PyCON HK

Introduction of Graalpython to execute multiple languages with one processing system at high performance

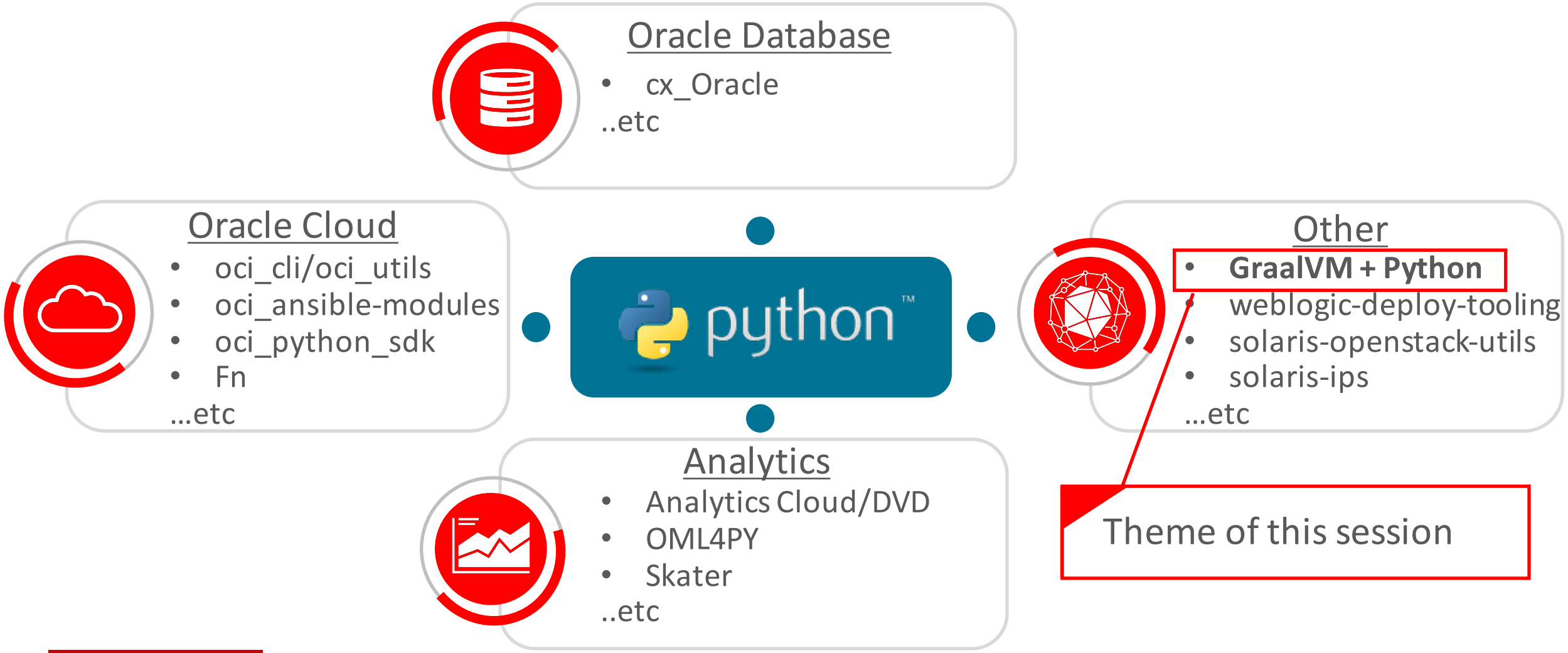
Amitpal Singh Dhillon

Regional Director, Asia-Pacific & Japan
Oracle Labs Research & Advanced Development

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Relationship between Oracle and Python

