Introduction to Virtual Machines with VMKit

Harris Bakiras,

Supervisors: Gaël Thomas, Gilles Müller
LIP6 REGAL TEAM – INRIA (Paris/France)
VMKit
a virtual machine substrate
Execution environments

Native code execution (C, C++)
- Source code
  - Compiler
    - Analyse
    - Optimise
    - Generate target specific binary code
  - Target specific binary code
- execution

Virtual machine execution (Java, C#)
- Source code
  - Compiler
    - Analyse
    - Generate generic byte-code
  - Generic Byte-code
- Virtual Machine
  - Byte-code analysis
  - Byte-code evaluation or
  - Just in time Compilation and optimization into native code (target specific)
- execution
Virtual Machines

Problems

Developing time is extremely long!

How to test an idea with different languages?

How to implement a new efficient virtual machine for new languages?

How to quickly extend existing languages?
VMKit: a substrate of virtual machine

VMKit's goal: help experiments on VM

Objective: factorize VM's common components

- Just In Time Compiler: native code generation on the fly
- Memory manager: allocates and collects automatically free memory
- Thread manager: creates and synchronizes threads

Java Personality (J3)  C# Personality (N3 not maintained)  R Personality (Reactor)

VMKit

Just In Time Compiler  Memory Manager  Thread Manager

Personalities

Harris Bakiras
VMKit: a substrate for Managed Runtime Environments
VMKit: a substrate of virtual machine

VMKit's goal: help experiments on VM

Objective: factorize VM's common components

- Just In Time Compiler: native code generation on the fly
- Memory manager: allocates and collects automatically free memory
- Thread manager: creates and synchronizes threads

Java Personality (J3)  C# Personality (N3 not maintained)  R Personality (Reactor)

VMKit

Just In Time Compiler  Memory Manager  Thread Manager

~ 25k loc  ~ 500k loc

Harris Bakiras  VMKit: a substrate for Managed Runtime Environments
VMKit: a substrate of virtual machine

VMKit's goal: help experiments on VM

Objective: factorize VM's common components

- Just In Time Compiler: native code generation on the fly
- Memory manager: allocates and collects automatically free memory
- Thread manager: creates and synchronizes threads

Java Personality (J3)          C# Personality (N3 not maintained)          R Personality (Reactor)

~ 25k loc

< 5% of the code

VMKit

Just In Time Compiler          Memory Manager          Thread Manager

~ 500k loc
VMKit
from a technical point of view
VMKit's Implementation

Implementation choice: relies on external state of the art components

- Just In Time Compiler (JIT): LLVM [Lattner & Adve – CGO’04]
- Memory manager: MMTk [Blackburn et Al. – ICSE’04]
- Thread manager: Posix

VMKit = glue between the different components

- Between JIT-C et memory manager = precise GC
- Between Thread and Memory managers = multi-threaded GC
Detailed Architecture

VMKit: a substrate for Managed Runtime Environments

Harris Bakiras

INRIA
Minimal VM (ToyVM) for the tutorial
ToyVM's Architecture

Personality Skeleton

ToyCompiler
MutatorThread
GVMaterializer (llvm)

ToyThread
Execute Code

ToyVM

VirtualMachine

MutatorThread

GC

Object layout

ToyRoot

VMKit
ToyVM's Architecture

Personality Skeleton

ToyCompiler

MutatorThread

GVMaterializer (llvm)

VirtualMachine

ToyThread

Execute Code

GC

ToyRoot

Object layout
ToyVM ← VirtualMachine

- Thread management
- Garbage collectors entry point
- Backtrace (execution stack browsing)
- Global variables tracing
- Exceptions management

ToyThread ← MutatorThread ← Thread

- Garbage collectors thread synchronization
- Execution stack Scan during GC
- Global variables tracing
- Main execution method

Provided

Developers charge
ToyVM's Architecture

Personality Skeleton

ToyCompiler

MutatorThread

GVMaterializer (llvm)

VirtualMachine

ToyThread

Execute Code

ToyVM

Object layout

ToyRoot

GC
ToyVM's Architecture

Personality Skeleton

ToyCompiler

MutatorThread

GVMaterializer (llvm)

ToyRoot

VirtualMachine

ToyThread

Object layout

Execute Code

GMKit

Harris Bakiras

VMKit: a substrate for Managed Runtime Environments
ToyRoot ← vmkit::gc
- Tag collectible objects (stack maps)

ToyRoot* F (ToyRoot* param) {
    TOY_PARAM(param);
    TOY_VAR(ToyRoot, val);
    val = g(param, val);
    return val;
}

ToyRoot* G (ToyRoot* a, ToyRoot* b) {
    TOY_PARAM(a);
    TOY_PARAM(b);
    TOY_VAR(ToyRoot, res);
    res = a.doSomething(b);
    return res;
}
ToyRoot (tag)

ToyRoot ← vmkit::gc
- Tag collectible objects (stack maps)

```
[ ....... ]
ToyRoot* F(ToyRoot* param) {
    TOY_PARAM(param);
    TOY_VAR(ToyRoot, val);
    [ init val ... ]
    val = g(param, val);
    return val;
}
```

```
Stack Maps
```

```
ToyRoot* G(ToyRoot* a, ToyRoot* b) {
    TOY_PARAM(a);
    TOY_PARAM(b);
    TOY_VAR(ToyRoot, res);
    res = a.doSomething(b);
    return res;
}
```

```
[ ....... ]
```
ToyRoot ← vmkit::gc
- Tag collectible objects (stack maps)

