Nokogiri のお母さん

@flavorjones

ZOMG!!!!!

HAPPY

THURSDAY!!

WELCOME TO

RubyConf X!

X-TREME

RUBYCONF!!!!!

Aaron Patterson

@tenderlove

aaron.patterson@gmail.com

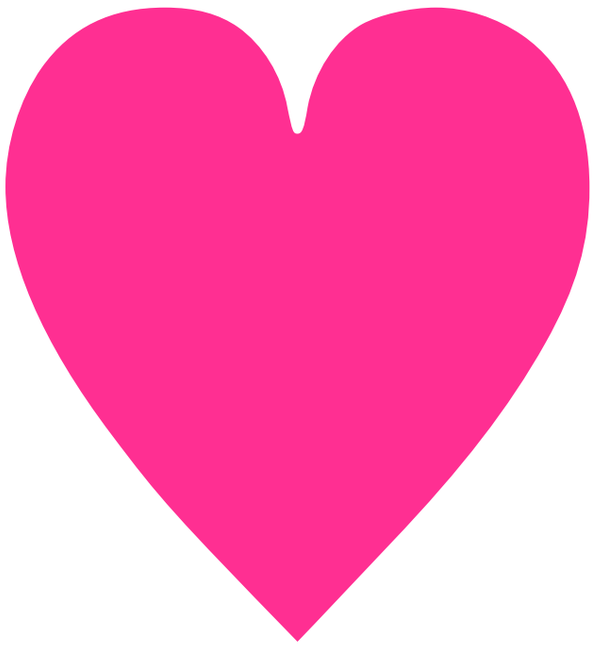
ruby committer

rails committer

Committer

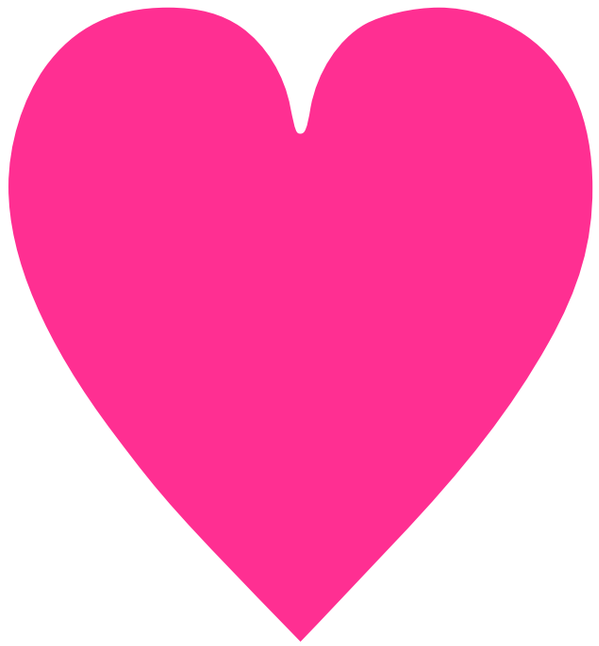
HOWTO

Ruby



Rails

RubyConf



WWFMD?

RubyConf 5k

RUBYCONF

FIVE THOUSAND

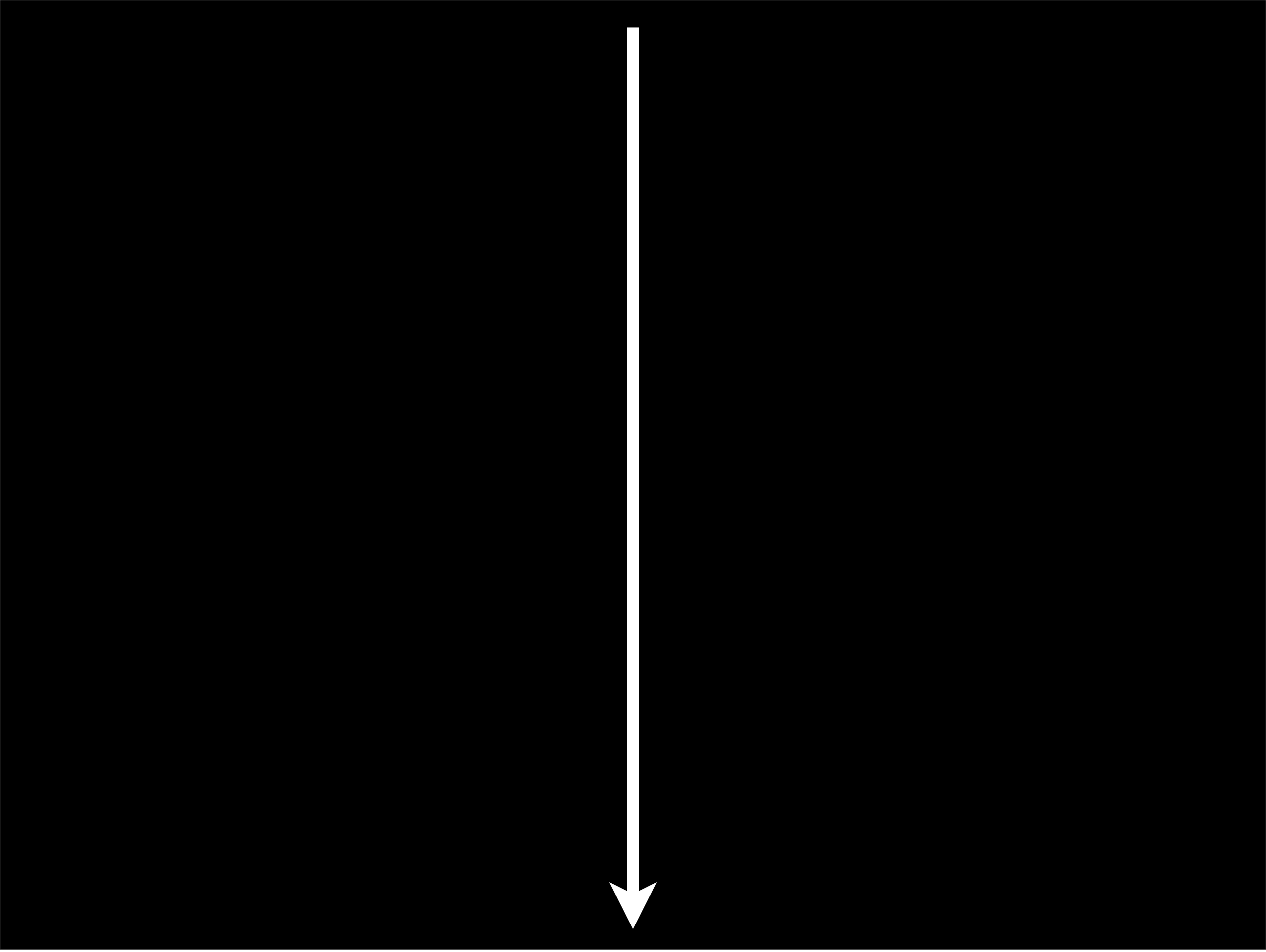
Thursday, November 11, 2010

ZOMG WHY IS THIS CODE SO SLOW?

Performance

Code Analysis

Story Form



IRL

IRL

IRL

Tools

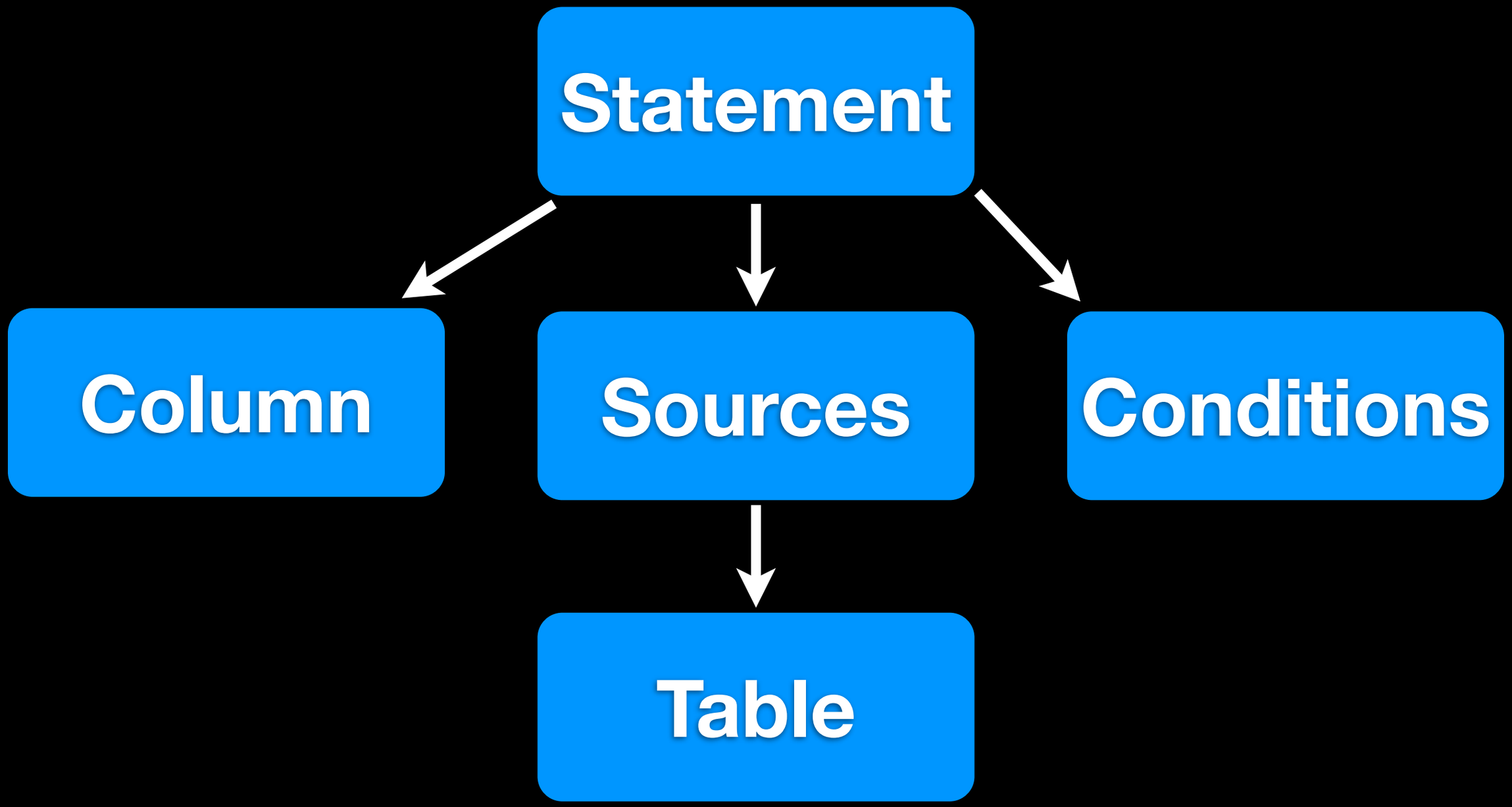
Theory

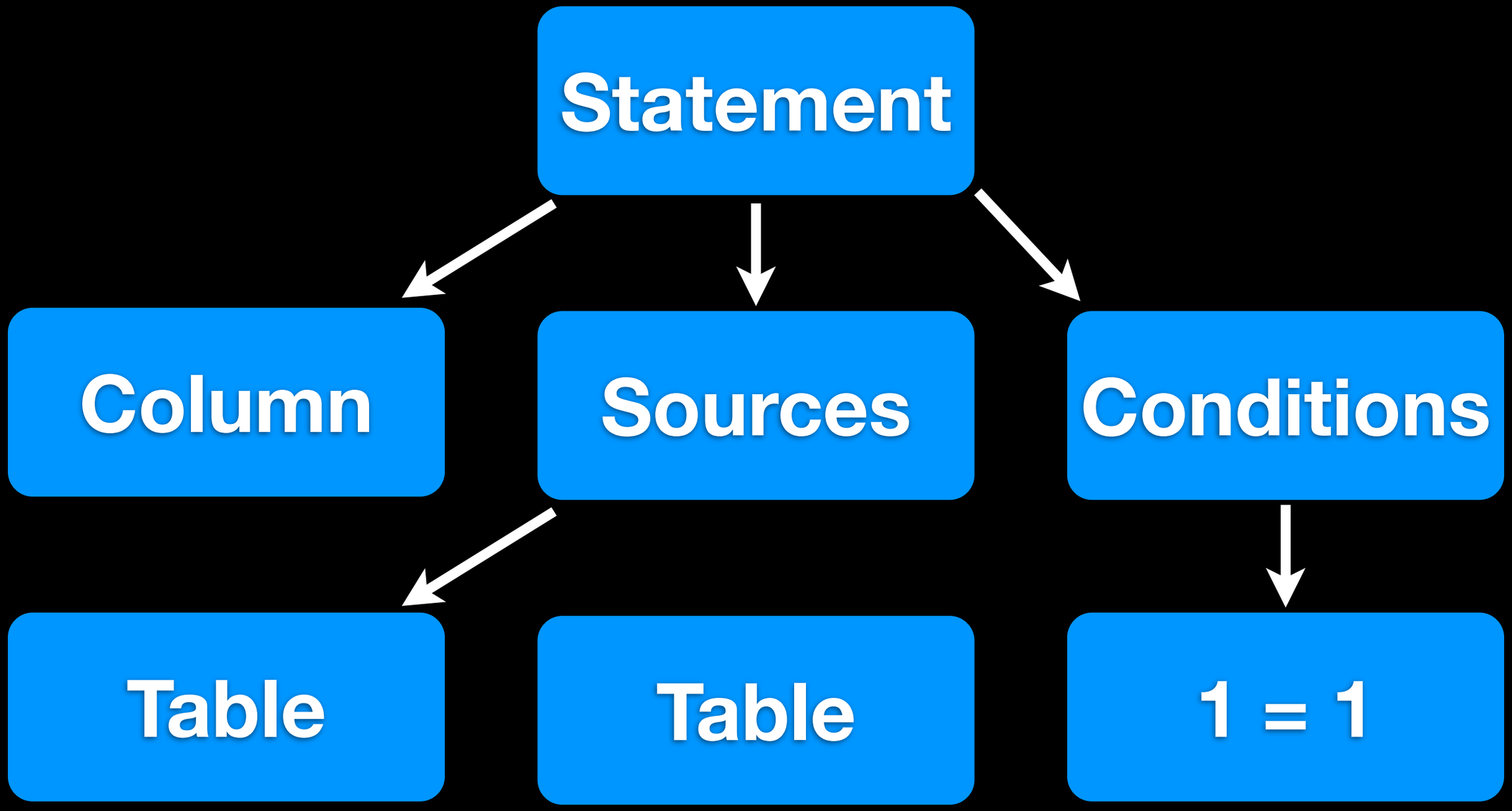
ARel

What is it?

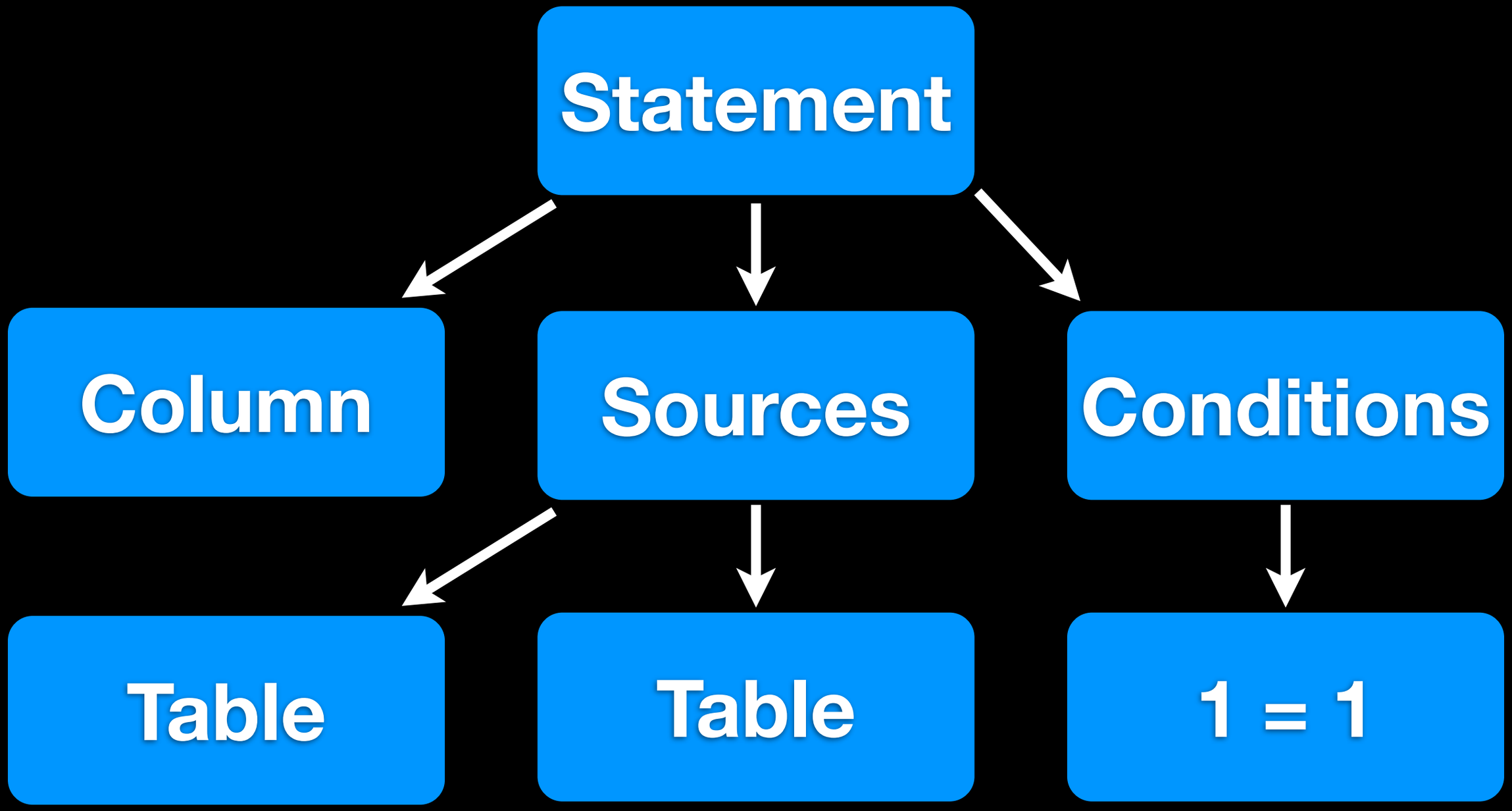
"Relational Algebra"

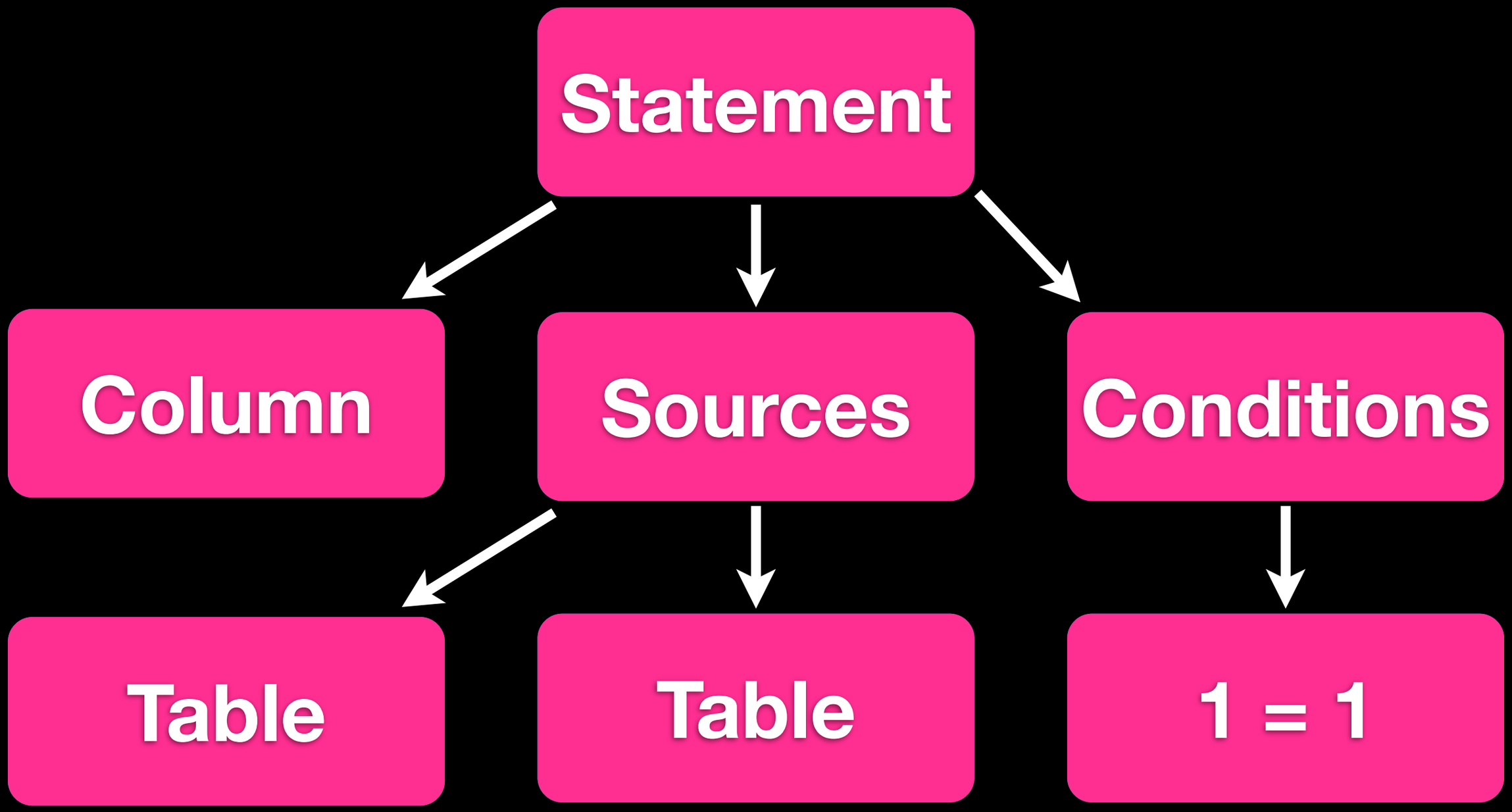
AST Manipulation


Statement
Column
Sources
Conditions
Table


Statement
Column
Sources
Conditions
Table
Table
1 = 1

AST Translation


Statement
Column
Sources
Conditions
Table
Table
1 = 1


Statement
Column
Sources
Conditions
Table
Table
1 = 1

```
SELECT COLUMN
FROM TABLE, TABLE
WHERE 1 = 1
```

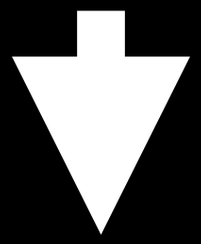
Relationship with Rails

User (YOU!)

ActiveRecord

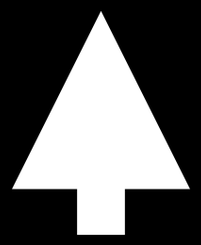
ARel

User (YOU!)



Records PLZ

ActiveRecord



SQL STMT

ARel

The More You Know™

Getting Started

Rails

Prepared Statement Caching

DEEPER UNDERSTANDING REQUIRED

ActiveRecord 5x slower than Rails 2.3.5

http://bit.ly/omgslow

5x Slower?!?

WTF?

Yes, 5x Slower

What could possibly go wrong?

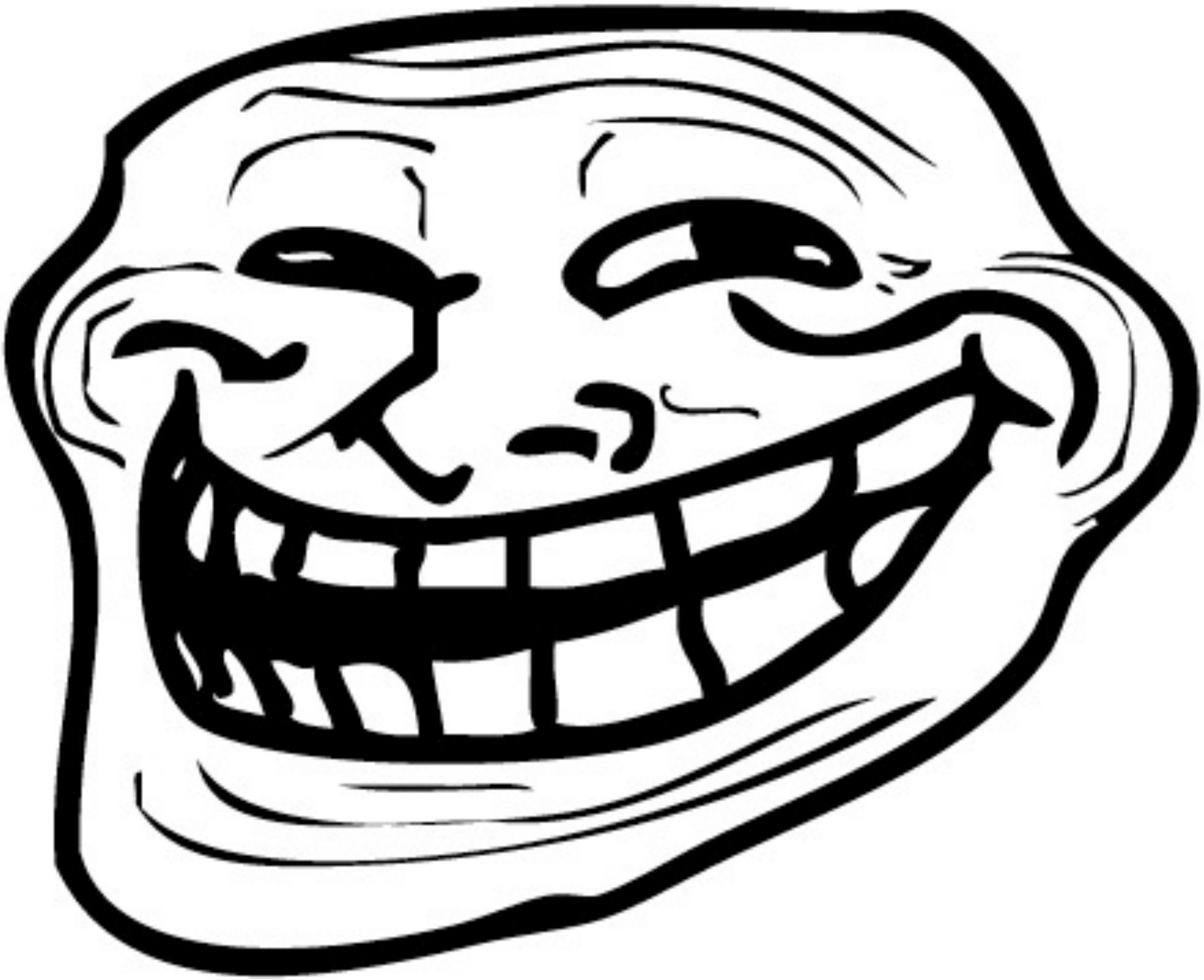
Motivation

Why do you care about speed?