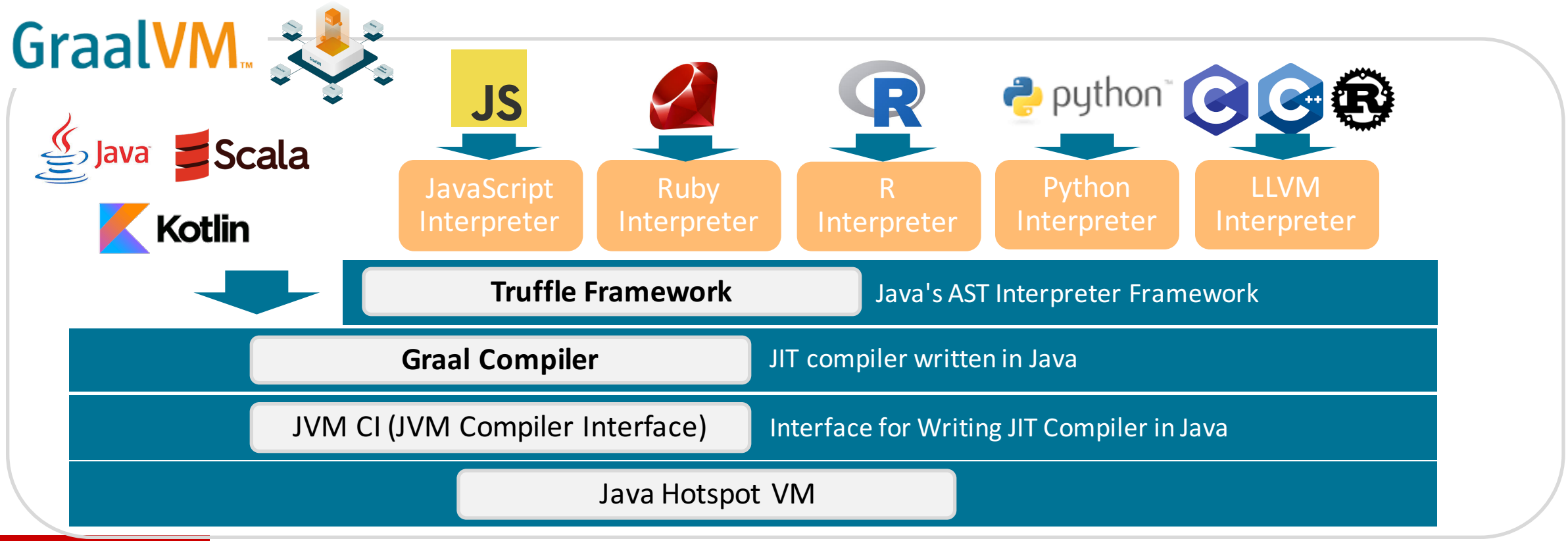


What is GraalVM

- A next generation VM that runs multiple languages fast with a single runtime

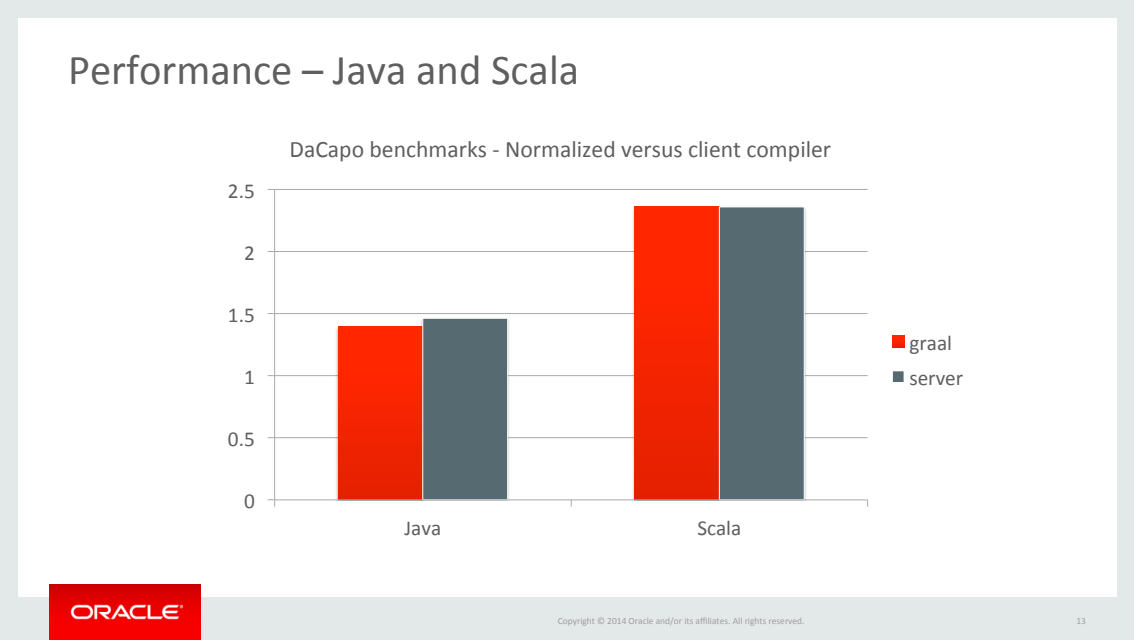
GraalVM Architecture

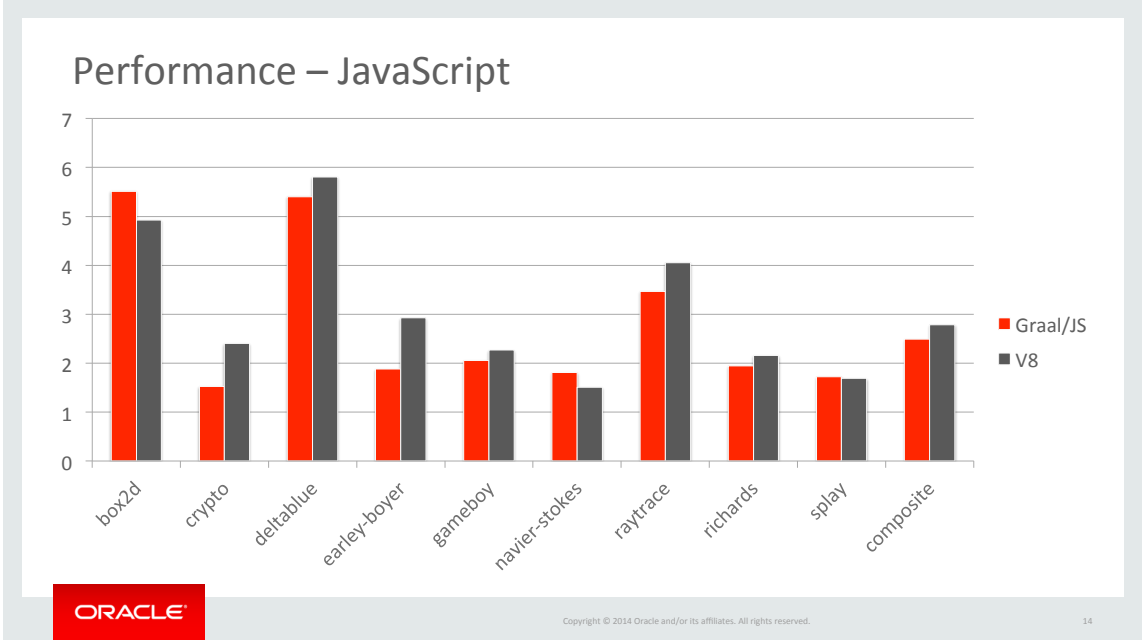
- Graal and Truffle for a High Performance Multilingual Executable VM Environment



