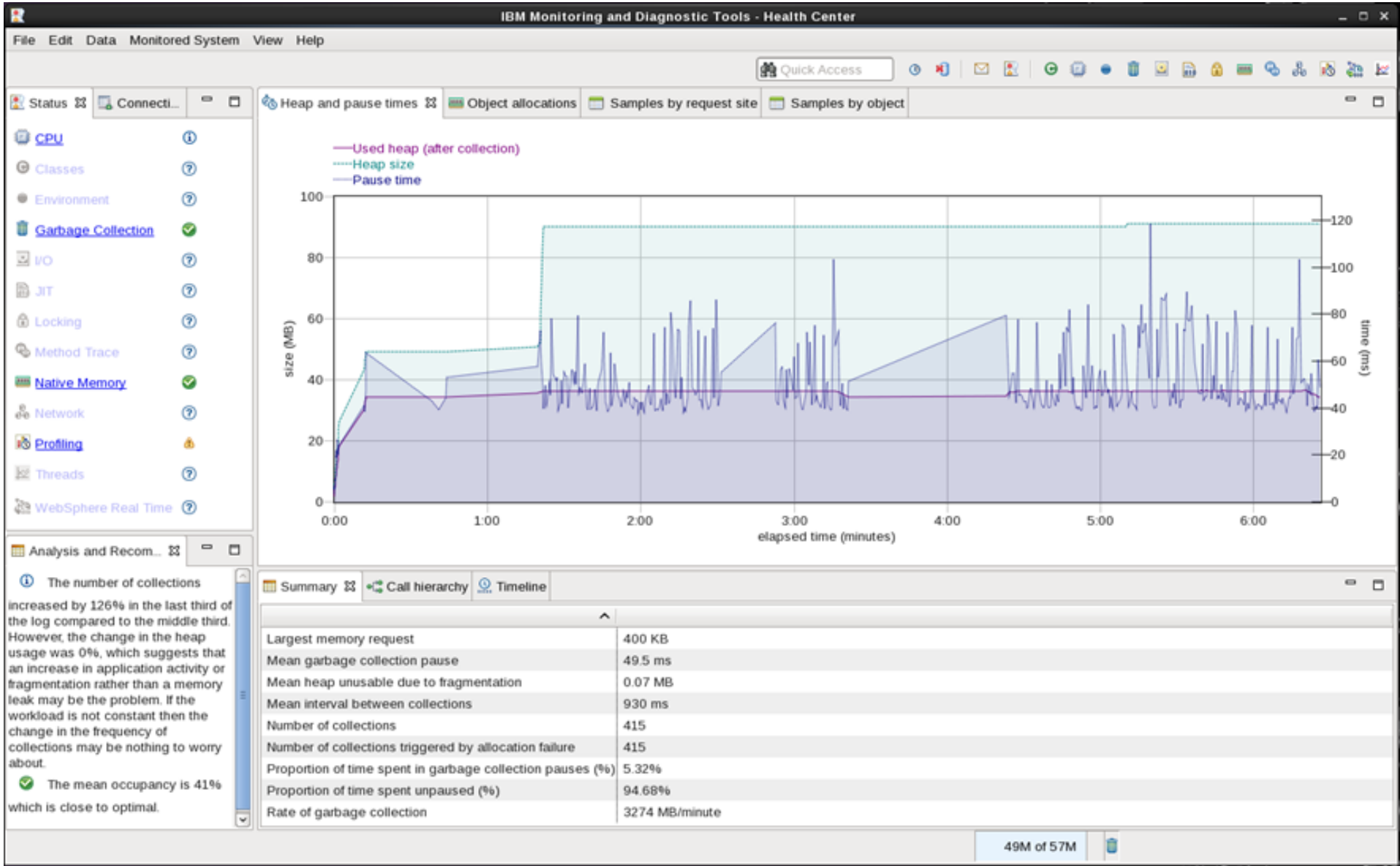
Garbage Collection and Memory Visualizer