Scaling Ruby

query_methods.rb (~/git/rails/a...lib/active_record/relation) - VIM1

```
        arel.join(join)
      end

      arel.joins(arel)
    end

    def build_arel
      arel = table

      arel = build_joins(arel, @joins_values) unless @joins_values.empty?

      (@where_values - ['']).uniq.each do |where|
        where = Arel.sql(where) if String === where
        arel = arel.where(Arel::Nodes::Grouping.new(where))
      end

      arel = arel.having(*@having_values.uniq.reject{|h| h.blank?}) unless @havi
ng_values.empty?

      arel = arel.take(@limit_value) if @limit_value
      arel = arel.skip(@offset_value) if @offset_value

      arel = arel.group(*@group_values.uniq.reject{|g| g.blank?}) unless @group_
values.empty?

      arel = arel.order(*@order_values.uniq.reject{|o| o.blank?}) unless @order_
values.empty?

      arel = build_select(arel, @select_values.uniq)

      arel = arel.from(@from_value) if @from_value
      arel = arel.lock(@lock_value) if @lock_value

      arel
    end

    private

query_methods.rb [2] 0-1, 195 of 267, (4572) 71% 0x0 (0)
```



When should I make my code faster?

When it isn't fast enough.

What is

"fast enough"?

Do people notice it?

In comparison to?

Finishes in a reasonable amount of time.

What code should I improve?

Only the code that matters.

Don't believe me.

Think Critically

Discovery

What to measure?

Breakdown

Post.find(1)

ARel....

find_by_sql()

execute()

log()

find_by_sql()

execute()

log()

Work per Time

Performance

Degraded

Benchmarking

Our Enemies

Time

Space

For Performance:

Things To Reduce

- Method calls
- Branching and looping
- Objects (memory consumption)

For Clean Code:

Things To Reduce

- Method calls
- Branching and looping
- Objects (memory consumption)

∴

(therefore)

Clean Code

≠

Performant code

Measurement is

Paramount

require 'benchmark'

```
require 'benchmark'

def fib n
  a = 0
  b = 1
  n.times { a, b = b, a + b }
  a
end
```

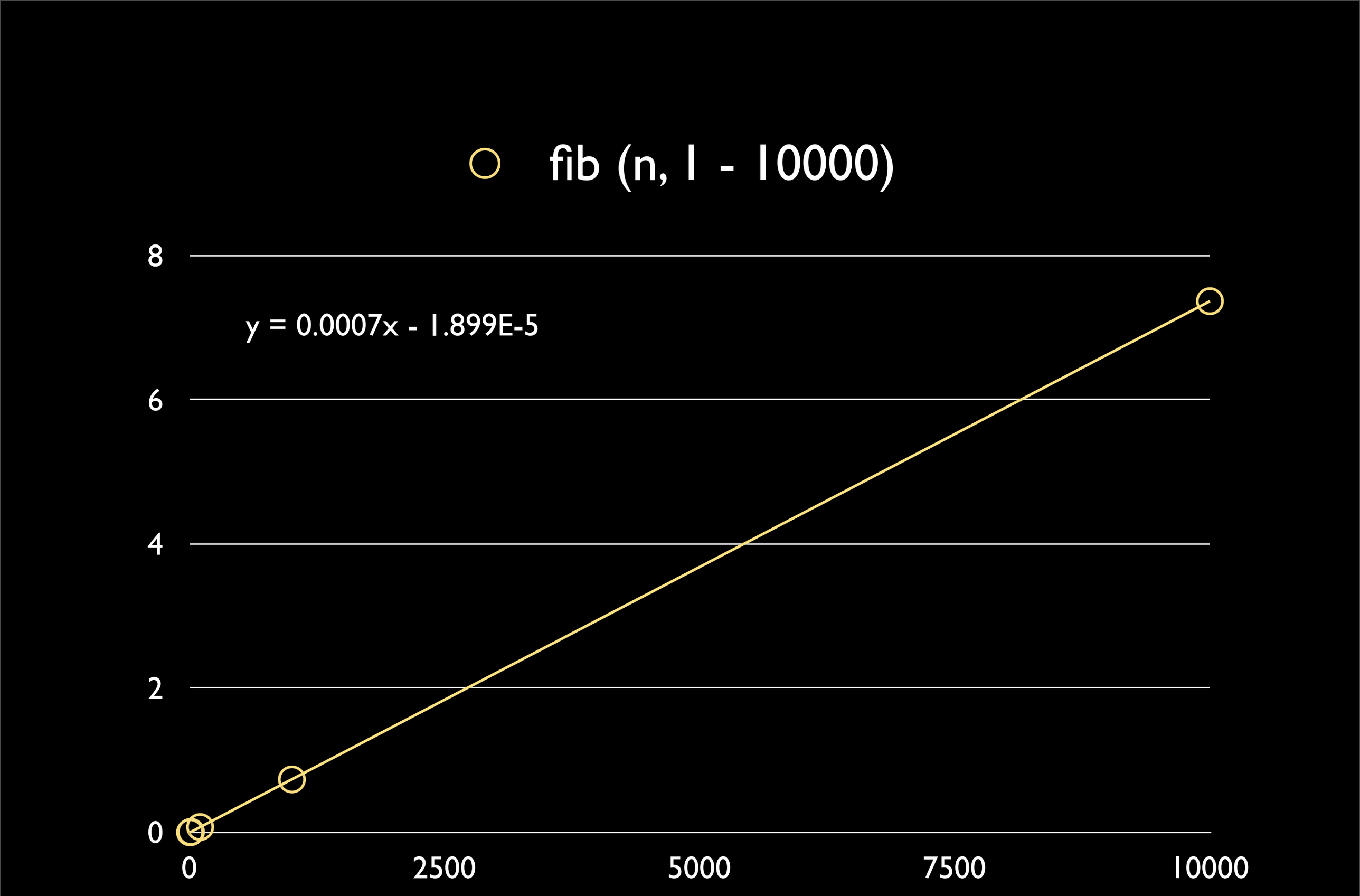
```
Benchmark.bm(7) do |x|
  x.report("fib") do
    3000.times do |i|
      fib(1000)
    end
  end
end
```

```
              user     system      total        real
fib       1.570000   0.000000   1.570000 (  1.570726)
```

user	1.570000
system	**0.000000**
total	1.570000
real	1.570726

```
Benchmark.bm(10) do |x|
  [1, 10, 100, 1000, 10000].each do |n|
    x.report("fib #{n}") do
      n.times { fib(1000) }
    end
  end
end
```

```
                 user     system      total        real
fib 1        0.000000   0.000000   0.000000 (  0.000671)
fib 10       0.010000   0.000000   0.010000 (  0.008352)
fib 100      0.070000   0.000000   0.070000 (  0.074577)
fib 1000     0.740000   0.000000   0.740000 (  0.734922)
fib 10000    7.330000   0.000000   7.330000 (  7.370046)
```


fib (n, 1 - 10000)
y = 0.0007x - 1.899E-5
8
6
4
2
0
0
2500
5000
7500
10000

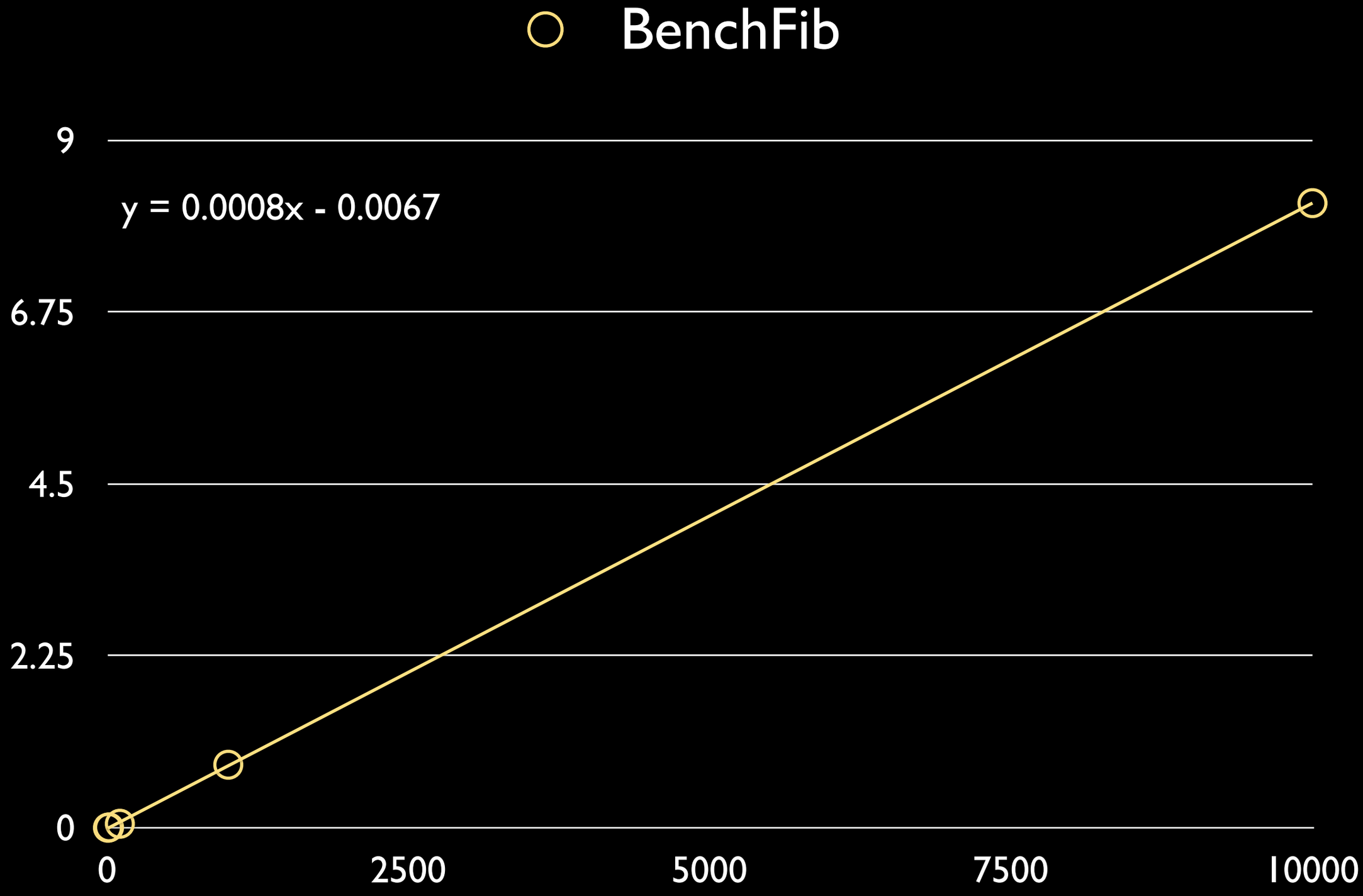
minitest/benchmark

```ruby
require 'rubygems'
require 'minitest/autorun'
require 'minitest/benchmark'

class BenchFib < MiniTest::Unit::TestCase
  def fib n
    a = 0
    b = 1
    n.times { a, b = b, a + b }
    a
  end

  def bench_fib
    assert_performance_linear 0.99 do |n|
      n.times { fib(1000) }
    end
  end
end
```

```
Benchmark.bm(10) do |x|
  [1, 10, 100, 1000, 10000].each do |n|
    x.report("fib #{n}") do
      n.times { fib(1000) }
    end
  end
end
```

```
assert_performance_linear 0.99 do |n|
  n.times { fib(1000) }
end
```

```
BenchFib  1   10  100  1000  10000
bench_fib  0.000571  0.005318  0.052582  0.825676  8.180719
```


BenchFib
9
y = 0.0008x - 0.0067
6.75
4.5
2.25
0
0
2500
5000
7500
10000

find_by_sql

```
def bench_find_by_sql
  assert_performance_linear 0.999 do |n|
    n.times do
      Post.find_by_sql(
        'SELECT * FROM posts WHERE id = 1')
    end
  end
end
```

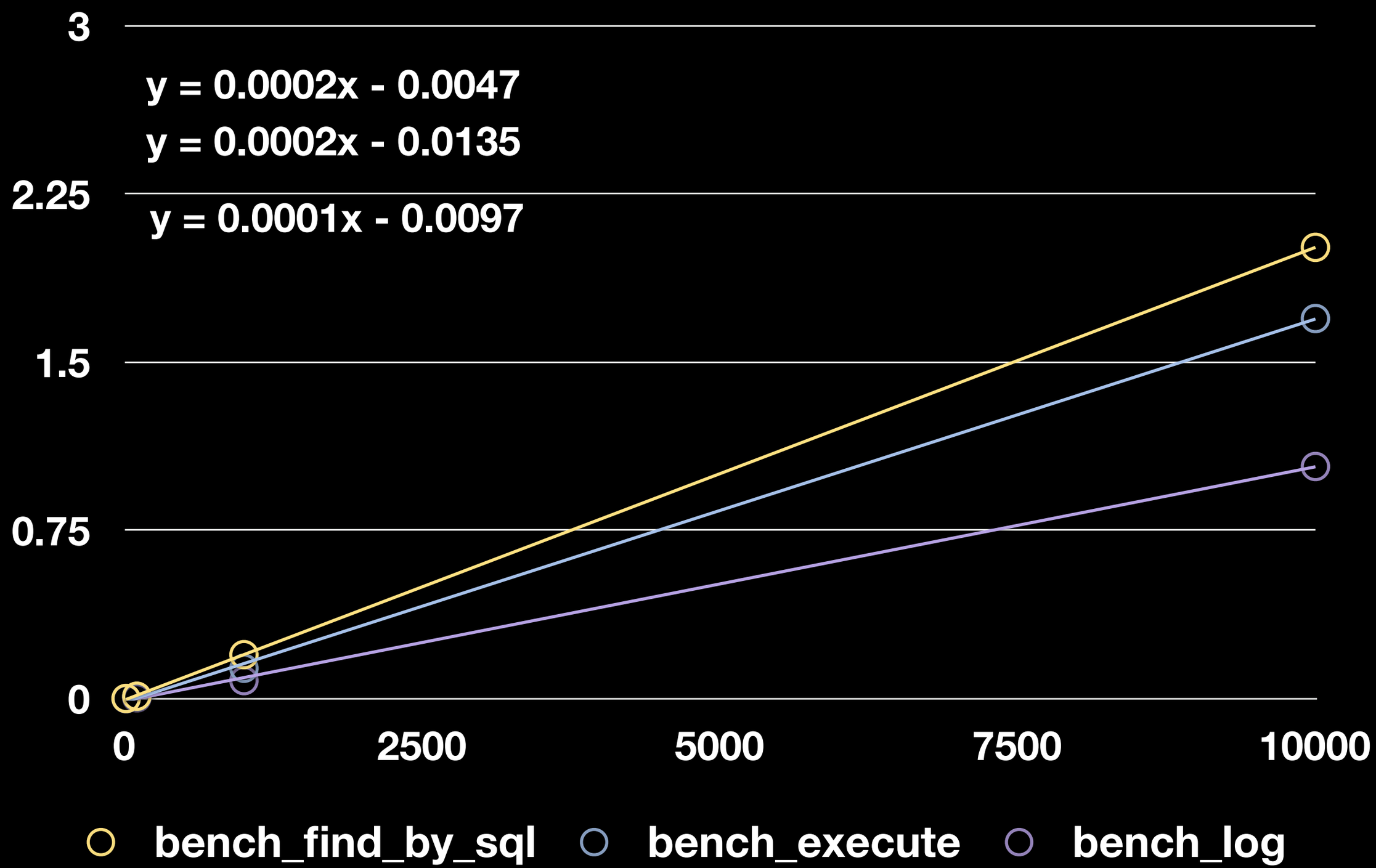
execute()

```
def bench_execute
  conn = Post.connection
  assert_performance_linear 0.999 do |n|
    n.times do
      conn.execute(
        'SELECT * FROM posts WHERE id = 1')
    end
  end
end
```

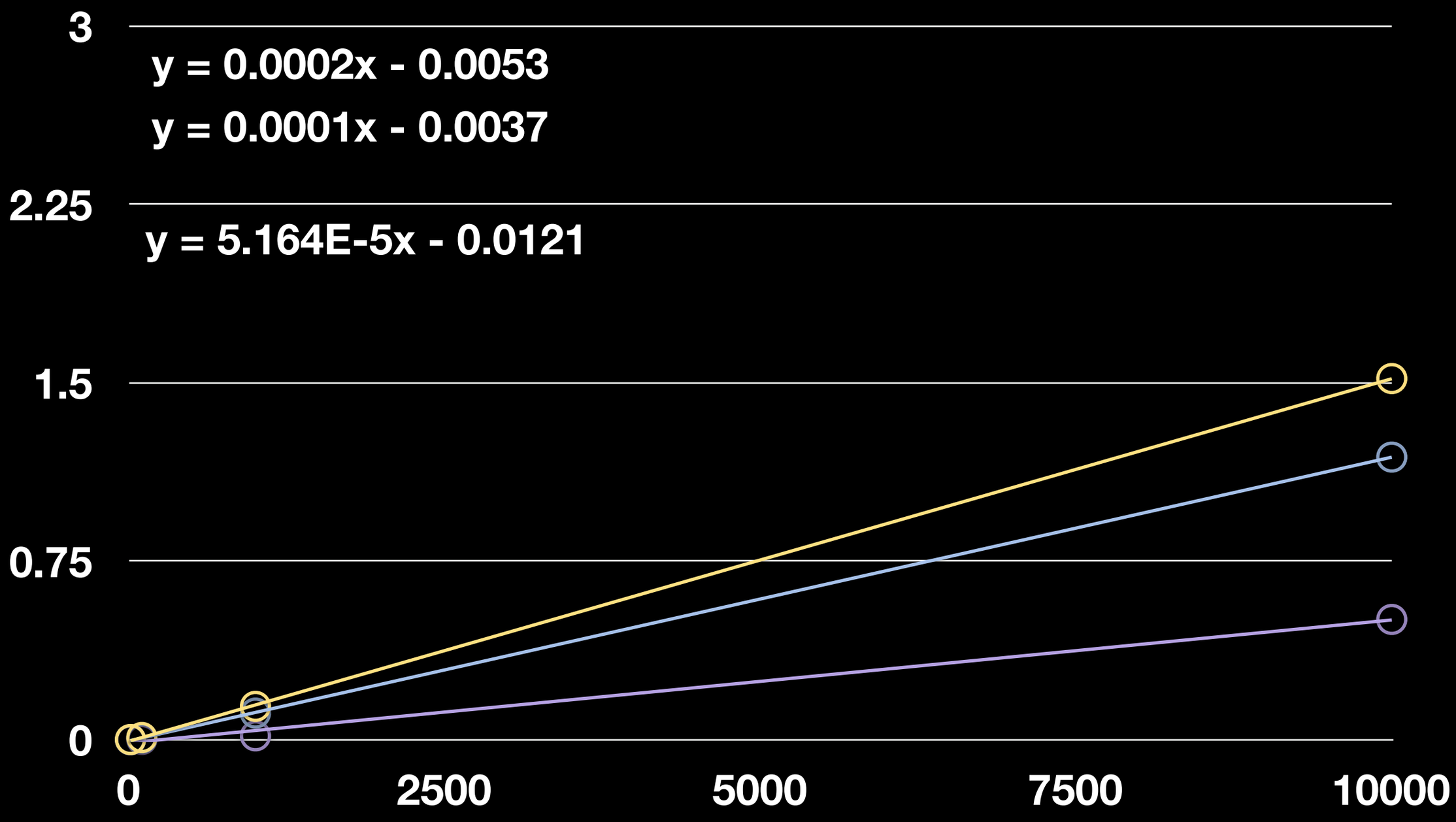
log()

```
def bench_log
  conn = Post.connection
  class << conn
    public :log
  end

  assert_performance_linear 0.999 do |n|
    n.times do
      conn.log('SQL', 'hi mom!') {}
    end
  end
end
```

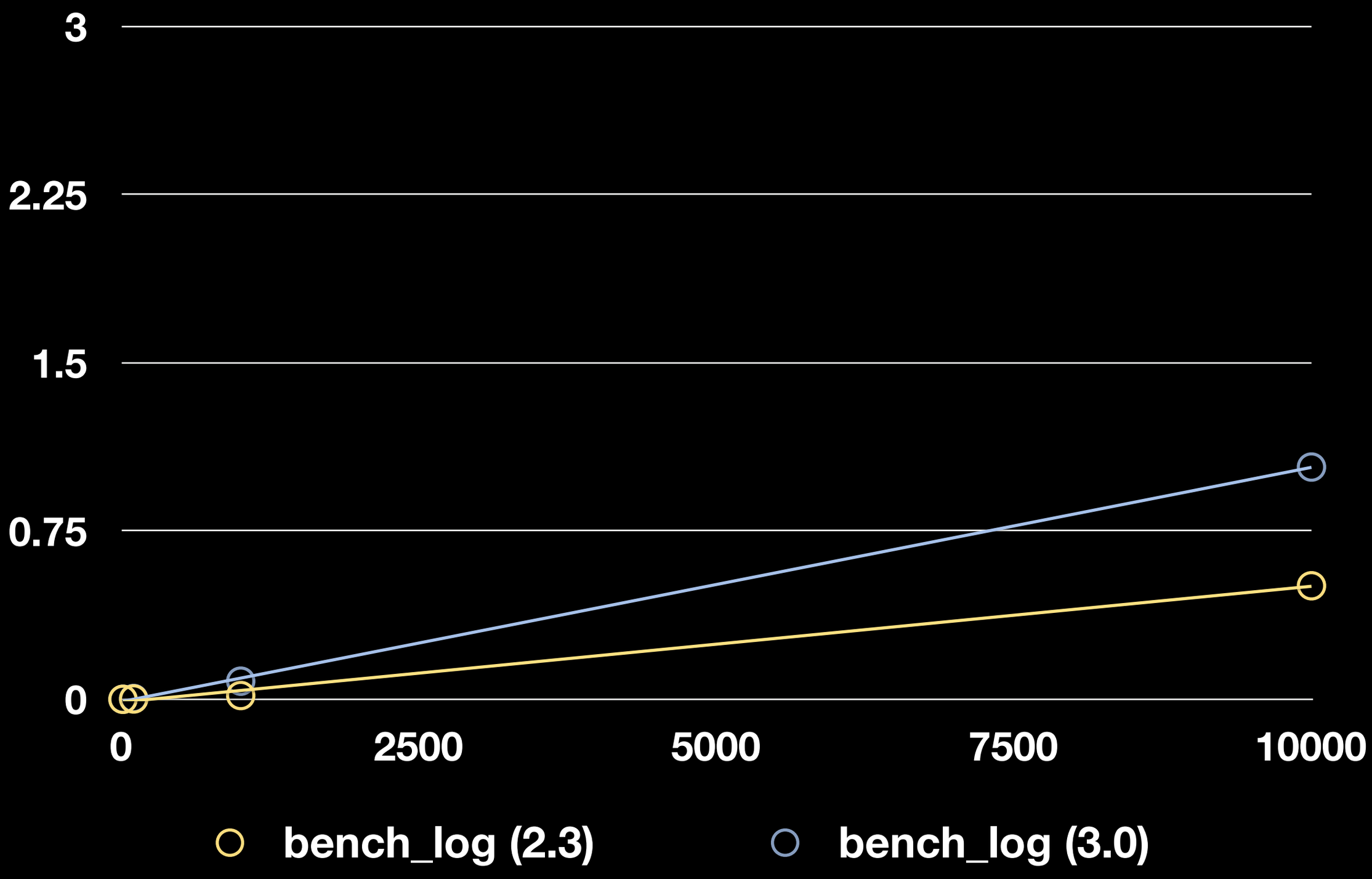
ActiveRecord 3.0 Beta



ActiveRecord 2.3.x



bench_log 2.3.x + 3.0



Δ find_by_sql()

=

Δ execute()