GraalVM performance

- Good performance has been reported in various languages (Java / Scala / R / JavaScript)

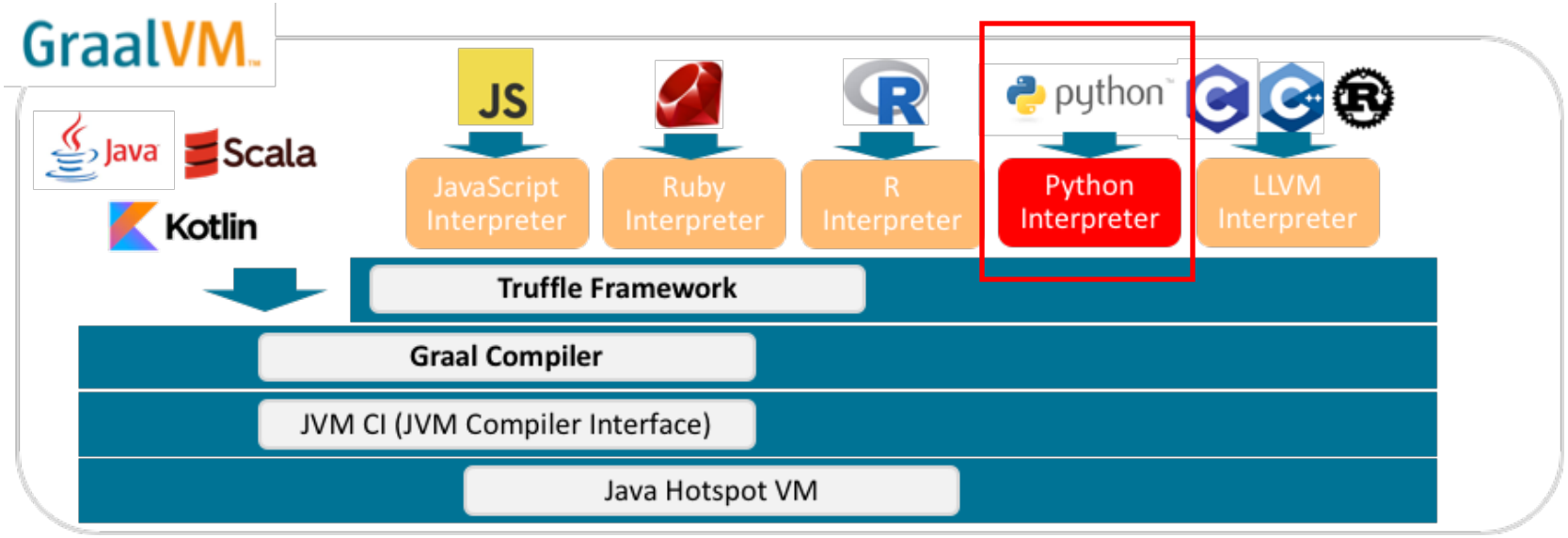




Ref.「**Graal VM: Multi-Language Execution**」**Thomas Wuerthinger**, Senior Research Director at Oracle
URL: https://www.slideshare.net/ThomasWuerthinger/jazoon2014-slides?from_action=save

What is graalpython

- Interpreter for Python 3 implementation built on GraalVM
- At the moment, "**early-stage experimental** implementation of Python"
- Ultimately, achieve full compatibility with Python 3.7 and aim for full support for Scipy and Scipy dependent libraries (Numpy, Pandas, Scikit-learn etc)



graalpython install & run

- Install with gu command of Graal and run GaalVM Python interpreter with graalpython command

Graalpython installation

```
$ gu -c install org.graalvm.python

$ graalpython --version

Graal Python 3.7.0 (GraalVM CE Native 1.0.0-rc15)
```

Run Graalpython

```
# Run a Python script
$ graalpython helloworld.py

# Run Python REPL
$ graalpython
Python 3.7.0 (Thu Apr 04 12:16:00 PDT 2019)
[GraalVM CE, Java 1.8.0_202] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
Please note: This Python implementation is
in the very early stages, and can run little more than
basic benchmarks at this point.
>>>
```

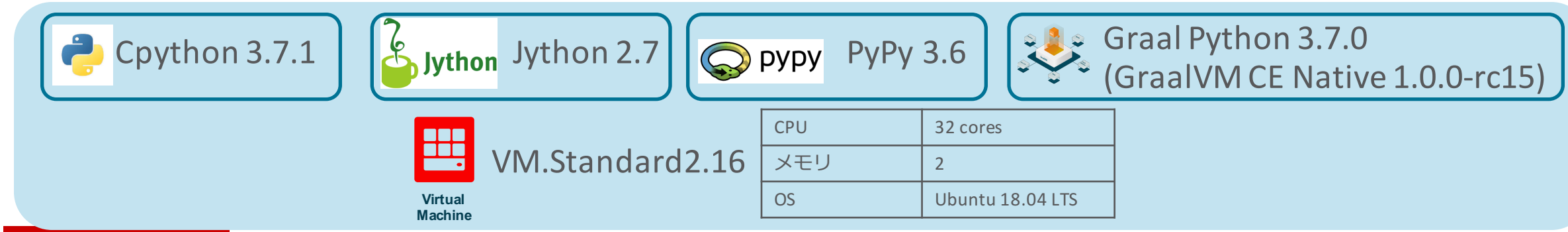
graalpython performance measurement

- In order to verify the high performance which is one of the attractiveness of graalpython, the following two benchmarks are executed on the Compute instance on OCI

