An OMR JIT for Lua

A Case Study in Simplified IL Generation Using OMR JitBuilder

Leonardo Banderali
OMR Intern
IBM Runtimes
leob@ca.ibm.com

About Me

- My name is Leonardo Banderali
- I’m a Software Engineering student from the University of Calgary
- Currently on internship at IBM on the OMR team

Overview

- Eclipse OMR Recap
- Current State of Compiler Development
- JitBuilder
- Lua Vermelha
- Challenges
- Current and Future Work

Eclipse OMR

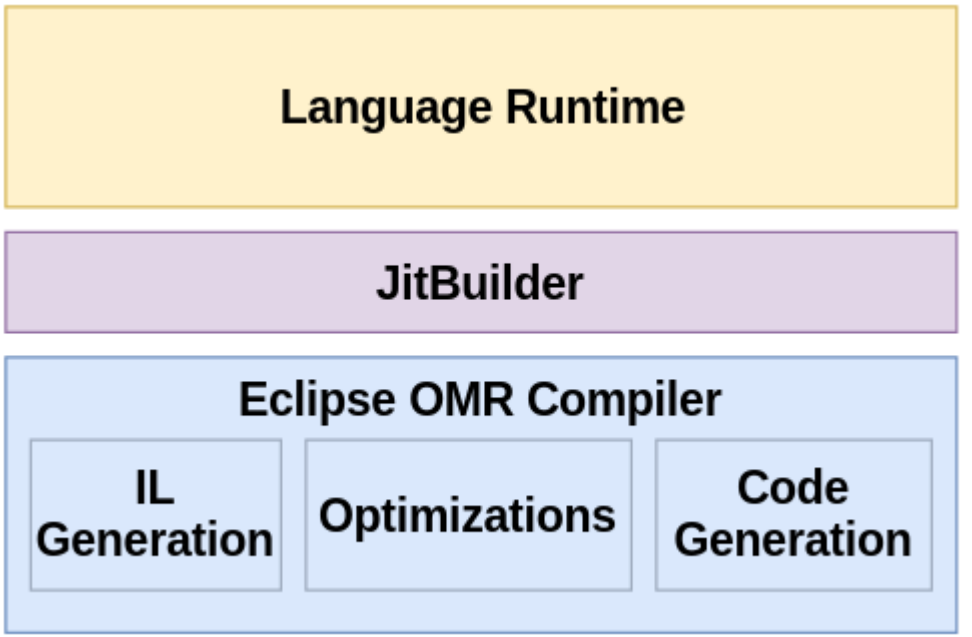
- A toolkit of cross platform components for building language runtimes
  - Garbage Collector
  - Compiler
  - JitBuilder
- Developed from core of IBM Java Virtual Machine implementation
- Generalized to support other languages

Current State of Compiler Development

- Difficult
- Expensive
  - Lots of boilerplate code
  - Generating IL is language specific, error prone
- Eclipse OMR compiler technology can help but
  - Developers must make many decisions
  - Using it requires non-trivial knowledge of technology

JitBuilder

- High level interface to Eclipse OMR compiler technology
- Simplify JIT development
  - Make it easier and faster



What JitBuilder Provides

- Sane defaults
  - E.g. JIT startup and shutdown, various optimizations, code generation
  - May not be optimal but suitable for most use cases
- Declarative interface for generating IL

