



TorqueBox

Bob McWhirter

Presented at **DevNexus**
22 March 2011



Bob McWhirter

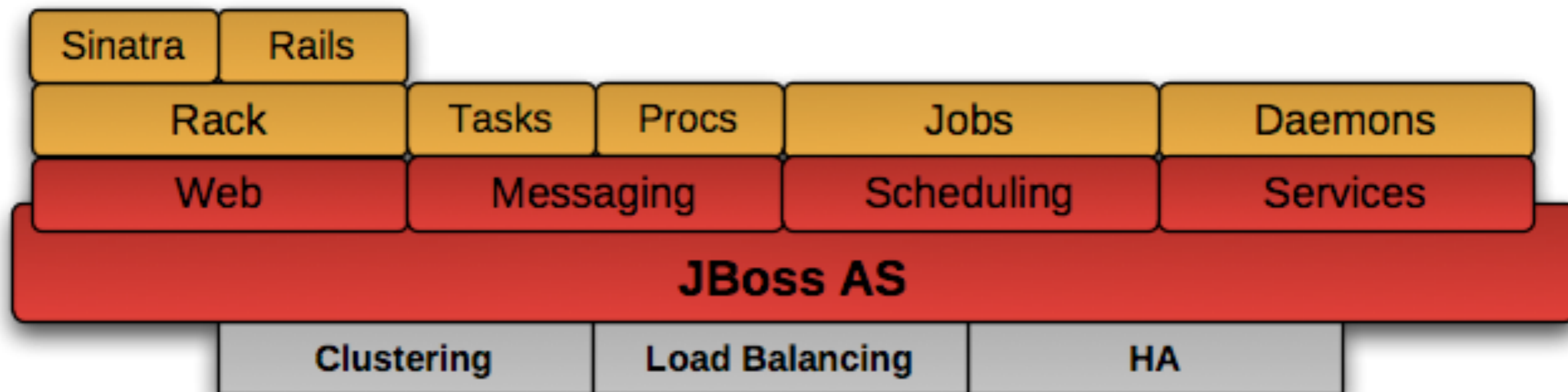
- Project lead of TorqueBox
- JBoss Fellow
- Founder of The Codehaus
- Founder of Drools
- Been with Red Hat ~4 years

but it's a team.



What is TorqueBox?

The mating of JRuby to
JBoss AS.



Goals

- Support Ruby web frameworks
 - Rails
 - Sinatra
 - Rack
- Go **beyond** the web
 - Messaging
 - Jobs
 - Services

But...

...Java already supports
messaging, jobs and
services.

That's right.

Why Ruby?

- No compilation
- Low ceremony
- Highly expressive
- Lots of shiny frameworks
- Few specifications (more fun!)
- Meta

Expressive

`anything.rb`

```
teams.  
  collect(&:members).  
  flatten.uniq.each &:promote!
```

Uhh...

com/foo/Anything.java

```
Set<Person> people = new HashSet<Person>();

for ( Team each : teams ) {
    people.addAll( each.getMembers() );
}

for ( Person each : people ) {
    each.promote();
}
```

Why *JRuby*

- Very fast runtime
- Real threads
- Java libraries
- Java tools
- Healthy community

Easy Install. Even on Windows

```
$ wget http://torquebox.org/torquebox-dev.zip
```

```
$ unzip torquebox-dev.zip
```

```
$ export TORQUEBOX_HOME=$PWD/torquebox-1*
```

```
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
```

```
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby
```

```
$ export PATH=$JRUBY_HOME/bin:$PATH
```

Easy Install. Even on Windows

```
$ wget http://torquebox.org/torquebox-dev.zip
```

```
$ unzip torquebox-dev.zip
```

```
$ export TORQUEBOX_HOME=$PWD/torquebox-1*
```

```
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
```

```
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby
```

```
$ export PATH=$JRUBY_HOME/bin:$PATH
```

Easy Install. Even on Windows

```
$ wget http://torquebox.org/torquebox-dev.zip
```

```
$ unzip torquebox-dev.zip
```

```
$ export TORQUEBOX_HOME=$PWD/torquebox-1*
```

```
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
```

```
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby
```

```
$ export PATH=$JRUBY_HOME/bin:$PATH
```

Easy Install. Even on Windows

```
$ wget http://torquebox.org
$ unzip torquebox-dev.zip

$ export TORQUEBOX_HOME=$PWD
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
$ export JRUBY_HOME=$TORQUEBOX_HOME/jruby
```

Make sure the jruby found in your path is in \$JRUBY_HOME/bin.

```
$ export PATH=$JRUBY_HOME/bin:$PATH
```

The Details

Builds upon and requires
JBoss AS 6.x.