=

Δ log()

$$\Delta \text{ execute()} - \Delta \text{ log()} = 0$$

Method Call

Analysis

perftools.rb

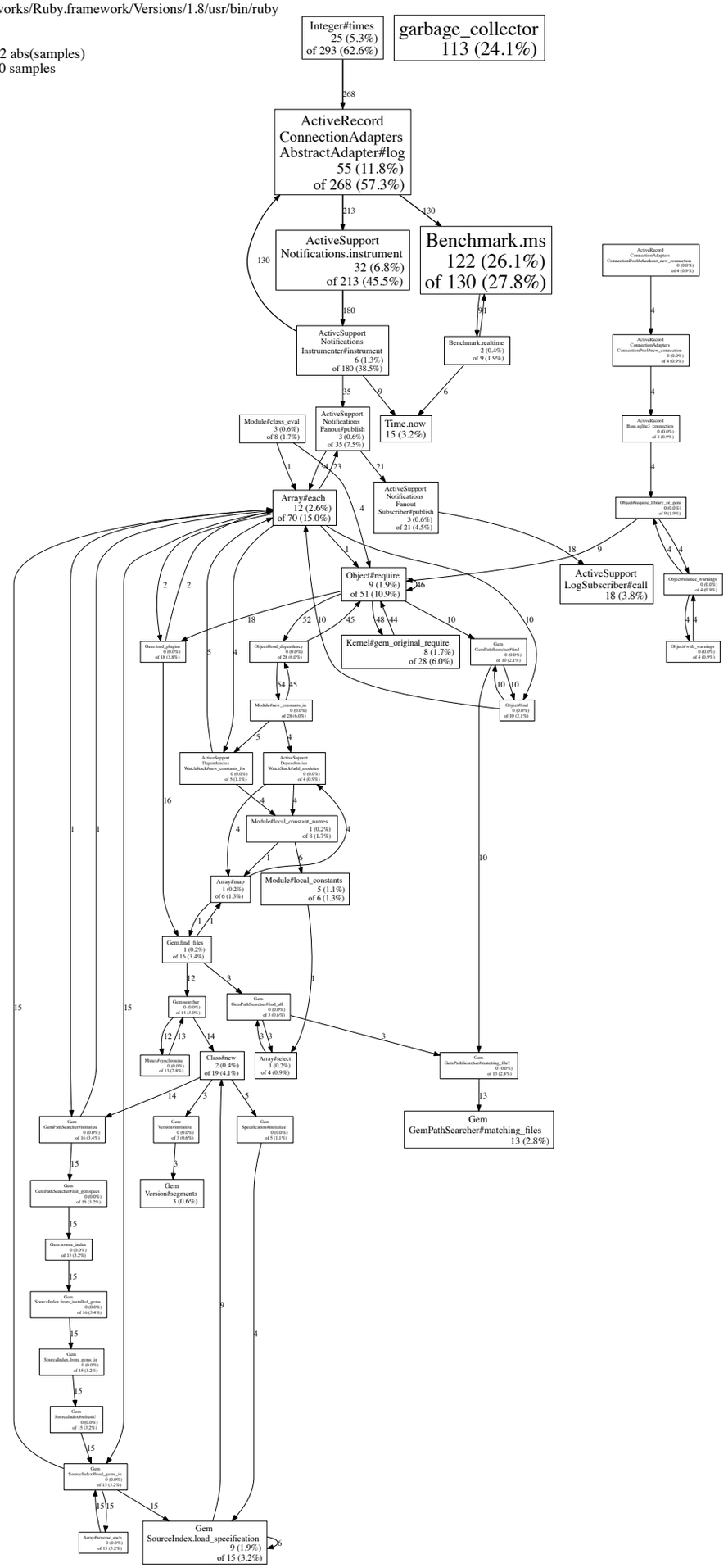
http://github.com/tmm1/perftools.rb/

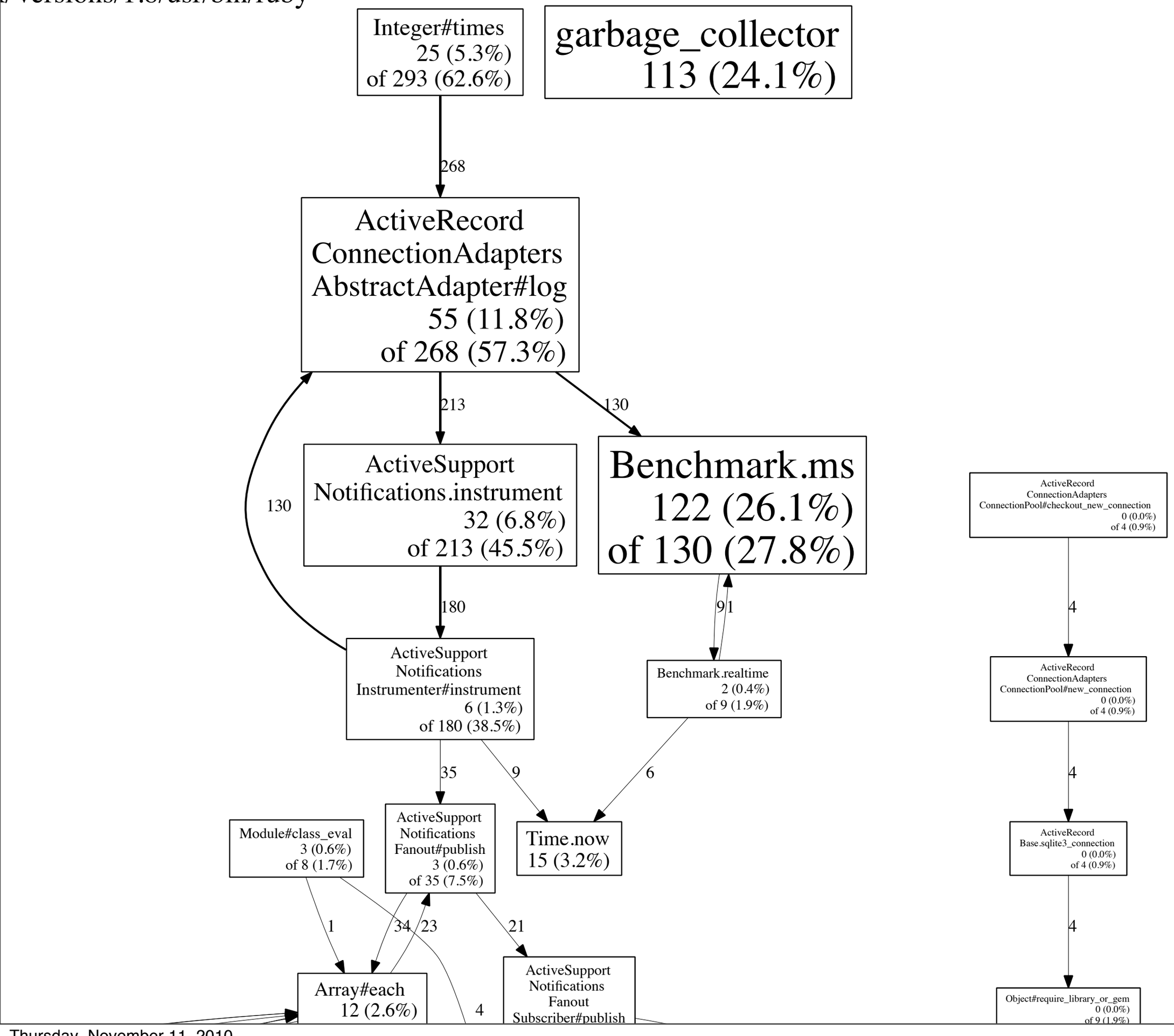
CPUPROFILE=/tmp/my_profile RUBYOPT="-r`gem which perftools | tail -1`" ruby ...

CPUPROFILE=/tmp/
my_profile RUBYOPT="-
r`gem which perftools | tail
-1`" ruby ...

Rails 3.0 Beta

/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/bin/ruby
Total samples: 468
Focusing on: 468
Dropped nodes with <= 2 abs(samples)
Dropped edges with <= 0 samples

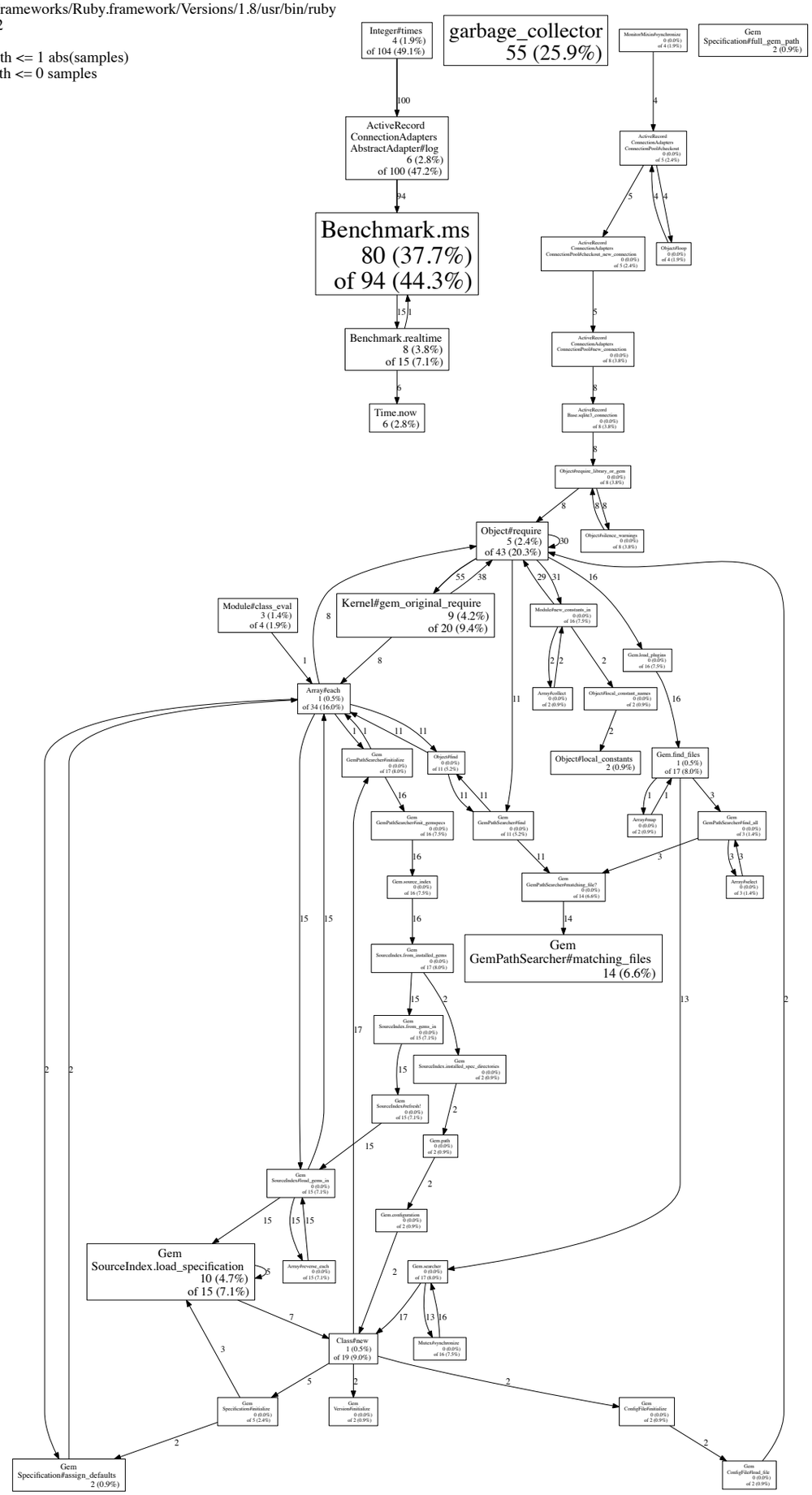



Integer#times
25 (5.3%)
of 293 (62.6%)
garbage_collector
113 (24.1%)
268
ActiveRecord
ConnectionAdapters
AbstractAdapter#log
55 (11.8%)
of 268 (57.3%)
213
130
130
ActiveSupport
Notifications.instrument
32 (6.8%)
of 213 (45.5%)
Benchmark.ms
122 (26.1%)
of 130 (27.8%)
180
9
1
ActiveSupport
Notifications
Instrumenter#instrument
6 (1.3%)
of 180 (38.5%)
Benchmark.realtime
2 (0.4%)
of 9 (1.9%)
35
9
6
Module#class_eval
3 (0.6%)
of 8 (1.7%)
ActiveSupport
Notifications
Fanout#publish
3 (0.6%)
of 35 (7.5%)
Time.now
15 (3.2%)
1
34
23
21
Array#each
12 (2.6%)
4
ActiveSupport
Notifications
Fanout
Subscriber#publish
ActiveRecord
ConnectionAdapters
ConnectionPool#checkout_new_connection
0 (0.0%)
of 4 (0.9%)
4
ActiveRecord
ConnectionAdapters
ConnectionPool#new_connection
0 (0.0%)
of 4 (0.9%)
4
ActiveRecord
Base.sqlite3_connection
0 (0.0%)
of 4 (0.9%)
4
Object#require_library_or_gem
0 (0.0%)
of 9 (1.9%)

```
Total: 468 samples
     122  26.1%  26.1%      130  27.8% Benchmark.ms
     113  24.1%  50.2%      113  24.1% garbage_collector
      55  11.8%  62.0%      268  57.3% ::AbstractAdapter#log
      32   6.8%  68.8%      213  45.5% ::Notifications.instrument
      25   5.3%  74.1%      293  62.6% Integer#times
      18   3.8%  78.0%       18   3.8%::LogSubscriber#call
      15   3.2%  81.2%       15   3.2% Time.now
      13   2.8%  84.0%       13   2.8% GemPathSearcher#matching_files
      12   2.6%  86.5%       70  15.0% Array#each
       9   1.9%  88.5%       15   3.2% SourceIndex.load_specification
       9   1.9%  90.4%       51  10.9% Object#require
```

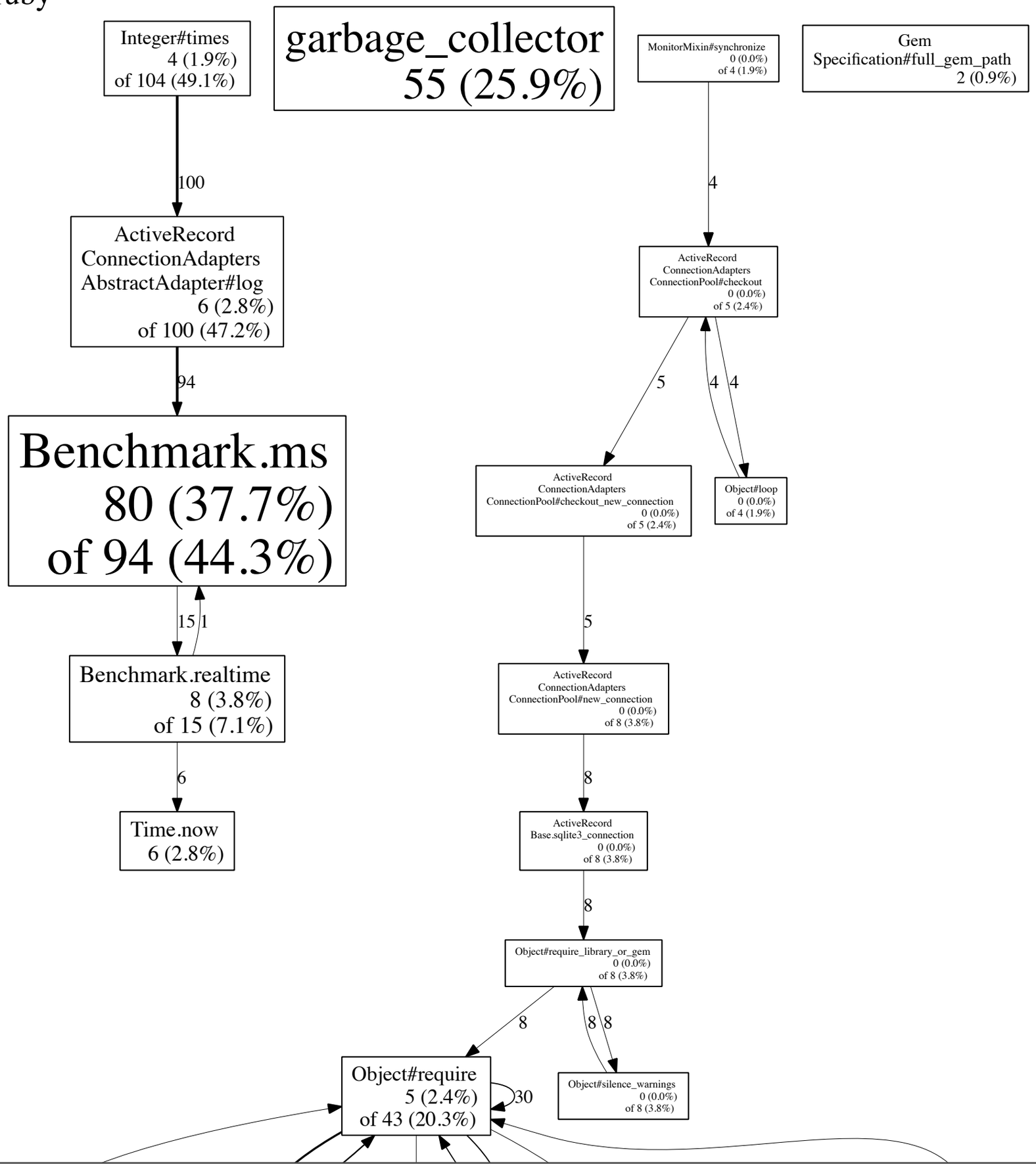
Rails 2-3-stable

/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/bin/ruby
Total samples: 212
Focusing on: 212
Dropped nodes with <= 1 abs(samples)
Dropped edges with <= 0 samples



orks/Ruby.framework/Versions/1.8/usr/bin/ruby

abs(samples)
samples



```
Total: 212 samples
      80  37.7%  37.7%       94  44.3% Benchmark.ms
      55  25.9%  63.7%       55  25.9% garbage_collector
      14   6.6%  70.3%       14   6.6% ::GemPathSearcher#matching_files
      10   4.7%  75.0%       15   7.1% ::SourceIndex.load_specification
       9   4.2%  79.2%       20   9.4% Kernel#gem_original_require
       8   3.8%  83.0%       15   7.1% Benchmark.realtime
       6   2.8%  85.8%      100  47.2% ::AbstractAdapter#log
       6   2.8%  88.7%        6   2.8% Time.now
       5   2.4%  91.0%       43  20.3% Object#require
       4   1.9%  92.9%      104  49.1% Integer#times
       3   1.4%  94.3%        4   1.9% Module#class_eval
       2   0.9%  95.3%        2   0.9% ::Specification#assign_defaults
       2   0.9%  96.2%        2   0.9% Gem::Specification#full_gem_path
       2   0.9%  97.2%        2   0.9% Object#local_constants
```

ruby-prof

Usage

```
result = RubyProf.profile do
  ...
end
printer = RubyProf::FlatPrinter.new(result)
printer.print(STDOUT, 0)
```

n = 1000

Rails 3.0 Beta

```
Total: 0.160831

 %self     total     self     wait    child    calls  name
 28.26      0.15     0.05     0.00     0.10     1000  <::Notifications>#instrument
  9.36      0.03     0.02     0.00     0.01     1000  <::Benchmark>#realtime
  9.16      0.09     0.01     0.00     0.07     1000  ::Instrumenter#instrument
  7.28      0.02     0.01     0.00     0.01     4000  <Class::Time>#now
  5.41      0.16     0.01     0.00     0.15     1000  ::AbstractAdapter#log
  5.10      0.01     0.01     0.00     0.01     1000  <::Notifications>#instrumenter
  2.92      0.00     0.00     0.00     0.00     4000  <Class::Time>#allocate
  2.73      0.02     0.00     0.00     0.01     1000  Array#each
  2.57      0.03     0.00     0.00     0.03     1000  <Module::Benchmark>#ms
  2.57      0.00     0.00     0.00     0.00     4000  Time#initialize
  2.55      0.03     0.00     0.00     0.02     1000  Notifications::Fanout#publish
  2.45      0.01     0.00     0.00     0.01     1000  LogSubscriber#call
  2.37      0.01     0.00     0.00     0.01     1000 ::Fanout::Subscriber#publish
  2.04      0.01     0.00     0.00     0.00     1000  LogSubscriber#logger
  1.84      0.00     0.00     0.00     0.00     1000 ::Fanout#listeners_for
  1.64      0.16     0.00     0.00     0.16        1  Integer#times
```

Rails 2-3-stable

```
Thread ID: 2148237740
Total: 0.051336

 %self     total     self     wait    child    calls  name
 27.00      0.03     0.01     0.00     0.01     1000  <Module::Benchmark>#realtime
 23.35      0.05     0.01     0.00     0.04     1000 ::AbstractAdapter#log
 11.67      0.01     0.01     0.00     0.00     2000  <Class::Time>#now
  7.66      0.03     0.00     0.00     0.03     1000  <Module::Benchmark>#ms
  5.37      0.00     0.00     0.00     0.00     1000 ::AbstractAdapter#log_info
  5.11      0.05     0.00     0.00     0.05        1  Integer#times
  4.52      0.00     0.00     0.00     0.00     2000  <Class::Time>#allocate
  3.89      0.00     0.00     0.00     0.00     2000  Time#initialize
  3.83      0.00     0.00     0.00     0.00     2000  Time#to_f
```

Methods in Common

- <Class::Time>#now
- <Class::Time>#allocate

n = 1000

	3.0 Beta	2-3 Stable
Time#now	4000	2000
Time#allocate	4000	2000

http://bit.ly/omgslow

"It's all fixed!"

Wait a few hours...

"It's better, but still 2x slower"

Post.find(1)

ARel....

find_by_sql()

execute()

log()

ARel....

Side note:
This is when Ryan told me to rewrite.

Superficial

Improvements

Limited Domain / System Knowledge

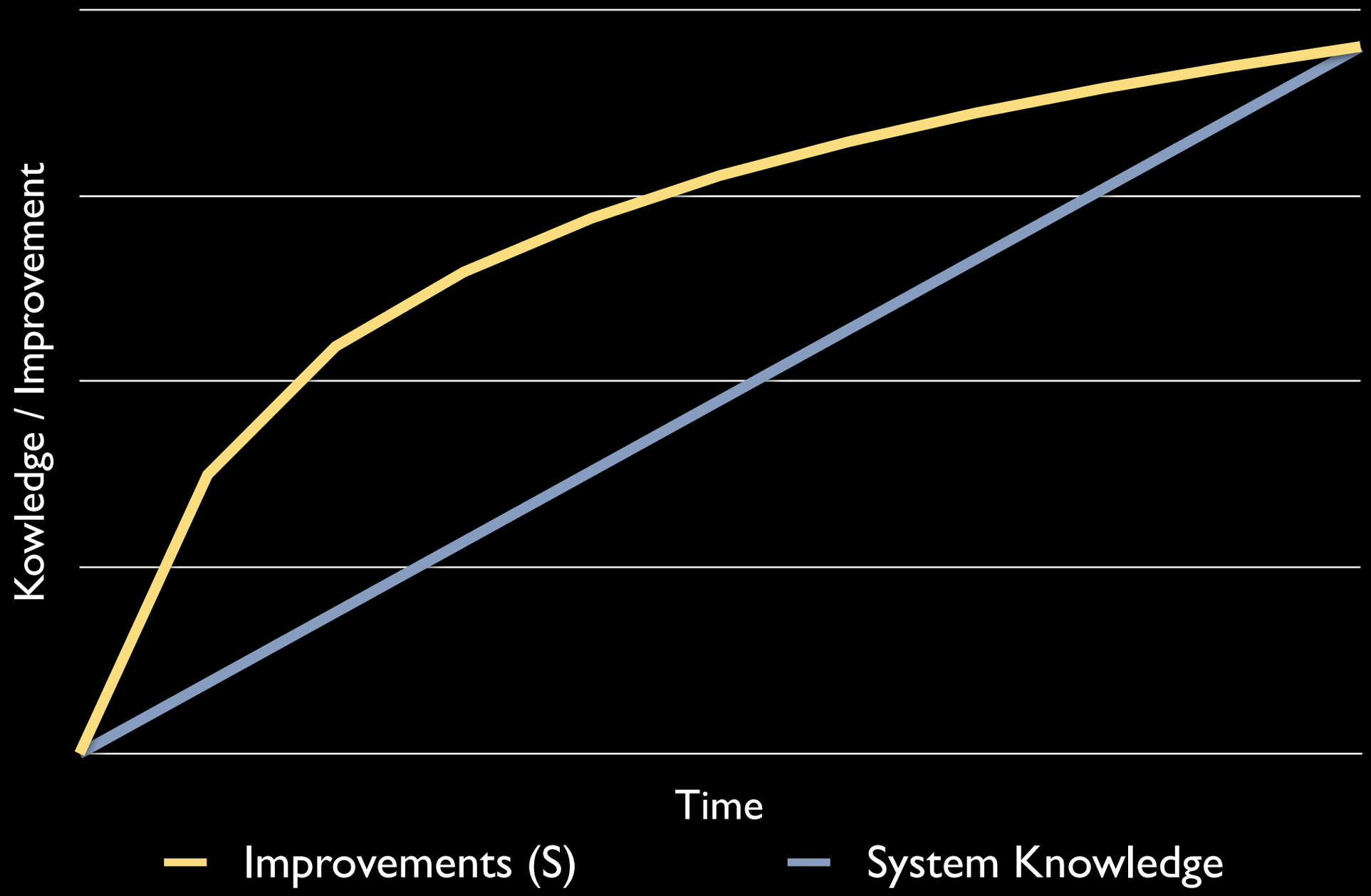
VM Tricks

See Results Quickly

Tapers off Over

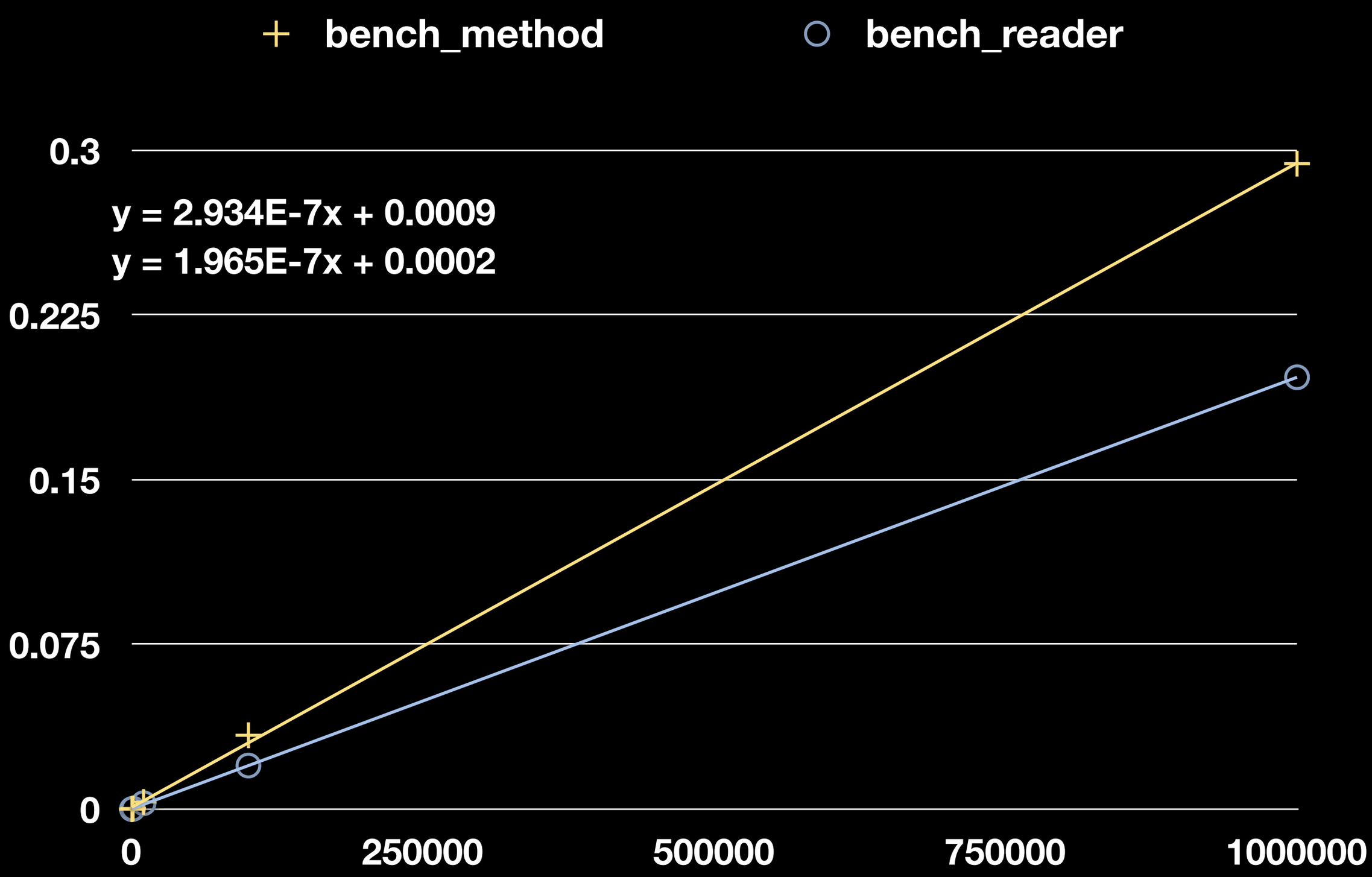
Time

System Impact



attr_*

```ruby
def some_attribute
  @some_attribute
end

# vs

attr_reader :some_attribute
```