  ① Some benchmark scripts run in the pyperfomance (*) library

  ② Python script to perform pi calculation by Monte Carlo method

- Compare three Python implementations (Cpython, Jython, PyPy) with graalpython



[Reference] Python Implementations

Implementation	Latest Version	Standard Python Compatible version	Feature
CPython	3.7.3	(Standard Python implementation)	Python language reference implementation https://www.python.org/
Jython	2.7.0	2.7	Java implementation of Python https://www.jython.org/
PyPy	3.6	3.6	A Python implementation of RPython designed to accelerate CPython https://pypy.org/index.html
graalpython	3.7 GraalVM 1.0.0-rc16	3.7 ※ However, because it is experimental, it is not yet fully compatible	Python implementation that runs on Graal VM https://github.com/graalvm/graalpython
Zippy	3.7	3.7	Python implementation that works on Truffle Framework https://github.com/securesystemslab/zippy

graalpython performance measurement – pyperformance

The following three scripts in pyperformance are slightly modified and executed because there are limitations on libraries that can be operated with graalpython

bm_spectral_norm.py

Script to solve "Hundred-Dollar, Hundred-Digit Challenge Problems"

https://pyperformance.readthedocs.io/benchmarks.html#spectral-norm

bm_nqueens.py

Script that solves N-Queen problem with simple solver

https://pyperformance.readthedocs.io/benchmarks.html#nqueens

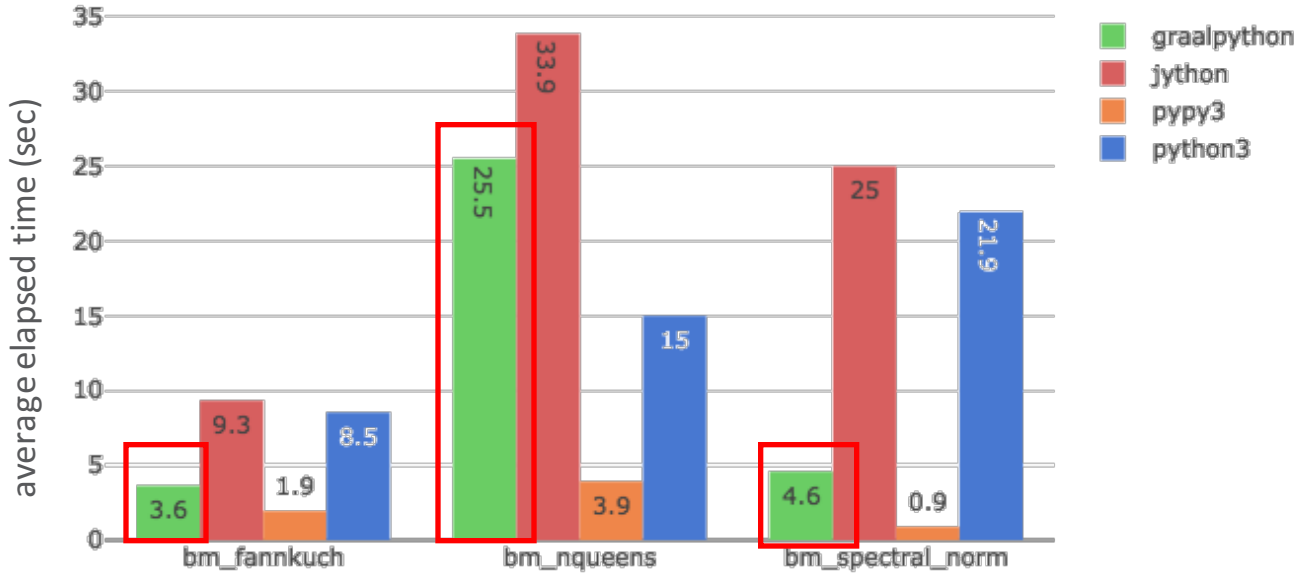
bm_fannkuch.py

Game script for computer language benchmark

https://pyperformance.readthedocs.io/benchmarks.html#fannkuch

Results

pyperformance benchmark tests



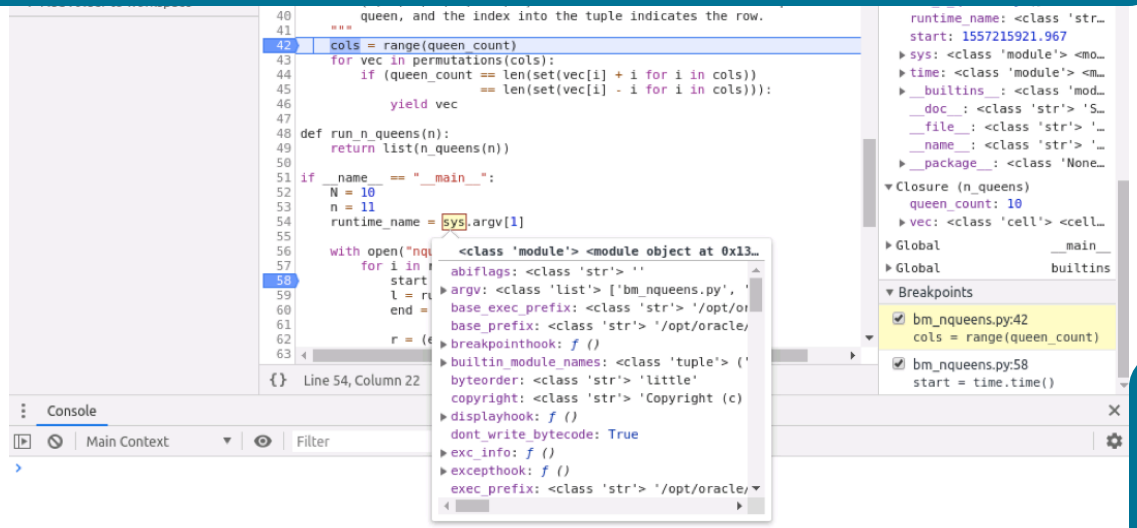
pyperformance tests

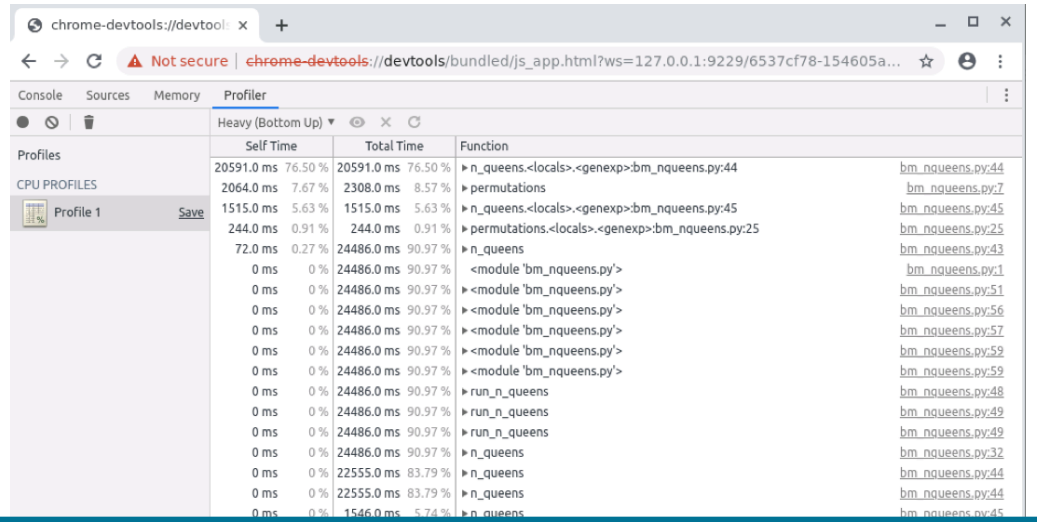
- graalpython outperforms PyPy but gives better performance than Cpython
- ”bm_nqueens.py" uses Generator extensively, and it seems that the execution speed of CPython, which is the standard implementation, is optimized and executed more than graalpython.

[Reference] GraalVM Debugging Tools - chrome-devtools

```
$ graalpython --jvm --inspect <your_python_script_name>.py
Debugger listening on port 9229.
To start debugging, open the following URL in Chrome:
    chrome-devtools://devtools/bundled/js_app.html?ws=127.0.0.1:9229/6537cf78-154605af4255
```

- Settings Break Point
- Executing Python functions on browser console
- Referencing to variables