Tight integration with the
JBoss stack.

The Competition

Warbler, Trinidad, Unicorn,
Thin, Passenger...

...all address only the web
question.

Web

Run Ruby web-apps within the context of the Servlet container.

Without compilation.

A rails application

```
RAILS_ROOT/  
  app/  
    models/  
      person.rb  
    views/  
      person/  
        index.html.haml  
        show.html.haml  
    controllers/  
      persons_controller.rb  
  lib/  
    my-java-components.jar  
  config/  
    database.yml  
    torquebox.yml
```

Deploy!

```
$ rake torquebox:deploy
```

But continue editing

Deployment does **not** create archives (by default).

Continue live-editing of running app:

models, views, controllers...

**Non-surprising.
Almost boring.**

Web (Java Integration)

```
class SomeController  
  
  def index  
    session[:password] = 'sw0rdfish'  
  end  
  
end
```

Web (Java Integration)

```
public class SomeServlet
{
    public void doGet(HttpServletRequest req,
                      HttpServletResponse resp)
    {
        request.getSession().getValue("password");
    }
}
```

Clustering

Ruby applications participate fully in AS clustering.

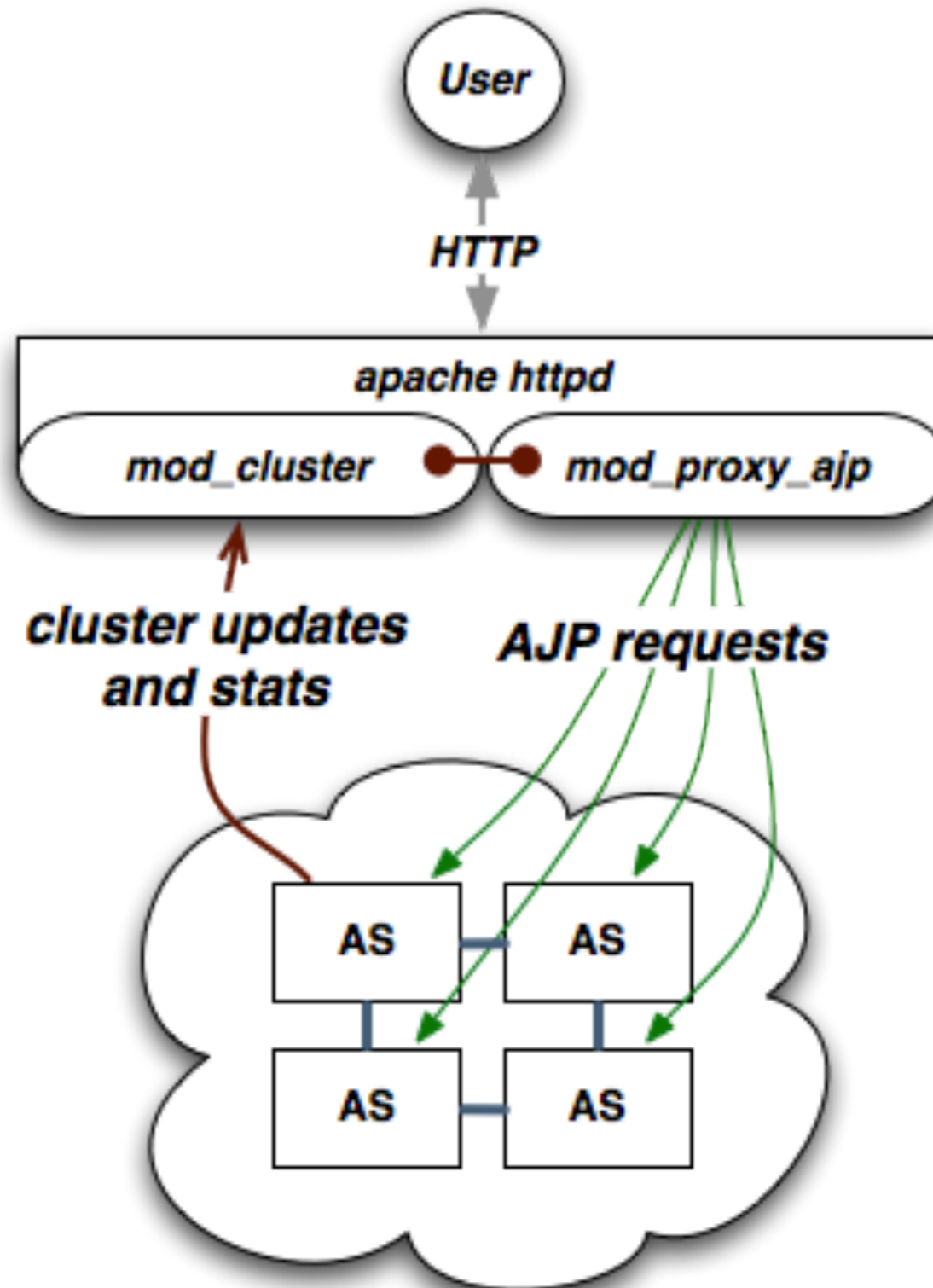
Can use JBoss **mod_cluster**.