bench_method
bench_reader
0.3
y = 2.934E-7x + 0.0009
y = 1.965E-7x + 0.0002
0.225
0.15
0.075
0
0
250000
500000
750000
1000000

attr_reader

```
case VM_METHOD_TYPE_IVAR: {
  if (argc != 0) {
    rb_raise(rb_eArgError,
     "wrong number of arguments (%d for 0)", argc);
  }
  val = rb_attr_get(recv, def->body.attr.id);
  break;
}
```

method call

```
case VM_METHOD_TYPE_ISEQ: {
  rb_control_frame_t *reg_cfp;
  int i;

  rb_vm_set_finish_env(th);
  reg_cfp = th->cfp;

  CHECK_STACK_OVERFLOW(reg_cfp, argc + 1);

  *reg_cfp->sp++ = recv;
  for (i = 0; i < argc; i++) {
      *reg_cfp->sp++ = argv[i];
  }

  vm_setup_method(th, reg_cfp, recv, argc, blockptr, 0 /* flag */, me);
  val = vm_exec(th);
  break;
}
```

vm_setup_method

- Check for stack overflow
- Pushing a stack frame
- Copying arguments

Predicate Methods

```
class Foo
  attr_reader :some_attribute

  def some_attribute?
    @some_attribute
  end
end
```

Predicate Methods

```
class Foo
  attr_reader :some_attribute
  alias :some_attribute? :some_attribute
end
```

Hash[] vs inject({})

inject({})

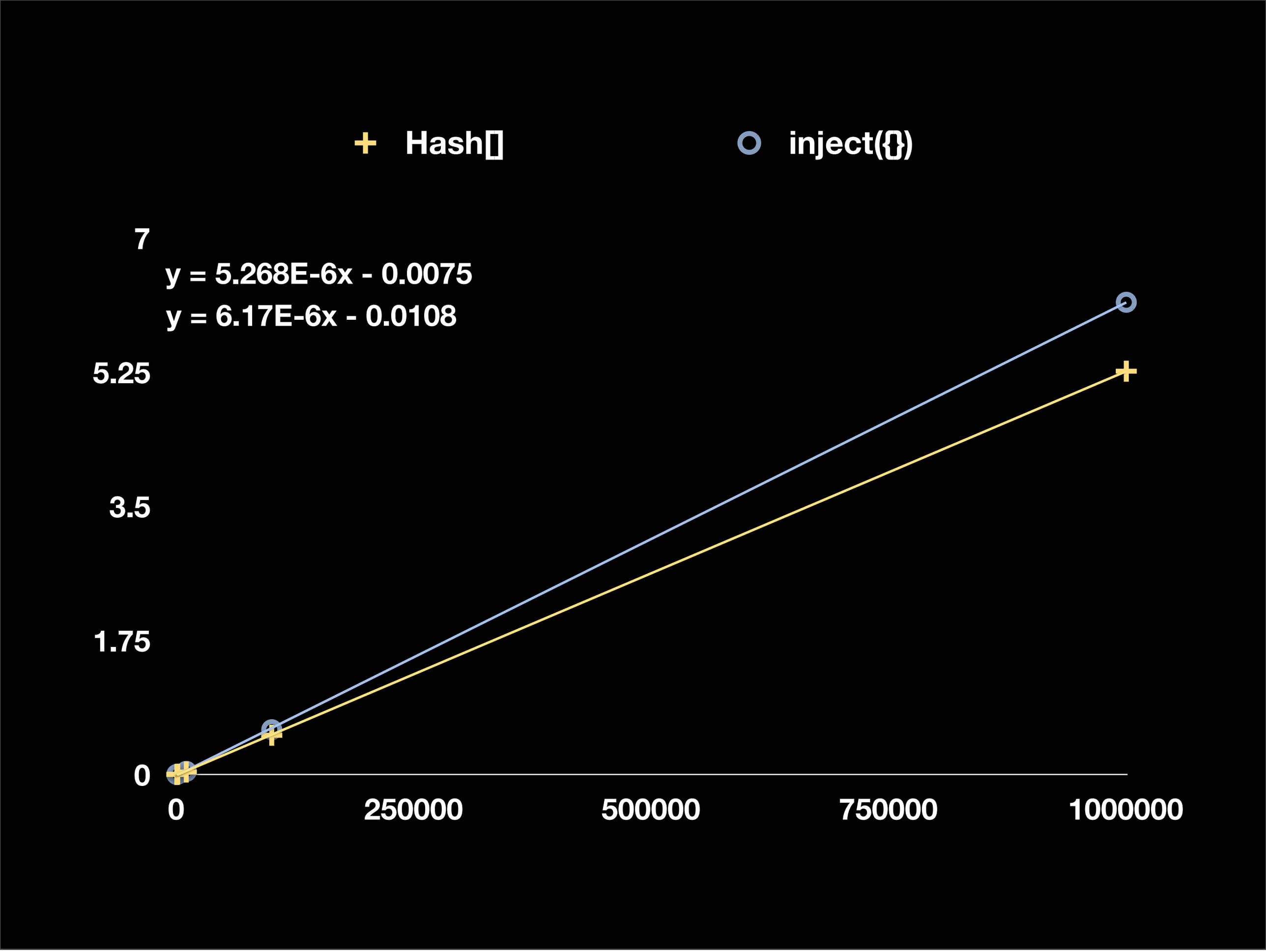
```
some_list.inject({}) do |hash,val|
  hash[val] = some_transform(val)
  hash
end
```

Hash[]

```
values = some_list.map { |val|
  [val, some_transform(val)]
}
Hash[values]
```

```
@list.inject({}) do |hash,val|
  hash[val] = val.length
  hash
end
```

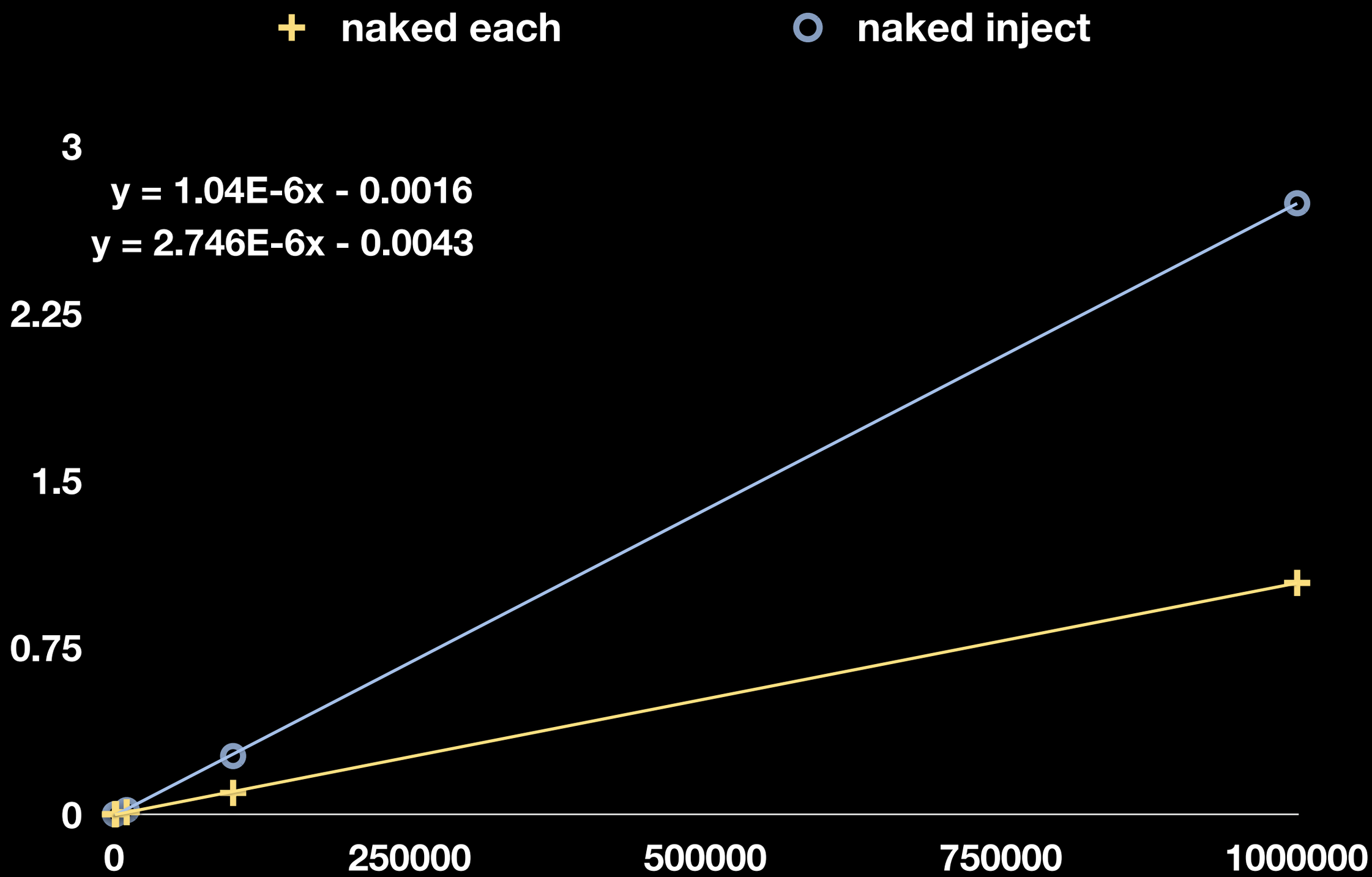
```
# vs

Hash[@list.map { |val| [val,val.length] }]
```


Hash[]
inject({})
y = 5.268E-6x - 0.0075
y = 6.17E-6x - 0.0108
7
5.25
3.5
1.75
0
0
250000
500000
750000
1000000

Strangeness

```
def bench_naked_each
  assert_performance_linear 0.999 do |n|
    m = nil
    n.times { @list.each { |v| m = v } }
  end
end
```

```
def bench_naked_inject
  assert_performance_linear 0.999 do |n|
    n.times { @list.inject { |m,v| m = v } }
  end
end
```


naked each
naked inject
y = 1.04E-6x - 0.0016
y = 2.746E-6x - 0.0043
3
2.25
1.5
0.75
0
0
250000
500000
750000
1000000

TANGENT

When to use inject()

When one calculation depends on the previous

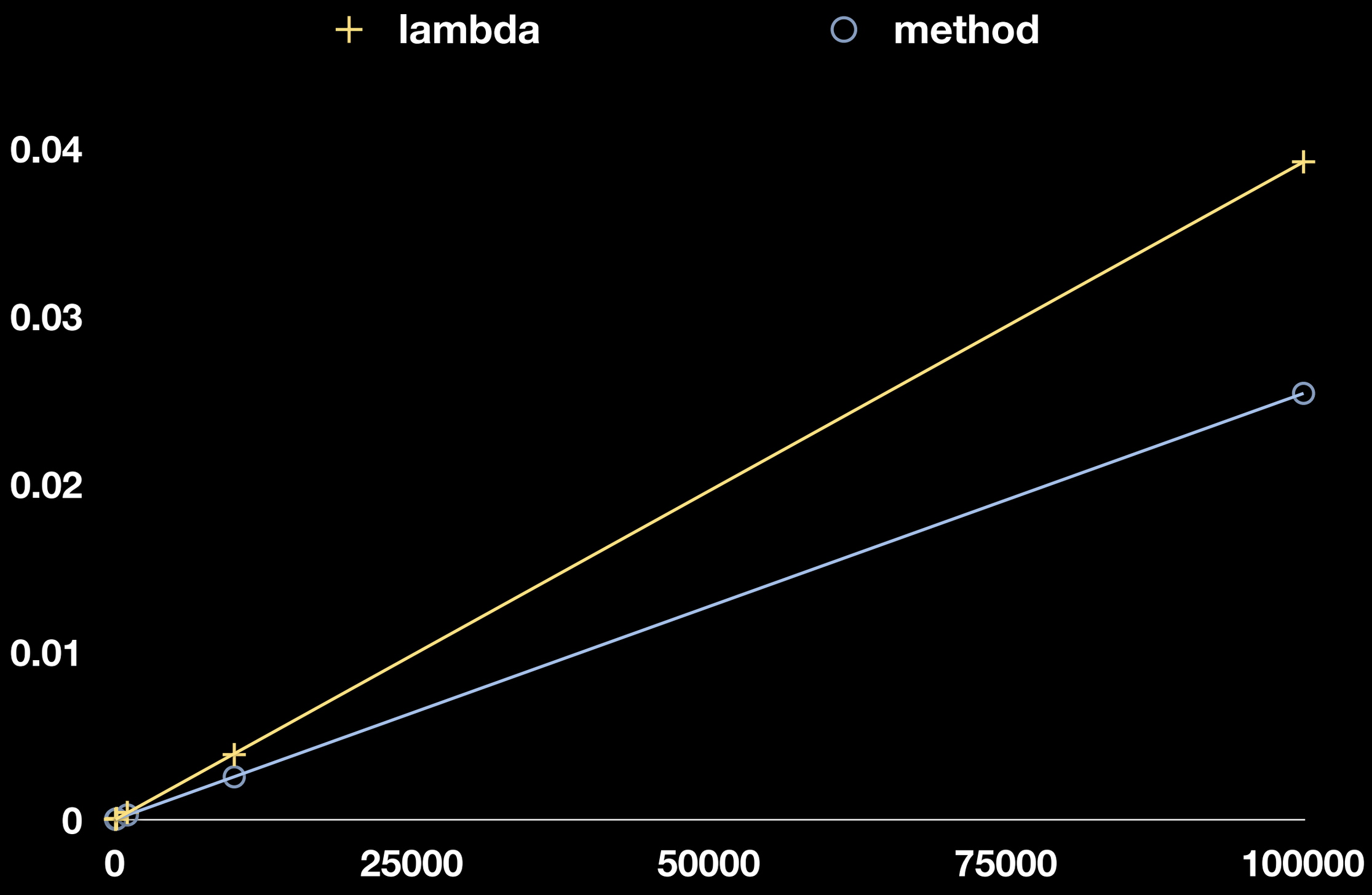
```
@list.inject({}) do |hash,val|
  hash[val] = val.length
  hash
end
```

```
# vs

Hash[@list.map { |val| [val,val.length] }]
```

```
%w{
  Foo
  Bar
  Baz
}.inject(Object) { |klass,string|
  klass.const_get(string.to_sym)
}
```

Proc Activation

```
lambda { ... }

# vs

class Callable
  def call; ... end
end
```


lambda
method
0.04
0.03
0.02
0.01
0
0
25000
50000
75000
100000

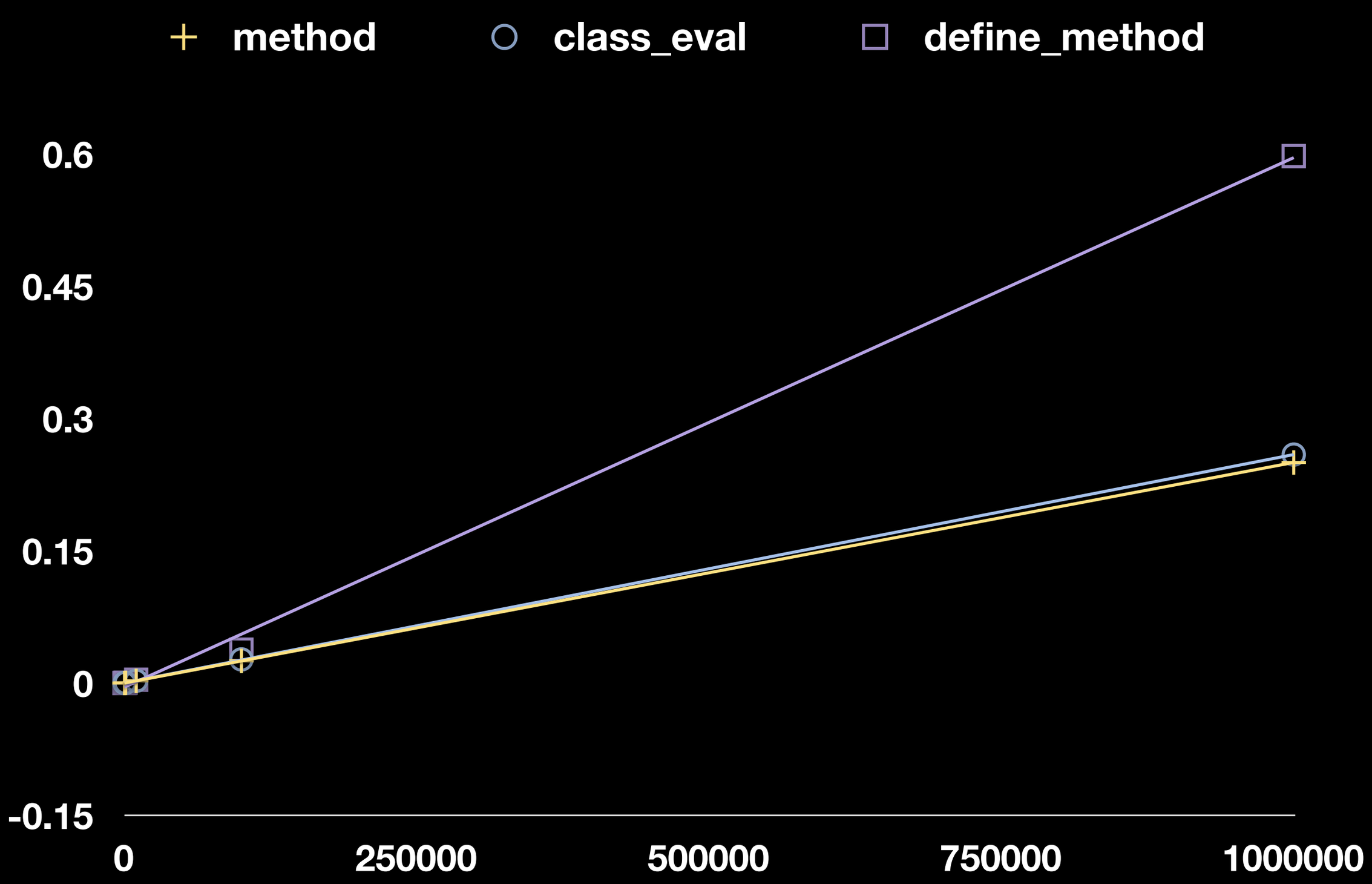
is_a?(Proc)

respond_to?(:call)

```
class Callable
  def call(...)
    ...
  end
end
```

define_method

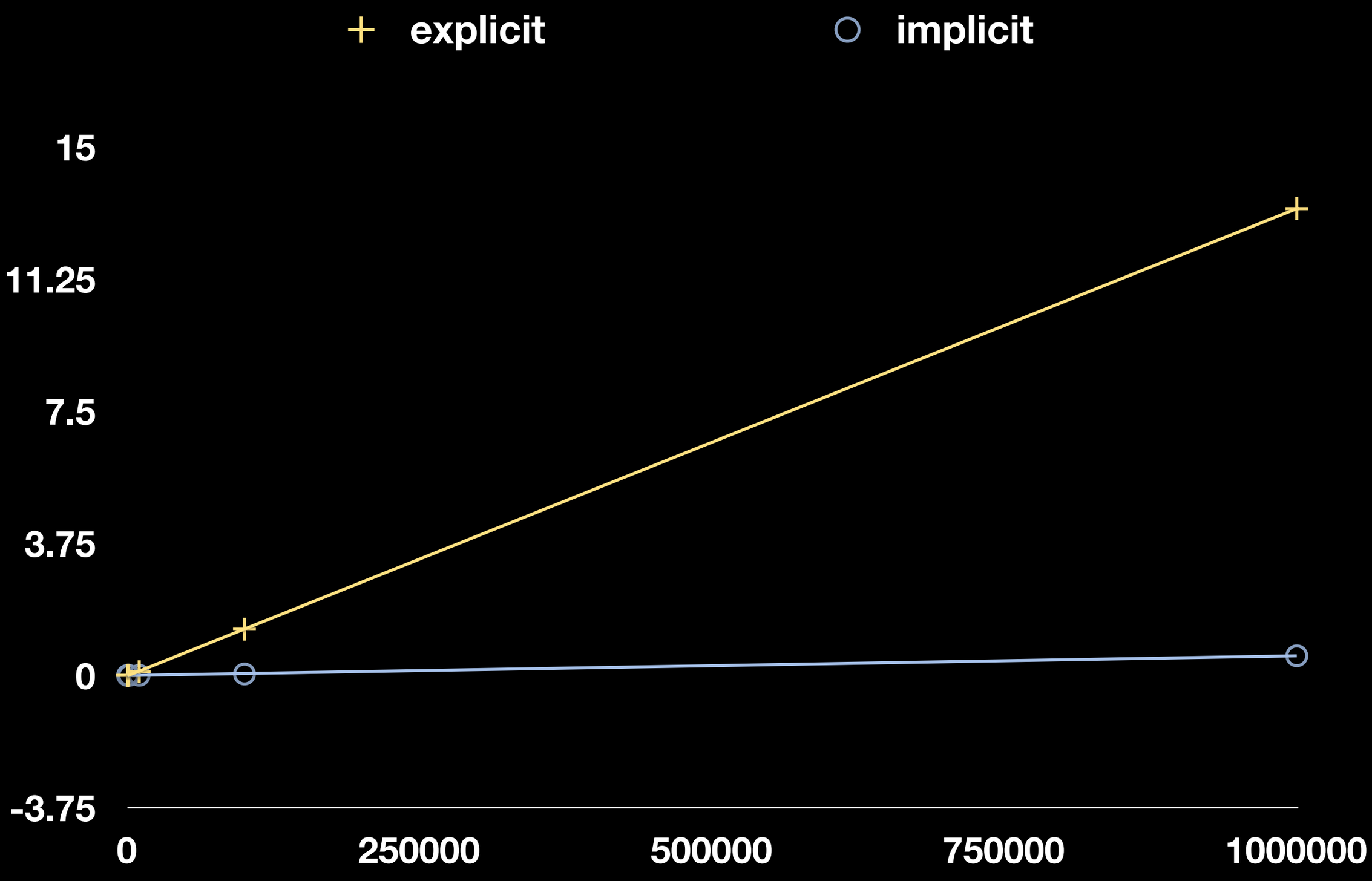
```
class Foo
  def foo; end
  define_method :bar do; end
  class_eval %{ def baz; end }
end
```


method
class_eval
define_method
0.6
0.45
0.3
0.15
0
-0.15
0
250000
500000
750000
1000000

