TypeProfile

A *type profile* is information which summarizes the type of some value at some *profile point* in the program being executed by the JVM. The summary is designed to allow the optimizing compiler to guess at future types at the same point in the program. In many cases, these profiles are necessary for attaining best code quality.

A profile point is a specific instance of a bytecode. Not all bytecodes perform profiling.

The type recorded at a profile point is the concrete class of an object operand to the bytecode. (I.e., it is the `_klass` object header field, of type `Klass*`.)

There is also an indication if nulls have been seen at the profile point.

Here are the bytecodes with their profiled operands:

Profile Point Bytecode	Type Profiled Operand
invokevirtual, invokeinterface	receiver
checkcast, instanceof	tested object
aastore	element value

Primitive values are not currently profiled, but see bug 8015418. Non-receiver arguments and return values are not currently profiled, but see bug 6919064.

Type profile structure

For background on profiles, see MethodData.

Within a `MethodData` block, each original instance of a type-profiled bytecode instruction (invokevirtual, checkcast, etc.) gets a `ReceiverTypeData` recor d. Each record has a few rows (`TypeProfileWidth`, default 2), each of which contains a count and an exact object type. The record also has an overall count, which may differ from the sum of the individual counts.

Failure modes

Type profiles have two failure modes: First, a method might be compiled before its profile exists and is "mature", so that no stable conclusions can be drawn about operands in that method. Second, a method might be used from many different contexts with independent operand types (as with `ArrayList .contains`), so that the profile becomes "polluted" by many independent types. A polluted profile contains the first few (2) encountered types with low counts, and a high total count that signals that the `ReceiverTypeData` structure was too small.

Polluted profiles stem from the fact that a method (containing generically reusable code) has only one profile structure, but the method is reused from a variety of contexts, providing a variety of operand types. Polluted profiles are discussed here.