mod_cluster

A reverse proxy implemented as an Apache module with JBoss awareness.

Constantly gathers load statistics and deployment availability for intelligent request distribution.

mod_cluster



mod_cluster

- Dynamic configuration
- Server-side load factor calculation
- Fine-grained web app lifecycle
- AJP (Apache JServ Protocol) is optional. HTTP[S] is also supported.

**Let's go
beyond the
web...**

Scheduled Jobs

Jobs

app/jobs/newsletter_sender.rb

```
class NewsletterSender

  def run()
    subscriptions = Subscription.find(:all)
    subscriptions.each do |e|
      send_newsletter( e )
    end
  end
end

end
```

Jobs

config/torquebox.yml

jobs:

monthly_newsletter:

description: first of month

job: NewsletterSender

cron: '0 0 0 1 * ?'

sandbox:

job: Sandbox

cron: '* /5 * * * * ?'

Messaging, Part 1

(async tasks)

Tasks

app/tasks/email_task.rb

```
class EmailTask < TorqueBox::Messaging::Task

  def welcome(payload)
    person = payload[:person]
    person ||= Person.find_by_id(payload[:id])
    if person
      # send the email
      person.welcomed = true
      person.save!
    end
  end
end

end
```

Tasks

```
EmailTask.async( :welcome, payload )
```

Tasks

Call them from your **controllers, models, and observers**, or even other **tasks**. Even in non-Rails apps!

Messaging, Part 2

(backgroundable)

Regular Class

```
class Something  
  
  def foo()  
  end  
  
  def bar()  
  end  
  
end
```

Blocking invocations

```
something = Something.new
```

```
something.foo
```

```
something.bar
```

Backgroundable

```
class Something

  include TorqueBox::Messaging::Backgroundable

  def foo()
  end

  def bar()
  end

end
```

Non-blocking invocations

```
something = Something.new
```

```
something.background.foo
```

```
something.background.bar
```

Choices

```
class Something

  include TorqueBox::Messaging::Backgroundable
  always_background :foo

  def foo()
  end

  def bar()
  end

end
```

Just do it right.

```
something = Something.new
```

```
something.foo
```

Messaging, Part 3

(low-level)

Destinations

`config/torquebox.yml`

`queues:`

`/queues/questions:`

`/queues/answers:`

`durable: false`

`topics:`

`/topics/new_accounts`

`/topics/notifications`

Processors

config/torquebox.yml

```
messaging:  
  /topics/print: PrintHandler  
  /queues/popular:  
    - PopularHandler  
    - AdultObserver:  
      filter: "age >= 18"  
      concurrency: 5  
  /queues/students:  
    PrintHandler:  
      config:  
        color: true
```

Processors

app/models/print_handler.rb

```
include TorqueBox::Messaging

class PrintHandler < MessageProcessor
  def initialize(opts)
    @color = opts['color']
  end
  def on_message(body)
    puts "Processing #{body} of #{message}"
  end
end
```

Queues

example

```
include TorqueBox
req = Messaging::Queue.new '/queues/questions'
res = Messaging::Queue.new '/queues/answers'

Thread.new do
  req.publish "What time is it?"
  puts res.receive( :timeout => 1000 )
end

puts req.receive
res.publish Time.now
```