Explicit Block

Parameters

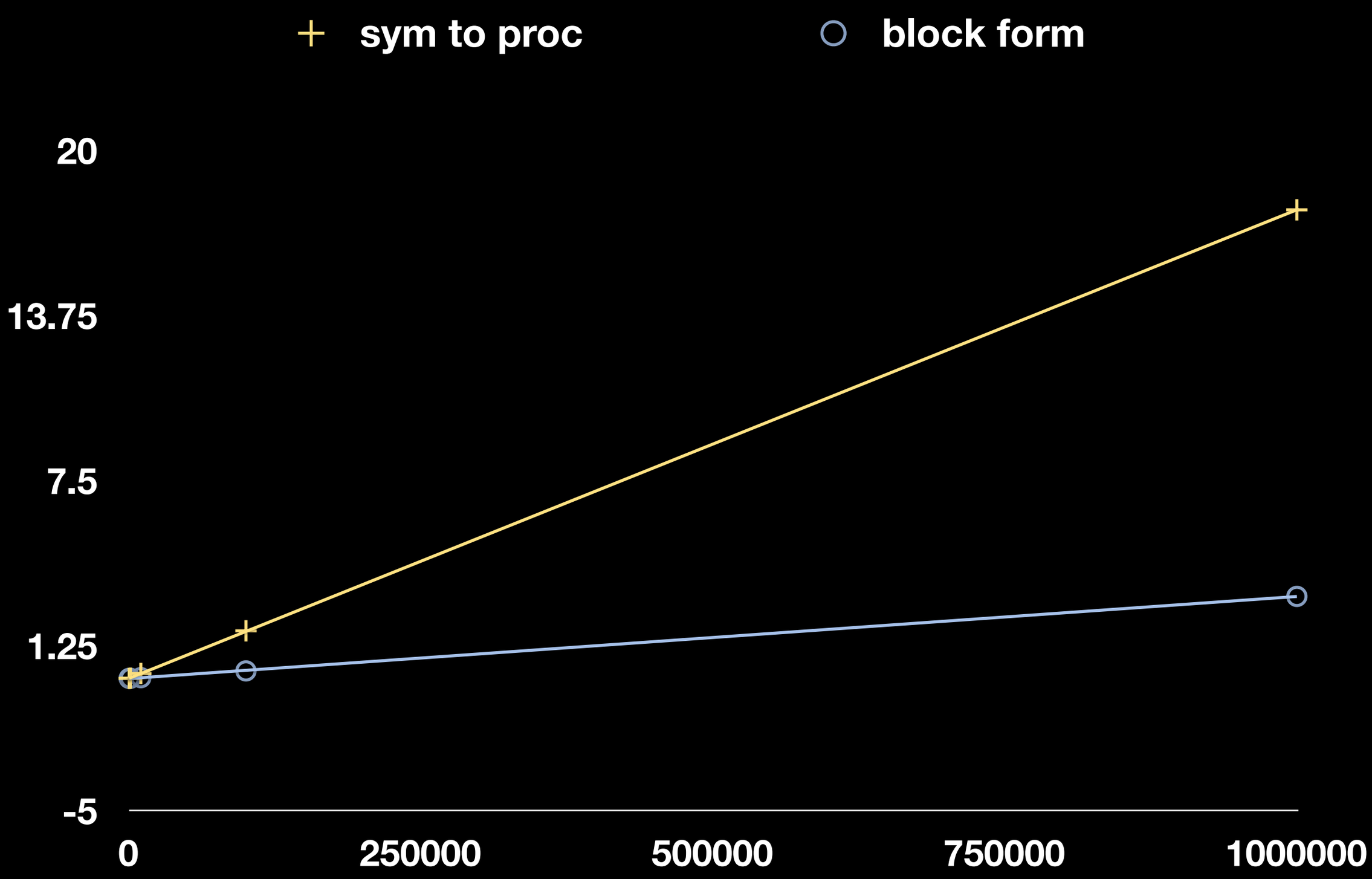
```
class Foo
  def explicit &block
    yield
  end

  def implicit
    yield
  end
end
```

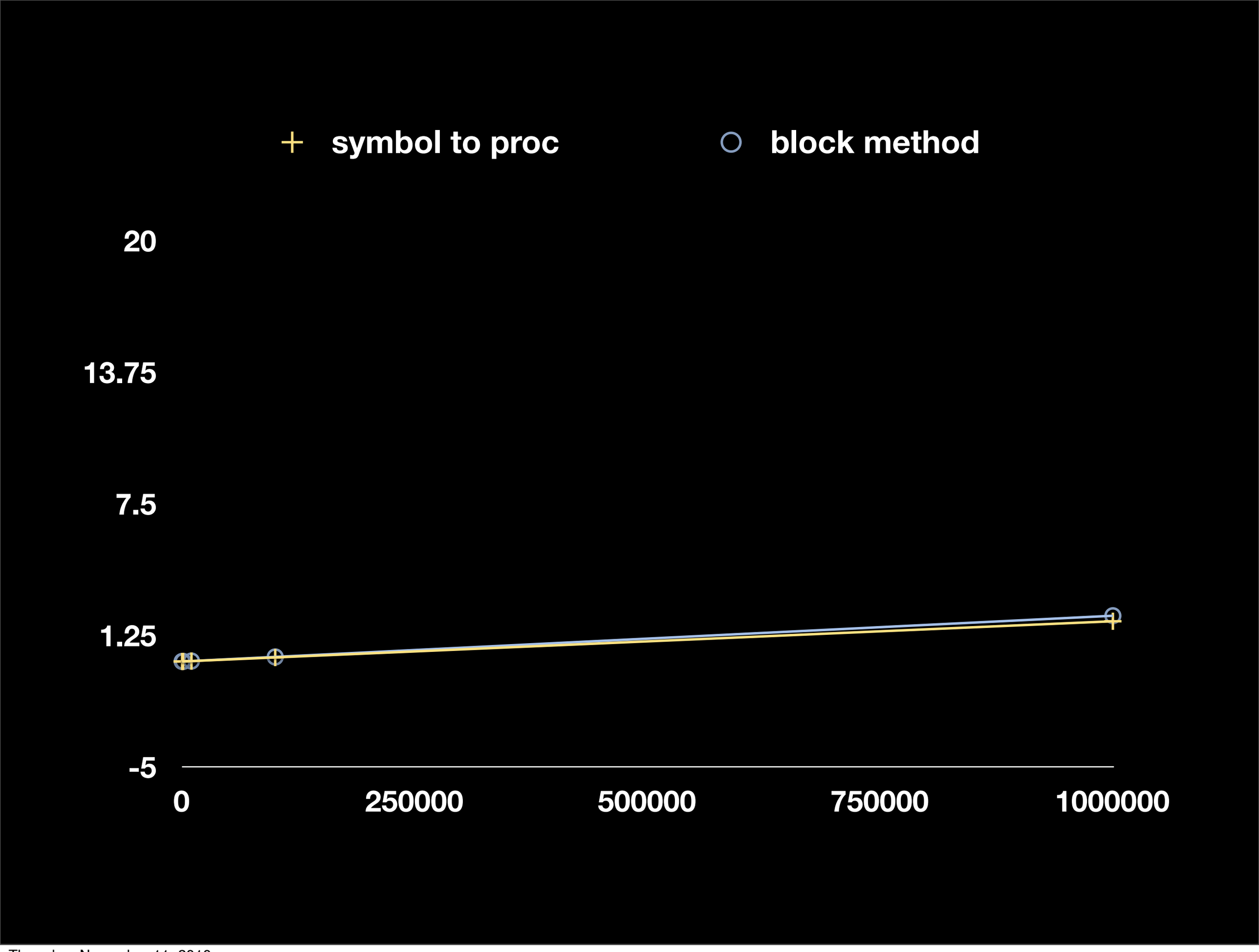

explicit
implicit
15
11.25
7.5
3.75
0
-3.75
0
250000
500000
750000
1000000

```
def sometimes_block
  if block_given?
    Proc.new.call
  end
end

sometimes_block { puts "hi" }
sometimes_block
```

Symbol to Proc

```
@list.map(&:to_i)

# vs

@list.map { |x| x.to_i }
```


sym to proc
block form
20
13.75
7.5
1.25
-5
0
250000
500000
750000
1000000


symbol to proc
block method
20
13.75
7.5
1.25
-5
0
250000
500000
750000
1000000

Know Your

Audience

return value

caching

```
def some_method
  @some_method ||= some_expensive_op
end
```

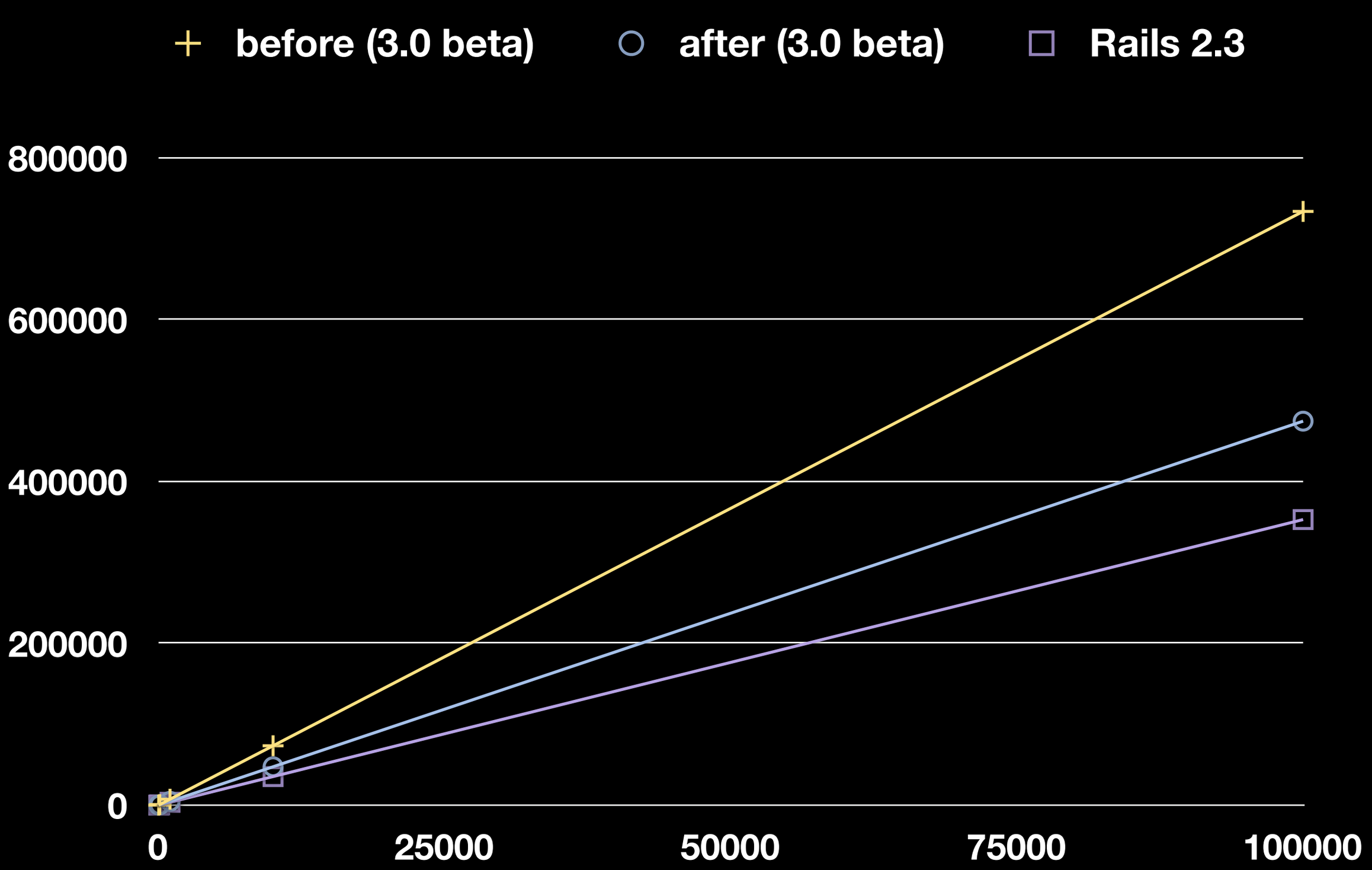
How many times?

Can the caller cache?

Made our improvements

Feeling better!

:-D


before (3.0 beta)
after (3.0 beta)
Rails 2.3
800000
600000
400000
200000
0
0
25000
50000
75000
100000

What do we do?

We have to go deeper

```
$ git grep 'include Relation' | wc -l
       6
$ git grep 'def bind' | wc -l
      12
$
```

Everything is_a

Relation

Everything

Responds to "bind"

Everything has a "relation"

bind() is recursively called on relation

How does it work?

Arel
Attribute#to_sql
14 (0.4%)
of 114 (3.0%)

5

24

Arel
e#bind
(0.1%)
(1.2%)

18

43

e_for
.5%)
.0%)

Class#new
632 (16.7%)
of 806 (21.2%)

32

49

149

Arel
Take#initialize
32 (0.8%)

Arel
Project#initialize
60 (1.6%)
of 149 (3.9%)

37

46

42

46

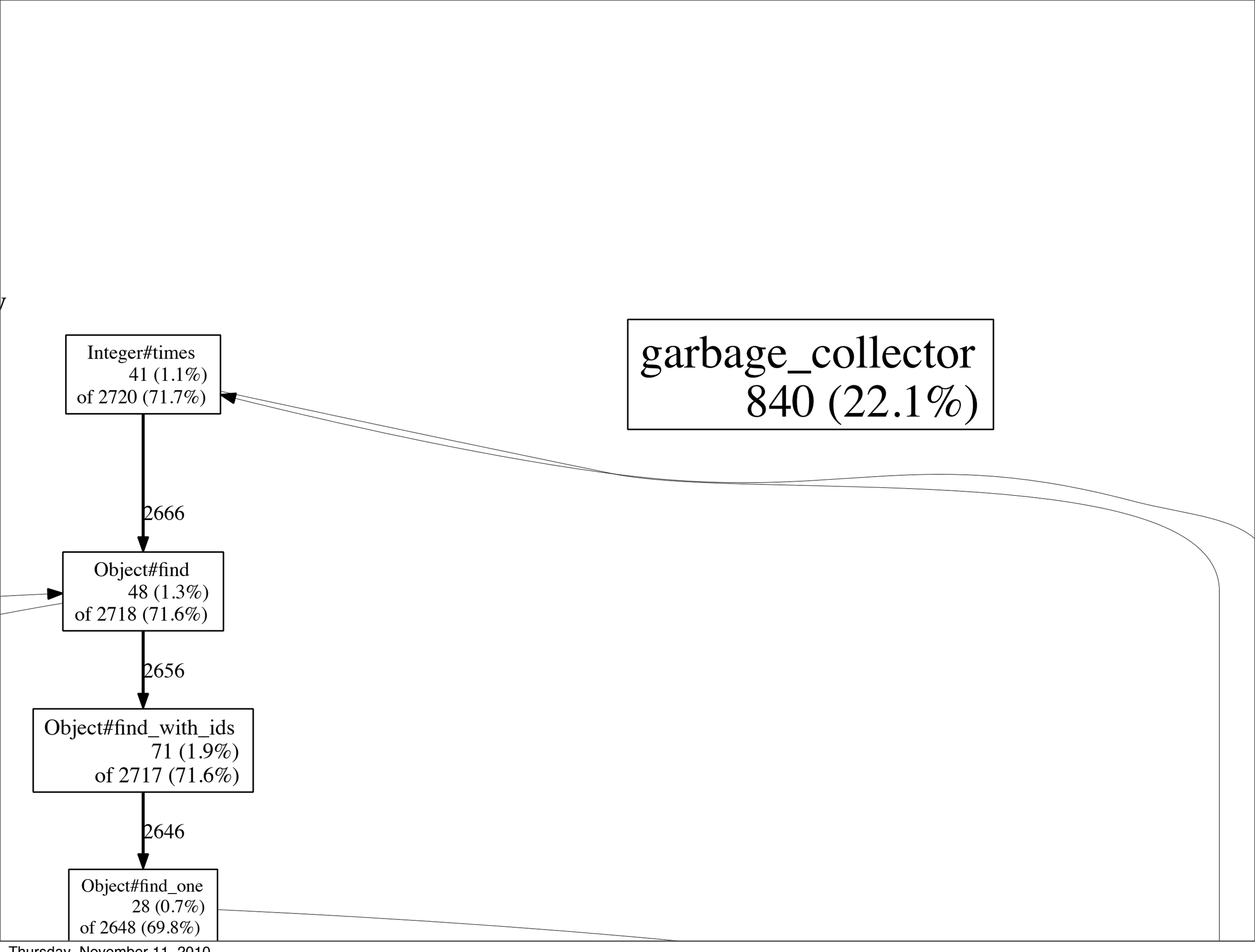
36

Array#map
13 (0.3%)
of 136 (3.6%)

17

14

Arel
Where#initialize


garbage_collector
840 (22.1%)
Integer#times
41 (1.1%)
of 2720 (71.7%)
2666
Object#find
48 (1.3%)
of 2718 (71.6%)
2656
Object#find_with_ids
71 (1.9%)
of 2717 (71.6%)
2646
Object#find_one
28 (0.7%)
of 2648 (69.8%)

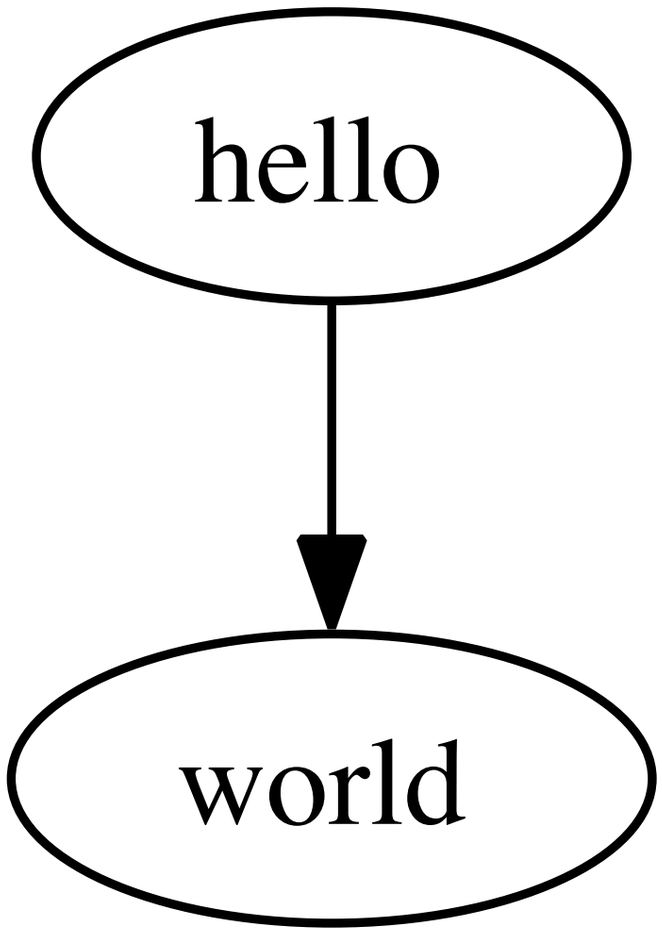
Data Structure

Analysis

Graphviz

graphviz.org

```
digraph "foo" {
  node [width=0.375,height=0.25];
  N1 [label="hello"];
  N2 [label="world"];
  N1 -> N2;
}
```

