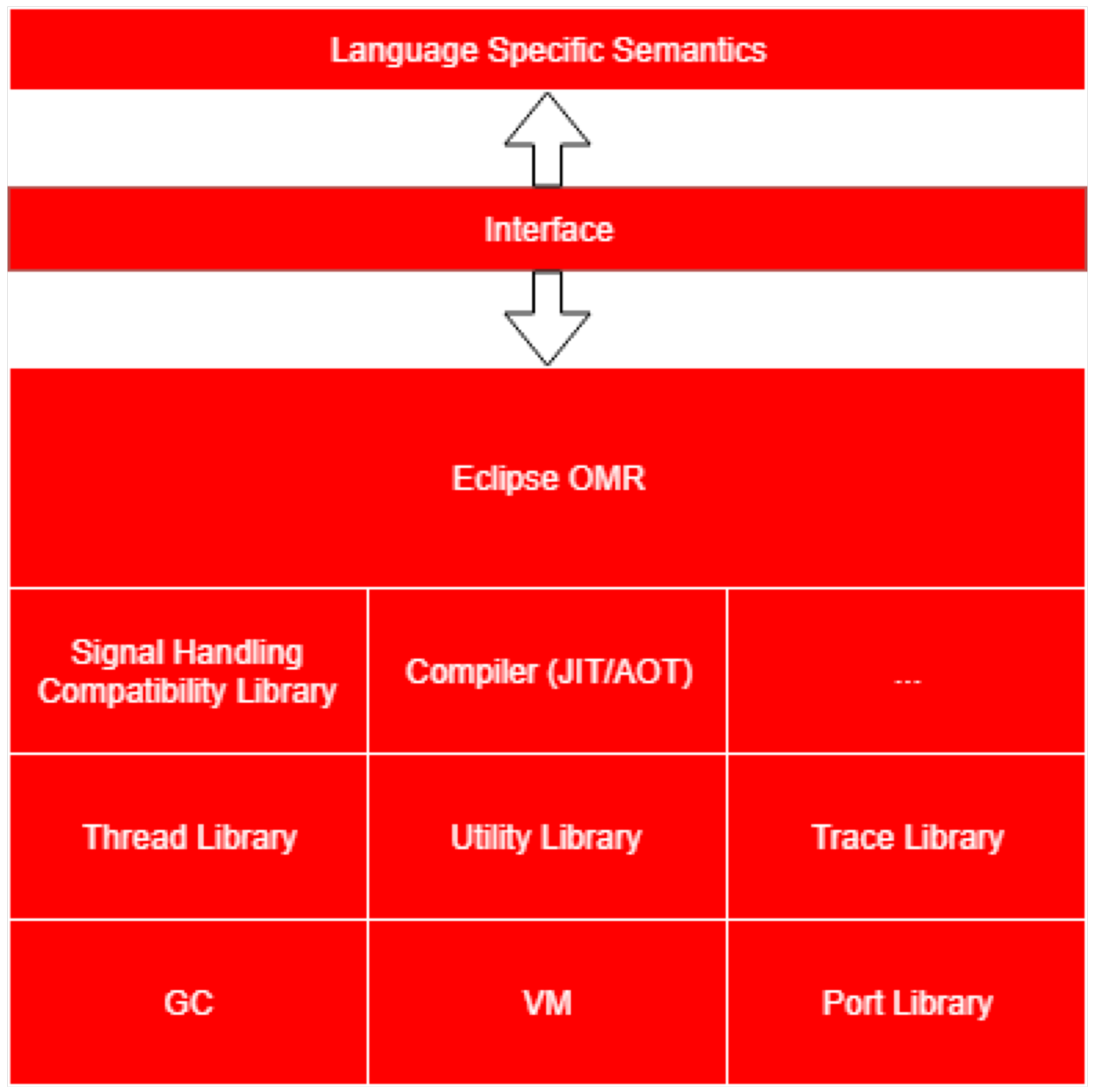
Python on OMR

Dayton J. Allen, David Bremner, DeVerne Jones
University of New Brunswick, Faculty of Computer Science
{dayton.allen, bremner, deverne.jones}@unb.ca
Mark Stoodley, Daryl Maier, Leonardo Banderali
IBM Canada
{mstoodle, maier}@ca.ibm.com, leob@ibm.com

Eclipse OMR & JitBuilder

OMR is an open-sourced project that provides several enterprise-class reusable runtime components to aid in the development of language runtimes.

JitBuilder is a static library with a high-level interface to the Eclipse OMR JIT compiler technology. The library exposes an API that encapsulates the details of the underlying intermediate representation used by OMR.


Language Specific Semantics

Interface

Eclipse OMR

Signal Handling Compatibility Library | Compiler (JIT/AOT) | ...