Queues

example

```
include TorqueBox  
req = Messaging::Queue.new '/queues/questions'  
res = Messaging::Queue.new '/queues/answers'  
  
Thread.new do  
  req.publish "What time is it?"  
  puts res.receive( :timeout => 1000 )  
end  
  
puts req.receive  
res.publish Time.now
```

Queues

example

```
include TorqueBox  
req = Messaging::Queue.new '/queues/questions'  
res = Messaging::Queue.new '/queues/answers'
```

```
Thread.new do  
  req.publish "What time is it?"  
  puts res.receive( :timeout => 1000 )  
end
```

```
puts req.receive  
res.publish Time.now
```

Queues

example

```
include TorqueBox
req = Messaging::Queue.new '/queues/questions'
res = Messaging::Queue.new '/queues/answers'

Thread.new do
  req.publish "What time is it?"
  puts res.receive( :timeout => 1000 )
end
```

```
puts req.receive
res.publish Time.now
```

Queues

“on the fly”

```
include TorqueBox
```

```
queue = Messaging::Queue.new '/queues/foo'  
queue.create
```

```
...
```

```
queue.destroy
```

Topics

- behavior is different, but interface is the same.
- all subscribers of a **topic** see each message, but only one subscriber will see any message from a **queue**
- use `TorqueBox::Messaging::Topic`

Services

run along, lil' daemon

Services

Long-running, non-web “daemons” that share the runtime environment and deployment lifecycle of your app.

Services

- Represented as a class with optional `initialize`(Hash), `start`() and `stop`() methods, which should each return quickly.
- Typically will start a long-running loop in a thread and respond to external events.

Services

config/torquebox.yml

```
services:  
  IrcBot:  
    server: freenode.net  
    channel: #torquebox  
    publish: /topics/irc
```

```
MyMudServer:
```

```
SomeOtherService:
```

Services

app/{models, etc}/my_service.rb

```
class MyService
  def initialize opts={}
    name = opts[:publish]
    @queue = Messaging::Queue.new(name)
  end
  def start
    Thread.new { run }
  end
  def stop
    @done = true
  end
end
```

Services

app/{models, etc}/my_service.rb

```
class MyService
  def initialize opts={}
    name = opts[:publish]
    @queue = Messaging::Queue.new(name)
  end
  def start
    Thread.new { run }
  end
  def stop
    @done = true
  end
end
```

Services

app/{models, etc}/my_service.rb

```
class MyService
  def initialize opts={}
    name = opts[:publish]
    @queue = Messaging::Queue.new(name)
  end
  def start
    Thread.new { run }
  end
  def stop
    @done = true
  end
end
```

Services

app/{models, etc}/my_service.rb

```
class MyService
  def initialize opts={}
    name = opts[:publish]
    @queue = Messaging::Queue.new(name)
  end
  def start
    Thread.new { run }
  end
  def stop
    @done = true
  end
end
```

Services

app/{models, etc}/my_service.rb

```
class MyService
  def run
    until @done
      @queue.publish(Time.now)
      sleep(1)
    end
  end
end
```

Services

app/{models, etc}/my_service.rb

```
class MyService
  def run
    until @done
      @queue.publish(Time.now)
      sleep(1)
    end
  end
end
```

Caching

~~Caching~~ NoSQL

Integration with **JBoss Infinispan**, a distributed/replicated object store.

Transparent Infinispan

Easily used for all of the implicit caching within Rails.

Replace in-memory, or **memcached** caches.

Opaque Infinispan

```
include ActiveSupport::Cache

myCache = TorqueBoxStore.new( :name => 'MyCache',
                             :mode => :replicated,
                             :sync => true)
```

I N J E C T I O N

(we must go deeper)

Injection?

Letting the container figure out how to help you wire up complex component models.

aka

Inversion of Control

Guice, PicoContainer, JBoss Microcontainer

Java CDI

```
package com.mycorp;  
  
@ApplicationScoped  
class Something {  
    @Inject  
    private Else elsewise;  
}  
  
@ApplicationScoped  
class Else {  
  
}
```

CDI Injection

```
class MyService
  def initialize opts={}
    @thing = cdi(com.mycorp.Something)
  end
end
```

CDI Injection

```
class MyService
  def initialize opts={}
    @thing = cdi(com.mycorp.Something)
  end
end
```

**But there's more than
just CDI you can inject.
There's also heroin,
queues, topics and
other things.**

But *not really* heroin.