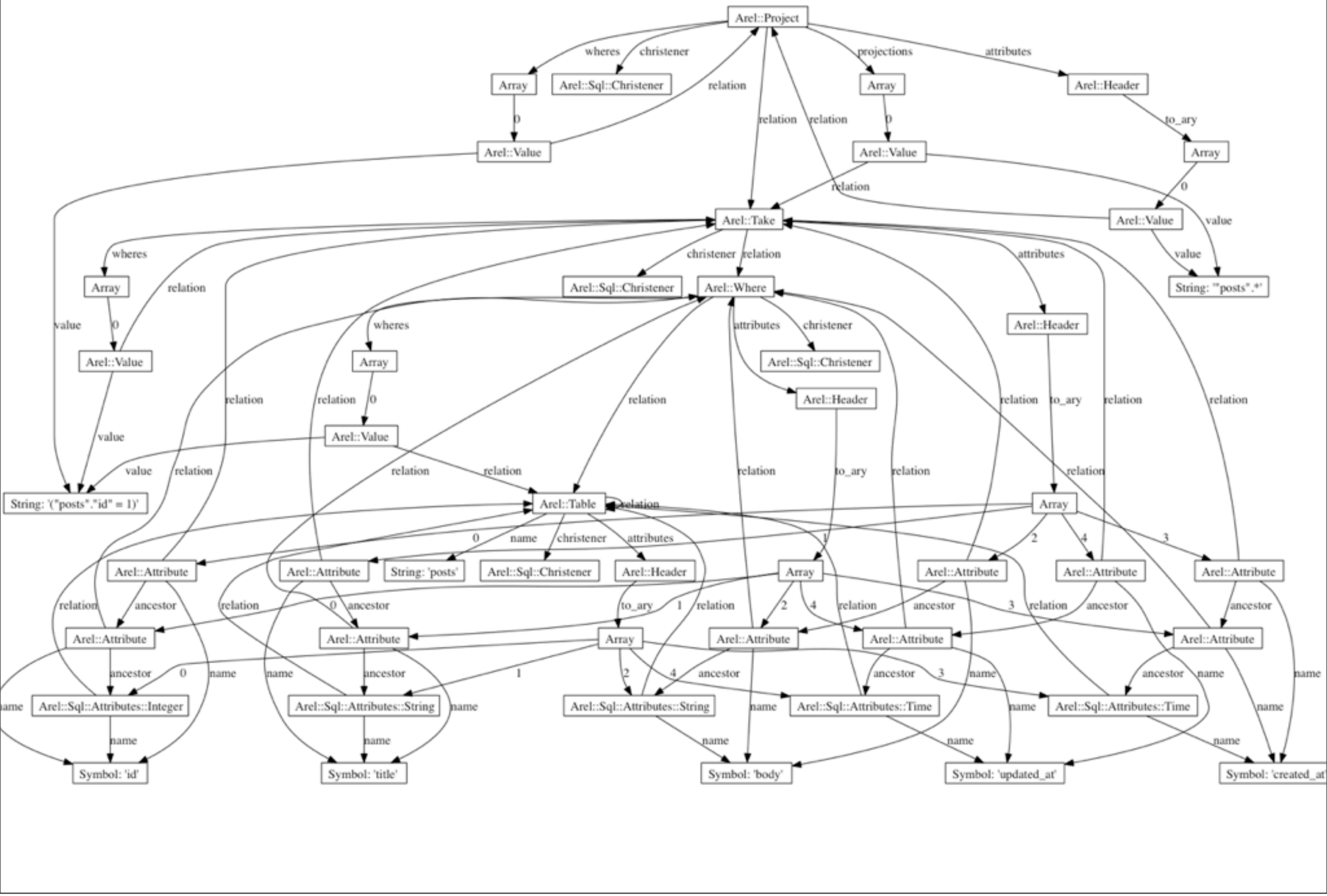

hello
world

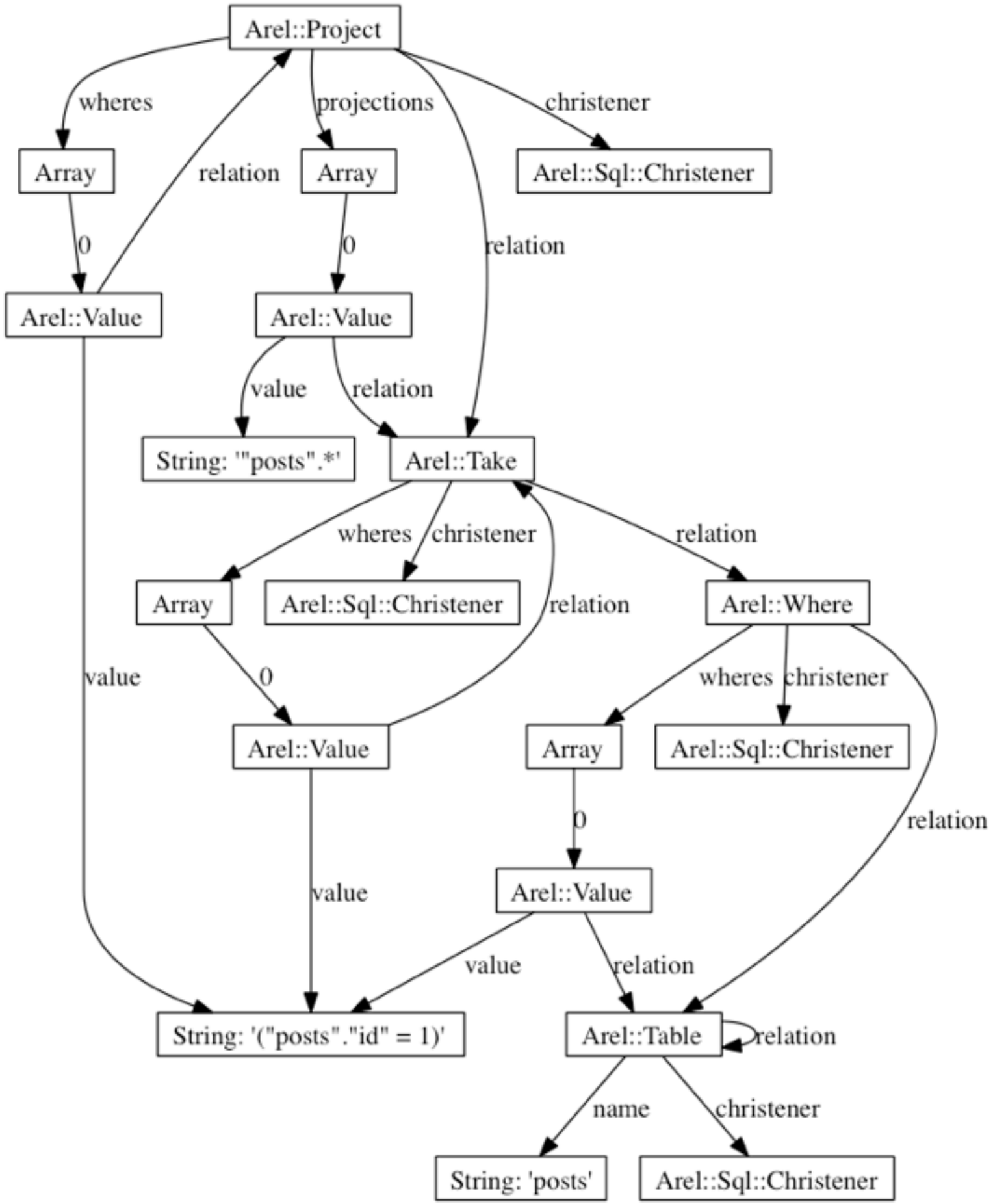
Visitor Pattern

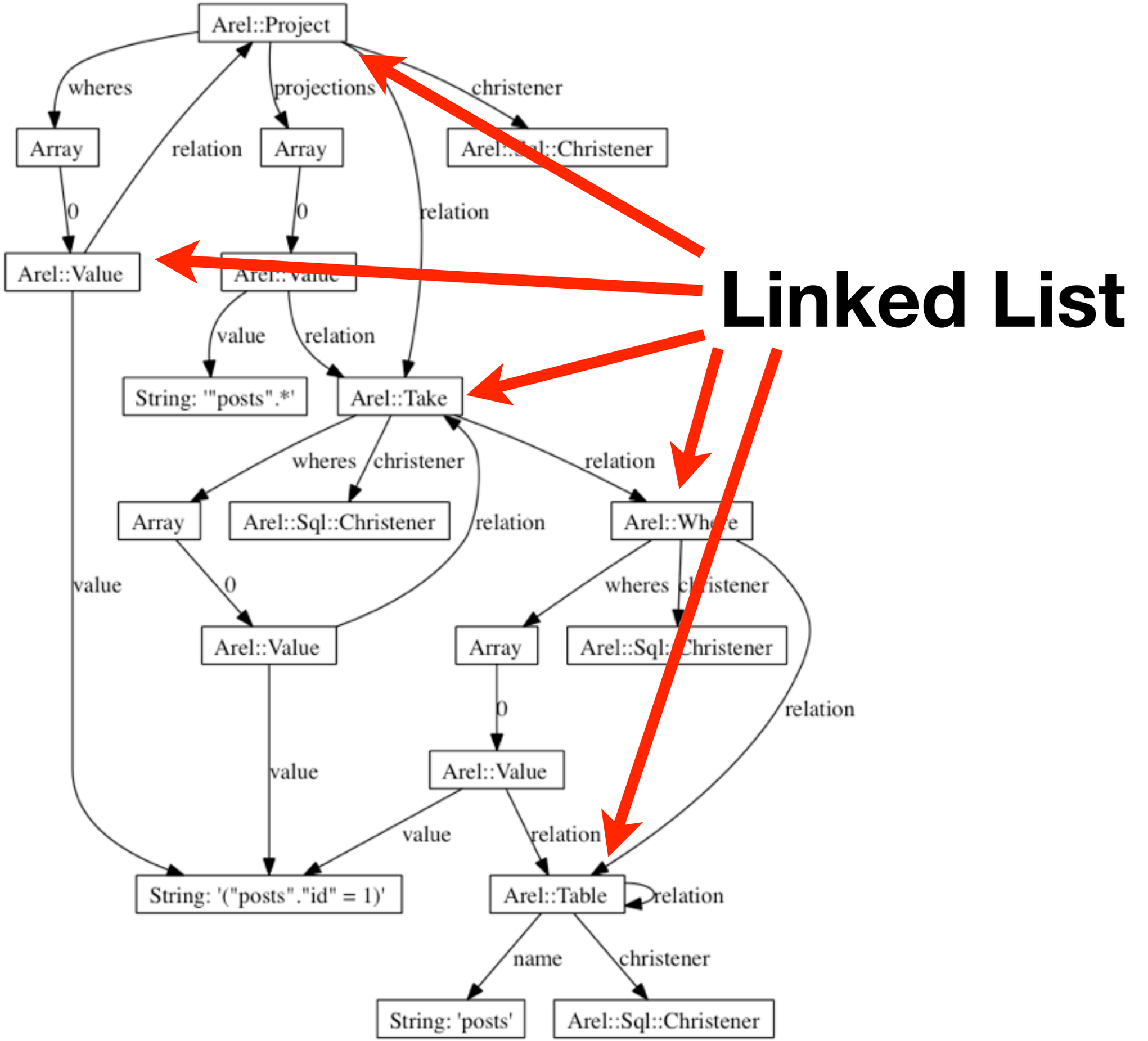
```
class Visitor
  def accept(object)
    method = object.class.name.split('::').join('_')
    send("visit_#{method}", object)
  end
end
```

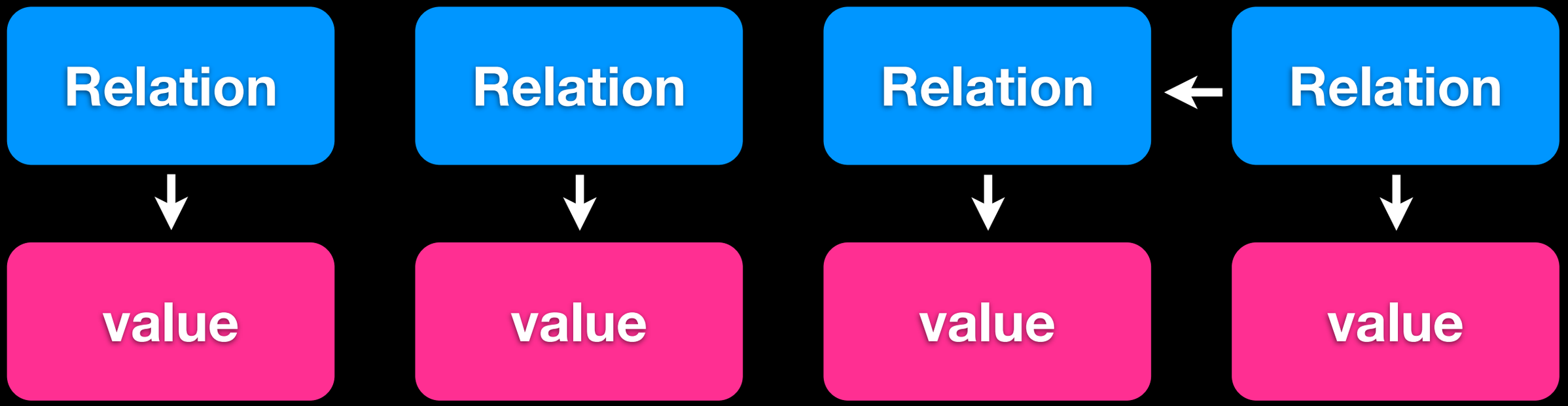
```
class Visitor
  def accept(object)
    method = object.class.name.split('::').join('_')
    send("visit_#{method}", object)
  end

  def visit_Arel_Alias(node)
    # keep track of the node called
    accept(node.attribute)
  end
end
```

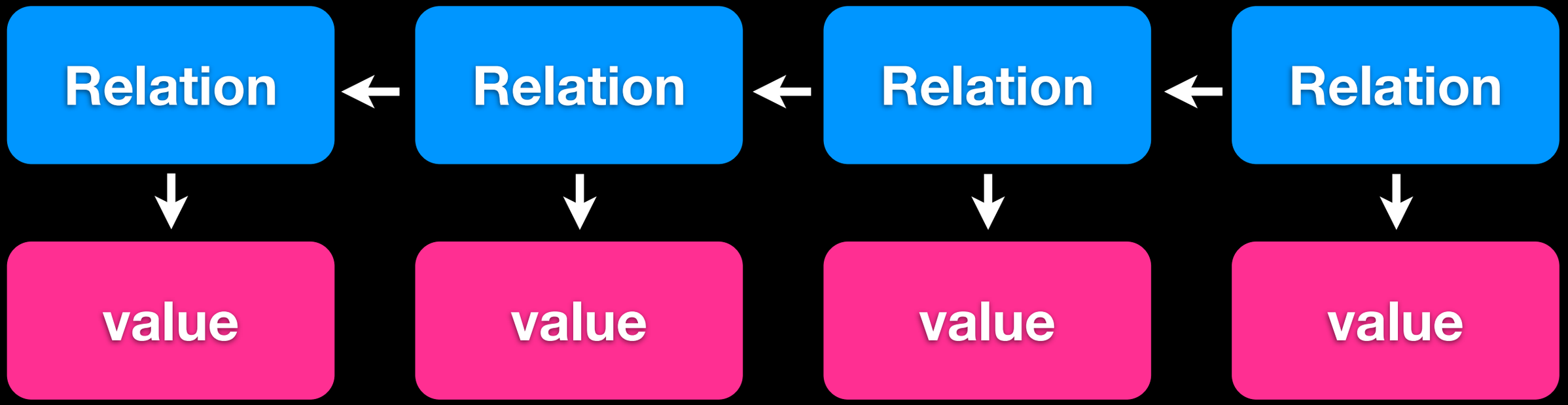
```ruby
class Visitor
  def accept(object)
    method = object.class.name.split('::').join('_')
    send("visit_#{method}", object)
  end

  def visit_Arel_Alias(node)
    # keep track of the node called
    accept(node.attribute)
  end

  def visit_Arel_Table(node)
    # keep track of the node called
    accept(node.name)
    node.columns.each { |c| accept(c) }
  end
end
```

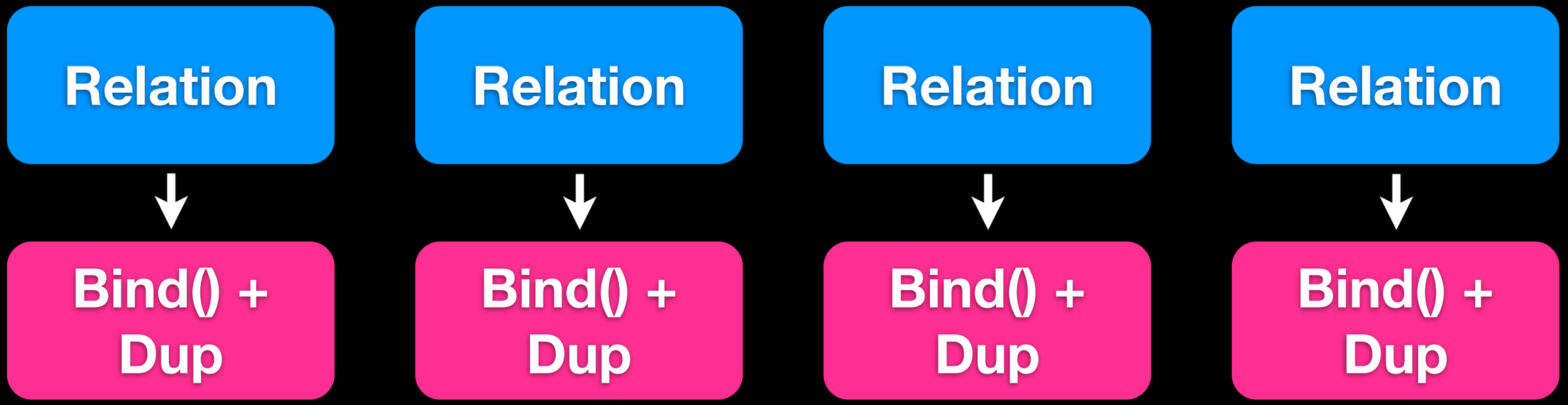

Arel::Project
wheres
christener
projections
attributes
Array
Arel::Sql::Christener
Arel::Header
relation
0
to_ary
Arel::Value
Array
Arel::Take
value
String: '"posts".*'
Arel::Where
Arel::Sql::Christener
Arel::Header
Array
Arel::Value
String: '("posts"."id" = 1)'
Arel::Table
String: 'posts'
name
Arel::Attribute
ancestor
Arel::Sql::Attributes::Integer
Arel::Sql::Attributes::String
Arel::Sql::Attributes::Time
Symbol: 'id'
Symbol: 'title'
Symbol: 'body'
Symbol: 'updated_at'
Symbol: 'created_at'

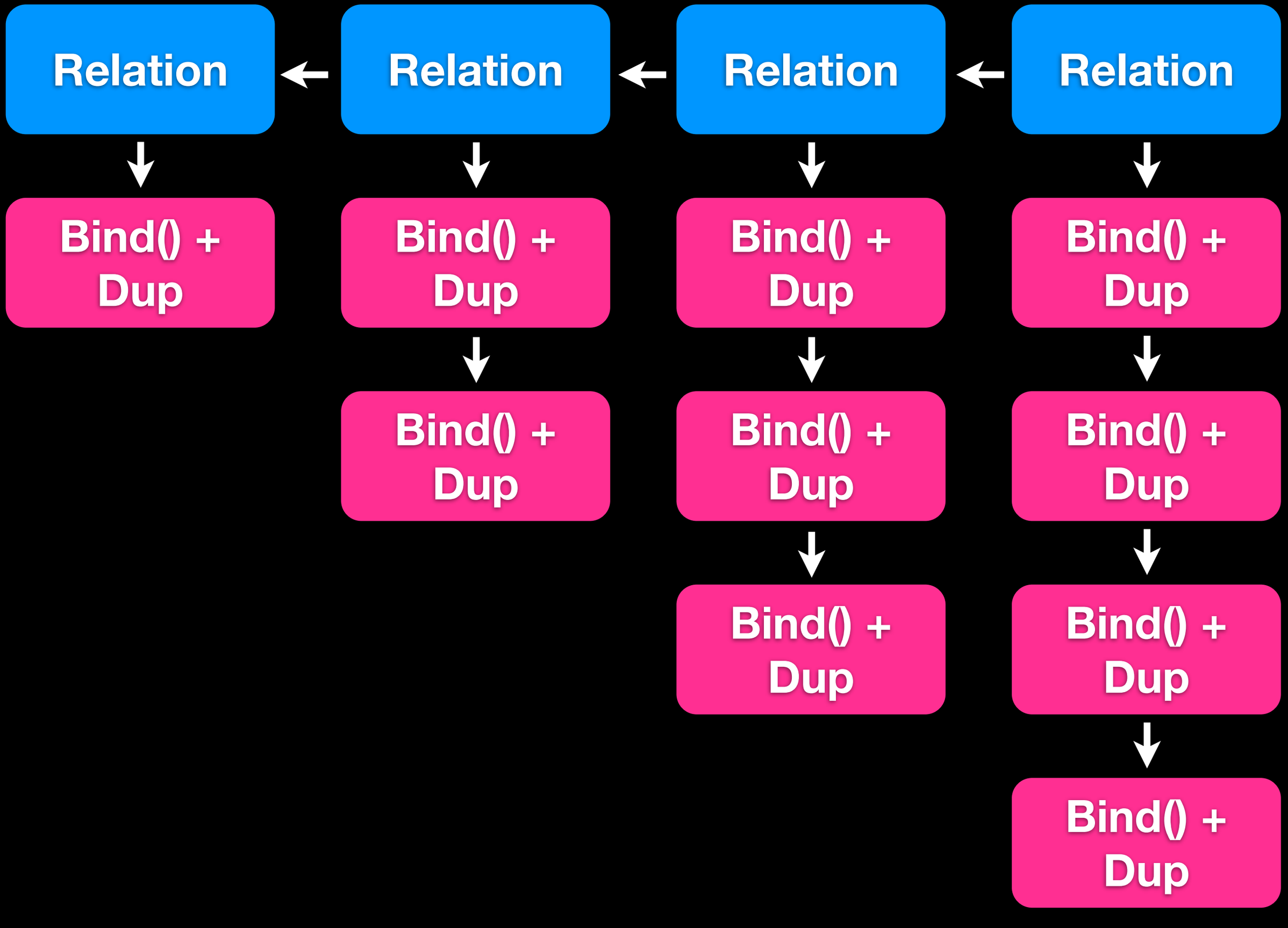

Arel::Project
wheres
projections
christener
Array
relation
Array
Arel::Sql::Christener
0
0
relation
Arel::Value
Arel::Value
value
relation
String: '"posts".*'
Arel::Take
wheres
christener
relation
Array
Arel::Sql::Christener
relation
Arel::Where
value
0
wheres
christener
Arel::Value
Array
Arel::Sql::Christener
0
relation
value
Arel::Value
value
relation
String: '("posts"."id" = 1)'
Arel::Table
relation
name
christener
String: 'posts'
Arel::Sql::Christener

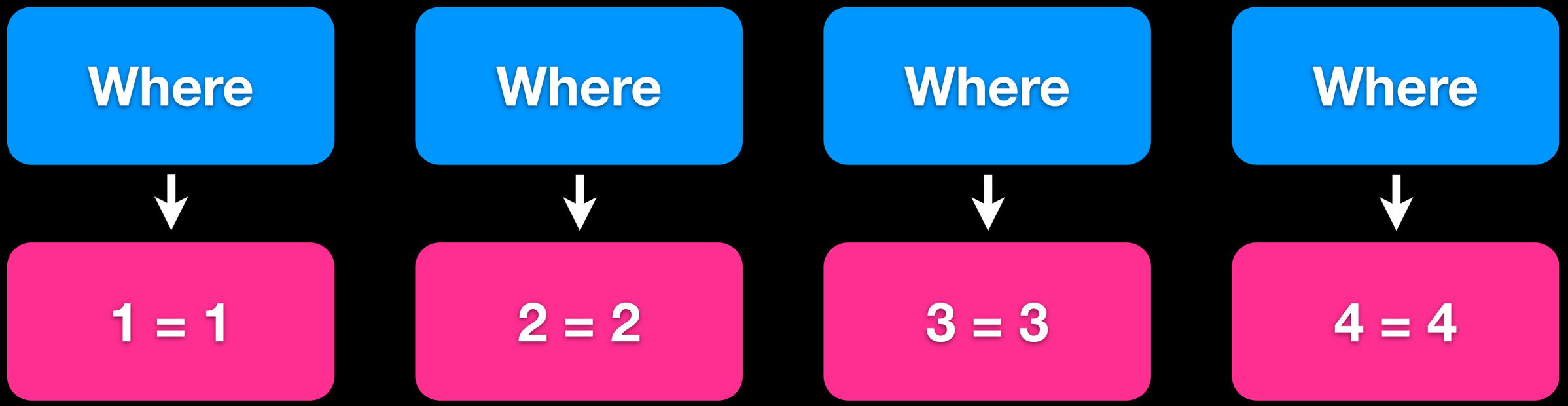

Arel::Project
wheres
projections
christener
Array
relation
Array
Arel::Sql::Christener
0
0
relation
Arel::Value
Arel::Value
value
relation
String: '"posts".*'
Arel::Take
wheres
christener
relation
Array
Arel::Sql::Christener
relation
Arel::Where
value
0
wheres
christener
Arel::Value
Array
Arel::Sql::Christener
0
relation
value
Arel::Value
value
relation
String: '("posts"."id" = 1)'
Arel::Table
relation
name
christener
String: 'posts'
Arel::Sql::Christener
Linked List

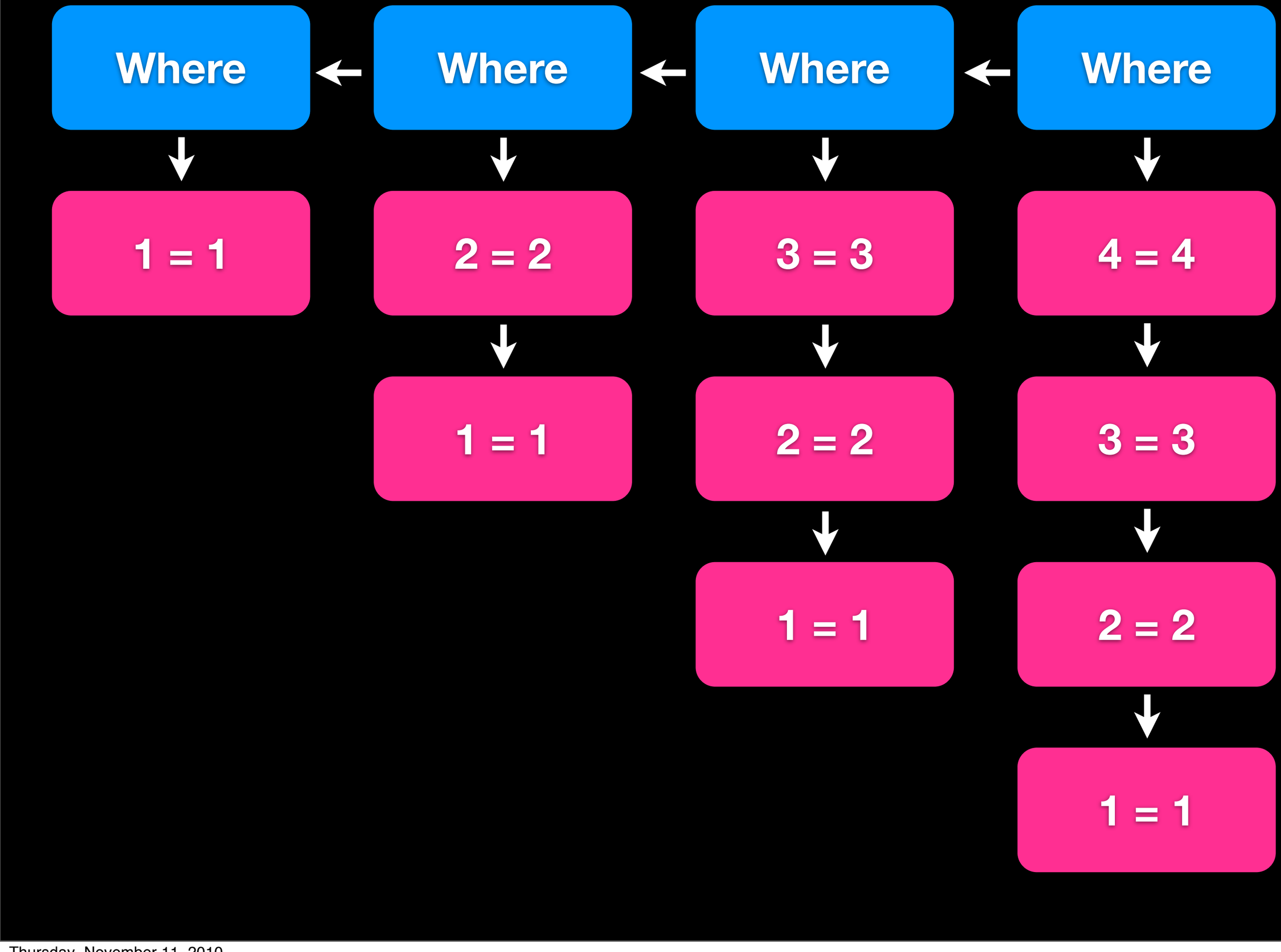

Relation
Relation
Relation
Relation
value
value
value
value


Relation
Relation
Relation
Relation
value
value
value
value


Relation
Relation
Relation
Relation
Bind() + Dup
Bind() + Dup
Bind() + Dup
Bind() + Dup


Relation
Relation
Relation
Relation
Bind() + Dup
Bind() + Dup
Bind() + Dup
Bind() + Dup
Bind() + Dup
Bind() + Dup
Bind() + Dup
Bind() + Dup
Bind() + Dup


Where
Where
Where
Where
1 = 1
2 = 2
3 = 3
4 = 4


Where
Where
Where
Where
1 = 1
2 = 2
3 = 3
4 = 4
1 = 1
2 = 2
3 = 3
1 = 1
2 = 2
1 = 1

Big O!