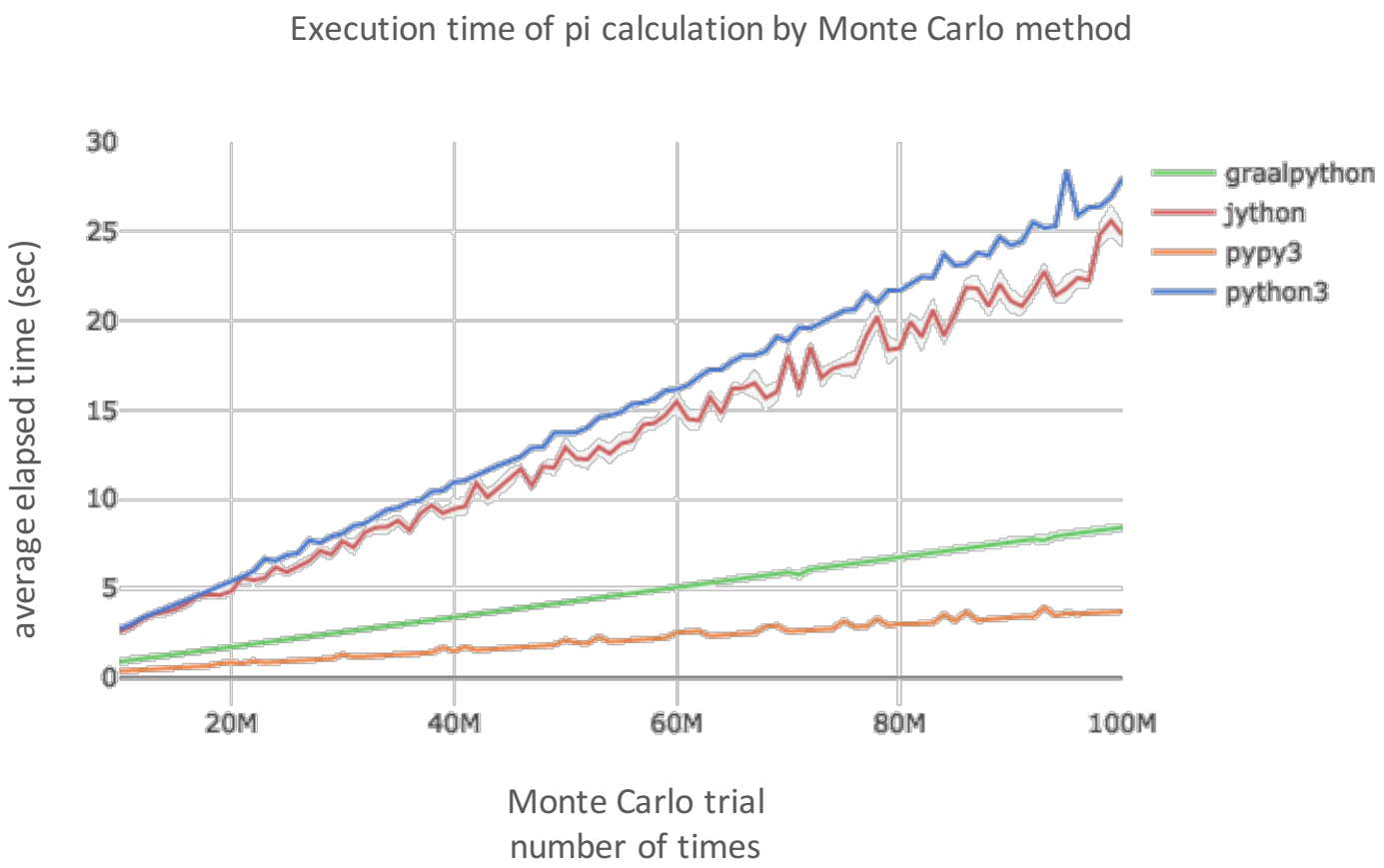




- Profiling Python scripts to see which functions have consumed how much time

graalpython performance measurement-Monte Carlo method 1

Execute script to approximate pi by Monte Carlo method (*)

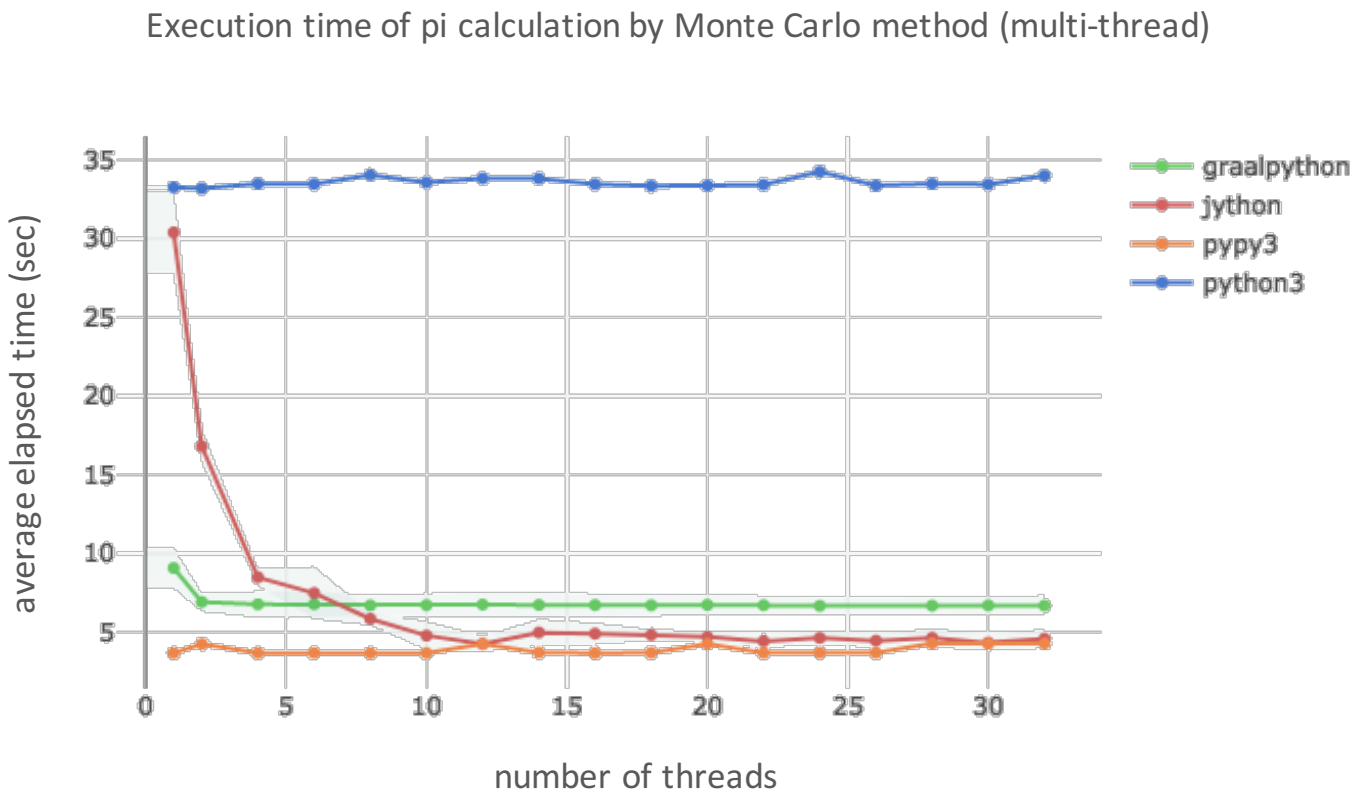


- Even in the Monte Carlo calculation, although it is not comparable to PyPy, the result is higher than CPython and Jython.
- A simple CPU bound process without I / O such as Monte Carlo calculation may be considered as an optimized calculation.

