Whole-program Devirtualization Optimizations

Felix Kwok (wkwok@sable.mcgill.ca)

September 5, 2000

This note explains how to use call graph information for whole-program devirtualization optimizations. The user should first be familiar with material in both Soot command-line options and Phase options.

1 Running whole-program optimizations

Soot provides tools for whole-program optimizations in the `wjop` pack. To use these tools, one must run `soot` in whole-program mode and must have turned on optimization. This is accomplished by the command-line option `-W`. Since we want Soot to output all the classes in our application, we should also use the `-app` option.

2 The `wjop` Pack

The `wjop` pack contains two transformers, `StaticMethodBinder` and `StaticInliner`. Only one transformer should be applied for each execution. By default, `StaticMethodBinder` is disabled and `StaticInliner` enabled. This can be changed by setting the `enabled` option for each transformer.

`StaticInliner` (phase `wjop.si`) does the following:

1. finds call sites which are monomorphic;
2. checks whether the call sites can be safely inlined. The inlining criteria are listed in Vijay Sundaresan's Master's thesis;
3. if the call site is safe to inline, inlines the body of the target into that of the caller.

`StaticMethodBinder` (phase `wjop.smb`) does the following:

1. finds call sites which are monomorphic (i.e. has only one target);
2. creates an new static method which has a body identical to the target, but whose first parameter is the object that used to be the receiver;
3. redirects the original call site to the newly-created static method.

By default, the call graph is build using CHA. Spark can be used instead by enabling it with the option `-p cg.spark on`. Spark can also simulate other analyses such as VTA (`-p cg.spark vta`) or RTA (`-p cg.spark rta`).

3 Including dynamically-loaded classes

If the program to be optimized loads classes dynamically using the `newInstance` method, Soot will be unable to tell statically which classes need to be resolved. In this case, the user will need to tell Soot explicitly which classes are loaded. This can be done using one of the following command-line options:

1. `--dynamic-dir` lets the user specify paths under which all classes are considered potentially dynamic.

2. `--dynamic-package` lets the user specify packages for which all class files belonging to the package or any subpackage thereof are considered potentially dynamic. For instance, saying

   ```
   --dynamic-package sun.security.provider
   ```

   will mark a class like `sun.security.provider.Provider` as potentially dynamic.

3. `--dynamic-class` lets the user specify specific dynamic classes.

These options may be specified multiple times to specify multiple dynamic directories, packages, or classes. Note: *The user must specify all potentially dynamic classes using one (or both) of the above, or the call graph may be incomplete.*

4 Examples

- ```
  java -mx300m soot.Main --app -W -p wjop.smb on -p wjop.si off
  spec.benchmarks._201_compress.Main
  ```

  This command runs `StaticMethodBinder` instead of `StaticInliner`. It does not include any dynamic packages. The `-mx300m` switch is present so that the virtual machine is allowed to use more memory (300 Mb) than the default value (since whole-program analysis usually uses a lot of memory). Note that the switch for allowing more memory usage may be different depending on the virtual machine used.

- ```
  java -mx500m soot.Main --app -W --dynamic-package java.text.resources
  --dynamic-package spec.benchmarks._213_javac SpecApplication
  ```

  This command runs `StaticInliner`. It uses CHA to find monomorphic sites. It analyzes library classes, and it includes all classes in the packages `java.text.resources`, `spec.benchmarks._213_javac`, or any of their subpackages, as potentially dynamic classes. It allows the virtual machine to use 500 Mb of memory.

History

- September 5, 2000: Initial version.
- May 31, 2003: Updated for Soot 2.0.