Thread Library | Utility Library | Trace Library

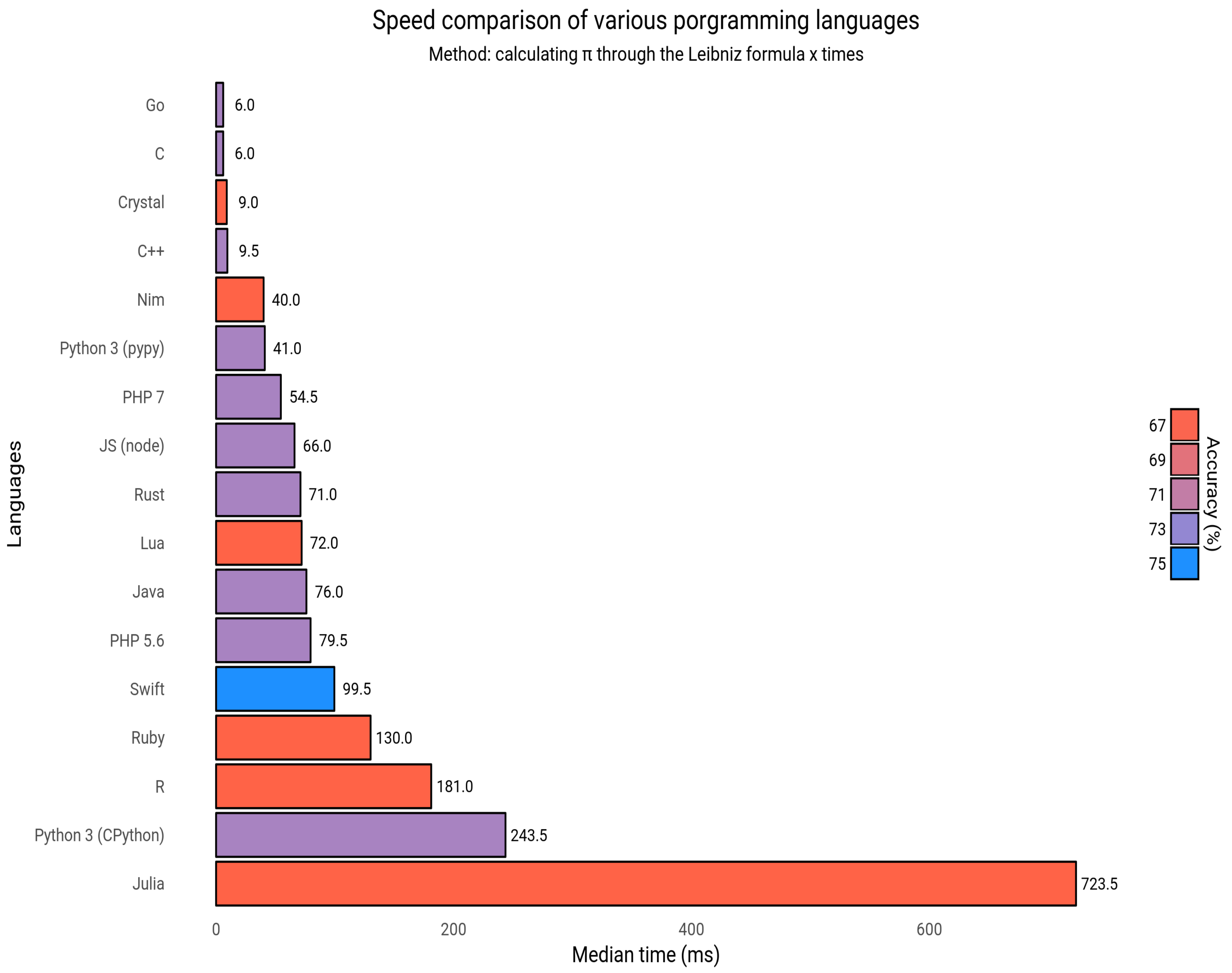
GC | VM | Port Library

Background

- A JIT compiler is a compiler that compiles code at runtime and can be thought of as a middle-ground between static compilation and interpretation, hence, they can be used to gain the best of both worlds.

- Since compilation occurs during runtime, JIT compilers are able to access runtime profile information and consequently make better optimization decisions than a static or ahead-of-time (AOT) compiler.

- Another advantage that JIT compilers have over AOT compilers is that they can perform de-optimization. Meaning, they are able to revert optimizations that result in incorrect or worse performing code.



Problem Statement

- Interpreted languages offer several advantages, among these are ease of use and fast start-up time.

- Despite these advantages, interpreted languages offer subpar performance when compared to their compiled counterparts.
  - For this reason, they are usually excluded from consideration for domains that demand high performance.

- Our goal is to bridge this performance gap by creating a low overhead interface between the CPython runtime and OMR's JIT component. This would allow developers to JIT compile Python methods without leaving the Python environment.

- Secondary goals include evaluating the reusability of OMR’s JIT compilation and Garbage Collection components.

*Graph obtained at https://github.com/niklas-heer/speed-comparison

UNB EST. 1785

IBM Centre for Advanced Studies - Atlantic

FACULTY OF COMPUTER SCIENCE