Cheap function entry instrumentation

Aditya Kumar
Ian Levesque
Sam Todd

Agenda

1. Motivation
2. Functionality
3. Infrastructure
4. Benefit

Motivation

Finding dead code in a large codebase

- Static analysis does not work
- Externally visible functions can’t be stripped by the linker

Instrumenting Function Entry

- One array for each module.
- Module hash to identify the module
  - Some bits can be used to encode instrumentation scheme (#futurework)
- Each function has 1 byte of storage
- Each function can use their byte for fun and profit!
  - Dead code detection
  - Null pointer check for function arguments
  - Cheap value profiling

Each array has the following format:

```
|—-——64bit—————-|———32bit———-—|—-Entry for each function ————|
|--Module hash———|---#of functions-|—-Byte array, with 0 or 1 ---------|
```

Instrumenting for dead code detection

Compiler adds code that stores 0x1 whenever the function is called
1 byte per function is cheap
Lock free

Each array has the following format:

```
|—-——64bit—————-|———32bit——-—|—-Entry for each function ————|
|--Module hash———|---#of functions-|—-Byte array, with 0 or 1 ---------|
$ cat test.ll

target triple = "arm64-apple-ios"

; Function Attrs: norecurse nounwind readnone
define i32 @foo(i32 %a) {
  %a.sroa.0.0.trunc = trunc i32 %a to i8
  %a.sroa.5.0.shift = lshr i32 %a, 8
  %bf.clear = and i8 %a.sroa.0.0.trunc, 1
  %bf.cast = zext i8 %bf.clear to i32
  %bf.lshr = lshr i8 %a.sroa.0.0.trunc, 1
  ret i32 %a
}

define i32 @bar(i32 %a) {
  %a.sroa.0.0.trunc = trunc i32 %a to i8
  %a.sroa.5.0.shift = lshr i32 %a, 8
  %bf.clear = and i8 %a.sroa.0.0.trunc, 1
  %bf.cast = zext i8 %bf.clear to i32
  %bf.lshr = lshr i8 %a.sroa.0.0.trunc, 1
  ret i32 %a
}
```

```
$ opt -instrfuncentry -S test.ll -o -

@_llvm_funcentry_array_5977508082728489289 = linkonce hidden global [14 x i8] c"I\D1\0EY\EE_\F4R\02\00\00\00\FF\FF", section "__DATA,__llvm_funcentry"

define i32 @foo(i32 %a) {
funcentry_set:
  store i8 0, i8* getelementptr inbounds ([14 x i8], [14 x i8]* @_llvm_funcentry_array_5977508082728489289, i32 0, i32 12)
  br label %0

; <label>:0:
  %a.sroa.0.0.trunc = trunc i32 %a to i8
  %a.sroa.5.0.shift = lshr i32 %a, 8
  %bf.clear = and i8 %a.sroa.0.0.trunc, 1
  %bf.cast = zext i8 %bf.clear to i32
  %bf.lshr = lshr i8 %a.sroa.0.0.trunc, 1
  ret i32 %a
}

define i32 @bar(i32 %a) {
funcentry_set:
  store i8 0, i8* getelementptr inbounds ([14 x i8], [14 x i8]* @_llvm_funcentry_array_5977508082728489289, i32 0, i32 13)
  br label %0

; <label>:0:
  %a.sroa.0.0.trunc = trunc i32 %a to i8
  %a.sroa.5.0.shift = lshr i32 %a, 8
  %bf.clear = and i8 %a.sroa.0.0.trunc, 1
  %bf.cast = zext i8 %bf.clear to i32
  %bf.lshr = lshr i8 %a.sroa.0.0.trunc, 1
  ret i32 %a
}
```

Cost models

- Skip standard library functions
- Skip functions smaller than a certain size
- Skip functions specified in a blacklist

Extensible

- Skip standard library functions
- Skip functions smaller than a certain size
- Skip functions specified in a blacklist

Data for post processing

- Dump list of global buffers in a file
- Dump Hashing, and function indices

```
cat mapping/func-entry-write-mapping.txt
MD5 5cf8c24cdb18bdac,foo,12
MD5 e413754a191db537,bar,13
```

```
cat mapping/func-entry-global-buffer.txt
_llvm_funcentry_array_5977508082728489289
_llvm_funcentry_array_7867504356345634132
```

Infrastructure to support dead code detection

- Collect data from the application and store it somewhere
- Use the mapping file (version specific) to index functions that were called
- Aggregate all the data for sufficiently long duration
- List functions which were called

Collecting data from production

- Instrument each function and get data from the field.
  - If a function hasn't been called in a while, it's probably dead
    - Manual inspection required

Caveats

- New functions may have been written
- Some functions which were detected have been deleted already
  - Or moved to another location (refactoring...)
- Functions which are rarely called may add to noise
  - Error reporting, exception handling functions

References

- Patch: https://reviews.llvm.org/D74362

- LLVM Xray https://llvm.org/docs/XRay.html
- Order File Instrumentation: https://reviews.llvm.org/D58751
- https://en.wikipedia.org/wiki/Profile-guided_optimization