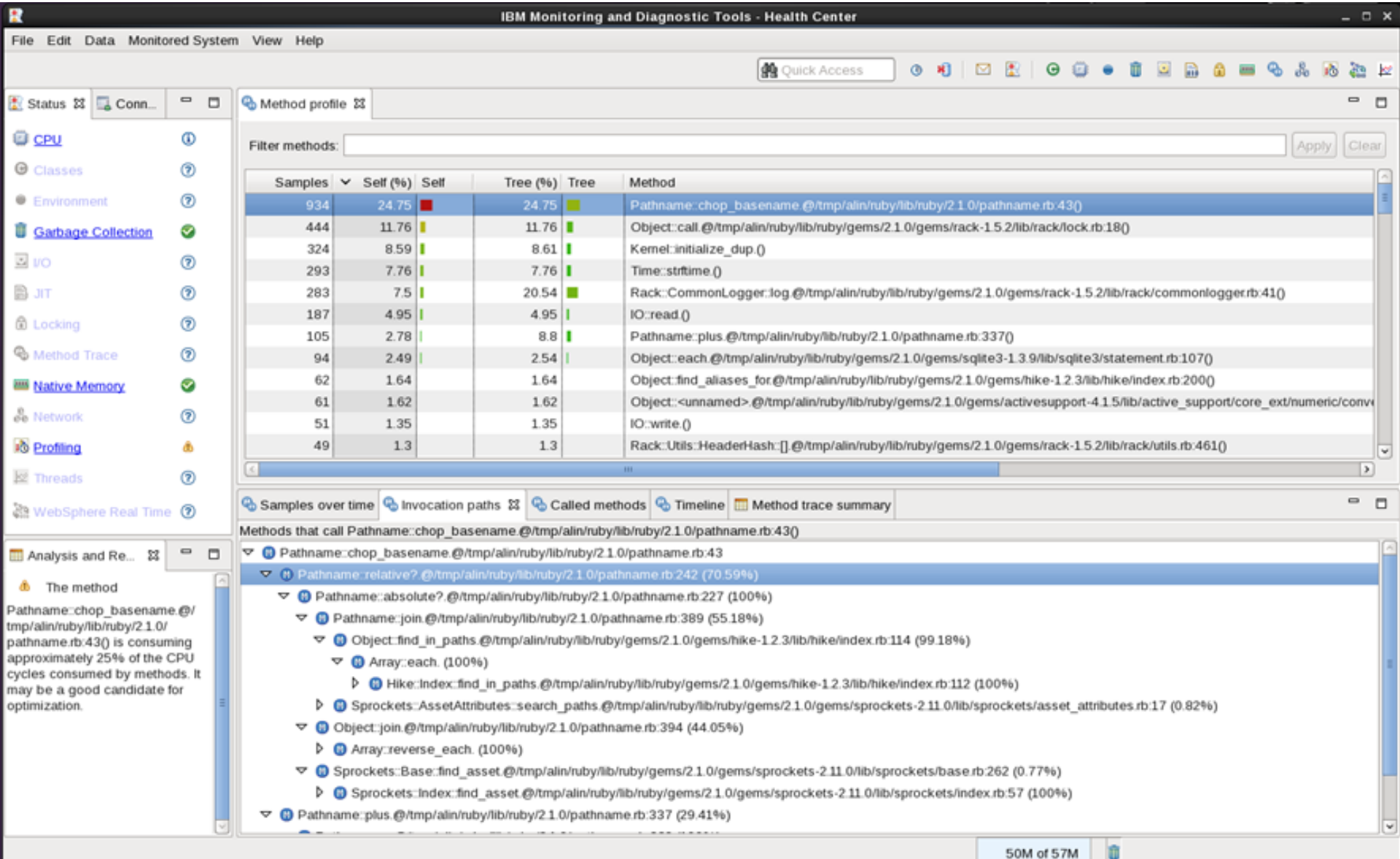
Health Center

- Provides a live view of runtime details
- Works with IBM JDK, IBM Node.js and all of our proof-of-concepts
- http://goo.gl/u3VITI

