Building a Distributed Data Ingestion System with RabbitMQ

Alvaro Videla - RabbitMQ
Alvaro Videla

- Developer Advocate at Pivotal / RabbitMQ
- Co-Author of RabbitMQ in Action
- Creator of the RabbitMQ Simulator
- Blogs about RabbitMQ Internals: http://videlalvaro.github.io/internals.html
- @old_sound — alvaro@rabbitmq.com — github.com/videlalvaro
About Me

Co-authored

RabbitMQ in Action

About this Talk

• Exploratory Talk

• A ‘what could be done’ talk instead of ‘this is how you do it’
Agenda

• Intro to RabbitMQ
• The Problem
• Solution Proposal
• Improvements
What is RabbitMQ
RabbitMQ
RabbitMQ
RabbitMQ

- Multi Protocol Messaging Server
RabbitMQ

- Multi Protocol Messaging Server
- Open Source (MPL)
RabbitMQ

- Multi Protocol Messaging Server
- Open Source (MPL)
- Polyglot
RabbitMQ

- Multi Protocol Messaging Server
- Open Source (MPL)
- Polyglot
- Written in Erlang/OTP
Multi Protocol

Community Plugins

http://www.rabbitmq.com/community-plugins.html
Polyglot
Polyglot
Polyglot

• Java
Polyglot

• Java
• node.js
Polyglot

• Java
• node.js
• Erlang
Polyglot

• Java
• node.js
• Erlang
• PHP
Polyglot

• Java
• node.js
• Erlang
• PHP
• Ruby
Polyglot

- Java
- node.js
- Erlang
- PHP
- Ruby
- .Net
Polyglot

• Java
• node.js
• Erlang
• PHP
• Ruby
• .Net
• Haskell
Polyglot

Even COBOL!!!11
Some users of RabbitMQ
Some users of RabbitMQ

• Instagram
Some users of RabbitMQ

- Instagram
- Indeed.com
Some users of RabbitMQ

- Instagram
- Indeed.com
- Telefonica
Some users of RabbitMQ

• Instagram
• Indeed.com
• Telefonica
• Mercado Libre
Some users of RabbitMQ

- Instagram
- Indeed.com
- Telefonica
- Mercado Libre
- NHS
Some users of RabbitMQ

- Instagram
- Indeed.com
- Telefonica
- Mercado Libre
- NHS
- Mozilla
The New York Times on RabbitMQ

This architecture - Fabrik - has dozens of RabbitMQ instances spread across 6 AWS zones in Oregon and Dublin.

Upon launch today, the system autoscaled to ~500,000 users. Connection times remained flat at ~200ms.

http://www.rabbitmq.com/download.html

Unix - Mac - Windows
Messaging with RabbitMQ

A demo with the RabbitMQ Simulator

https://github.com/RabbitMQSimulator/RabbitMQSimulator
http://tryrabbitmq.com
RabbitMQ Simulator
The Problem
Distributed Application
Distributed Application
{ok, Connection} = amqp_connection:start(#amqp_params_network{host = "localhost"}),

{ok, Channel} = amqp_connection:open_channel(Connection),
Data Producer

Declare an Exchange

amqp_channel:call(Channel, #'exchange.declare'{exchange = <<"events">>,
                     type = <<"direct">>}),
Data Producer

Publish a message

amqp_channel:cast(Channel,
    #'basic.publish'{
        exchange = <<"events">>,
    #amqp_msg{props = #'P_basic'{delivery_mode = 2},
        payload = <<"Hello Federation">>},
    )
Data Consumer

Obtain a Channel

{ok, Connection} =
    amqp_connection:start(#amqp_params_network{host = "localhost"}),
{ok, Channel} = amqp_connection:open_channel(Connection),
Data Consumer

Declare Queue and bind it

amqp_channel:call(Channel, #'exchange.declare'{exchange = <<"events">>,
                type = <<"direct">>}),

#'queue.declare_ok'{queue = Queue} =
    amqp_channel:call(Channel, #'queue.declare'{exclusive = true}),

amqp_channel:call(Channel, #'queue.bind'{exchange = <<"events">>,
                queue = Queue}),
Data Consumer

Start a consumer

amqp_channel:subscribe(Channel, #'basic.consume'{queue = Queue,
no_ack = true}, self()),
receive
    #'basic.consume_ok'{} -> ok
end,
loop(Channel).
Data Consumer

Process messages

```erlang
loop(Channel) ->
    receive
        #{'basic.deliver'{}, #amqp_msg{payload = Body}} ->
            io:format(" [x] ~p\n", [Body]),
            loop(Channel)
    end.
```
Ad-hoc solution
A process that replicates data to the remote server
Possible issues
Possible issues

• Remote server is offline
Possible issues

- Remote server is offline
- Prevent unbounded local buffers
Possible issues

• Remote server is offline
  • Prevent unbounded local buffers
  • Prevent message loss
Possible issues

- Remote server is offline
- Prevent unbounded local buffers
- Prevent message loss
- Prevent unnecessary message replication
Possible issues

• Remote server is offline
  • Prevent unbounded local buffers
  • Prevent message loss
• Prevent unnecessary message replication
  • No need for those messages on remote server
Possible issues