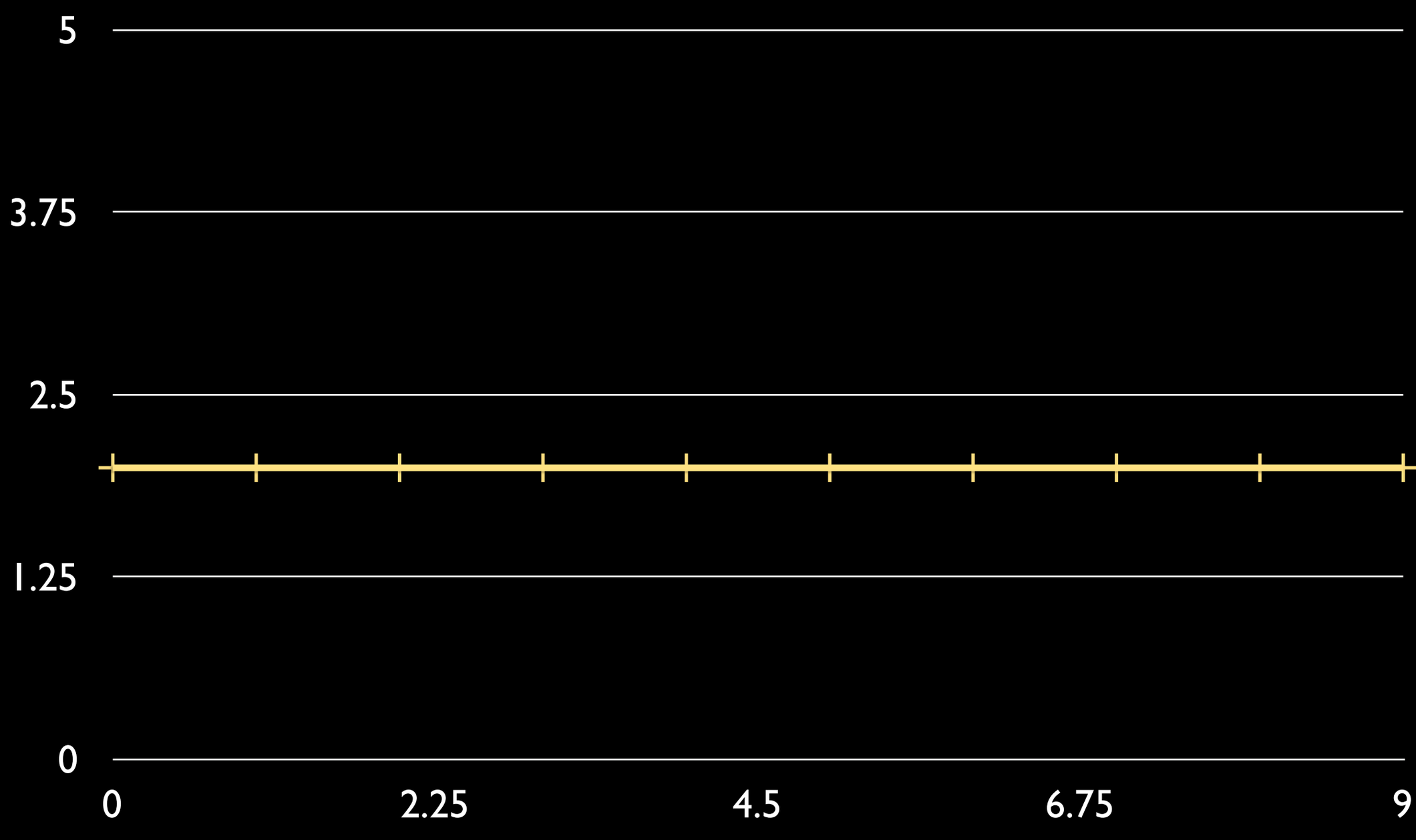
OMG

ZOMG

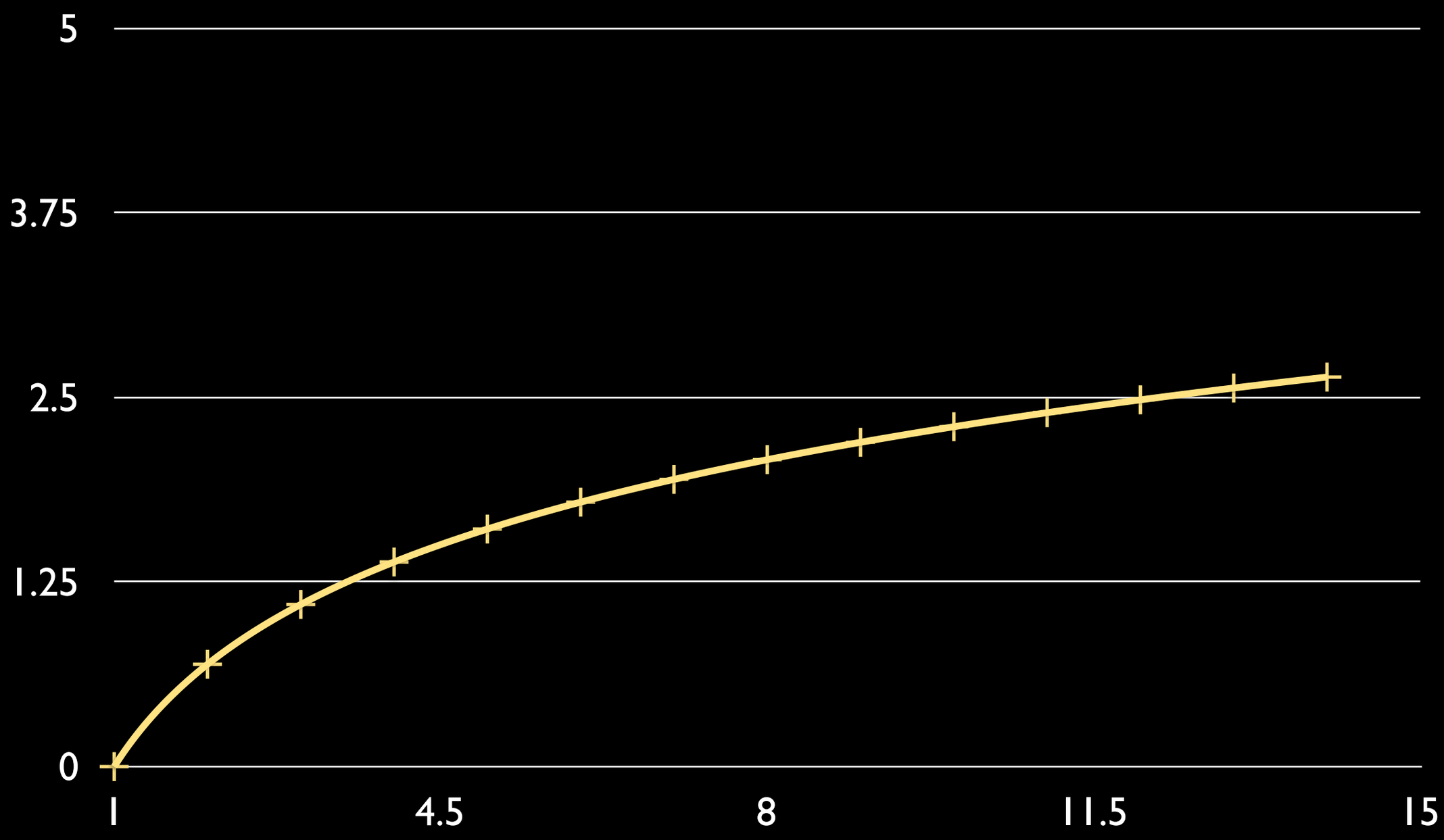
Mathematical

Representation

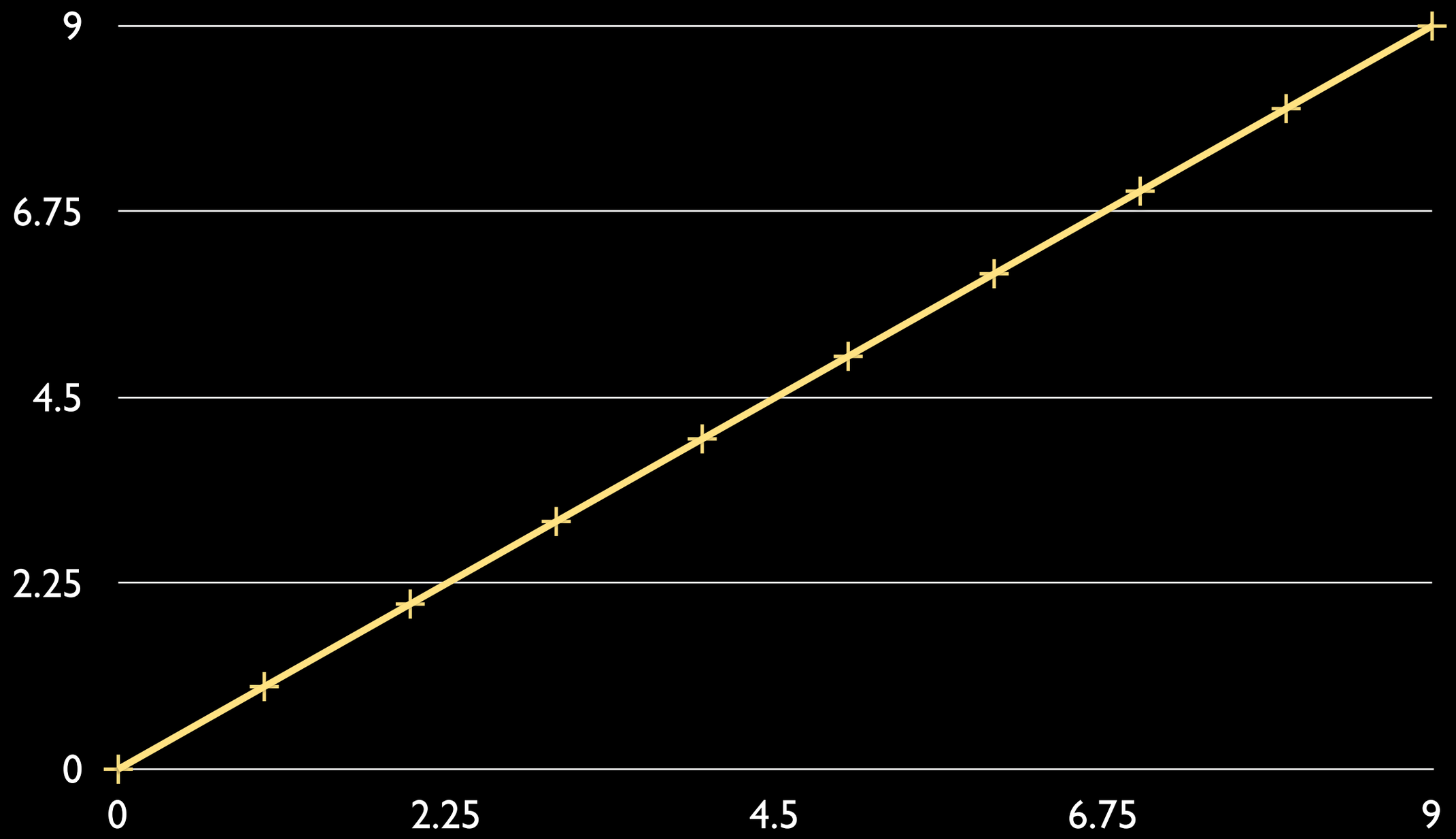
y = 2



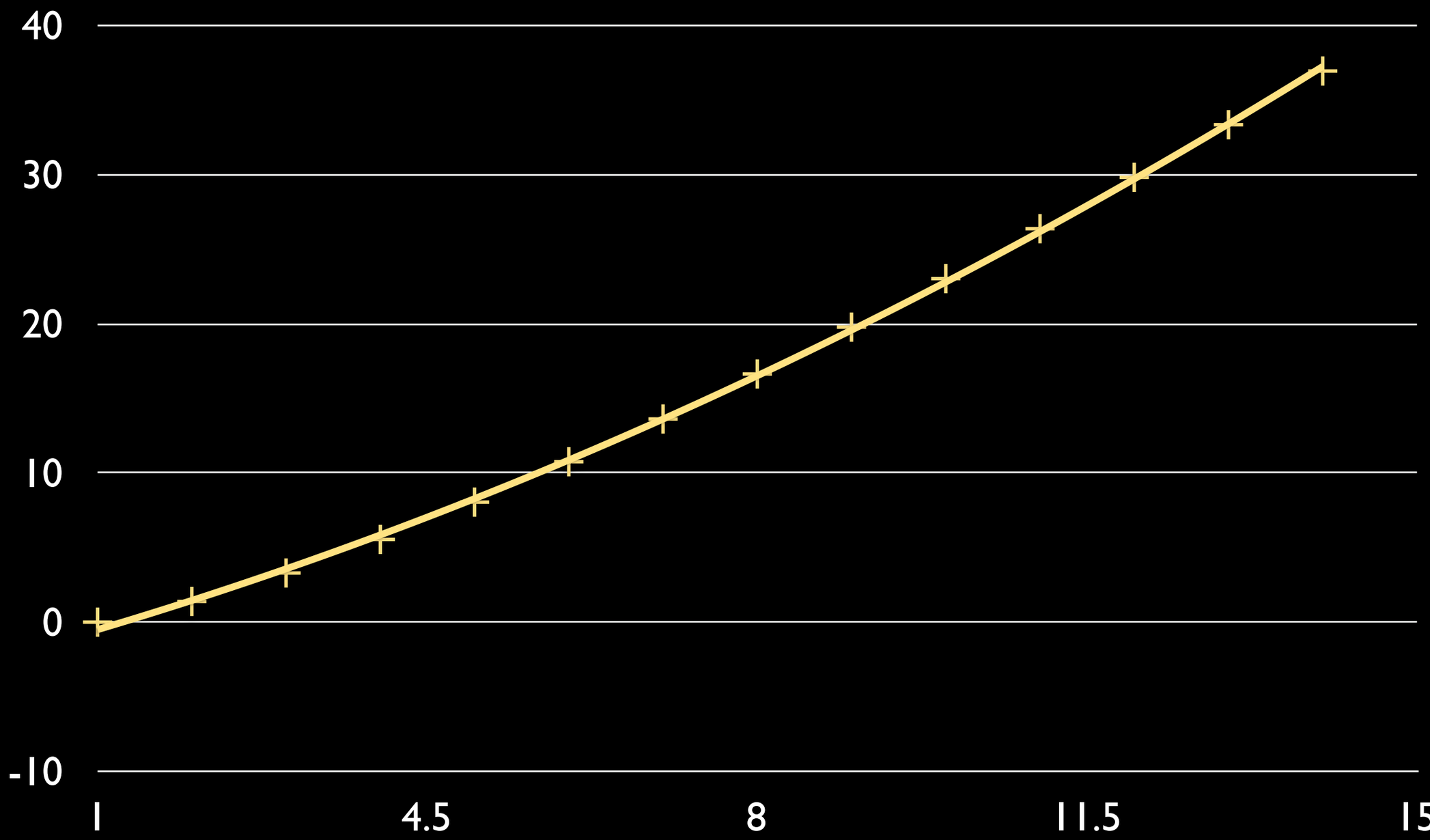
y = log(n)



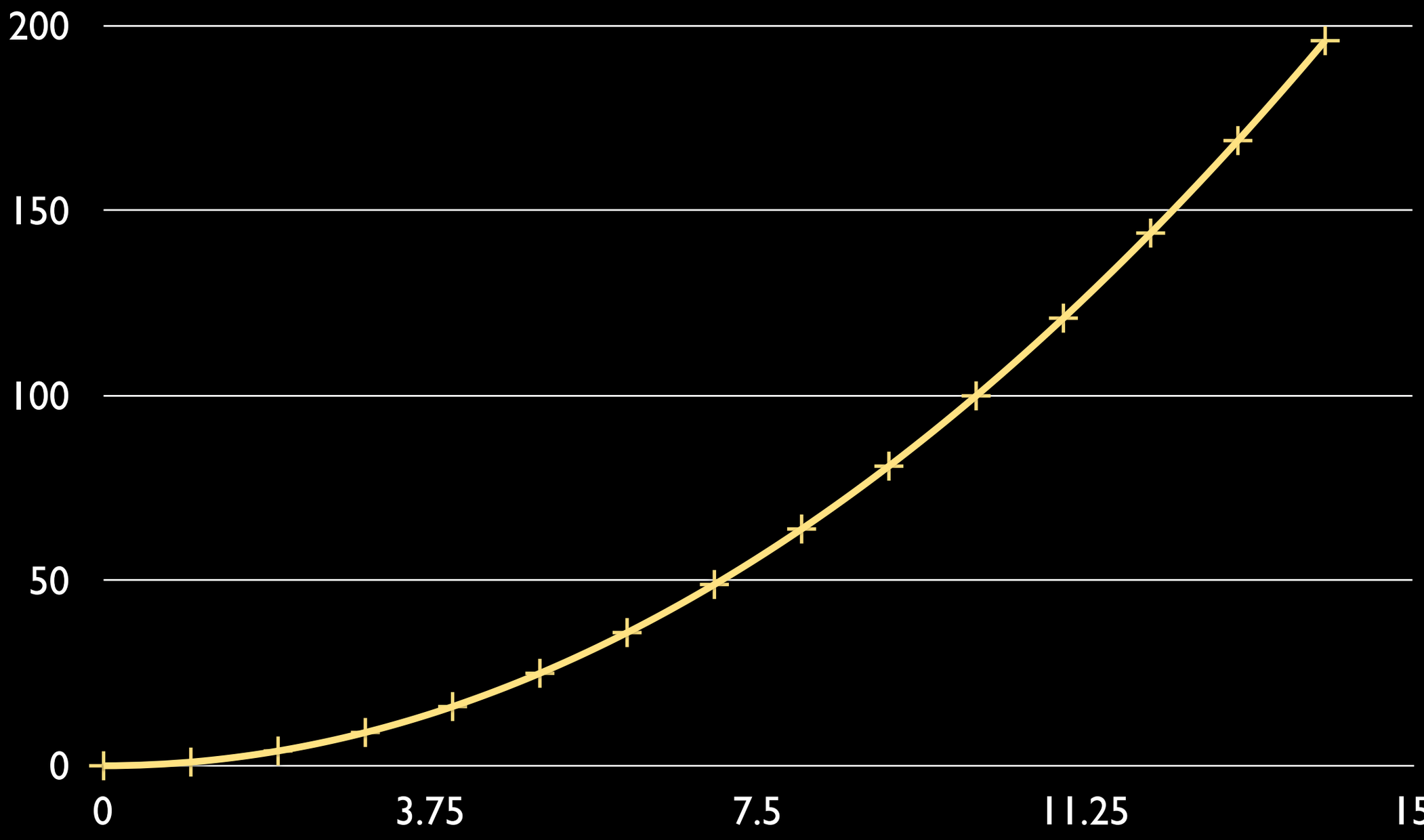
y = x



y = n log(n)



