CS6004: Code Optimization for Object-Oriented Languages

Spring 2024 (Due: April 30th, 2024)

Assignment PA4: Transform Me If You Can.

---

1 Assignment Objective

Perform any program analysis, use its result to transform Java class files and then measure the improvement within the Java Virtual Machine.

2 Detailed Specification

Your objectives in this assignment are follows:

- **Generating the analysis result**: First you need to generate result for any analysis intended for a particular optimization (eg, Null-Check Analysis to remove redundant null-checks from a program). You can also use the analysis which you generated either in PA2, PA3 or even an existing Soot analysis.
- **Transforming the class files**: Next you need to use the analysis result to transform the class files and generate new class files. For example, suppose you need to transform your code such that all the iadd operations need to be replaced by imul operation. We provide a sample code on the next page which will do that transformation. (Note that this is just for a simple illustration; we do not want you to change program semantics in any form.)
- **Measure the improvement**: Finally you need to run the transformed class files in the JVM and measure the improvement for the transformed code. Improvement can be measured by putting some counters in interpreter's code (eg, count the number of monomorphic callsites) or by measuring the difference in the run time of a program. Note that to see the performance improvement in run time you might have to run your program long enough.
- **Use OpenJ9 as the VM**: We will be using the Eclipse Openj9 JVM for running and measuring the improvements for different programs.
- **Building OpenJ9**:
  1. For building Openj9 VM you can follow the steps at link: `https://github.com/eclipse-openj9/openj9/blob/master/doc/build-instructions/Build_Instructions_V8.md`
  2. Make sure to change the branch of the following repositories before building:
     (a) **openjdk:** `v0.26.0-release`
     (b) **openj9:** `v0.26.0-release`
     (c) **omr:** `v0.26.0-release`

3. The interpreter code is present in the file "`runtime/vm/BytecodeInterpreter.hpp`" file. You would see how different bytecodes are handled and evaluated by the interpreter.
4. Make sure to rebuild your VM every time when you make some changes.

- **Disabling JIT of OpenJ9 VM**: To observe the performance improvement you need to disable the just-in-time compilation while executing your testcases. So in that case only interpretation will happen. You can disable by supplying the flag: `-Xint`.

    1. Example: `$openj9-java -Xint Main` (assuming "openj9-java" aliases to the java binary in the built VM.)

3 Transformation code

```java
@Override
protected void internalTransform(Body body, String phaseName, Map<String, String> options) {
    // Iterate over all units (instructions) in the method body
    PatchingChain<Unit> units = body.getUnits();
    // Iterate over instructions and replace iadd with imul
    Iterator<Unit> unitIt = units.snapshotIterator();
    while (unitIt.hasNext()) {
        Unit unit = unitIt.next();
        if (unit instanceof Stmt) {
            Stmt stmt = (Stmt) unit;
            // Check if it's an iadd operation
            if (stmt instanceof AssignStmt) {
                AssignStmt assignStmt = (AssignStmt) stmt;
                Value rightOp = assignStmt.getRightOp();
                if (rightOp instanceof AddExpr && rightOp.getType() instanceof IntType) {
                    AddExpr addExpr = (AddExpr) rightOp;
                    if (addExpr.getOp1().getType() instanceof IntType && addExpr.getOp2().getType
                        () instanceof IntType) {
                      // Create a new multiplication expression
                      MulExpr mulExpr = Jimple.v().newMulExpr(addExpr.getOp1(), addExpr.getOp2());
                      // Replace iadd with imul
                      assignStmt.setRightOp(mulExpr);
                    }
                }
            }
        }
    }
}
```

4 Submission

You need to submit the report on moodle individually. Your submission must be named `rollnum-pa4.pdf`, where rollnum is your roll-number in small letters.

1. We should be able to reproduce your effort by following the instructions in your report. All your code (Soot as well as OpenJ9) should be in a shared Github repository.

2. We may have a video and/or a viva at the end.

3. You may form groups of two (optional).

5 Plagiarism Warning

You are allowed to discuss publicly on piazza, but are supposed to do the assignment completely individually in your group. If plagiarism is found:

- First instance: 0 marks in the assignment
- Second instance: FR grade in the course
- Third instance: report to institutional committee

-*-*-*- Do the assignment honestly, enjoy learning the course. -*-*-*-