Stack Maps

Call stack

Parameters

Local vars

Parameters

Local vars

ToyRoot* F (ToyRoot* param) {
  TOY_PARAM(param);
  TOY_VAR(ToyRoot, val);
  [ init val … ]
  val = g(param, val);
  return val;
}

ToyRoot* G (ToyRoot* a, ToyRoot* b) {
  TOY_PARAM(a);
  TOY_PARAM(b);
  TOY_VAR(ToyRoot, res);
  res = a.doSomething(b);
  return res;
}

[ ……. ]
ToyRoot (tracer)

ToyRoot ← vmkit::gc
- Tag collectible objects (stack maps)
- Object tracing method
ToyRoot (tracer)

ToyRoot ← vmkit::gc
- Tag collectible objects (stack maps)

- Object tracing method

Stack Maps

Call stack

stack

param

tracer

val

tracer

ToyRoot 1

markAndTrace

markAndTrace

markAndTrace

ToyRoot 2

ToyRoot 3

F

tracer
ToyRoot (allocation)

ToyRoot ← vmkit::gc

- Tag collectible objects (stack maps)

- Object tracing method

- GC objects allocation
  - Override `operator new`
  - Call to `new forbidden` for gc objects (opaque parameter)
ToyVM's Architecture

Personality Skeleton

ToyCompiler

ToyThread

ToyVM

GVMaterializer (llvm)

MutatorThread

VirtualMachine

GC

Object layout

Harris Bakiras

VMKit: a substrate for Managed Runtime Environments
ToyVM's Architecture

Personality Skeleton

ToyCompiler

MutatorThread

GVMaterializer (llvm)

ToyThread

Execute Code

ToyVM

VirtualMachine

GC

Object layout

ToyRoot

Harris Bakiras

VMKit: a substrate for Managed Runtime Environments
ToyCompiler $\leftarrow$ GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)
ToyCompiler (load IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)
ToyCompiler (load IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)
ToyCompiler (load IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)
ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)
  ▫ JIT compiling

```
module

IR
toyVM

ToyCompiler

pass manager

execution engine

ToyThread

execute(...) {
  ...
  func = compiler.jitCompile(...);
  func(...);
  ...
}
```
ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

ToyCompiler

module

IR

ToyVM

pass manager

execution engine

ToyThread

jitCompile

LLVM func

No body

execute(...)

{
    ...
    func = compiler.jitCompile(...);
    func(...);
    ...
}
ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

ToyCompiler

module

IR

ToyVM

Materialize

GenerateCode

pass manager

execution engine

ToyThread

execute(...)

{  
  ...  
  func = compiler.jitCompile(...);
  func(...);
  ...

}
ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

• JIT compiling

```
ToyCompiler

module

IR toyVM

GenerateCode

LLVM func No opt

pass manager

execution engine

Materialize

jitCompile

ToyThread

execute(...) {
    ...
    func = compiler.jitCompile(...);
    func(...);
    ...
}
```

Harris Bakiras

VMKit: a substrate for Managed Runtime Environments
ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

ToyCompiler

module

IR

toyVM

Materialize

GenerateCode

LLVM func

opt

pass manager

execution engine

jitCompile

ToyThread

execute(...)

{
  ...
  func = compiler.jitCompile(...);
  func(...);
  ...
}
ToyCompiler (optimize IR)

ToyCompiler $\leftarrow$ GVMaterializer

- Retrieving ToyVM's IR (LLVM intermediate representation)
  - JIT compiling

```
execute(...)
{
  func = compiler.jitCompile(...);
  func(...);
}
```

IR

ToyCompiler

Materialize

pass manager

ToyThread

execute(...)
{
  ...  
  func = compiler.jitCompile(...);
  func(...);
  ...  
}

LLVM func

toyVM

opt

module

IR

toyVM

 execution engine

LLVM func

No body
ToyCompiler (optimize IR)

ToyCompiler ← GVMaterializer
  Retrieving ToyVM's IR (LLVM intermediate representation)
  ✔ JIT compiling

ToyCompiler

module

IR

toyVM

pass manager

 execution engine

jitCompile

LLVM func opt

ToyThread

execute(...)
{
  ...
  func = compiler.jitCompile(...);
  func(...);
  ...
}
ToyCompiler (IR to native)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)
- JIT compiling

ToyCompiler

module

IR

toyVM

ToyThread

execute(...) {
    ...
    func = compiler.jitCompile(...);
    func(...);
    ...
}

pass manager

native func

opt

LLVM func

opt

execution engine
ToyCompiler (IR to native)

ToyCompiler ← GVMaterializer
  Retrieving ToyVM's IR (LLVM intermediate representation)
  ✓ JIT compiling

ToyCompiler

module

IR

toyVM

pass manager

execution engine

ToyThread

execute(...) {
  ...
  func = compiler.jitCompile(...);
  func(...);
  ...
}
ToyCompiler

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling
  - Generation of IR
  - Optimization of IR
  - Conversion of IR into native code
Questions ?
Let's code!

VMKit: a substrate for Managed Runtime Environments

Harris Bakiras