Towards Efficient Dynamic Role Dispatch and a Role Type System

By Lars Schütze
Review Benchmarks 1st TAB

Heap Pressure Benchmark

4 – 5 orders of magnitude slower

Do not scale

Memory Overflow

Runtime Factor normalized to ROP (lower is better)

# Transactions

2500 10000 250000 1000000 2250000 4000000

OT/J (callin) OT/J (callout) Scroll LyRT
Review Outlook 1st TAB

- Graal JIT compiler
  - Partial Evaluation
  - Hotspot VM: No custom code generation

- Dispatch in ObjectTeams
  - Indirections: mapping roles to OOP
  - Role-aware control- and dataflow analysis

- Truffle AST interpreter
  - Self-specialization and Partial Evaluation
  - Profiling in DSL
  - Emphasizes need for a type system for roles
Research Questions

- **Dispatching Functions of Role-playing Objects**
  - What impact have **role-features** on application performance?
  - How can a compiler **detect** these features and generate **performant** code?
  - Requirements for **fast dispatch** for role-playing objects?

- **Type System for Roles**
  - What are **types of role-playing** objects?
  - How can we **capture** this in a type system?
  - How can a compiler/runtime use that information for **specialization**?
Dynamic Dispatch in Object Teams
Object Teams Indirections

- Same code pattern always used

```java
class Role playedBy BaseClass {
    void roleMethod() {
        ...;
        base.someMethod();
    }
}
```

Call all bindings:
```
TeamManager:
getTeamsAndCallinIds

_bound_method_id, callin_id, joinpoint_id
switch (id):
    case ...
        buildStack1();
        LIFTING();
        ParameterMapping();
        RoleMethod1();
    case ...
        buildStack2();
        RoleMethod2();
```

Indirections everywhere:
- `callAllBindings`
- `callBefore`
- `callReplace`
- `callAfter`

```
someMethod
```

Diagram:

- `callAllBindings`
- `getTeamsAndCallinIds`
- `_OT$callAllBindings`
- `callBefore`
- `callReplace`
- `callAfter`
- `callOrig`
- `callOrig`
ObjectTeams Dispatch

- **Compiler Problems**
  - No use-def graph for Teams, Roles and Base
  - Open world: Never know everything

- **Dispatching Problems**
  - Recursion may terminate program
  - Huge context required by functions
  - Concumes lots of inlining budget of JVM

- **Runtime Problems**
  - Representation of data types
  - High pressure on garbage collector

```java
class Role {
    playedBy BaseClass {
        void roleMethod() {
            ... base.someMethod();
        }
        void roleMethod() { replace someMethod();
    }
}
```
ObjectTeams Dynamic Dispatch

- **Runtime Evaluated Invocation**
  - Bytecode instruction
  - JVM function pointers
  - Type safe

- **Bootstrap**
  - Predefine type of call target
  - Push parameters on stack

- **Self-adaptive Callsite**
  - Change target dynamically
  - Invalidate call target externally

---

**Reasons**
- Family Polymorphism
- Implicit Inheritance

---

**Same observation in**
- SCROLL
- NextEJ / EpsilonJ
- Role4J
Benchmark Results Dynamic Dispatch

- **Benchmarks**
  - Two consecutive loops over List of Accounts
    - 2.6x speedup
    - Lots of inlining potential
  - Random access to List of Accounts
    - 1.6x speedup
  - Dynamic benchmark: many invalidations
    - 7x – 10x slowdown

- **Potential improvements**
  - Polymorphic Inline Caches (PIC)
  - LRU cache for MethodHandles
General Observations

- Observation
  - Player-role data-structures are inadequate

- Reason
  - Types may be different per object
  - Objects have multiple different types simultaneously
  - Hard to capture in static class-centric object-oriented world

Is everything a role?

**No**, could also be Phases, Facets, Parts, …
Towards a Type System for Roles
Why a Role Type System?

What could go wrong?
- Application may crash
- Undefined behavior

Problem
- Types differ per role-playing object
- Object types control-flow dependent
- Ambiguous function calls

Solution
- Static Analysis
- Role-play graph
- Dispatch Description Language
- Type System

---

Requirements for a Role Type System

- **General Questions**
  - Type of a role-playing object?
  - Role types, compartment types and natural types
    - Relation?
    - Special types?
  - Ambiguity resolution?

- **Requirements**
  - Detect ambiguous function calls
  - Dependent Types
  - Intersection Types
  - Decidable
Foundations for a Role Type System

- Dependent Object Types (DOT) [2]
  - Minimal formal foundation for Scala
  - Sound, machine checked
  - Intersection types, dependent types

\[ \Gamma \vdash d_1 : T_1 \quad \Gamma \vdash d_2 : T_2 \]
\[ \text{dom}(d_1) \cap \text{dom}(d_2) = \emptyset \]
\[ \Gamma \vdash d_1 \land d_2 : T_1 \land T_2 \]

Allow intersections with joint labels.
Provide Total Order.

Outlook
Future Work

Read, Write, Collaborate

10.2016

LASSY

Benchmark

10.2017

2nd TAB

Faster dispatch in Object Teams

10.2018

Type System for Roles

Role features and their runtime behavior

12.2019

Prototype e.g. in Truffle
Backup: Invoke Dynamic ObjectTeams
• Callin ≡ Advice as in aspect-oriented programming
• Modifiers before, after, replace

```java
class Role playedBy BaseClass {
    void roleMethod() { ...; base.someMethod(); } // replace
    void someMethod() { replace roleMethod(); }
}
```
Same code pattern always used

```c
class Role playedBy BaseClass {
    void roleMethod() {
        ...; base.someMethod();
    }
    void roleMethod() ← replace someMethod();
}
```

**Indirections everywhere:**

```c
bound_method_id, callin_id, joinpoint_id
switch( id ):
    case ...
    buildStack1();
    LIFTING();
    ParameterMapping();
    RoleMethod1();
    case ...
    buildStack2();
    RoleMethod2();
```