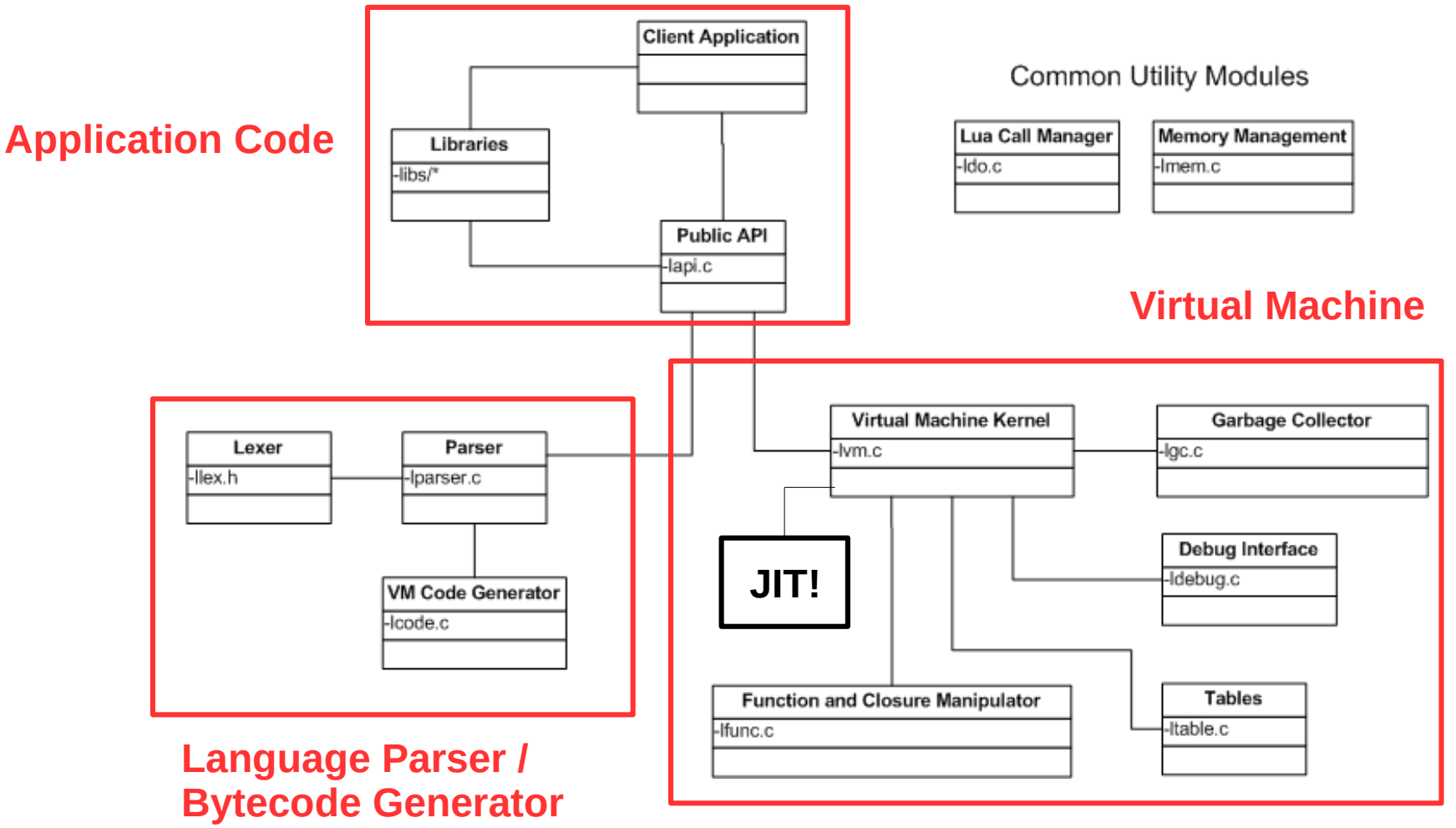
Lua Vermelha

- A proof of concept for JitBuilder
- Objectives
  - Gain experience using JitBuilder
  - Help JitBuilder mature
  - Show that JIT creation can be simplified
- Create a JitBuilder JIT for PUC-Rio Lua
  - Integrate JIT into existing VM
  - Keep changes to the VM minimal

Why (PUC-Rio) Lua

- Popular scripting language
  - Many use cases
- About PUC-Rio VM
  - Standard/Most used Lua implementation
  - Register-based bytecode interpreter, but no JIT
  - Written in C
- Different from all other VMs currently using OMR compiler technology

PUC-Rio Lua VM Architecture



Glasberg, M. S., & Bresler, J. (2006). The Lua Architecture. *Advanced Topics in Software Engineering*.

VM Integration

- JitBuilder startup and shutdown functions wrapped in C functions
  - Called in creation/destruction of Lua State object
- JIT dispatch done directly from interpreter
  - On CALL instruction
  - JIT invoked
  - If function is compiled, execute compiled body
  - Else interpret

```
LUA_API lua_State *lua_newstate (/* ... */) {
/* ... */
if (luaD_rawrunprotected(L, f_luaopen, NULL) != LUA_OK
    || luaJ_initJit() != 0)
```

```
LUA_API void lua_close (lua_State *L) {
  luaJ_stopJit();
  /* ... */
```

```
  /* JIT dispatch */
  lua_JitFunction f = p->compiledcode;
  if (!f)
    f = luaJ_compile(p);
  if (f)
    f(L);  // L is lua_State object (holds VM state)
```

JIT Implementation

- Attempt to mimic behaviour of interpreter
- Create JitBuilder representation equivalent to VM implementation

```
TValue *rb = k + GETARG_Bx(instruction);

// ~~~~~~~~~~~~~~~~~~~~~~~~~~ becomes ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

auto arg_b = GETARG_Bx(instruction);
builder->Store("rb",
builder->      IndexAt(typeDictionary()->PointerTo(luaTypes.TValue),
builder->              Load("k"),
builder->              ConstInt32(arg_b)));
```

Interfacing with the VM

- JITed code must be able to interact with objects allocated by VM
- JitBuilder facility for defining structs

```
typedef struct LClosure {
  GCObject *next;
  /* ... */
  UpVal *upvals[1];
} Lclosure;
```

```
TR::IlType pGCObject_t = Address;
TR::IlType pAddress = PointerTo(Address);
// ...

luaTypes.LClosure = DefineStruct("LClosure");
DefineField("LClosure", "next", pGCObject_t);
/* ... */
DefineField("LClosure", "upvals", pAddress);
CloseStruct("LClosure");
```

Challenges

- Inter-operating C and C++
  - C++11 in particular
- Unravelling complex macros in VM
- JitBuilder incompatibilities with C/C++ ABI
  - Struct member alignment (compiler specific)
- No JitBuilder implementation of unions

Current and Future Work

- Improving Lua Vermelha JIT
  - Support more instructions/opcodes
  - Improve performance
- Improving JitBuilder
  - Existing API
  - New features
  - Better support for overlaying structs
  - Unions

Thank You!

Eclipse OMR

http://www.eclipse.org/omr

https://github.com/eclipse/omr

https://developer.ibm.com/open/omr

Mark Stoodley (Creator of JitBuilder)

- IBM OMR Project Lead
- mstoodle@ca.ibm.com
- GitHub: mstoodle

Leonardo Banderali (Me)

- IBM OMR Intern
- leob@ca.ibm.com
- GitHub: Leonardo2718

Related Works

- OMR compiler based JITs
  - Ruby
  - Python
- JitBuilder based JITs
  - SOMpp
- Lua JITs
  - LuaJIT project
  - llvm-lua

About Lua

- Scripting language
- Embeddable language
- Dynamically typed
- Garbage collected
- Fast (?)
- Different implementations exist
  - PUC-Rio Lua, LuaJIT, etc.

Other Challenges

- Initial attempt to hook up to the C API failed