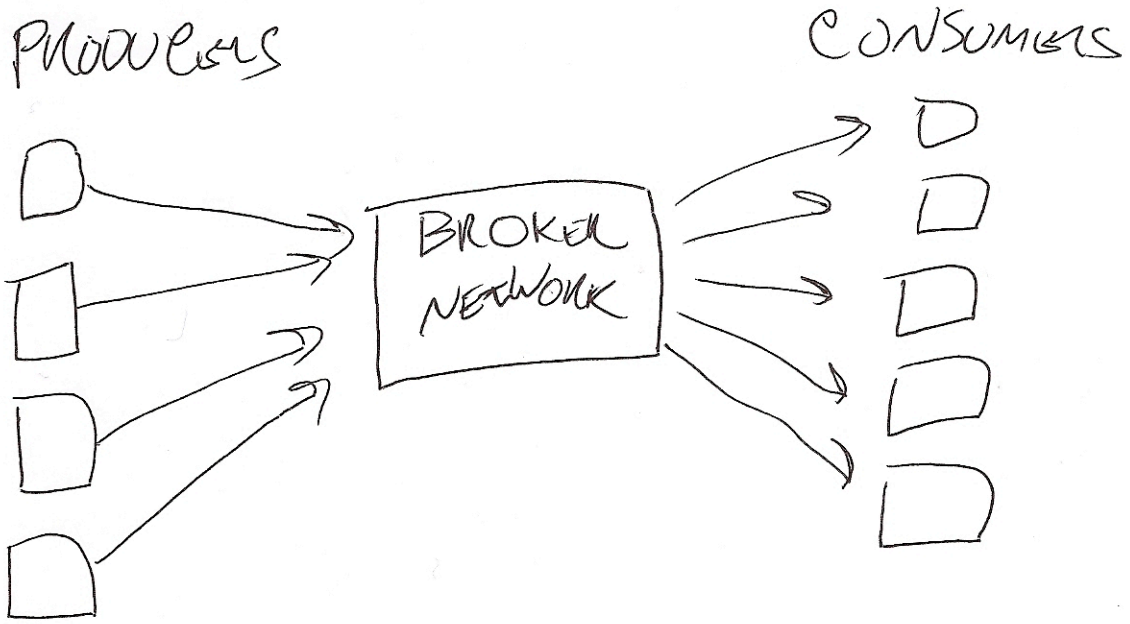
ActiveMessaging

Cliff Moon
Powerset, Inc.

Messaging Architecture



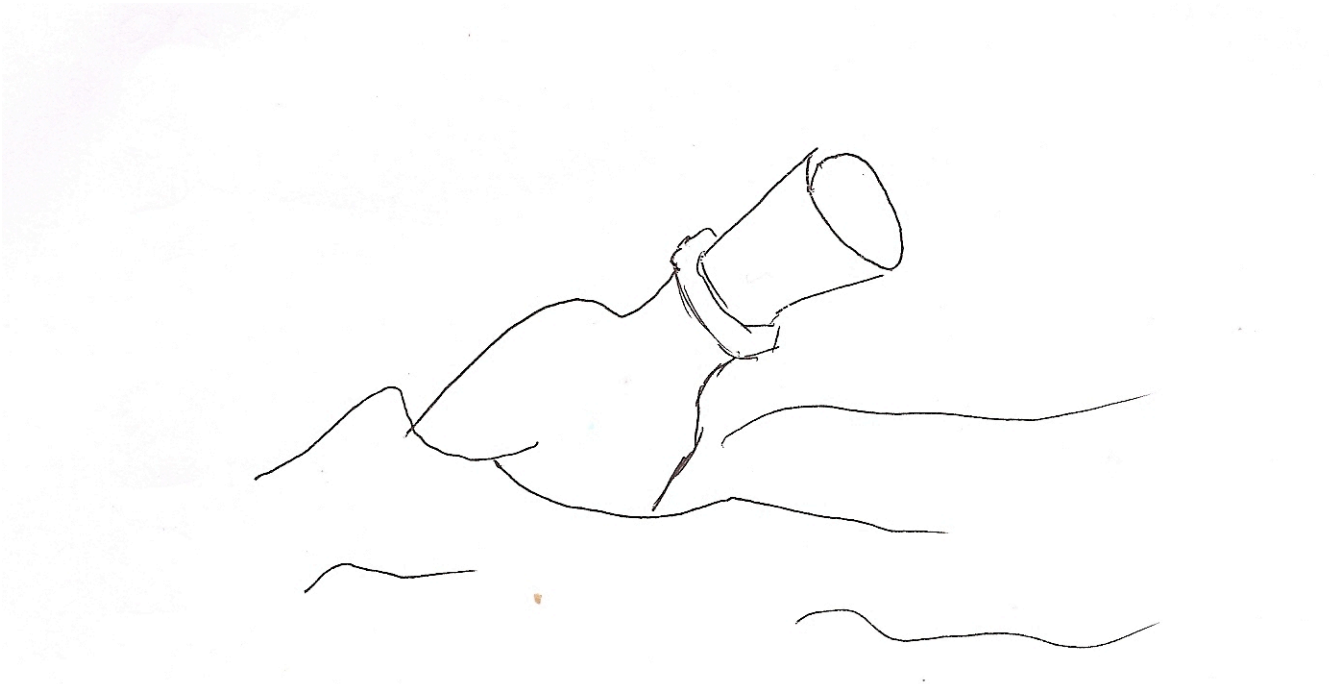
ActiveMessaging...

