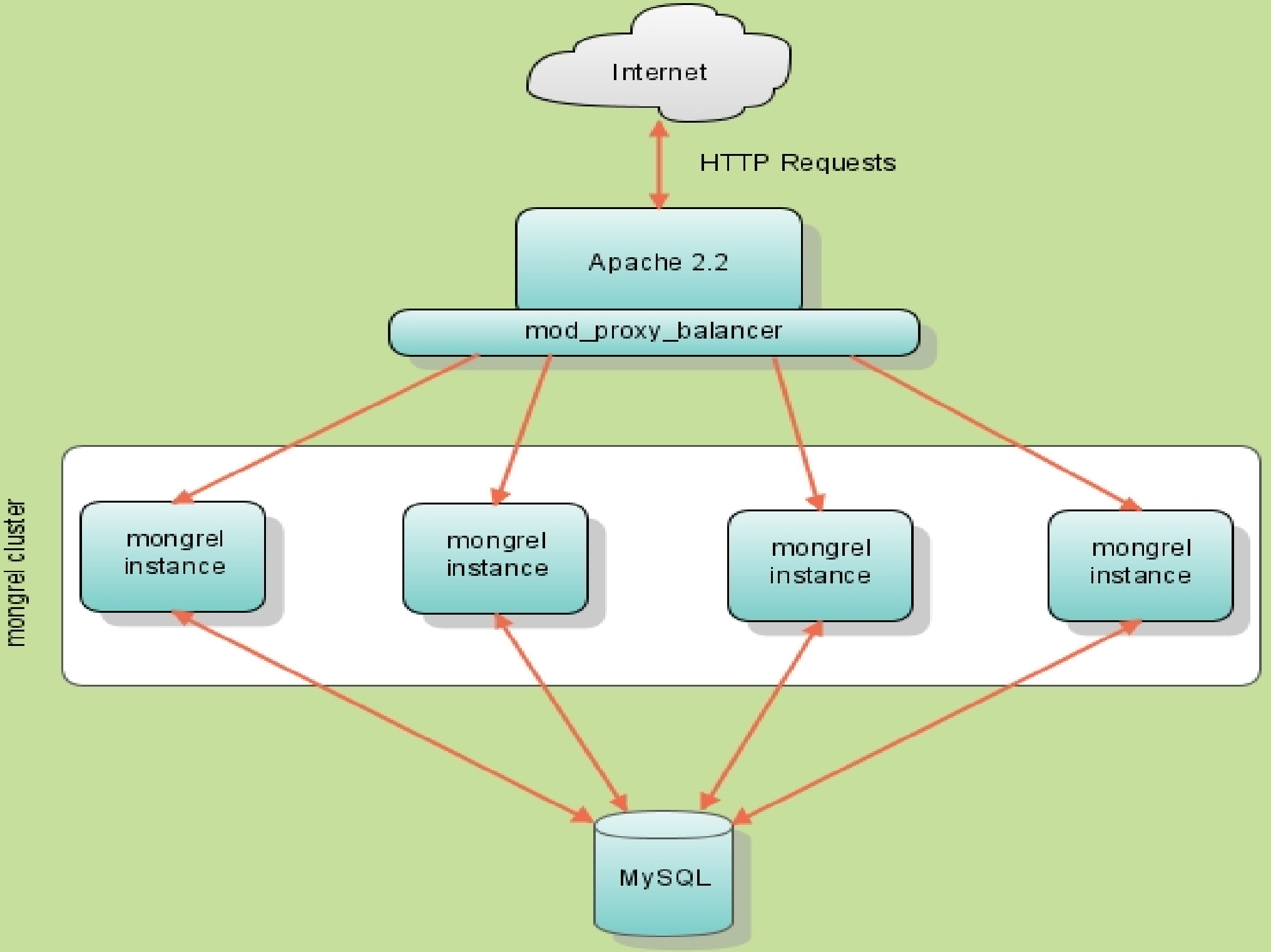
A Small Talk on Getting Big.
Scaling a Rails App & all that Jazz.

How did this Happen?


Internet
HTTP Requests
Apache 2.2
mod_proxy_balancer
mongrel cluster
mongrel instance
mongrel instance
mongrel instance
mongrel instance
MySQL

Too Much Time in the Application.
Too Much Time in the Database.