Health Center – GC Statistics

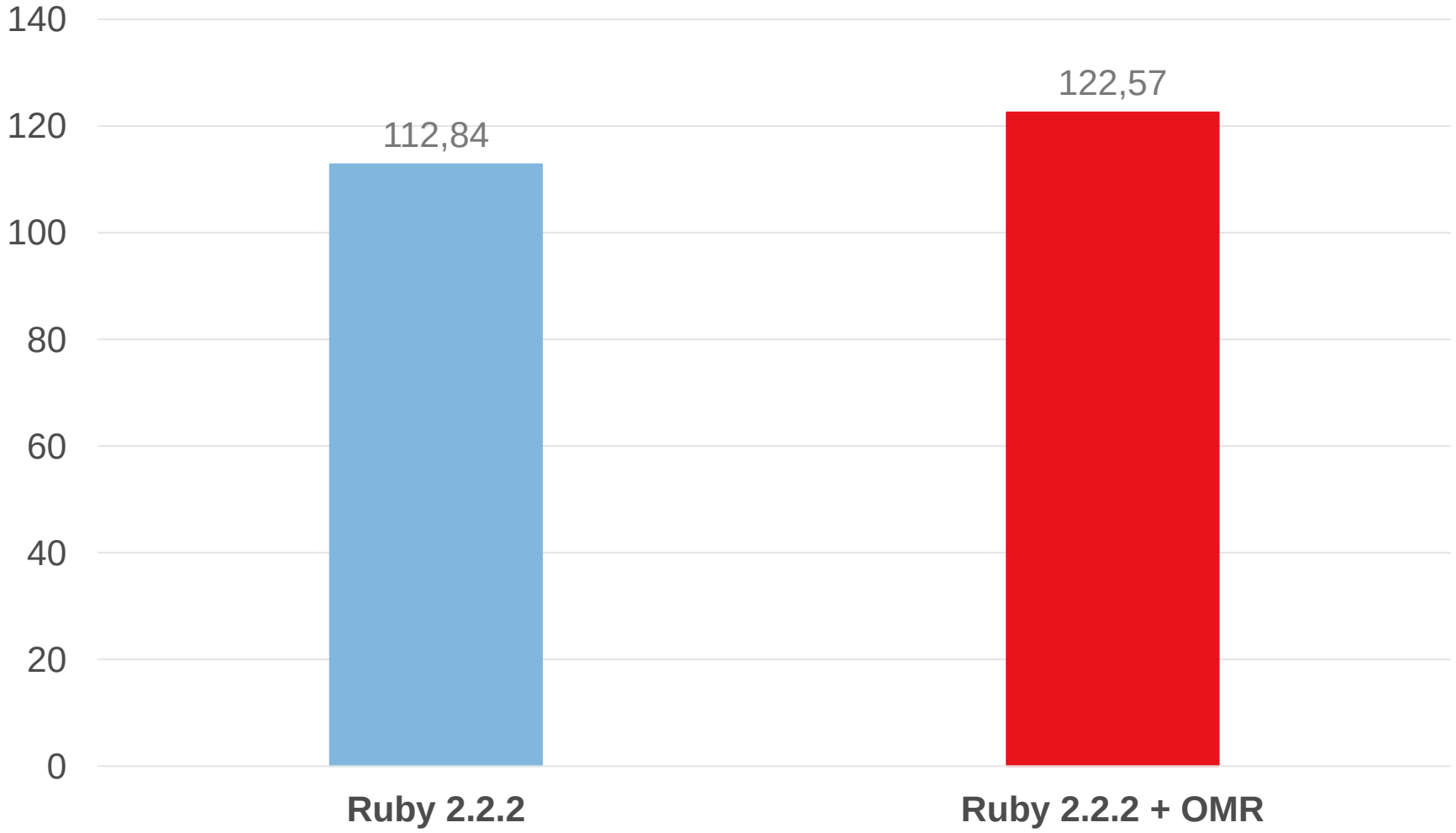


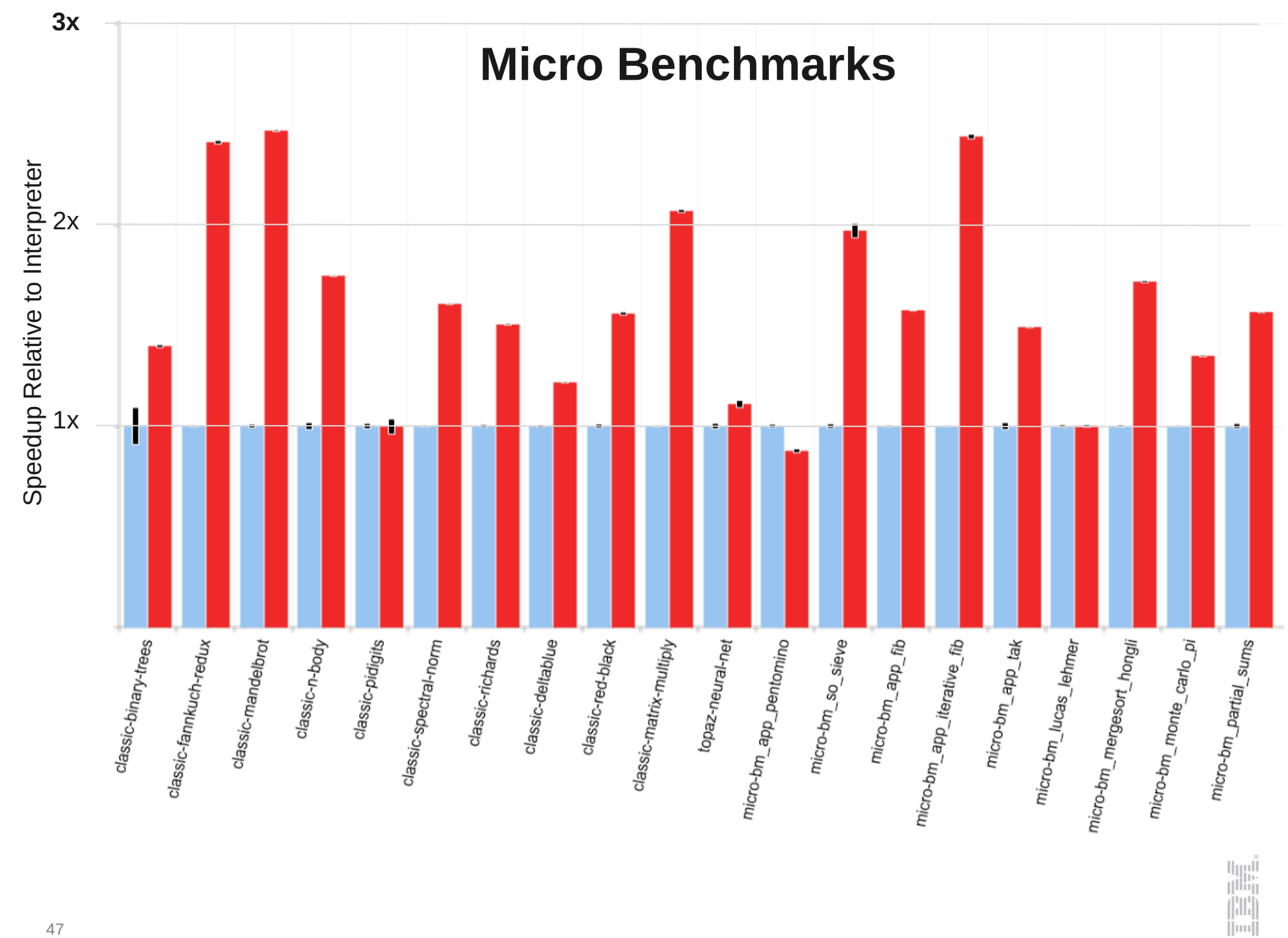
Health Center – Ruby Method Profiling



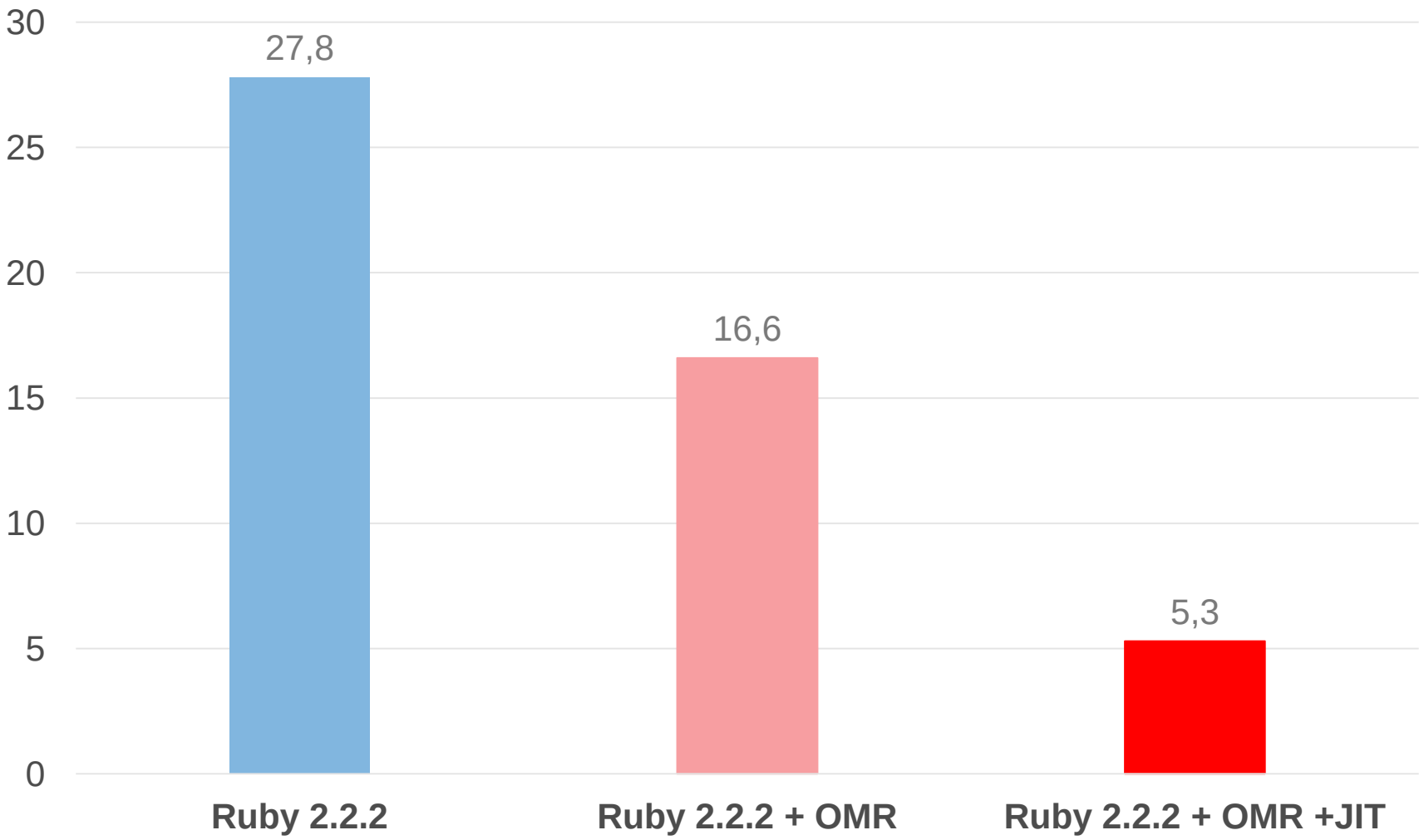
Performance Results

Is Ruby Fast Yet?




Micro Benchmarks
Speedup Relative to Interpreter
3x
2x
1x
classic-binary-trees
classic-fannkuch-redux
classic-mandelbrot
classic-n-body
classic-pidigits
classic-spectral-norm
classic-richards
classic-deltablue
classic-red-black
classic-matrix-multiply
topaz-neural-net
micro-bm_app_pentomino
micro-bm_so_sieve
micro-bm_app_fib
micro-bm_app_iterative_fib
micro-bm_app_tak
micro-bm_lucas_lehmer
micro-bm_mergesort_hongli
micro-bm_monte_carlo_pi
micro-bm_partial_sums

pow(2,n)


30
25
20
15
10
5
0
27,8
16,6
5,3
Ruby 2.2.2
Ruby 2.2.2 + OMR
Ruby 2.2.2 + OMR +JIT

Parting Thoughts

- Almost no one starts a new project saying:

Parting Thoughts

- Almost no one starts a new project saying:
  - First, I'll write the firmware from scratch…

Parting Thoughts