(*) ref. **Monte Carlo method (Wikipedia)**
https://en.wikipedia.org/wiki/Monte_Carlo_method

graalpython performance measurement-Monte Carlo method 2

Execute script that approximates pi by multi-thread (*) by Monte Carlo method

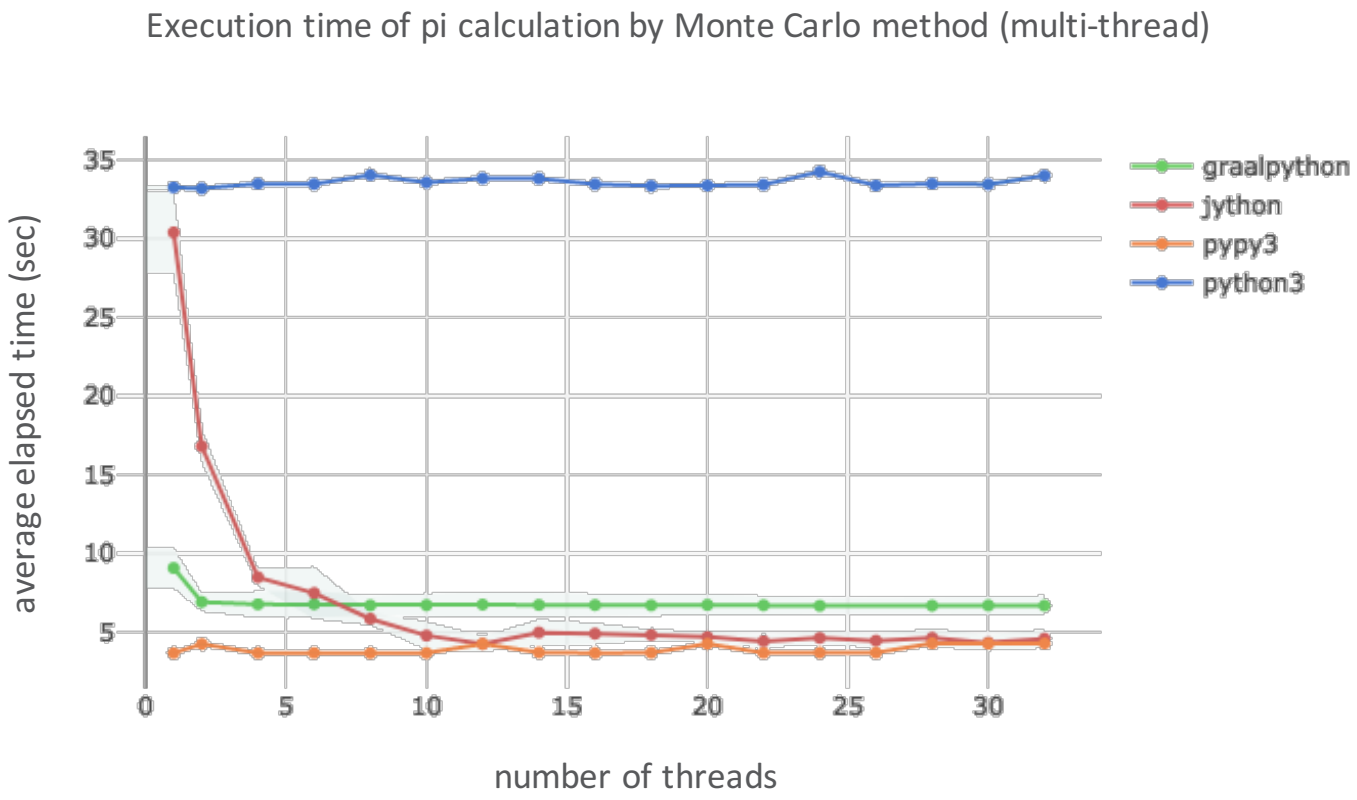


- Graalpython seems to be a processing system with GIL as well as CPython and PyPy, because the processing time hardly changes even if the number of threads is increased.
- On the other hand, Jython that does not have GIL has a shorter processing time as the number of threads increases, and it can be seen that multi-thread execution can be performed with multiple cores.

(*) Implementation using threading's threading.Thread class, which is a Python standard library
https://docs.python.org/3/library/threading.html#threading.Thread

graalpython performance measurement-Monte Carlo method 2

Execute script that approximates pi by multi-thread (*) by Monte Carlo method



- Graalpython seems to be a processing system with GIL as well as CPython and PyPy, because the processing time hardly changes even if the number of threads is increased.
- On the other hand, Jython that does not have GIL has a shorter processing time as the number of threads increases, and it can be seen that multi-thread execution can be performed with multiple cores.