- is a compatibility layer for messaging
- has adapters for a number of popular brokers
- provides infrastructure to get up and running quickly

Role of the broker

- decoupling consumers and producers
- impedance matching
- integration point

Message in a bottle



Messaging is good for...

- reporting / logging
- polyglot environments
- work queues
- broadcasting
- poor man’s erlang...

Not so great for...

- RPC
- Anything requiring synchronicity
- Apps that need a persistence guarantee

Fighting with AMQ

- :ack => ‘client’
- Durable subscriptions are the devil
- Virtual topics/queues
- Message groups

Persistence

- Effectively busted in 5.0. Use a 5.1 snapshot.
- Purging and deleting queues/topics from JMX/web interface rarely works.
- Durable subscriptions are the devil and cannot be counted upon to work right.

Networks of Brokers

- Each broker NEEDS a unique name
- Broker networks can get into weird states
- Topics are replicated via durable subscriptions. Which are the devil.

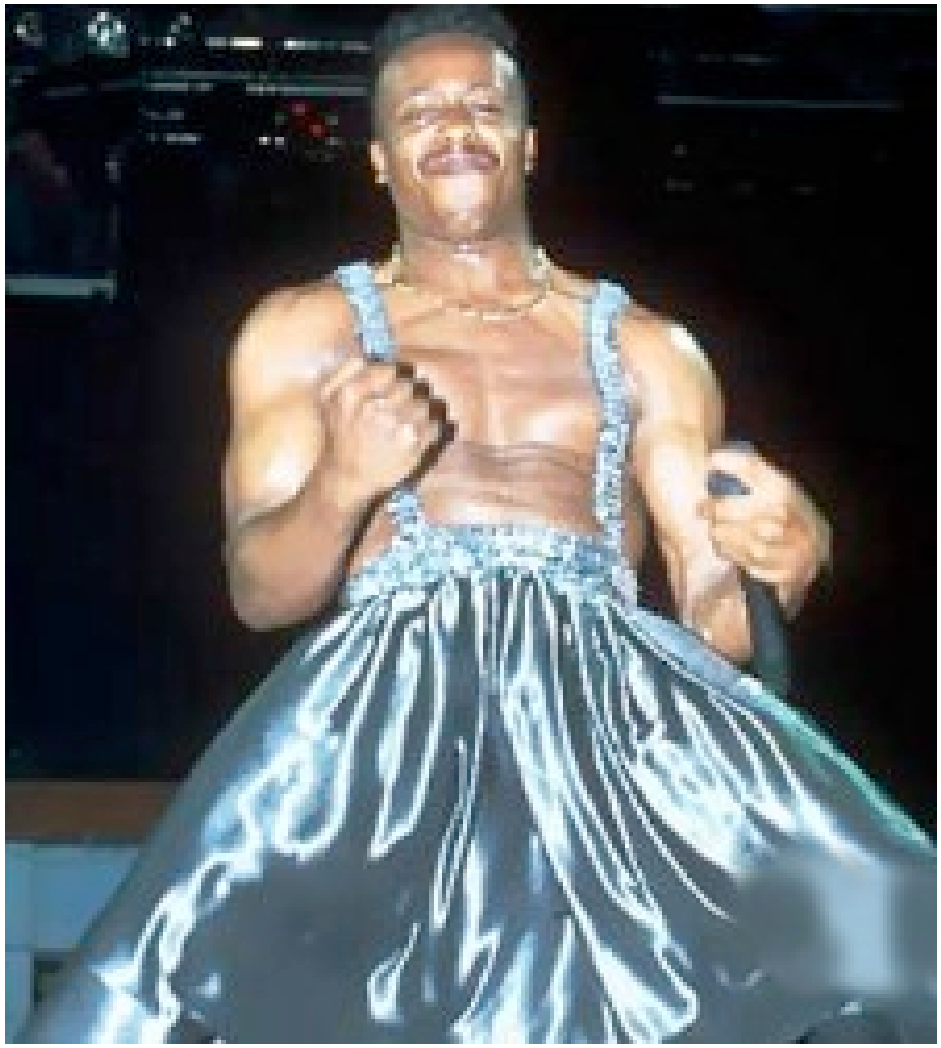
ActiveMessaging Layout

- app/processors - standalone message consumers
- config/broker.yml - broker connection configs
- config/messaging.rb - processor group and destination config

Examples

- One fake
- One real

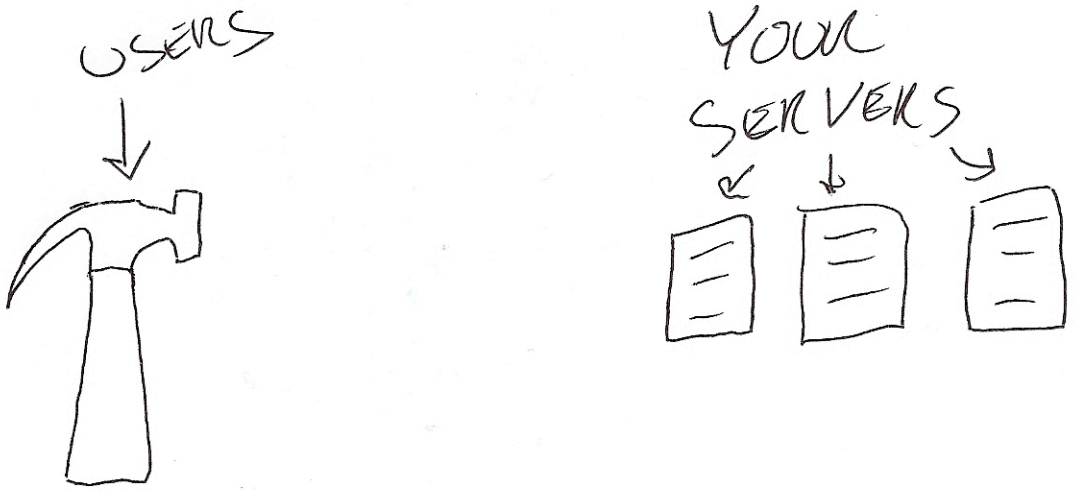
Can’t touch this.



The youtube of dance videos

- built on rails for quick time to market
- traffic pattern is 1 video upload to 10 views
- transcoding done on app servers using backgroundrb
- all is well until...

Hammer Time.



Traffic patterns change

- Dancing with the stars world tour
- app servers fill up with transcoding jobs
- CPU and disk seek go through the roof
- Cannot serve existing videos
- EPIC FAIL

A solution

- Upload app stores videos in shared datastore
- Sends message to encoding queue
- Encoders are hosted on either a dedicated server or a slice of an app server
- Encoders insert metadata into the DB and/or notify the user

Hammer Time Again.

- Work queue starts to get big since encoders cannot keep up
- You can either:
  - Add more encoders
  - Wait for them to catch up after the storm
- You can still serve video!
  - Since you know how many concurrent encoders your datastore can handle, right?

Query Collection the old way

- Queries are inserted into a live database
- cron job runs export script
- Posts CSV batch to analytics app
- Changes propagate between 2 database schemas and 2 apps

Query Collection today: Voyeur

- ActiveMessaging from Merb
- sends messages to virtual topics, arranged logically as data streams
- analytics apps pull from virtual topic queues
- virtual topics allow us to have multicast AND load balancing
- messages are asynchronous: ACID is expensive.

broker.yml

```
1  #
2  # broker.yml
3  #
4  # Simple yaml file for the env specific configuration
5  # See the wiki for more information: http://code.googl
6  #
7  development:
8      ##############################
9      # Stomp Adapter Properties #
10     ##############################
11     adapter: stomp
12     host: localhost
13     login: ""
14     passcode: ""
15     port: 61613
16     reliable: true
17     reconnectDelay: 5
```

messaging.rb

```
5  ActiveMessaging::Gateway.define do |s|
6    #s.destination :orders, '/queue/Orders'
7    #s.filter :some_filter, :only=>:orders
8    #s.processor_group :group1, :order_processor
9    s.processor_group :clickstream, :clickstream_processor
10   s.processor_group :querystream, :querystream_processor
11   s.processor_group :mousestream, :mouse_processor
12   s.processor_group :belushinatorstream, :belushinator_processor
13   s.processor_group :componenterrorstream, :componenterror_processor
14   s.processor_group :errorstream, :errorstream_processor
15
16   s.destination :clickstream, '/queue/Consumer.voyeur.VirtualTopic.Clickstream'
17   s.destination :querystream, '/queue/Consumer.voyeur.VirtualTopic.Querystream'
18   s.destination :mousestream, '/queue/Consumer.voyeur.VirtualTopic.Mousestream'
19   s.destination :belushinatorstream, '/queue/Consumer.voyeur.VirtualTopic.Belushinato
20   s.destination :componenterrorstream, '/queue/Consumer.voyeur.VirtualTopic.Component
21   s.destination :errorstream, '/queue/Consumer.voyeur.VirtualTopic.Errorstream'
22 end
```

processor

```ruby
class ClickstreamProcessor < ApplicationProcessor

  subscribes_to :clickstream, :ack => 'client'

  def on_message(message)
    clickstream = YAML.load(message)
    write_records(timestamp) do |r|
      r.record('clicks', 1)
      r.record('request_latency', clickstream[:latency])
      r.record('uniques', clickstream[:unique])
    end

    Clickstream.create(clickstream)
    return nil
  end

  protected

  def setup_recording(c)
    c.setup('clicks', 'absolute')
    c.setup('request_latency', 'seconds')
    c.setup('uniques', 'absolute')
  end
end
```

the producer - part 1

```
216   def record_clickstream
217     time = Time.now.to_f + @_clickstream_time
218     Stenographer.record(:clickstream, {
219       :session_id => session_id,
220       :ip_address => anon_ip_address,
221       :logged_in => !current_email.nil?,
222       :user_agent => request.env["HTTP_USER_AGENT"],
223       :language => request.env["HTTP_ACCEPT_LANGUAGE"],
224       :uri => request.env["REQUEST_URI"],
225       :referrer => request.env["HTTP_REFERER"],
226       :timestamp => Time.now,
227       :latency => time,
228       :unique => @unique
229     }, :click_id)
230   end
231 end
232
```

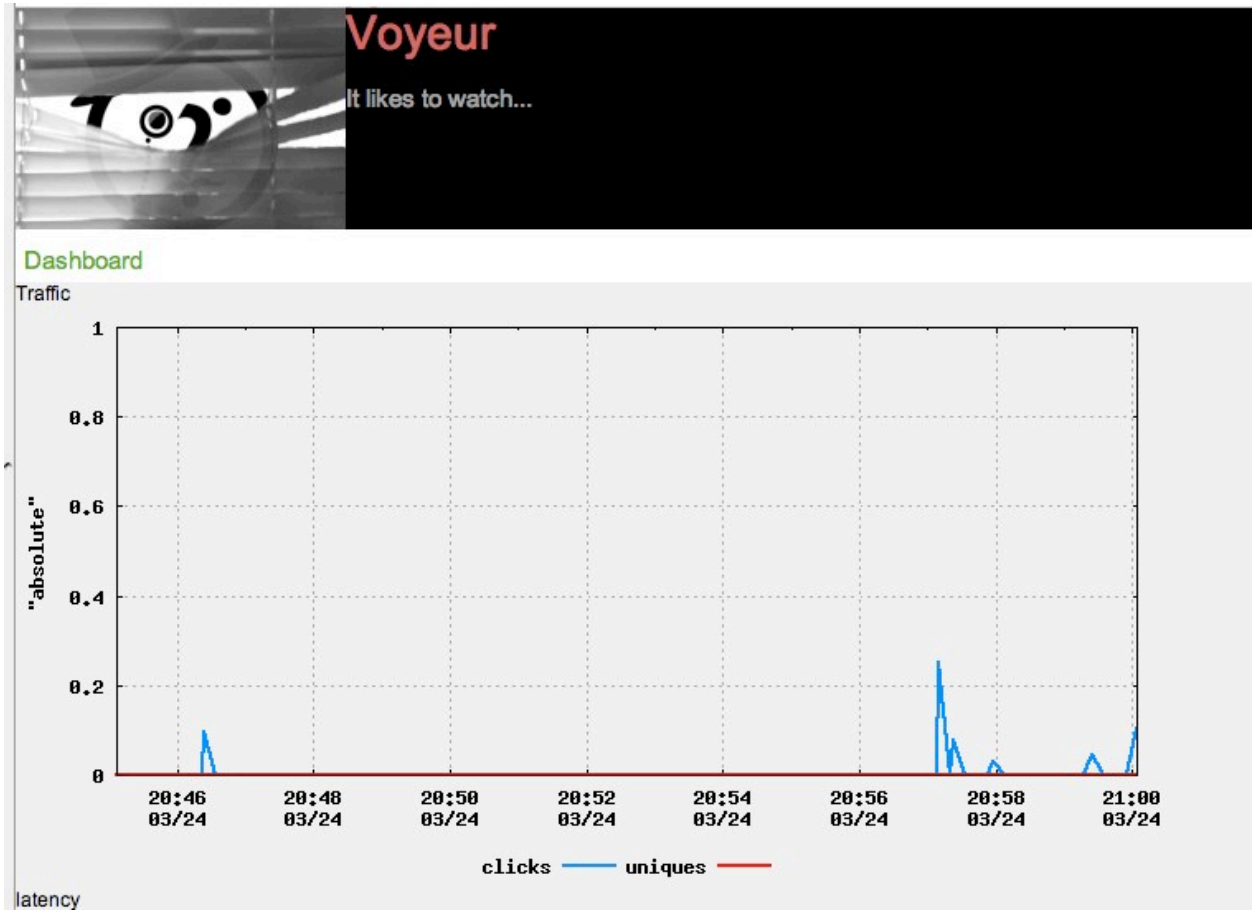
the producer - part 2

```ruby
class Stenographer
  include ActiveMessaging::MessageSender

  def record(destination, message={}, id_name=:id, with_group=false)
    message[id_name] = Digest::SHA1.hexdigest(message.to_yaml)
    headers = {}
    headers['JMSXGroupID'] = message[id_name] if with_group
    if Powerset.config[:voyeur]
      publish destination.to_sym, message.to_yaml, headers
    end
    message[id_name]
  end

  class <<self
    def record(destination, message={}, with_group=false)
      Stenographer.new.record(destination, message, with_group)
    end
  end
end
```

Demo


Voyeur
It likes to watch...
Dashboard
Traffic
"absolute"
1
0.8
0.6
0.4
0.2
0
20:46 03/24
20:48 03/24
20:50 03/24
20:52 03/24
20:54 03/24
20:56 03/24
20:58 03/24
21:00 03/24
clicks
uniques
latency

Summary

- Databases make crappy integration points
- Scalability is about efficiently distributing work across a cluster
- This is one solution. There are many.

Lets go drink beer.