Debugging a flaky Go test with Mozilla rr

Mon, Feb 22, 2021

> This is how you debug a test that only fails once every 1000 times.

The Test

```go
package my

import (
	"math/rand"
	"testing"
	"time"
)

func init() {
	rand.Seed(time.Now().UnixNano())
}

func TestRandFail(t *testing.T) {
	if n := rand.Intn(1000); n == 50 {
		t.Fatalf("finally got %d", n)
	}
}
```

- This is obviously a pedagogical example.

Get the newest version of rr

```
git clone https://github.com/rr-debugger/rr.git
cd rr
git checkout 5.4.0 # change this to the latest release (DO NOT BUILD HEAD)
mkdir build
cd build
cmake ..
make -j8
sudo make install
```

- **Warning**: `rr` does not work in VirtualBox.
- https://github.com/rr-debugger/rr/wiki/Building-And-Installing

Compile your failing test to a binary

```
go test -gcflags 'all=-N -l' -c
```

- The `-gcflags 'all=-N -l'` disables optimizations and inlining.

Install the following `rrloop` script.

```sh
#!/bin/sh

while :
do
  rr $@
  if [ $? -ne 0 ]; then
    echo "encountered non-zero exit code: $?";
    exit;
  fi
done
```

- `rrloop` is a wrapper around `rr` which keeps looping until it sees a non-zero exit code.
- This works because `rr` exits using the recorded process' exit code.

Record the test execution in a loop until it fails

```
echo -1 | sudo tee -a /proc/sys/kernel/perf_event_paranoid
echo 0 | sudo tee -a /proc/sys/kernel/kptr_restrict

rrloop record ./my.test
```

- If you have multiple tests, you can pass `-test.run=TestRandFail` to only run a specific one.
- The `--chaos` flag often increases the chances of the failure.

Example Output:

```
...
rr: Saving execution to trace directory `/home/icholy/.local/share/rr/my.test-11
8'.
PASS
rr: Saving execution to trace directory `/home/icholy/.local/share/rr/my.test-11
9'.
PASS
rr: Saving execution to trace directory `/home/icholy/.local/share/rr/my.test-12
0'.
PASS
rr: Saving execution to trace directory `/home/icholy/.local/share/rr/my.test-12
1'.
PASS
rr: Saving execution to trace directory `/home/icholy/.local/share/rr/my.test-12
2'.
--- FAIL: TestRandFail (0.00s)
    my_test.go:15: finally got 50
FAIL
encountered non-zero exit code: 0
```

- We're only interested in the last trace file (in this case `/home/icholy/.local/share/rr/my.test-122`).
- Depending on how many runs it takes before the test fails, you might need to stop and clear the trace directory.

Debug the execution trace

```
go install github.com/go-delve/delve/cmd/dlv@latest
dlv replay /home/icholy/.local/share/rr/my.test-122
```

Find the test function and set a breakpoint

```
(dlv) funcs TestRand
my.TestRandFail
(dlv) b my.TestRandFail
Breakpoint 1 set at 0x50b173 for my.TestRandFail() ./my_test.go:13
(dlv) c
> my.TestRandFail() ./my_test.go:13 (hits goroutine(6):1 total:1) (PC: 0x50b173)
Current event: 414
     8:
     9: func init() {
    10:         rand.Seed(time.Now().UnixNano())
    11: }
    12:
=>  13: func TestRandFail(t *testing.T) {
    14:         if n := rand.Intn(100); n == 50 {
    15:                 t.Fatalf("finally got %d", n)
    16:         }
    17: }
```

- See rev and rewind commands.

Connect Visual Studio Code

Start delve in server mode

```
dlv replay /home/icholy/.local/share/rr/my.test-122 --headless --listen=:2345 -
-log --api-version=2
```

Remotely connect using the following `launch.json` configuration:

```
{
    "name": "Replay Trace",
    "type": "go",
    "request": "attach",
    "mode": "remote",
    "remotePath": "${workspaceFolder}",
    "port": 2345,
    "host": "127.0.0.1"
}
```

- **Note**: vscode doesn’t support reverse commands. See https://github.com/golang/vscode-go/pull/89

Comments