- Almost no one starts a new project saying:
  - First, I'll write the firmware from scratch…
  - First, I'll write the file system from scratch…

Parting Thoughts

- Almost no one starts a new project saying:
  - First, I'll write the firmware from scratch …
  - First, I'll write the file system from scratch …
  - First, I'll write the display drivers from scratch …

Parting Thoughts

- Almost no one starts a new project saying:
  - First, I'll write the firmware from scratch …
  - First, I'll write the file system from scratch …
  - First, I'll write the display drivers from scratch …

- We would like to make these statements just as unlikely:
  - First, I'll write the cross platform port library from scratch …
  - First, I'll write a garbage collector from scratch …
  - First, I'll write the JIT compiler from scratch …

Thank you!


5

Contact Info

Charlie Gracie
OMR GC Architect
crgracie@ca.ibm.com
@crgracie

Mark Stoodley
OMR Project Lead
mstoodle@ca.ibm.com
@mstoodle

John Duimovich
CTO IBM Runtimes
duimovic@ca.ibm.com
@jduimovich

Questions?

Trademarks, Copyrights, Disclaimers

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Additional Important Disclaimers

- THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
- WHILST EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.
- ALL PERFORMANCE DATA INCLUDED IN THIS PRESENTATION HAVE BEEN GATHERED IN A CONTROLLED ENVIRONMENT. YOUR OWN TEST RESULTS MAY VARY BASED ON HARDWARE, SOFTWARE OR INFRASTRUCTURE DIFFERENCES.
- ALL DATA INCLUDED IN THIS PRESENTATION ARE MEANT TO BE USED ONLY AS A GUIDE.
- IN ADDITION, THE INFORMATION CONTAINED IN THIS PRESENTATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM, WITHOUT NOTICE.
- IBM AND ITS AFFILIATED COMPANIES SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.
- NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:
- - CREATING ANY WARRANT OR REPRESENTATION FROM IBM, ITS AFFILIATED COMPANIES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS