How we can do things go faster. Concurrent and asynchronous technologies overview

Anton Mishchuk

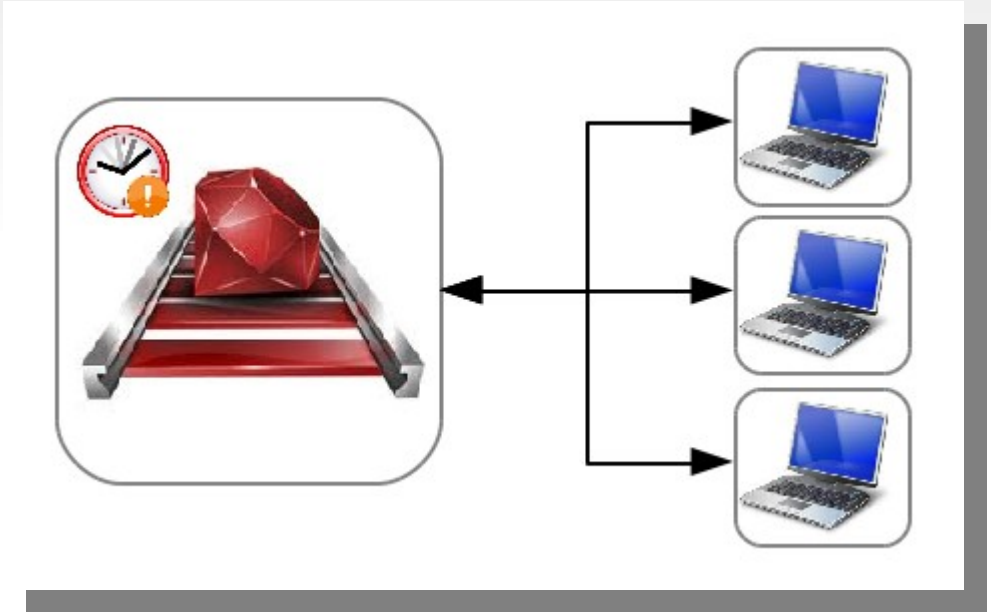
Itransition, Minsk, 2012

The ultimate Ruby apllication

Contents

- Asynchronous execution of long tasks
- Reactor pattern and EventMachine
- HTML5 WebSocket
- Pub/sub systems and services

Few connections
Many calculations



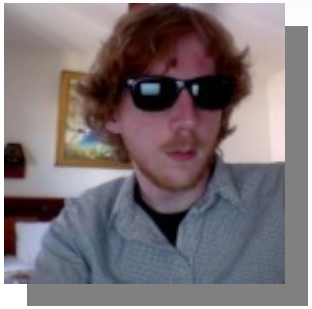
Asynchronous executing long tasks in the background Problems

- Disk manipulating
- Data processing
- Sending massive newsletters
- Statistics gathering
- Http downloads/uploads
- Warming caches
- Updating search index
- Slow insert statements

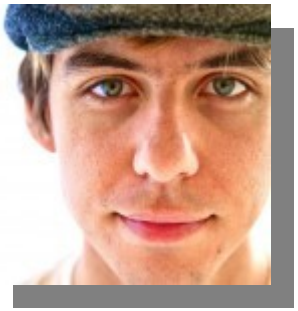
Asynchronous executing long tasks in the background Solutions

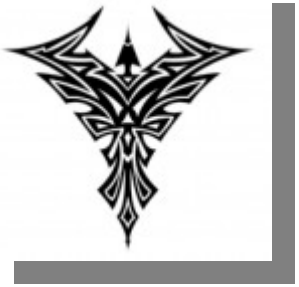
- Delayed_job (tobi, collectiveidea)

- Resque (defunkt)



- Beanstalkd with Stalker (kr, adamwiggins)





Main idea

- Store the tasks and execute them later in different process

Delayed_job

shopify.com - hosted ecommerce solution

- Queue storage – database (serialize Ruby objects)
- Supports many backends (ActiveRecord, DataMapper, Mongoid, Redis)
- Has job priorities
- Retries failed tasks
- Simple integration with Rails app

Delayed_job

- The best way:

```
class Work
  def perform
    #long running method
  end
end
Delayed::Job.enqueue Work.new("do it")
```

- Be carefull:

```
gem 'delayed_job_active_record'
class Calculation
  def calculate
    #long running method
  end
  handle_asynchronously :calculate
end
Calculation.new.calculate
```

Resque

github.com - social coding

- Inspired by DJ but designed for huge queues
- Based on Redis storage. Jobs are persisted as JSON objects
- Supports multiple queues
- Includes a Sinatra app for monitoring

```
class Worker
  @queue = :new_work
  def self.perform(work_id, params)
    work = Work.find(work_id)
    work.do_work(params)
  end
end

class Work
  def async_work(params)
    Resque.enqueue(Worker, self.id, params)
  end
end

Work.new.async_work('do_it')
console> QUEUE=:new_work rake resque:work
```

Resque monitor

Overview | Working | Failed | Queues | Workers | Stats

Queues

The list below contains all the registered queues with the number of jobs currently in the queue. Select a queue from above to view

Name	Jobs
main	6388
failed	0

1 of 1 Workers Working

The list below contains all workers which are currently running a job.

	Where	Queue	Processing
	antonmi-1005PXD:6629	MAIN	ResqueTask NaN days ago

Beanstalkd (with Stalker)

postrank.com - aggregator of social engagement

- Uses own memcashed like storage
- Has priorities and timeouts
- Before and error tasks
- Simple & fast

```
class AnyClass
  def self.work(data)
    #long work with data
  end
end

Stalker.enqueue('stalker_job', data: id)

job "stalker_job" do |args|
  AnyClass.work(args[:data])
end
```

What is the best solution?

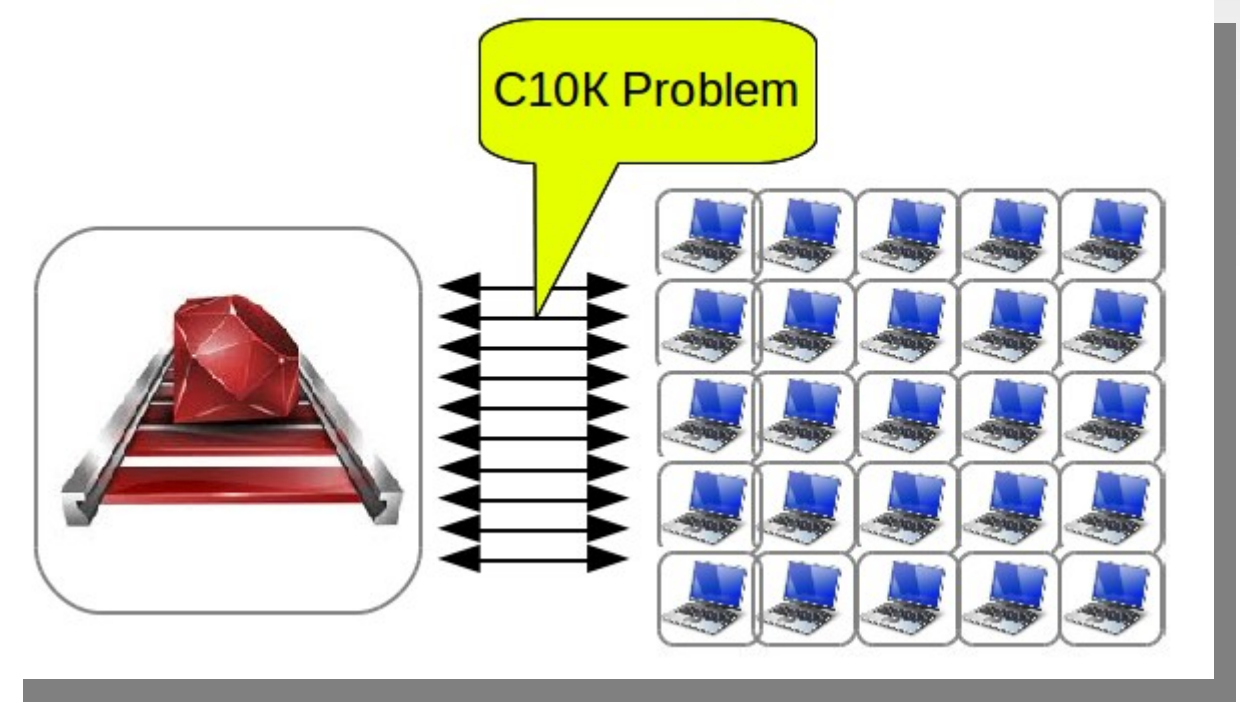
- It depends …

	Enqueue, jobs/sec	Work, jobs/sec
delayed_job	200	120
resque	3800	300
beanstalkd	9000	5200

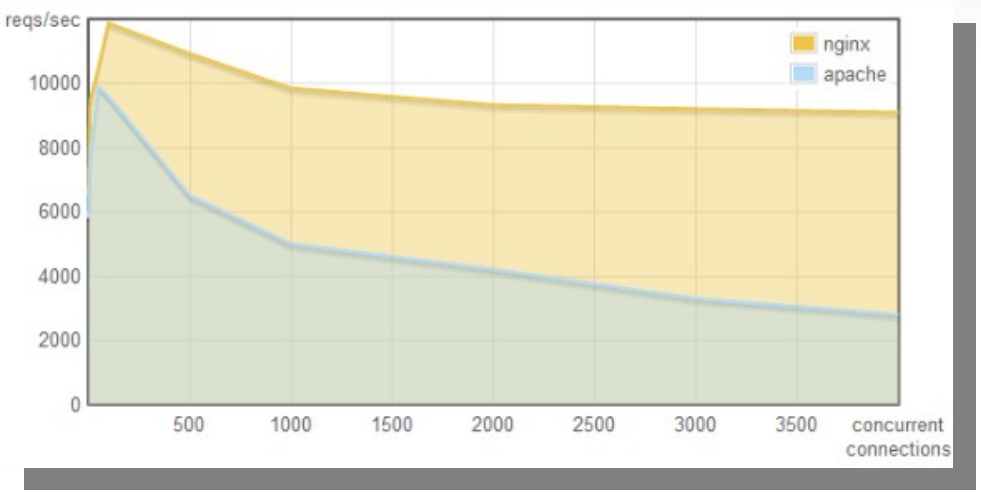
http://adam.heroku.com/past/2010/4/24/beanstalk_a_simple_and_fast_queueing_backend/

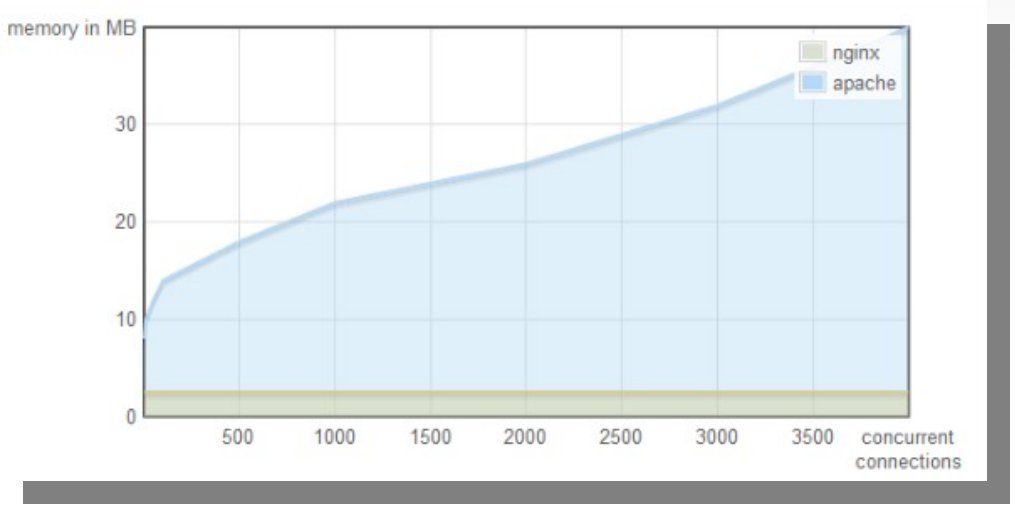
	Enqueue,10000 tasks	Work, 10000 tasks
delayed_job	620 sec	1370 sec
resque	33 sec	310 sec
beanstalkd	6 sec	173 sec

Many connections
Few calculations



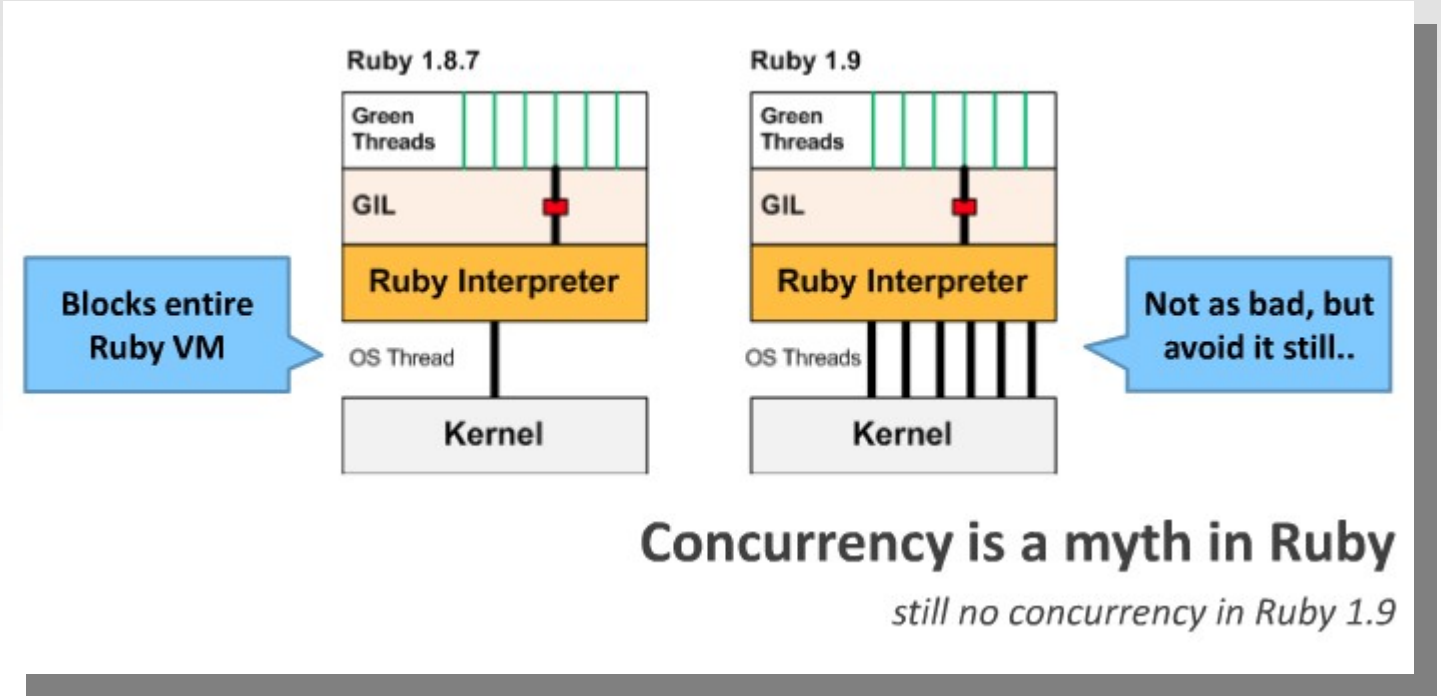
Threads or Events? Servers





From Carlos Vasquez presentation:
http://www.slideshare.net/carlosforero3/ruby-eventmachine-emwebsocket?src=related_normal&rel=4675626

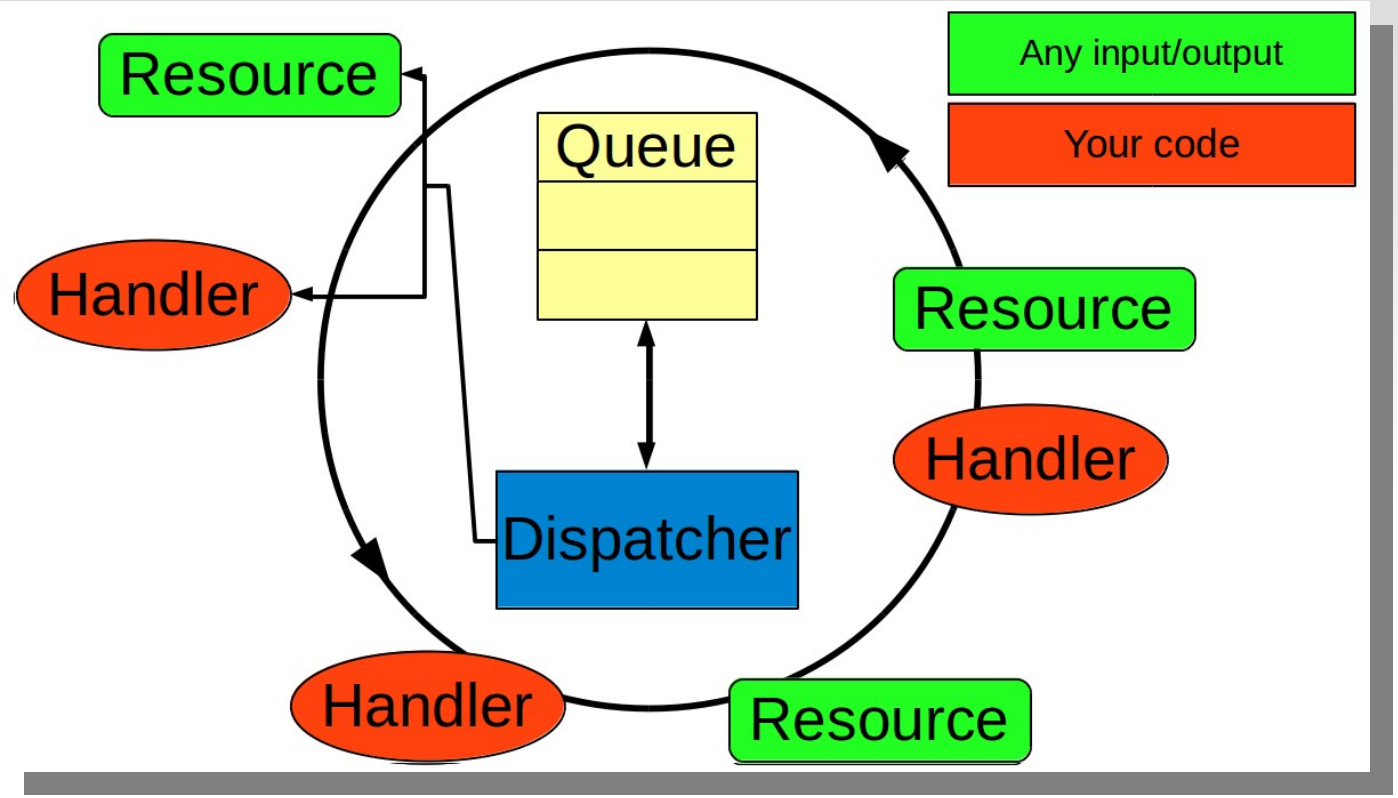
Threads in Ruby



GIL (Global Interpreter Lock) is a mutual exclusion lock held by a programming language interpreter thread to avoid sharing code that is not thread-safe with other threads.

From Ilya Grigorik presentation:
http://www.slideshare.net/igrigorik/no-callbacks-no-threads-railsconf-2010

Reactor



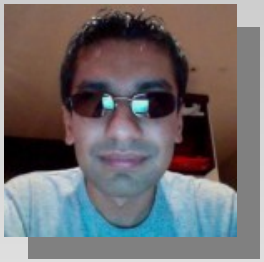
- Reactor is a single threaded while loop (”reactor loop”)
- The code in the loop ”reacts” to incoming events
- If event handler takes too long, other events cannot fire
- Ideal for I/O based application

Reactor pattern implementations

- Ruby: EventMachine
- Javascript: Node.js
- Python: Twisted
- Java: Jboss Netty
- C#: Interlace
- ...

EventMachine

tmm1 (Aman Gupta)



- Most web application are I/O bound, not CPU bound
- Basic idea – instead of waiting a response from the network use that time to process other request
- EventMachine provides event-driven I/O using the Reactor pattern

EventMachine API

- EM.run
- EM.stop
- EM.next_tick
- EM.schedule
- EM.defer
- EM.system
- EM.connect
- EM.start_server
- EM.popen
- EM.watch
- EM::TickLoop
- EM::Deferrable
- EM.Callback
- EM::Timer
- EM::PeriodicTimer
- EM::Queue
- EM::Channel
- EM::Iterator
- EM::Connection
- EM::Protocols

Let's code

Protocol Implementation

- SMTP
- HTTPClient
- HTTPClient2
- Postgres
- MemCache
- Stomp
- Socks4
- ObjectProtocol
- SASLauth
- LineAndText
- LineText2
- …..

- MySQL
- Redis
- Beanstalk
- HTTPRequest
- PubSubHubbub
- Proxy
- WebSocket
- IRC
- Cassandra
- Solr
- SSH
- …..

Ilya Grigorik (igrigorik)

- Social & Google+
  Analytics at Google
- Founder & CTO of
  PostRank (Google)
- Open-source evangelist
- Web engineer
- Photographer

- www.igvita.com

- em-http-request
- em-websocket
- em-synchrony
- etc, etc ...

em-http-request

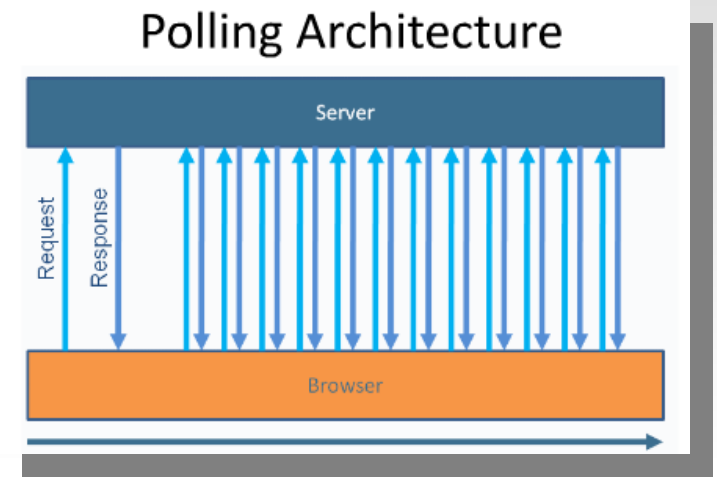
- Keep-Alive and HTTP pipelining support
- Auto-follow 3xx redirects with max depth
- Automatic gzip & deflate decoding
- Streaming response processing
- Streaming file uploads
- HTTP proxy and SOCKS5 support
- Basic Auth & OAuth
- Connection-level & Global middleware support

Real-time

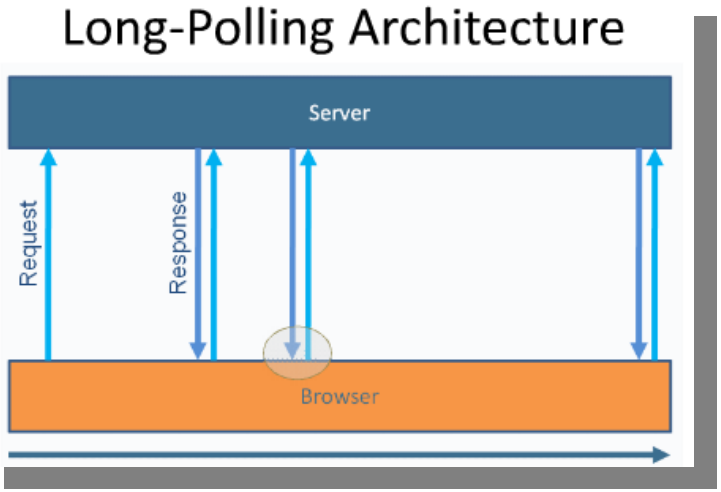
- Notifications
- Chat
- Stocks
- Games
- Collaboration
- etc …
- etc ...

Real-time?

- Polling (AJAX)



- Long Polling (Comet)



From ffdead presentaion: http://www.slideshare.net/ffdead/the-html5-websocket-api

HTTP overhead

```
GET /PollingStock//PollingStock HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5)
Gecko/20091102 Firefox/3.5.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us
Accept-Encoding: gzip,deflate Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300      691 chars
Connection: keep-alive
Referer: http://localhost:8080/PollingStock/
Cookie: showInheritedConstant=false;
    showInheritedProtectedConstant=false; showInheritedProperty=false;
    showInheritedProtectedProperty=false; showInheritedMethod=false;
    showInheritedProtectedMethod=false; showInheritedEvent=false; showInheritedStyle=false;
    showInheritedEffect=false
HTTP/1.x 200 OK
X-Powered-By: Servlet/2.5 Server: Sun Java System Application Server 9.1_02
Content-Type: text/html;charset=UTF-8
Content-Length: 321
Date: Sat, 07 Nov 2009 00:32:46 GMT
```

Total 871 chars

1.000 clients polling every second:

$1.000 \cdot 871 \cdot 8 = 7$ Mbps !!!!!

Websocket

- WebSocket protocol - connection established by upgrading from HTTP to WebSocket protocol
- True full-duplex communication channell
- Proxy/Firewall friendly:

  - runs via port 80/443

  - integrates with cookie based authentication
- Secure connection via Secure WebSockets

Websocket

- Each message has only 2 bytes of overhed

  (0x00<data>0xFF)
- No lattency from establishing new connection
- No polling overhaed (only sends messages whe there is something to send)

1.000 clients send message every second:

$1.000 \cdot 2 \cdot 8 =$ **16** kbps (was **7** Mbps)

WebSocket API

- onopen
- onclose
- onmessage
- onerror
- send
- close

em-websocket

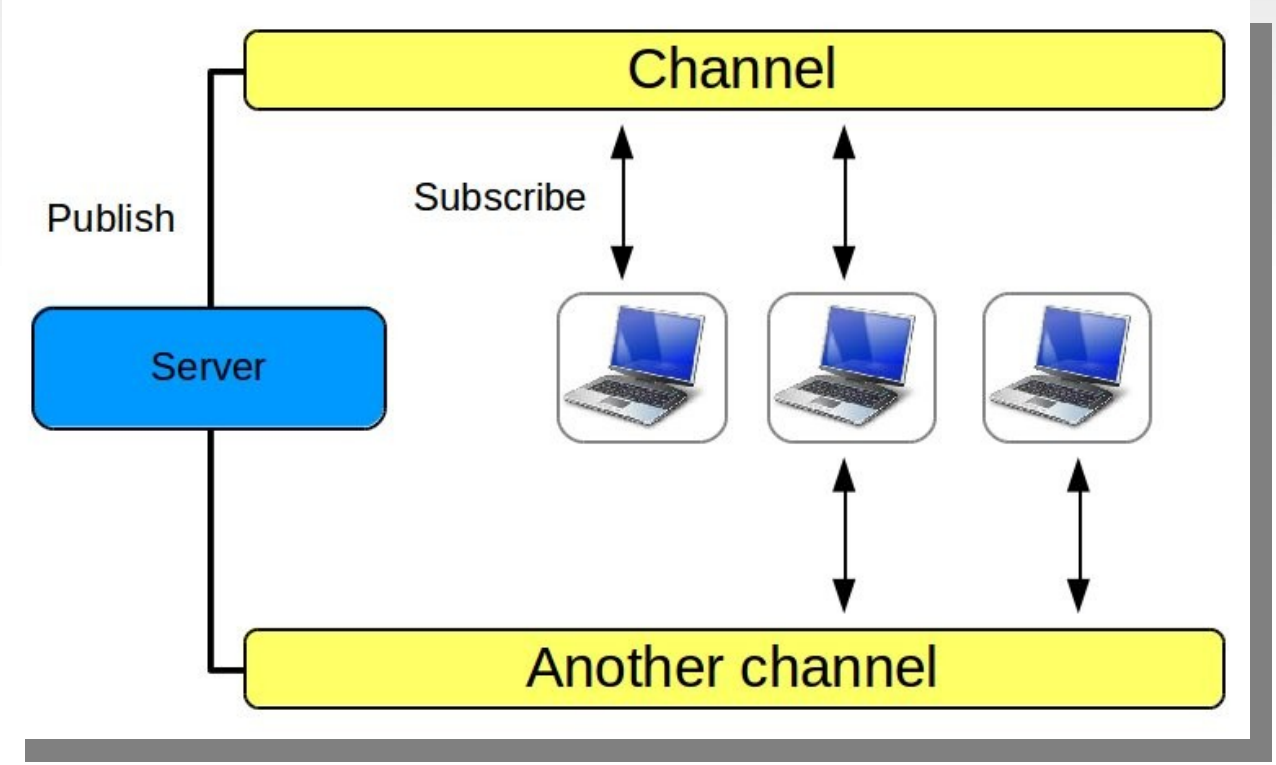
- See example

EventMachine in Rails

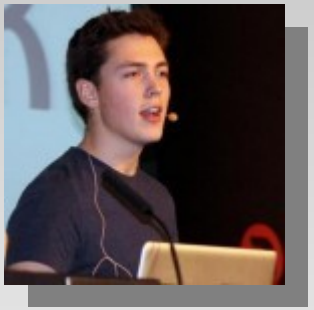
- Run EM in a thread
- Use EM enabled webservers:

  Thin, Goliath, Rainbow
- See example

Publish Subcribe pattern



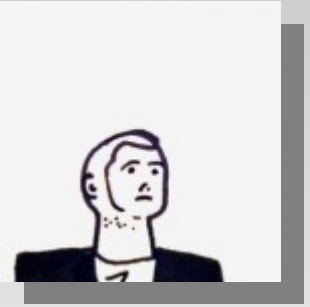
Juggernaut (maccman)



- Node.js server
- Redis storage
- Ruby client
- Supports the following protocols:

  WebSocket, Adobe Flash Socket, ActiveX HTMLFile , XHR, ...
- Horizontal scaling
- Reconnection support
- SSL support

Faye
(jcoglan)



- Node.js or EventMachine servers
- Bayeux protocol (primarily over HTTP)
- Memory or Redis storage
- Faye::WebSocket
- etc, etc …

- private_pub gem (Ryan Bates)

Private Pub

- In the view:

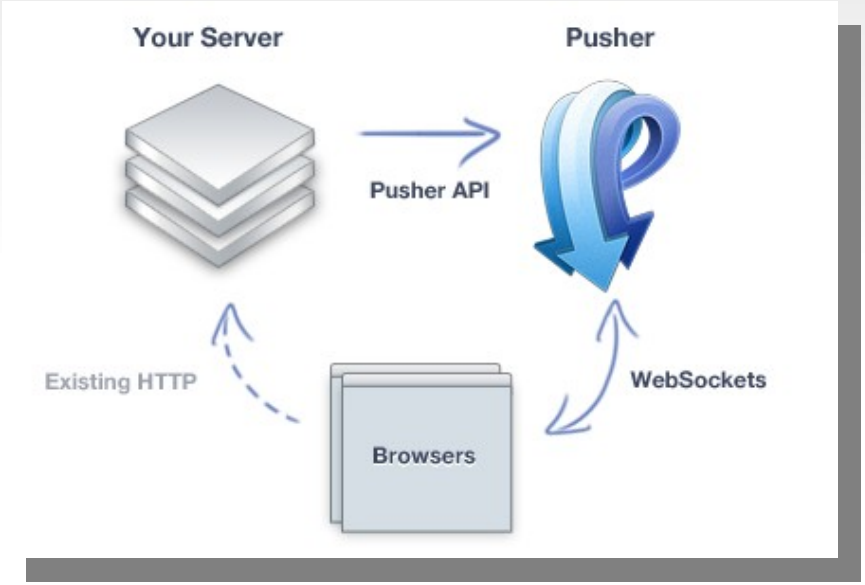
```
PrivatePub.subscribe("/messages", function(data, channel){
    //do smth with data
})
```

- In the controller:

```
PrivatePub.publish_to "/messages/", @message
```

Pusher.com
realtime messaging platform

- Pub/Sub model with channels, events and webhooks
- Rich suite of libraries
- WebSockets with fallback to Flash
- Public, private and presence channel
- Tools for monitoring and debugging



Pusher client libraries

- Javascript
- iOS – Objective C
- iOS – Appcelerator Titanium
- Android – Java
- ActionScript
- .Net & Silverlight
- Ruby
- Ardunio

Pusher publisher libraries

- Node.js
- Java
- Groovy/Grails
- Clojure
- Python
- Ruby
- VB.NET
- C#
- PHP
- Perl
- Coldfushion

Pusher Channels

- Public channels – can be subscribed by anyone.
- Private channels – allow controll access to the data you are broadcasting
- Presence channels – let you register user information on subscription and let other members of channel know who is online

Pusher events

- Connection events: connecting, connected, disconnected, unavailable, failed
- Channel events: subscription_succeeded, subscription_error, member_added, member_removed
- Custom events for channel(s)

```
var pusher = new Pusher('API_KEY');
var channel = pusher.subscribe('APPL');
channel.bind('new-price',
  function(data) {
    // add new price into the APPL widget
  }
);
```

Pusher webhooks

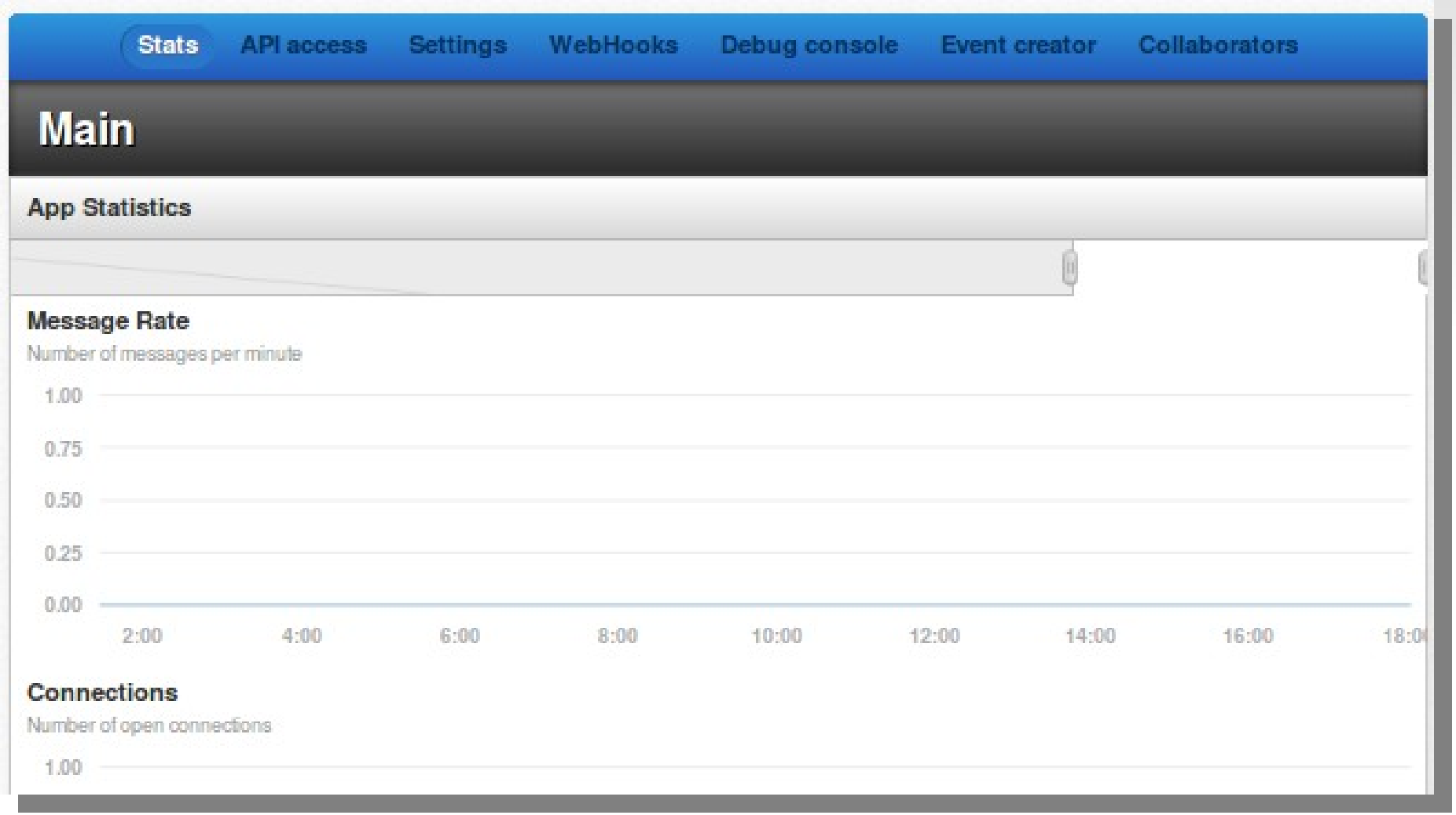
- Allow your server to be notified about events occuring within Pusher
- Pusher sends HTTP POST request

  to the specified url

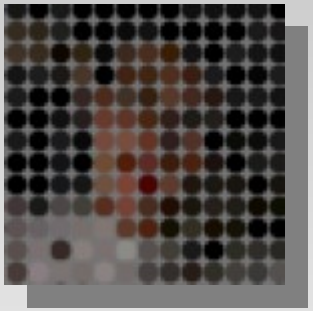
```
{
  "time_ms": 1327078148132
  "events": [
    { "name": "event_name", "some": "data" }
  ]
}
```

- Examples: channel_occupied, channel_vacated

Pusher monitoring an debugging



pusher-gem (mlaughran)



- Configure app

```
Pusher.app_id = '16661'
Pusher.key = '7382735ce49c18967856'
Pusher.secret = '1d767e0dc26bcbd11329'
```

```
<script type="text/javascript">
  var pusher = new Pusher('7382735ce49c18967856');
  var demo_channel = pusher.subscribe('demo_channel');
  demo_channel.bind('show_message', function(data){
      console.log(data);
      $('#message').html(data);
  })
</script>
```

- Trigger events

```
Pusher['demo_channel'].trigger('show_message', "Hello Pusher")
```

- Use EM

```
def pusher
  EM.schedule do
    EM::Timer.new(5) do
      push = Pusher['demo_channel'].trigger_async('show_message', "Use EM")
      push.callback {puts "Message sent"}
    end
  end
end
```

Pusher Pricing

Sandbox	Bootstrap	Startup	Big Boy	Enterprise
FREE	$19/month	$49/month	$199/month	Contact us
20 Max Connections	100 Max Connections	500 Max Connections	5,000 Max Connections	Millions Connections
100,000 Messages per day	200,000 Messages per day	1 million Messages per day	10 million Messages per day	Billions Messages per day
~~Encryption~~ SSL Protection	~~Encryption~~ SSL Protection	Encryption SSL Protection	Encryption SSL Protection	Premium support 24/7 Telephone

Defend Ruby from ”Ruby is slow” people

References
Delayed Job and others

- https://github.com/tobi/delayed_job
- https://github.com/collectiveidea/delayed_job
- http://railscasts.com/episodes/171-delayed-job-revised
- https://github.com/defunkt/resque
- http://railscasts.com/episodes/271-resque
- http://kr.github.com/beanstalkd/

  https://github.com/adamwiggins/stalker
- http://railscasts.com/episodes/243-beanstalkd-and-stalker
- http://adam.heroku.com/past/2010/4/24/beanstalk_a_simple_and_fast_queue

References EventMachine

- https://github.com/eventmachine/eventmachine
- http://www.youtube.com/watch?v=mPDs-xQhPb0
- http://www.viddler.com/v/cfadc37f
- https://github.com/igrigorik/em-synchrony
- http://www.igvita.com/2010/03/22/untangling-evented-code-with-ruby-fibers/
- http://www.igvita.com/2009/12/22/ruby-websockets-tcp-for-the-browser/
- https://github.com/igrigorik/async-rails
- http://www.slideshare.net/jweiss/eventmachine
- http://www.slideshare.net/igrigorik/no-callbacks-no-threads-railsconf-2010
- http://www.slideshare.net/autonomous/ruby-concurrency-and-eventmachine

References EventMachine

- http://www.slideshare.net/carlosforero3/ruby-eventmachine-emwebsocket?sr
- http://www.slideshare.net/kbal11/ruby-19-fibers?src=related_normal&rel=481
- http://www.slideshare.net/KyleDrake/fast-concurrent-ruby-web-applications-w
- http://www.scribd.com/doc/28253878/EventMachine-scalable-non-blocking-i-o
- http://code.macournoyer.com/thin/
- http://www.igvita.com/2011/03/08/goliath-non-blocking-ruby-19-web-server/
- http://www.slideshare.net/ismasan/websockets-and-ruby-eventmachine
- http://www.slideshare.net/ffdead/the-html5-websocket-api

References Pub/Sub

- http://juggernaut.rubyforge.org/
- https://github.com/maccman/juggernaut
- http://faye.jcoglan.com/
- http://railscasts.com/episodes/260-messaging-with-faye
- https://github.com/ryanb/private_pub
- http://railscasts.com/episodes/316-private-pub
- https://github.com/pusher/pusher-gem/tree/master/examples
- http://pusher.com
- https://github.com/pusher/pusher-gem

Thank you

- Questions?