• Remote server is offline
  • Prevent unbounded local buffers
  • Prevent message loss
• Prevent unnecessary message replication
  • No need for those messages on remote server
  • Messages that became stale
Can we do better?
RabbitMQ Federation
RabbitMQ Federation
RabbitMQ Federation

- Supports replication across different administrative domains
RabbitMQ Federation

• Supports replication across different administrative domains

• Supports mix of Erlang and RabbitMQ versions
RabbitMQ Federation

- Supports replication across different administrative domains
- Supports mix of Erlang and RabbitMQ versions
- Supports Network Partitions
RabbitMQ Federation

- Supports replication across different administrative domains
- Supports mix of Erlang and RabbitMQ versions
- Supports Network Partitions
- Specificity - not everything has to be federated
RabbitMQ Federation

max_hops=no of levels-1
RabbitMQ Federation

max_hops=5
RabbitMQ Federation
RabbitMQ Federation

- It's a RabbitMQ Plugin
RabbitMQ Federation

- It’s a RabbitMQ Plugin
- Internally uses Queues and Exchanges Decorators
RabbitMQ Federation

- It’s a RabbitMQ **Plugin**
- Internally uses **Queues** and **Exchanges Decorators**
- Managed using **Parameters** and **Policies**
Enabling the Plugin

rabbitmq-plugins enable rabbitmq_federation
Enabling the Plugin

rabbitmq-plugins enable rabbitmq_federation

rabbitmq-plugins enable rabbitmq_federation_management
Federating an Exchange

rabbitmqctl set_parameter federation-upstream my-upstream \
'{"uri":"amqp://server-name","expires":3600000}''
Federating an Exchange

rabbitmqctl set_parameter federation-upstream my-upstream \ '{"uri":"amqp://server-name","expires":3600000}'

rabbitmqctl set_policy --apply-to exchanges federate-me "^amq\." \ '{"federation-upstream-set":"all"}'}
Federating an Exchange
Configuring Federation
rabbitmqctl set_parameter federation-upstream \ 
name 'json-object'
Config Options

rabbitmqctl set_parameter federation-upstream \\name ‘json-object’

json-object: {  
  ‘uri’: ‘amqp://server-name/’,  
  ‘prefetch-count’: 1000,  
  ‘reconnect-delay’: 1,  
  ‘ack-mode’: on-confirm
}

Prevent unbound buffers

expires: N // ms.
message-ttl: N // ms.
Prevent message forwarding

max-hops: N
Speed vs No Message Loss

ack-mode: on-confirm
ack-mode: on-publish
ack-mode: no-ack
AMQP URI:

amqp://user:pass@host:10000/vhost

http://www.rabbitmq.com/uri-spec.html
Config can be applied via

- CLI using `rabbitmqctl`
- HTTP API
- RabbitMQ Management Interface
RabbitMQ Federation
Scaling the Setup
The Problem
The Problem

- Queues contents live in the node where the Queue was declared
The Problem

• Queues contents live in the node where the Queue was declared

• A cluster can access the queue from every connected node
The Problem

- Queues contents live in the node where the Queue was declared
- A cluster can access the queue from every connected node
- Queues are an Erlang process (tied to one core)
The Problem

• Queues contents live in the node where the Queue was declared
• A cluster can access the queue from every connected node
• Queues are an Erlang process (tied to one core)
• Adding more nodes doesn’t really help
Enter Sharded Queues
Enter Sharded Queues
Pieces of the Puzzle

- modulo hash exchange (consistent hash works as well)
- good ol’ queues
Sharded Queues

messages:

1 2 3 4 5 6 7 8 9

images exchange

images 1: 1 4 7
images 2: 2 5 8
images 3: 3 6 9
Sharded Queues
Sharded Queues
Sharded Queues
Sharded Queues

• Declare Queues with name: `nodename.queuename.index`
Sharded Queues

- Declare Queues with name: `nodename.queuename.index`
- Bind the queues to a consistent hash exchange
Sharded Queues

- Declare Queues with name: `nodename.queuename.index`
- Bind the queues to a partitioner exchange
- Transparent to the consumer (virtual queue name)
We need more scale!
Federated Queues
Federated Queues

- Load-balance messages across federated queues
- Only moves messages when needed
Federating a Queue

rabbitmqctl set_parameter federation-upstream my-upstream \
'{"uri":"amqp://server-name","expires":3600000}'}
Federating a Queue

```
rabbitmqctl set_parameter federation-upstream my-upstream \ 
  '{"uri":"amqp://server-name","expires":3600000}'

rabbitmqctl set_policy --apply-to queues federate-me "^images\." \ 
  '{"federation-upstream-set":"all"}'
```
With RabbitMQ we can
With RabbitMQ we can

- Ingest data using various protocols: AMQP, MQTT and STOMP
With RabbitMQ we can

• Ingest data using various protocols: AMQP, MQTT and STOMP

• Distribute that data globally using Federation
With RabbitMQ we can

• Ingest data using various protocols: AMQP, MQTT and STOMP
• Distribute that data globally using Federation
• Scale up using Sharding
With RabbitMQ we can

- Ingest data using various protocols: AMQP, MQTT and STOMP
- Distribute that data globally using Federation
- Scale up using Sharding
- Load balance consumers with Federated Queues
Credits

world map: wikipedia.org

federation diagrams: rabbitmq.com
Questions?
Thanks

Alvaro Videla - @old_sound