unicorns die with bullets made of glitter
Once upon a time...
Sunshine, Lollipops and vmg

github.com/vmg
twitter.com/vmg
These are horror stories from the GitHub Systems team.

(the rainbows are just to make them less awful)
I HAVE A LOT OF SPARE TIME
The Dumb Tale of a Dumb Garbage Collector
Mark Sweep
The Garbage Collector
Garbage Collector
Garbage Collector

Conservative
Garbage Collector
Conservative
Non-deterministic
Garbage Collector
Conservative
Non-deterministic
Stop The World
Garbage Collector
Conservative
Non-deterministic
Stop The World
Mark and Sweep
Garbage Collector
Conservative
Non-deterministic
Stop The World
Mark and Sweep
Basically a shitshow.
Mark
1. Walk the object graph

2. Look for things that look like Ruby objects
Ruby handles raw pointers to C extensions.
Ruby handles raw pointers to C extensions.

The Garbage Collector must find roots in the stack, the heap and in registers.
“This kinda looks like a pointer”
“This kinda looks like a pointer”

“I guess…”
“This kinda looks like a pointer”

“I guess…”
1. Go through every single object

2. Free the ones that have not been marked
The Problem
The Problem

There is memory allocated by Ruby that the Ruby GC cannot detect.
int a_ruby_function(VALUE rb_foo)
{
    char *str = RSTRING_PTR(rb_foo);
    return do_something(str);
}
int a_ruby_function(VALUE rb_foo)
{
    char *str = RSTRING_PTR(rb_foo);
    return do_smthing(str);
}
int do_something(char *str) {
    /* do stuff */
    return strlen(str);
}
int do_smthing(char *str)
{
    /* do stuff */
    return strlen(str);
}
Ung! This no look like Ruby object. Mark no touchie!
Hello World

No root.

String object with no roots?
Set fire. Yes. Fire.
String object with no roots?
Set fire. Yes. Fire.
No root.

String object with no roots?
Set fire. Yes. Fire.
int do_smthing(char *str)
{
    /* do stuff */
    return strlen(str);
}
int do_smthing(char *str)
{
    /* do stuff */
    return strlen(str);
}

*sound of an explosion*

“What was that?”

“Dunno.”
HAPPY ENDINGS:
(sorry about this slide)
1. Drink Yourself To Sleep Every Night
2. Stress the Garbage Collector in Critical Paths
mallopt(M_PERTURB, 0x33);
mallopt(M_PERTURB, 0x33);

C land

Ruby land

GC.stress = true
Bonus Points

Use Valgrind/ASan instead of tampering.
Bonus Points

Use **Valgrind/ASan** instead of tampering.

Your Rails test suite

Heat death of the Universe
3. Static Analysis
Our time is running out

“Oh dear! Oh dear! I shall be too late!
...This damned GVL!”
How do you feel about timeouts?
THERE ARE NO TIMEOUTS IN RUBY
(just ask the Timeout Rabbit)
require 'timeout'
require 'timeout'

Timeout::timeout(5) {
  # Do something
}

require 'timeout'

Timeout::timeout(5) {
  # Do something
}


oh hey
“Sorry I’m late. Couldn’t get here any faster.”
x = Thread.current
y = Thread.start {
  begin
  sleep sec
  rescue => e
  x.raise e
  else
  x.raise exception, "execution expired"
  end
}
return yield(sec)
1 x = Thread.current
2 y = Thread.start {
3     begin
4     | sleep sec
5     rescue => e
6     | x.raise e
7     else
8     | x.raise exception, "execution expired"
9     end
10 }
11 return yield(sec)
Beautiful

also:

wrong
MRI 1.9.3

Ruby Process

Native Thread

Native Thread

Native Thread

THE KERNEL
Use the force, kid
“Unix, you moron. Unix is the force.”
“Unix, you moron. Unix is the force.”
MRI 1.8.7

Use signals & the power of Unix
MRI 1.8.7
Use signals & the power of Unix

MRI 1.9.3
Release the GVL
“I, too, had a dreadful problem with timeouts”
You see, I was using JRuby...
JRuby?
Sounds like you have two problems now.
(shut up, kid)
You see, I was using JRuby, and the GitHub API you guys keep breaking...
Duh.
That’s probably because the thread is sleeping or stuck in the GVL.
You fool!
JRuby has native threads
and fine-grained locking!

*smack*
JRUBY 1.6.7

```ruby
Timeout::timeout(5) {
  Timeout::timeout(60) {
    # Do something
  }
}
```

Longest timeout clobbers the short one
Longest timeout clobbers the short one
I LIKE THIS BUG
Because JRuby wins
Simple Bug
Complex Bug
Painful Bug
Kernel Bug

NESTED TIMEOUTS IN THE JVM
Now Leaving:
Realm of Bugs

Valley of Common Sense
Valley of Common Sense
Mountains of Reason
Country of
WHAT THE
HELL ARE YOU
DOING

MRI Timeouts
in Green Threads
MORAL OF THE STORY
Sometimes

WORSE

IS

BETTER

*this doesn’t apply to the MRI GC
Computers break twice as often.
If you build simpler systems, they will break in simpler ways.
...and they lived happily ever after.
The end.

github.com/vmg
twitter.com/vmg