y = n^2

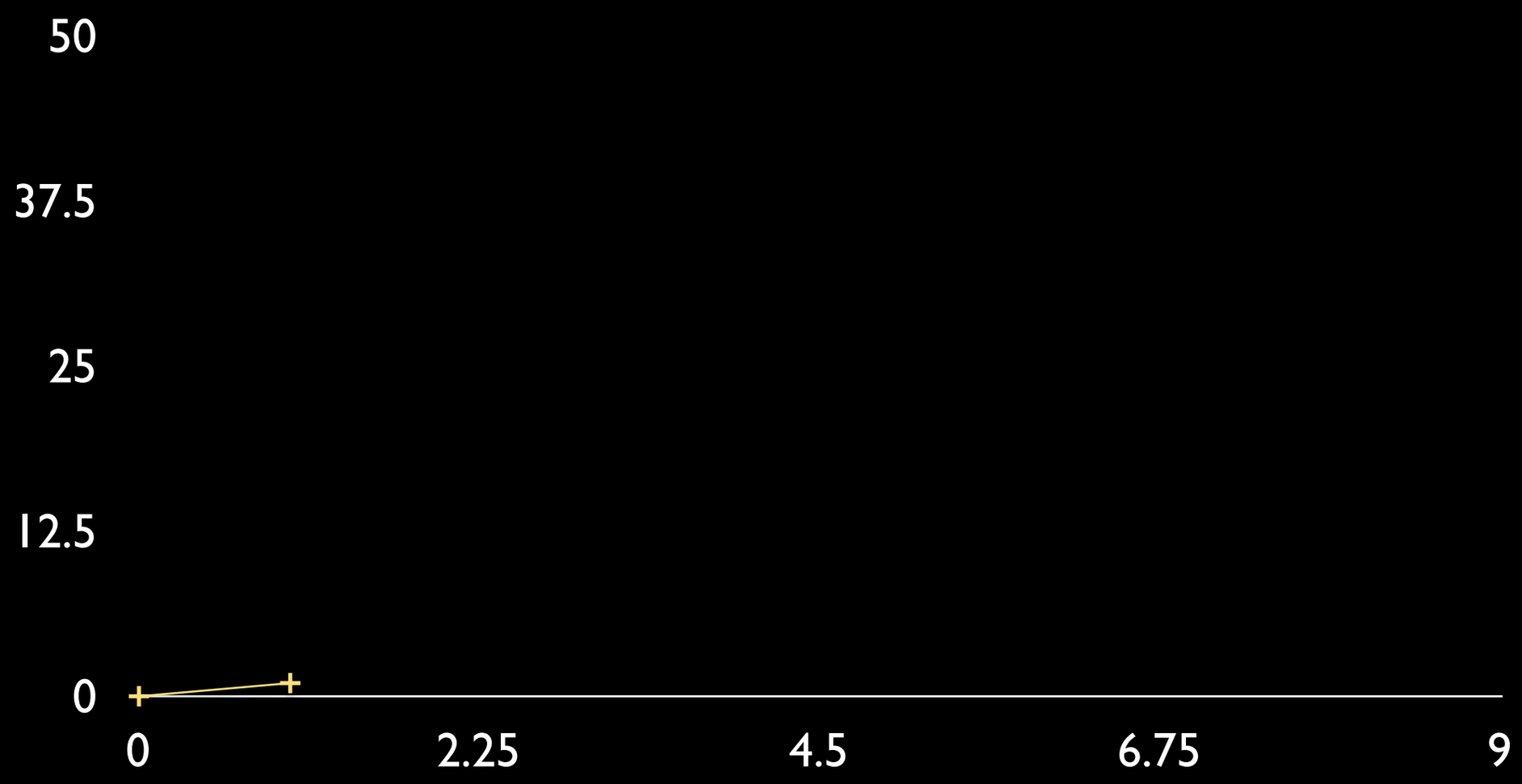


Finding Big O

- Give input
- Measure output
- Plot Results

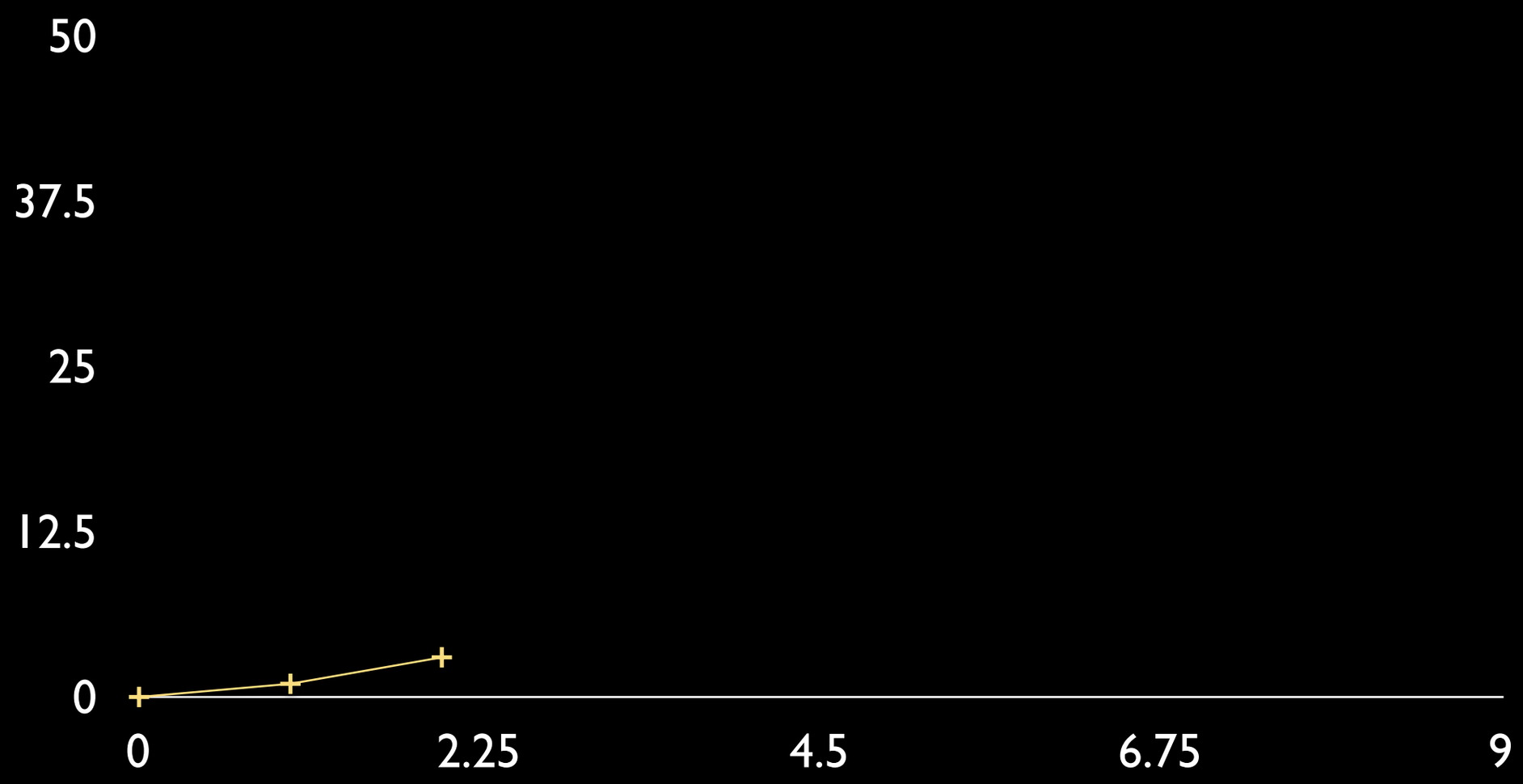
ARel's Big O

+ objects and funcalls



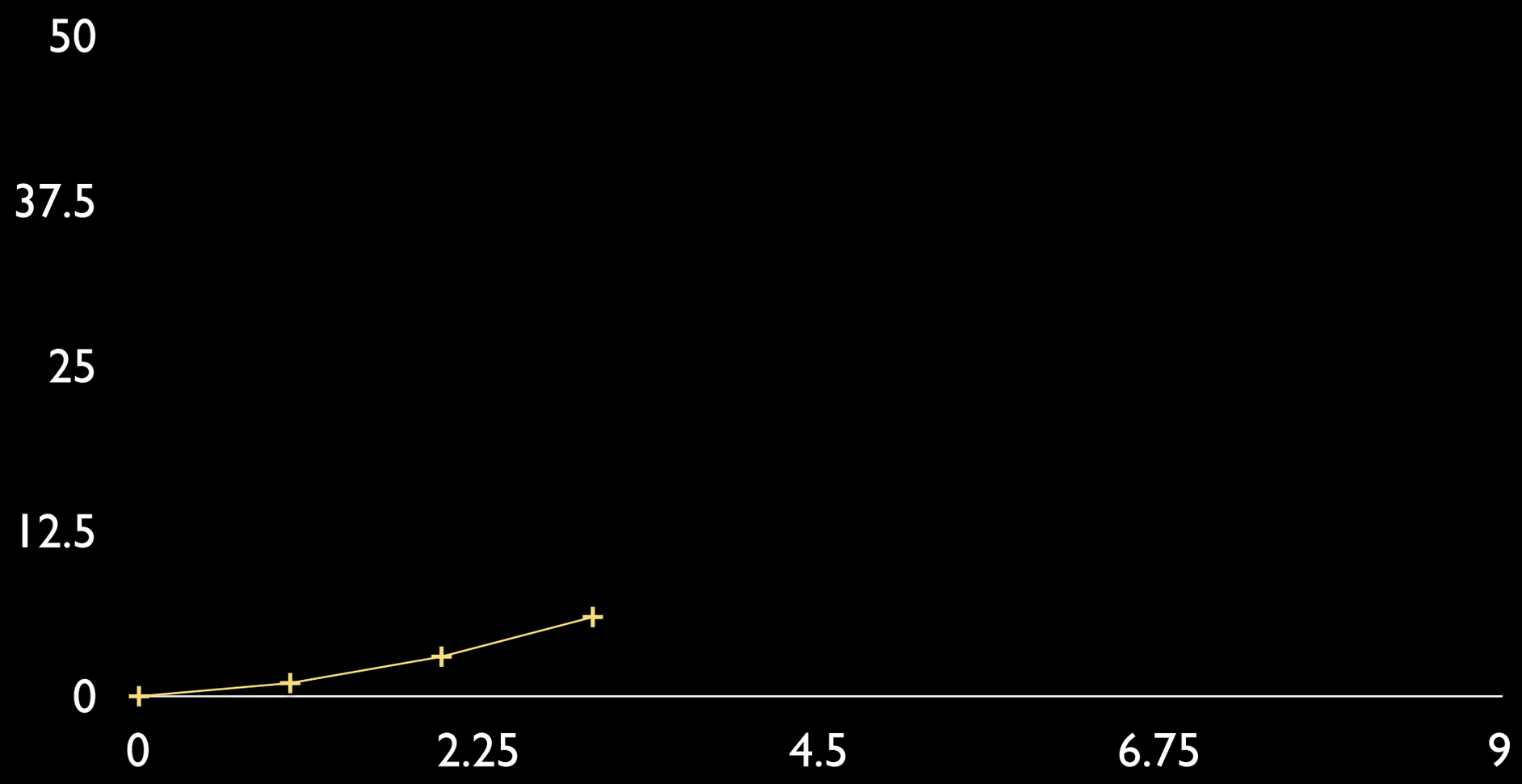
ARel's Big O

+ objects and funcalls



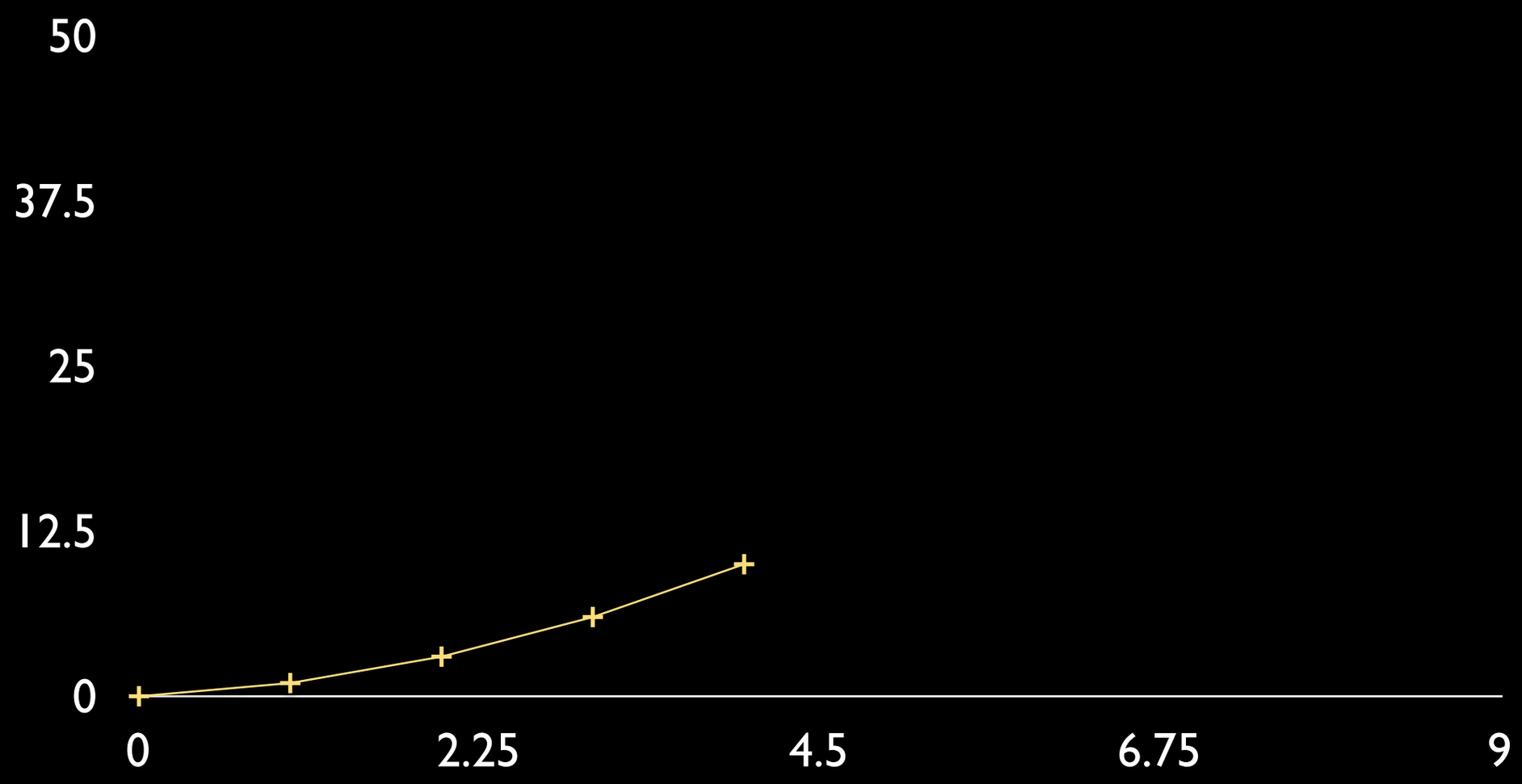
ARel's Big O

\+ objects and funcalls



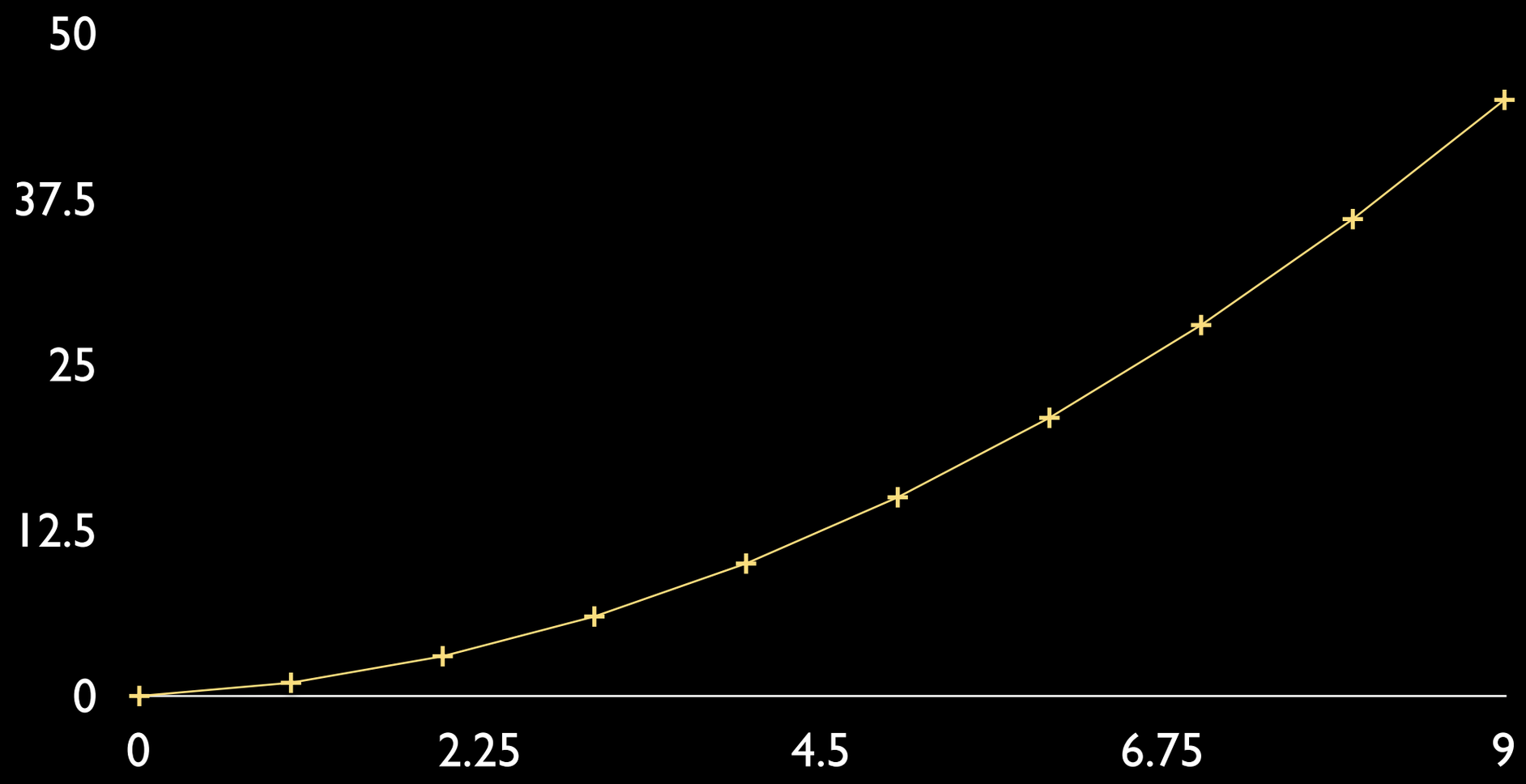
ARel's Big O

\+ objects and funcalls



ARel's Big O

+ objects and funcalls



$$y=\frac{1}{2}\left(n^2+n\right)$$

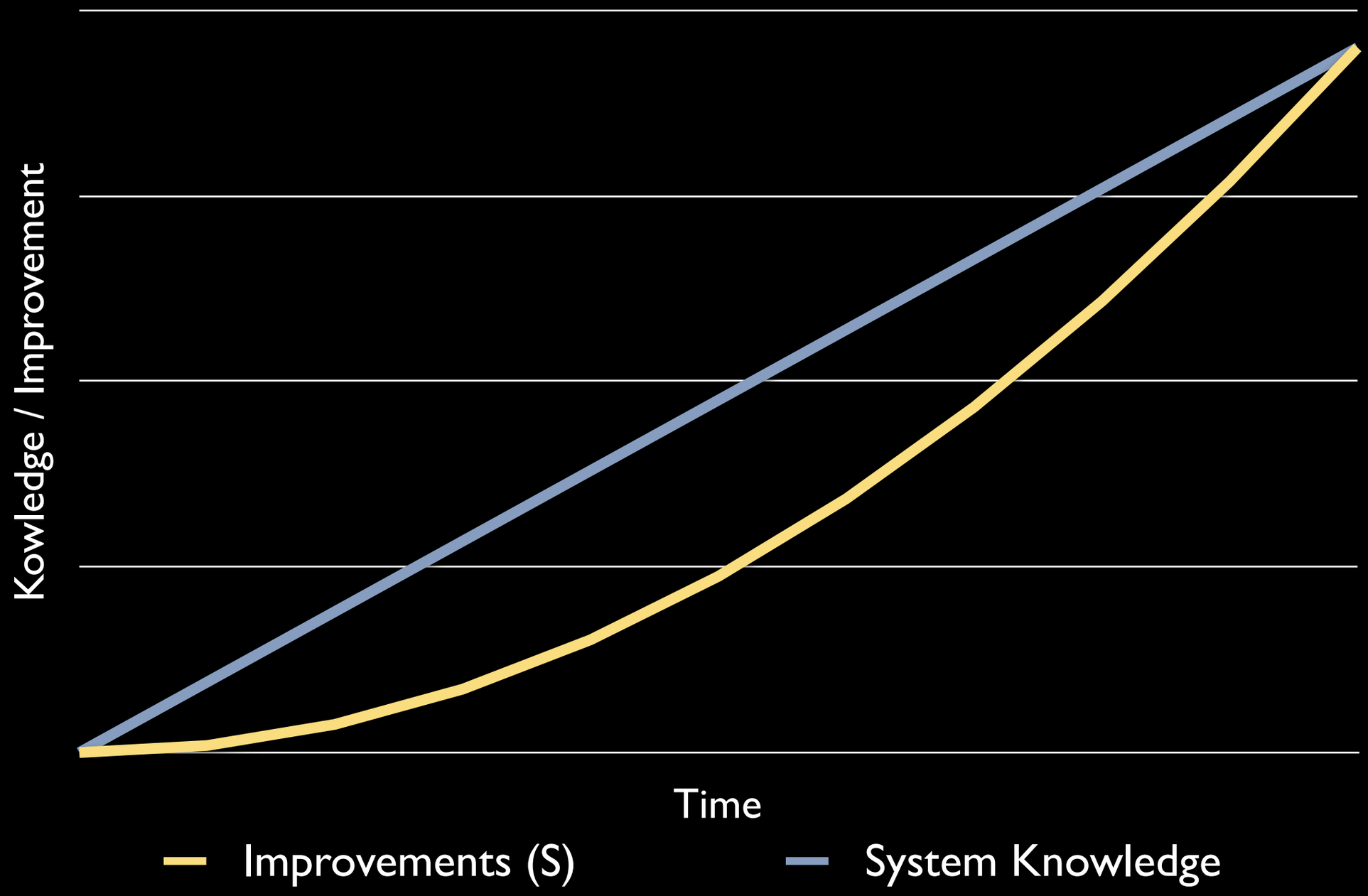
$$O(n^2)$$

ActiveRecord/Arel takes over 2 minutes to generate a pseudo-complex SQL query.

http://bit.ly/omgslow2

Deep Improvements

System Impact



AST + Visitor

O(n)

Should I rewrite?

- Clear solution
- Tests are numerous (Rails)
- Public API is limited

YES

6 Weeks Later...

ARel Today

Data Sheet

- O(n)
- 6 Weeks to Rewrite
- 2x faster (for simple queries)
- Adapter Specific code is DRY

flog (before)

```
2533.1: flog total
   6.8: flog/method average

 116.6: OracleCompiler#select_sql
  78.0: PostgreSQLCompiler#select_sql
  64.9: main#none
  59.4: GenericCompiler#insert_sql
  52.4: Join#joins
```

flog (after)

```
1864.6: flog total
   6.5: flog/method average

  81.4: main#none
  75.9: Dot#none
  59.1: Oracle# lib/arel/visitors/oracle.rb:6
  54.0: ToSql#lib/arel/visitors/to_sql.rb:31
  51.6: ToSql#none
```

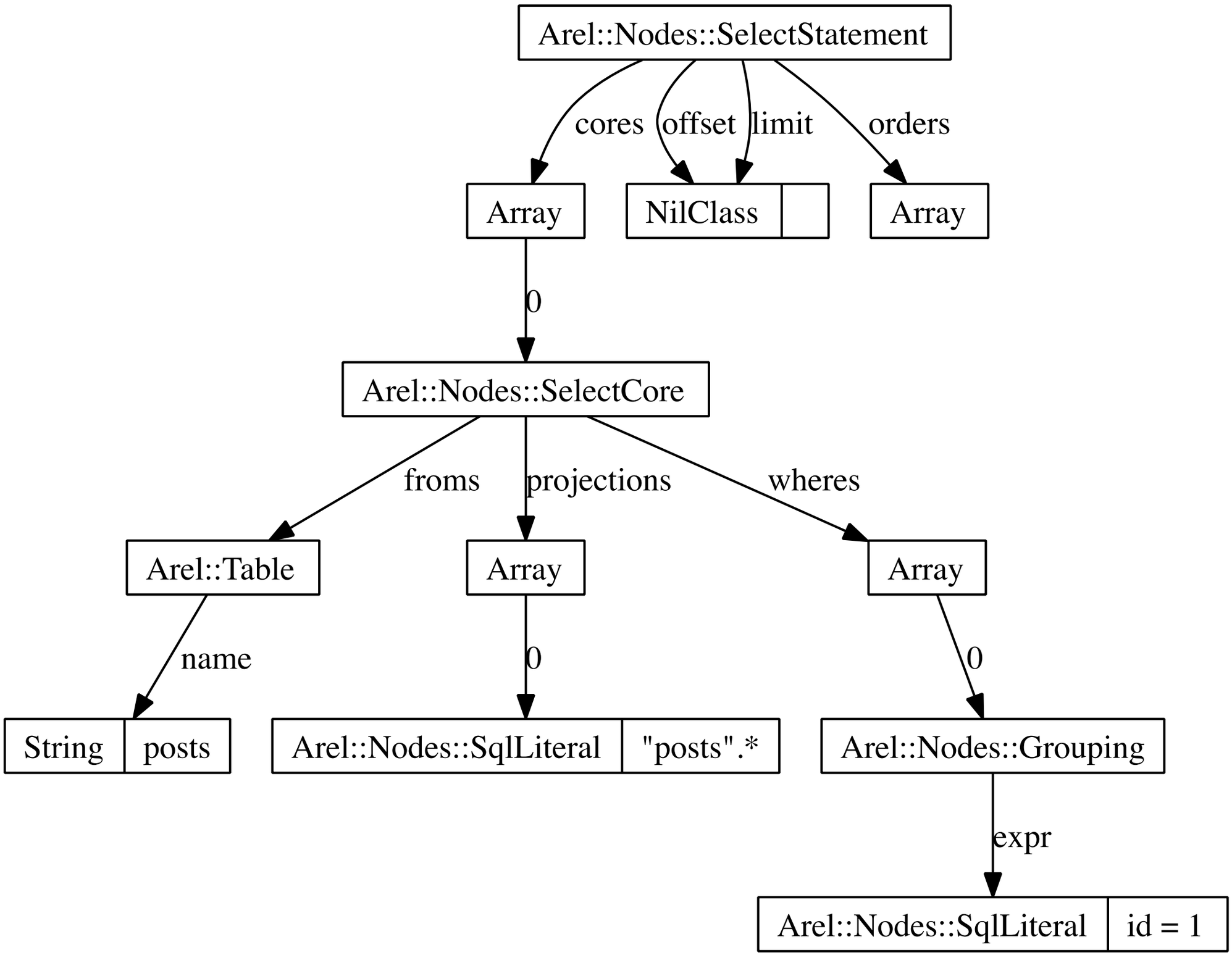
flay (before)

```
Total score (lower is better) = 684

12 complaints
```

flay (after)

```
Total score (lower is better) = 420

7 complaints
```

```
Post.where("id = 1").to_dot
```


Arel::Nodes::SelectStatement
cores
offset
limit
orders
Array
NilClass
Array
0
Arel::Nodes::SelectCore
froms
projections
wheres
Arel::Table
Array
Array
name
0
0
String
posts
Arel::Nodes::SqlLiteral
"posts".*
Arel::Nodes::Grouping
expr
Arel::Nodes::SqlLiteral
id = 1

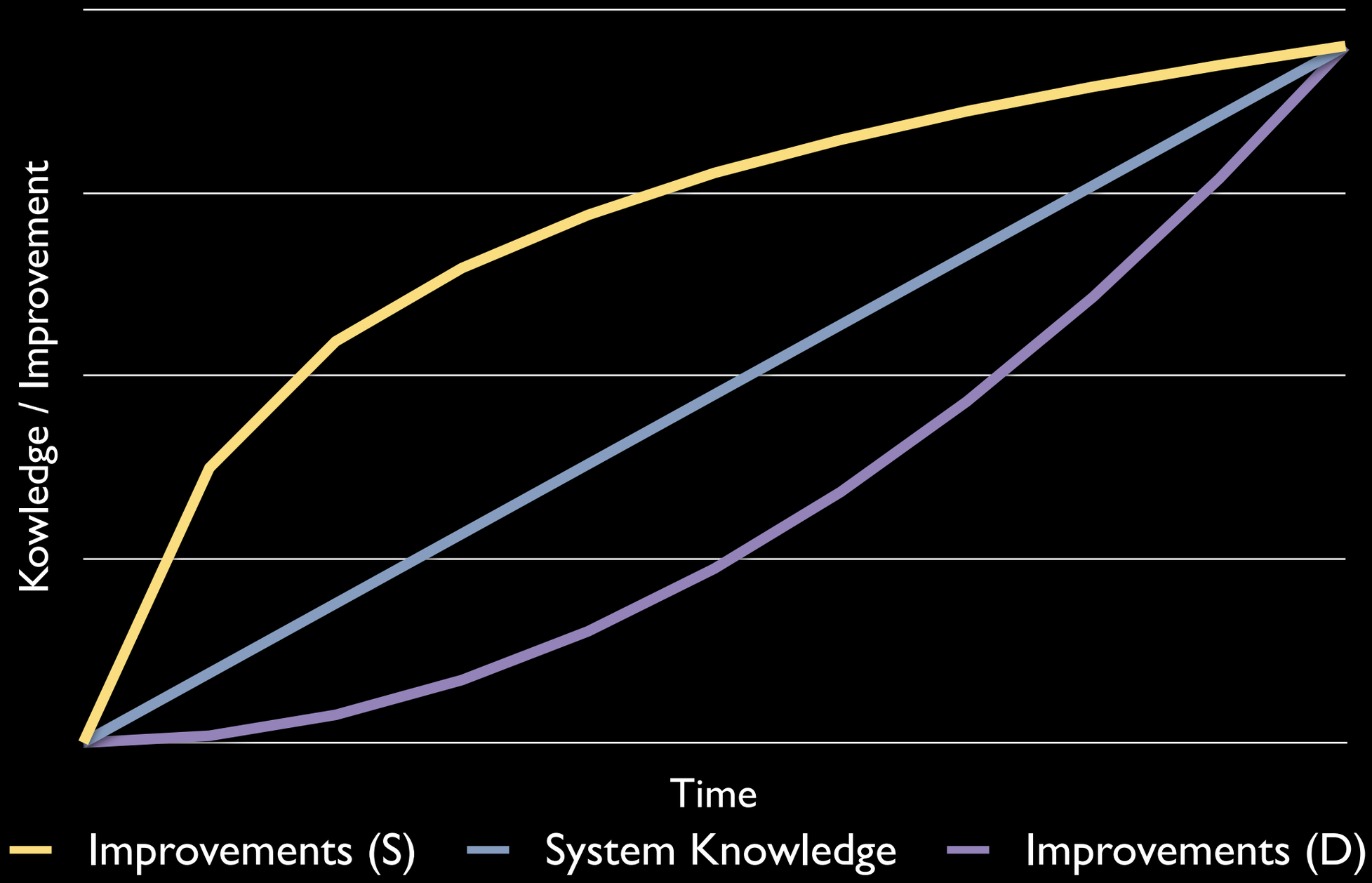
ARel Tomorrow

Conclusion

AKA:

Things I've Learned

System Impact



When Should I Rewrite?

Rewrite Timeline

Rewrite Timeline

We emphasize

the art

of Code

We should not forget

the Science

Learn

The Specific

But Embrace

The Generic

Photo Credits

- DHH: http://www.flickr.com/photos/pdcawley/250813158/
- Matz: http://www.flickr.com/photos/kakutani/4127354831/
- Chad Fowler: http://www.flickr.com/photos/fraserspeirs/3386558579/

Thanks @ebiltwin

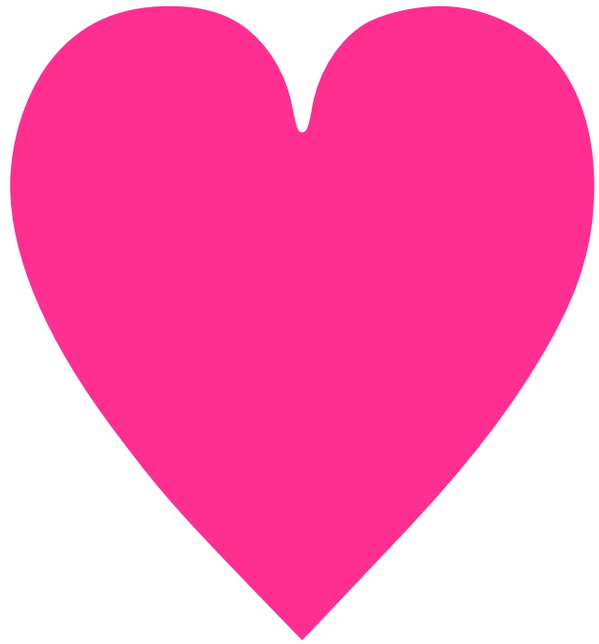
Thank you!

One More Thing...

It's RubyConf 10!

Give Ryan a **Kiss**!

<3



Questions?