(*) Implementation using threading's threading.Thread class, which is a Python standard library
https://docs.python.org/3/library/threading.html#threading.Thread

graalpython and GIL (Global Interpreter Lock)

- GIL (Global Interpreter Lock) is a protection mechanism to limit the number of threads that can be executed at one time on the interpreter to one
  - It can simplify low-level processing such as memory management and cooperation of C extension
- Graalpython uses GIL in the C extension part (*),
  and multi-thread processing is executed by one CPU core as CPython

Implementation	With or without GIL
CPython	With GIL
Jython	**Without GIL**
PyPy	With GIL ※ However, Software Transactional Memory (STM) can avoid GIL http://doc.pypy.org/en/latest/stm.html
graalpython	With GIL

(*) Ref: https://github.com/graalvm/graalpython/blob/master/graalpython/com.oracle.graal.python.cext/include/pystate.h#L378-L432

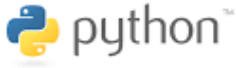
Another attraction of graalpython (Polyglot) 1

- The attraction of graalpython is not only high performance, but also the ability to run Polyglot (multilingual execution) programs easily and without any loss of performance

※ However, as of May 2019 graalpython does not support Scikit-Learn

[Example]
data analysis with R and Python

tidyverse library Modern data shaping used



Machine Learning Model Execution with Scikit-Learn Library

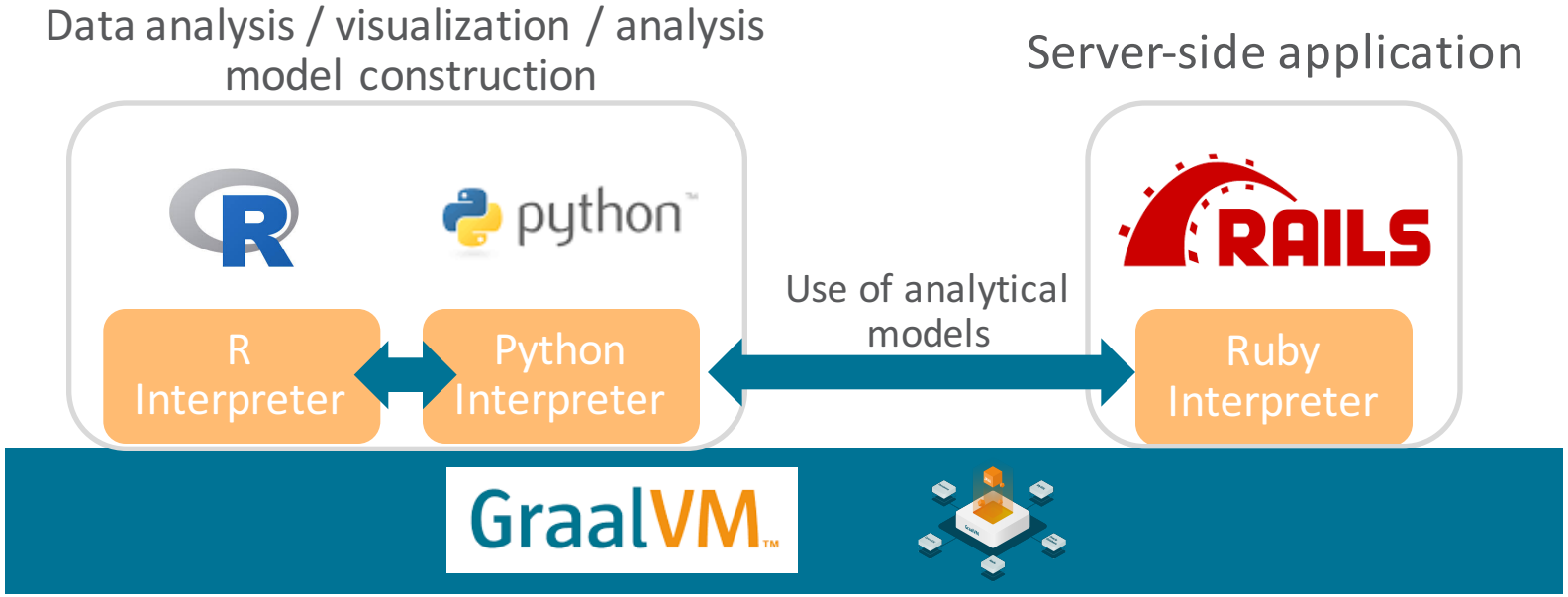
sample_polyglot.py

```
import polyglot
import sklearn

df = polyglot.eval(string="
        library(tidiverse)
        iris %>% dplyr::group_neset(Species) %>% dplyr::summarise_all(mean) …
")
…

clt = sklearn.ensemble.RandomForestClassifier()
clt.fit(df["X"], df["y"]
clf.feature_importances_
```

Another attraction of graalpython (Polyglot) 2

- Furthermore, combining with JavaScript and Ruby makes server-side collaboration easy and reduces the restriction on programming language selection for creating applications

- For example, the following machine learning model embedded application can be configured in the future



Summary

- graalpython is currently an experimental implementation, but performs better than CPython (the standard Python implementation)
- Not only performance but also Polyglot is one of the attractiveness of graalpython
  - In the area of Analytics, which is the main usage scene of Python, it is also possible to analyze combining R and Python, for example
  - It is possible to use different languages according to the characteristics of the program on a single Graal VM

Try graalpython ! GraalVM + python