(Drugs are bad, m'kay?)

Destination Injection

```
class MyService
  def initialize opts={}
    @inbound = topic( "/topics/questions" )
    @outbound = queue( "/queues/answers" )
  end
end
```

Destination Injection

```
class MyService
  def initialize opts={}
    @inbound = topic( "/topics/questions" )
    @outbound = queue( "/queues/answers" )
  end
end
```

JNDI Injection

```
class MyService
  def initialize opts={}
    @factory = naming("java:comp/env/jdbc/myDS")
  end
end
```

JNDI Injection

```
class MyService
  def initialize opts={}
    @factory = naming("java:comp/env/jdbc/myDS")
  end
end
```

JBossMC Injection

```
class MyService
  def initialize opts={}
    @heroin = mc( "SomeMCBean" )
  end
end
```

JBossMC Injection

```
class MyService
  def initialize opts={}
    @heroin = mc( "SomeMCBean" )
  end
end
```

Why MC Beans?

All internal plumbing of **JBoss AS** is stitched together using MC.

Grab the **WebServer**, the **CacheManager**, whatever.

But that's not all!

BackStage

Dashboard to inspect and control Ruby components.

And a RESTful API.

TorqueBox::Backstage

Apps

Queues

Topics

Msg. Processors

Jobs

Services

Name	App	Status	Messages	Delivering
ExpiryQueue	n/a	Running	0	0
DLQ	n/a	Running	0	0
/queues/a-kitchen-sink-queue	n/a	Running	7	0
MessageProducerTask	kitchen-sink	Paused	0	0
Backgroundable	kitchen-sink	Running	0	0

StompBox

Easy Heroku-esque
git-based deployments.

StompBox

Stomp Box :: Dashboard

Repositories Managed

- ballast-sinatra (master →)
- ballast-sinatra (test →)
- buyappalachian.org (torquebox →)

Pushes Received

	Date	Status	Commit	Repo
Push	February 09 - 16:13	received deploy	2dee90f	buyappalachian.org
Commits	<p>Lance fb656070c7e8370d1ca8ccd47b9392fc26ce20b6 2011-02-09T13:12:20-08:00 Add tmp dir for auto deployment</p> <hr/> <p>Lance 2dee90fff7d87821126734889623e7cd9a06bd76 2011-02-09T13:12:50-08:00 Merge branch 'torquebox' of github.com:lance/buyappalachian.org into torquebox</p>			

**But how does it
perform?**

BENchmarks

Real-world Rail application:

Redmine

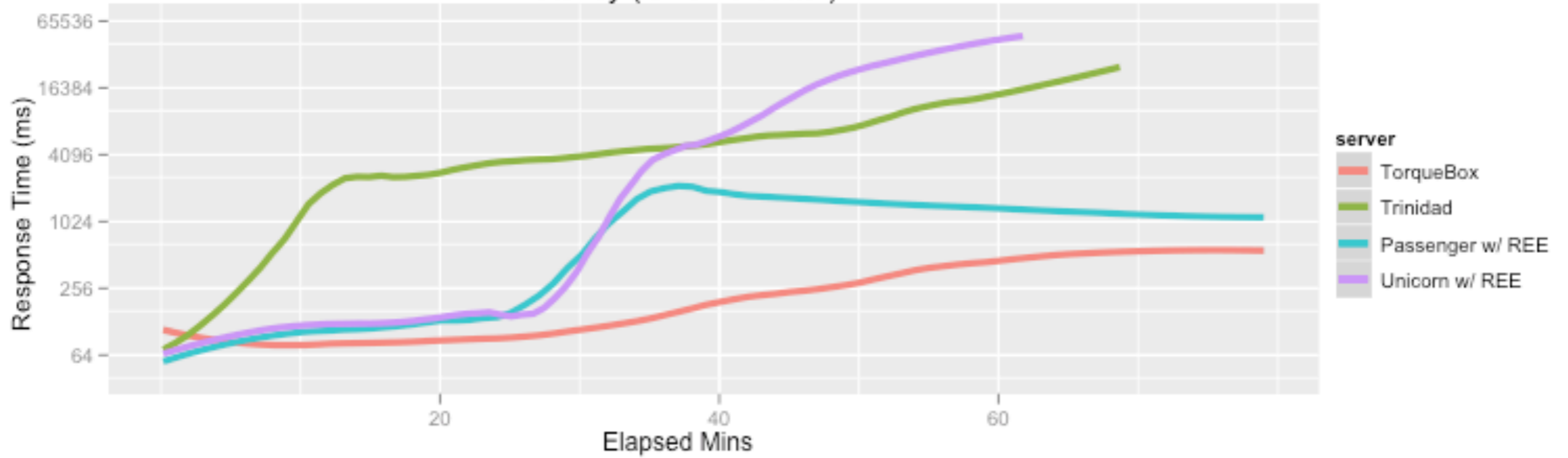
Comparisons:

TorqueBox, Trinidad, Passenger,
Unicorn, Glassfish, Thin

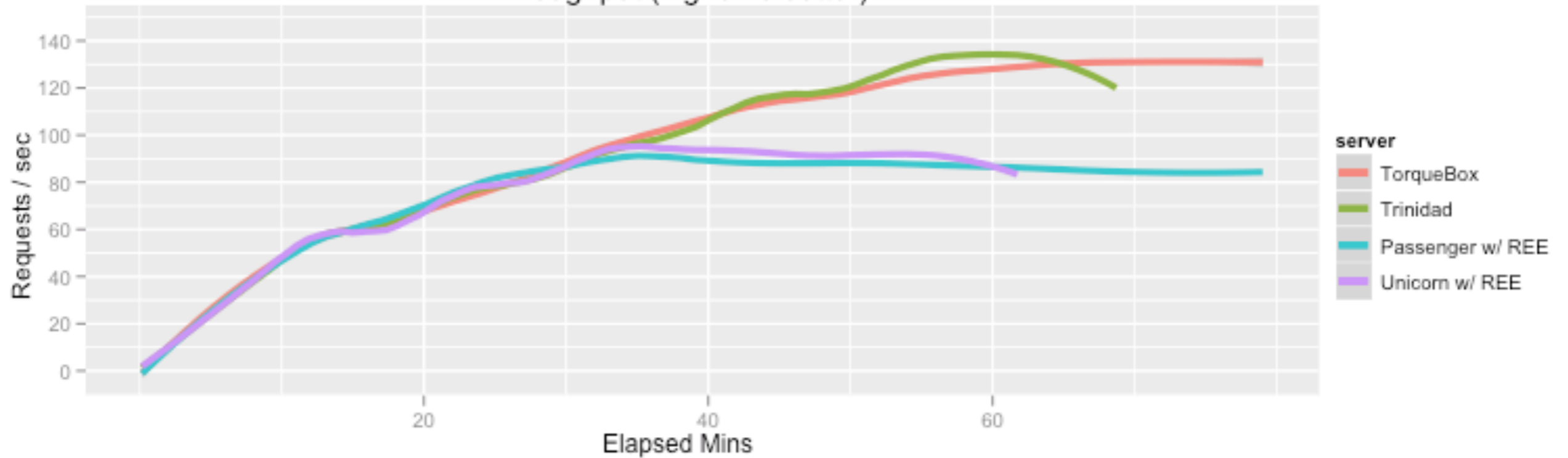
Runtimes:

JRuby, MRI, RubyEE

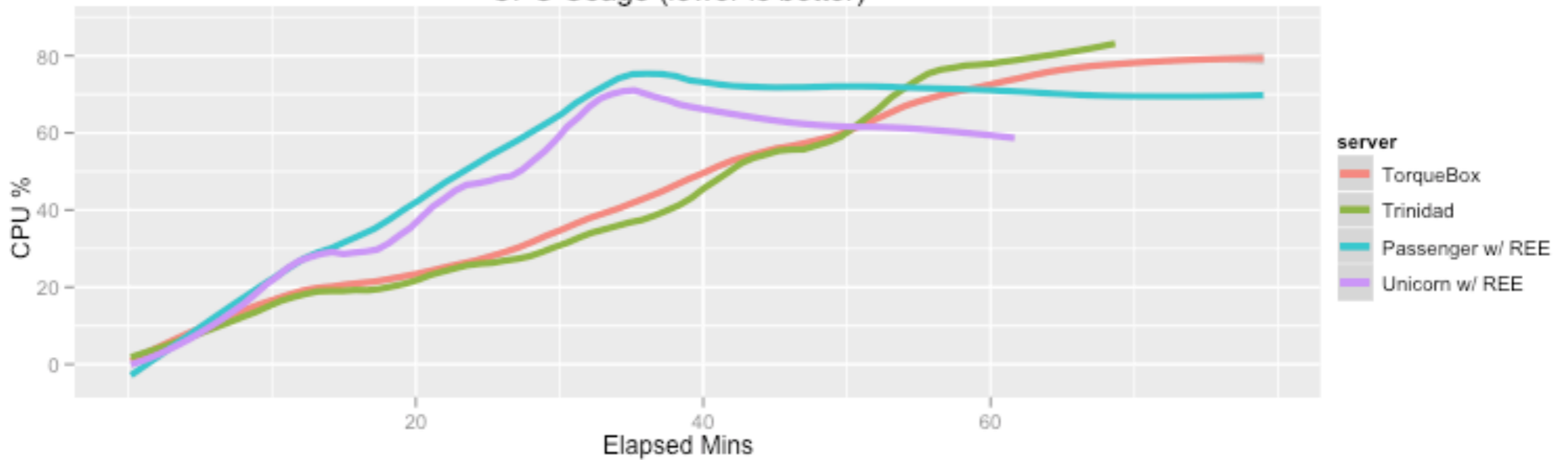
Latency (lower is better)



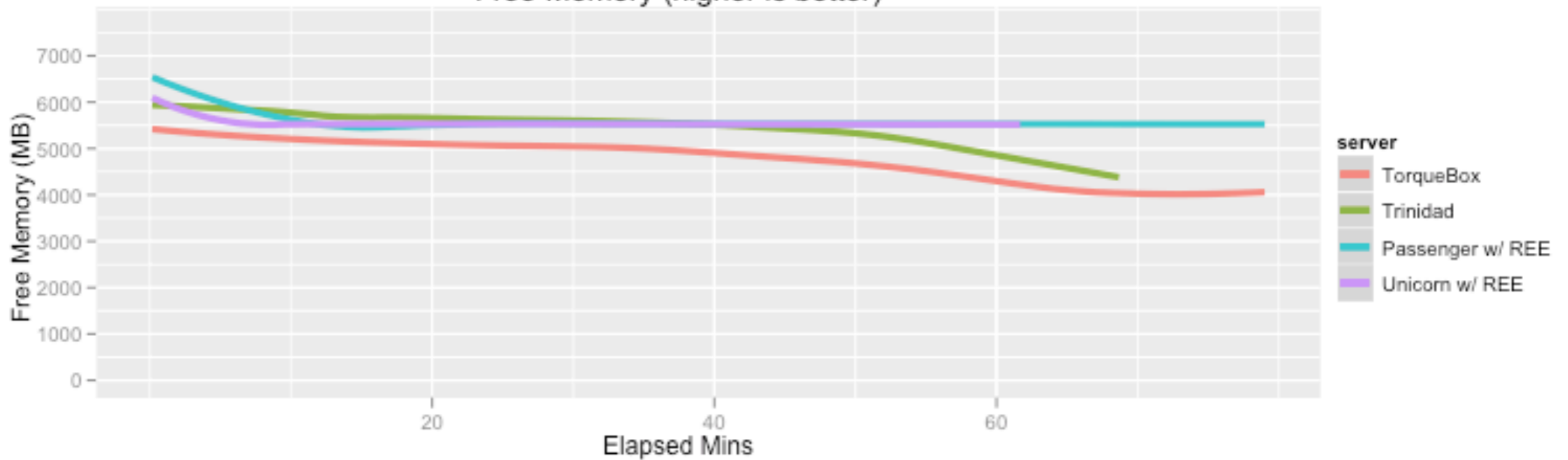
Throughput (higher is better)



CPU Usage (lower is better)



Free Memory (higher is better)



Roadmap

May - 1.0.0.Final

Then...

AS7

Authentication

Transactions

Drools/jBPM/etc

Mobicents

Resources

- <http://torquebox.org/>
- <http://github.com/torquebox>
- #torquebox on FreeNode
- @torquebox

Thanks, and don't
forget to pick up
some stickers.

