ORACLE®

Modern Stored Procedures Using GraalVM Oracle Labs

Matthias Brantner < matthias.brantner@oracle.com>



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



Stored Procedures & UDFs





Problems

ORACLE

- Often vendor specific languages (e.g., PL/SQL, Transact-SQL)
- Hard to find developers (\$\$\$)
- Relatively small ecosystems & marketplaces
- Lacking behind with tool support
- Hard to manage within VCSs



JavaScript UDF Demo



Features

- High-performance JavaScript / TypeScript
- MySQL and Oracle Database
- Driver for executing SQL (also ORM support)
- Support for querying JSON tables
- User-defined JavaScript functions
 - Scalar UDFs
 - Aggregation
 - Table functions
- Deployment tool (allows for integration with JS CIs)

ORACLE

Proliferation of Languages

Programming Language Market Share (TIOBE 8/17)



Embedding New Stored Procedure Languages is Hard

- Choose or implement
- Integrate
 - Integrate with query runtime for UDFs
 - Access to data (no copy, data conversion)
 - Manage & secure system resources
 - Provide driver for executing SQL
 - Provide tooling
- Maintain

Repeat for each new language

State-of-the-Art Language Implementation

Prototype a new language

Parser and language work to build syntax tree (AST), AST Interpreter

Write a "real" VM

In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, ...

People complain about performance

Define a bytecode format and write bytecode interpreter

Performance is still bad

Write a JIT compiler Improve the garbage collector

ORACLE

Truffle & Graal

Prototype a new language

Parser and language work to build syntax tree (AST), AST Interpreter

Truffle

Optimize AST via profiling and node rewriting

Graal

Just-in-time compile using partial evaluation (first Futamura projection)



Profiling, Node Rewriting and Compilation



Truffle



Graal

Deoptimization, Node Rewriting and Recompilation





Embedding GraalVM



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JavaScript Performance (Octane 1.0 benchmark suite)

Speed-up normalized vs Nashorn JDK9, higher is better



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

R Performance

Speedup over latest GNU R version on simple benchmarks







Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Specialization of Data Conversion (Oracle Number => IEEE 754 double)



Self-rewriting conversion code

specialize(length, exp, mantissaBytes);

if (length == 2 && exp == 0xc1) { result = mantissaBytes[0] - 1;}
else specialize(length, exp, mantissaBytes);

if (length == 2 && exp == 0xc1) { result = mantissaBytes[0] - 1;}
else if (length == 3 && exp == 0xc2) {
 result = (mantissaBytes[0] - 1) * 100 + (mantissaBytes[1] - 1;});}
else specialize(length, exp, mantissaBytes);

if (length == 2 && exp == 0xc1) { result = mantissaBytes[0] - 1;}
else if (length == 3 && exp == 0xc2) {
 result = (mantissaBytes[0] - 1) * 100 + (mantissaBytes[1] - 1;});}
else genericConversion(length, exp, mantissaBytes);

Conclusion

- Stored procedures and their challenges
- Demo of JavaScript UDF development and deployment
- Proliferation of languages
- GraalVM to simplify implementation of languages
- GraalVM embedded in various data processing platforms
- Speculative optimization for data conversion

Thank You! Questions?

matthias.brantner@oracle.com



Integrated Cloud Applications & Platform Services



ORACLE®