A Really Cool Trick

```
class ApplicationController < ActionController:Base
    include ResponseManagementAdditions
    after_filter :show_runtimes

    around_filter { |controller, action| controller.total_runtime  =
    Benchmark.measure(&action).real }
end
```

A Really Cool Trick

```
module ResponseManagementAdditions
    def show_runtimes
      return if response.headers['Status'] == '304 Not Modified' || !
(response.body && response.body.respond_to?(:sub!))

      db_runtime = ((0 + (@db_rt_before_render || 0) +
(@db_rt_after_render || 0)) * 1000).truncate
      rendering_runtime = ((@rendering_runtime || 0) * 1000).truncate
      total_runtime = ((@total_runtime || 0) * 1000).truncate

      response.body.gsub!(/<\/body><\/html>$/, "<!-- served to you
through a copper wire by #{HOSTNAME} at
#{Time.now.to_s(:short)} in #{total_runtime} ms (d #{db_runtime} / r
#{rendering_runtime}). thank you, come again. -->
\n</body></html>")
    end
end
```

Two Ways People Design Sites.

~~Over-architect.~~
Under-architect.

Move Fast. Scale Quickly.

Too Much Time in the Application.

Abstract Long Running Processes
to Daemons.

An Ugly but Amazingly Simple Queuing System.

```
class FooDaemon < TwitterDaemon::Base
  before_startup :set_fugly_dist_idx

  def process
    unprocessed_content do |c|
      increment_counter(:total)

      # Do work ...

      break unless running?
    end
  end

  ...
```

```
...

def unprocessed_content(&block)
    loop do
      content = ContentModel.pending_content("substring(truncate(id,
0),-2,1) = #{@fugly_dist_idx}")
      messages.each { |message| yield message }
      sleep 1 if messages.nil? || messages.empty?
    end
  end

  def set_fugly_dist_idx
    @fugly_dist_idx = ARGV.find { |v| v.match(/[0-9]/) }
    raise "You need to specify a dist idx between 0 and 9." unless
@fugly_dist_idx
    @fugly_dist_idx = @fugly_dist_idx.to_i
  end
end
```

A Better Queuing System.

Starling.

Distributed Queuing.
Transactional Playback.
Fast.
Simple.
Speaks Memcache's Language.
100% Pure Ruby.

Not Open Source.

(yet ...)

Too Much Time in the Database.

The Basics. Database 101.
(I shouldn't need this slide in here.)

Index everything you will query on.
Avoid complex joins.
Use joint indices when you must join tables.
Avoid scanning large sets of data.

Cache.

Cache.

Cache.

CACHE!

But How?
That Sounds Hard.

Turns Out it Isn't.

Serialize, Denormalize.

```
class User < ActiveRecord::Base
  serialize :following_ids

  def following_ids
    # this accessor is overwritten because we want to lazily set the
    # friends_ids column, rather than running a gigantic slow migration.
    RAILS_DEFAULT_LOGGER.debug "loading following_ids"
    ids = read_attribute(:following_ids)

    if ids.nil? || !ids.kind_of?(Array)
      ids = connection.select_values("SELECT DISTINCT followed_user_id
FROM followed_users WHERE user_id =
#{self.id}").map(&:to_i).compact
      update_attribute(:following_ids, ids)
    end

    ids
  end

  ...
```

```
  def following_ids_add(the_id)
    ids = self.following_ids.dup
    ids << the_id
    write_attribute(:following_ids, ids)
  end

  def following_ids_delete(the_id)
    ids = self.following_ids.dup
    ids.delete(the_id)
    write_attribute(:following_ids, ids)
  end

end # End Class
```

Oh yeah, and Cheat.
(It's ok!)

Thing about your application.
How can you cheat and get away with it?

Is your data delivered in real time?
Is your data static content?
How do users interact?

Interestingness.
(Little things that don't deserve other space.)

It's OK to use Monit to kill processes
if they get too big.

Ensure you can deploy frequently.

Ensure you can roll back easily.

Scale where it matters.

Some code is ugly. It's OK.
(who needs a hug?)

Ensure your users can give feedback easily.

Use the Community.

Make an API.
(Scale your Developer-base.)

We run on Edge (but with Piston).

A Cool Trick. Gems in Vendor.

```
Rails::Initializer.run do |config|

  # Load Gems from the /vendor/gems folder first, if they exist.
  config.load_paths += Dir["#{RAILS_ROOT}/vendor/gems/**"].map do |dir|
    File.directory?(lib = "#{dir}/lib") ? lib : dir
  end

  ...
```

Personal Pet Peeve.

It's 2007. Every spammer has your email address. Put it on your goddamn webpage so people can get ahold of you about interesting things.

Questions?

Britt Selvitelle

IM & Email
anotherbritt@gmail.com