RabbitMQ

ABHISHEK GUPTA[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: abhigupt@iu.edu*

project-001, March 15, 2017

RabbitMQ provides simple yet powerful messaging platform which allows applications pass messages in a reliable and fault tolerant way. RabbitMQ implements Advanced Message Queuing Protocol (AMQP) and is written in Erlang programming language. It runs on all major operating systems and supports SDK in all major programming languages[1] including objective-C, swift and node.js. When we look for messaging, we look for certain features: asynchronous messaging, large scale, reliability, clustering, multi-protocol, highly available, fault tolerant. RabbitMQ[2] fulfills these requirements and provides a distributed, persistent, highly available, fault tolerant messaging system which can scale as data grows.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/blob/master/paper1/S17-IO-3005/report.pdf

This review document is provided for you to achieve your best. We have listed a number of obvious opportunities for improvement. When improving it, please keep this copy untouched and instead focus on improving report.tex. The review does not include all possible improvement suggestions and for each comment you may want to check if it applies elsewhere in the document.

Please format your lines to be 80 characters long. This helps with reading the paper.

Abstract: "simple yet powerful" is subjective, so leave it out. Everything else looks good.

INTRODUCTION

RabbitMQ based[2]

Citation

Citation should come right after RabbitMQ

off AMQP (Advanced Message Queuing Protocol[3])

Grammar

Which defines how client applications connect to message brokers. Message brokers receive messages from producers and make it available to consumers. The producer posts messages to exchanges and broker agent reads these messages from exchange and writes to queues.

Grammar

Consumers can further read messages from the queue. It also supports a notion of acknowledgements where consumers can post an acknowledgement back to broker and which in turn can post messages back to the producer.

Some grammar issues in the preceding paragraph need to be addressed. In addition, it would be good to start by explaining what problems this producer/broker/consumer scheme is better suited to, compared to standard producer/consumer message passing schemes someone with a CS background might be familiar with.

Overall, a good overview of RabbitMQ, but there are some places where the paper can be more clear. In addition, there are minor language issues like not using singular/plural consistently, not using always articles ("a", "the"), and not capitalizing where appropriate. Finally, you need to review where in the text to best place references to your your sources. See below for a more detailed breakdown.

Assessment: Revisions required. Please address the review comments by end of March.

Looking back at the history[4] of development of rabbitMQ,

Term

Capitalize names properly.

it started with IBM MQseries in 1993, 1997 Microsoft MQ, 2001 java messaging service

Term

.

How do these other techs relate to RabbitMQ?

It was in 2003 where JPMorgan created the first version AMQP

Grammar

which became the base for RabbitMQ technologies formed in 2006.

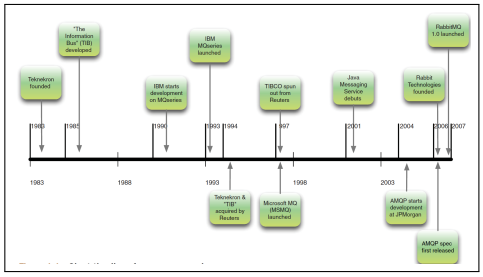

Fig. 1. Timeline for evolution of RabbitMQ [4]

Citation

The citation needs to be inside the caption.

ARCHITECTURE

[4]At high level

Grammar

Citation

A reference needs to go after the statement that refers to it, not before.

, producer publishes the message to the broker. Consumer consumes or subscribes the messages from the broker. Broker is the middle man who has the knowledge of exchanges and queues. It maintains the bindings between exchanges and queues. The consumers read the messages from the queue. Following

Grammar

table shows different exchanges supported by RabbitMQ and corresponding default exchanges[2]

Citation

Please, leave a space between the reference and the word it follows

.

Name	Default pre-declared names
Direct exchange	(Empty string) and amq.direct
Fanout exchange	amq.fanout
Topic exchange	amq.topic
Headers exchange	amq.match (and amq.headers in RabbitMQ)

TYPE OF EXCHANGES

The sections below explains various types of exchanges supported by RabbitMQ.

So far, you haven't explained what exchanges are, so it's difficult to follow.

Default Exchange

It is created by the broker and all queues are bound to default exchange unless specified separately. For example we have a queue called demoqueue, all messages assigned to demoqueue will be routed by default exchange.

Direct exchange

It is used to route messages to queue with a given routing key. For example a message queue has routing key K. A message with same routing key K will be delivered to same message queue.

Fanout exchange

Delivers messages to all queues that are bound to a given exchange ignoring the routing key. For example if there are 5 queues bound to a

Grammar

exchange E. Messages to exchange E are delivered to all 5 queues.

Topic exchange

It delivers the messages to a given queue, not just based on the routing key but a pattern specified in the message called topic. It can be useful in scenario

Grammar

when consumer/application decides to which topic(s) it is interested in. Further, it can subscribe to those topics(or messages).

Header exchange

This exchange ignores the routing key but uses the header parameter to decide which messages goes to which queue. A message is matched when a value in header

Grammar

matches with the one specified in the queue binding. Exchanges have other important attributes apart from routing key and exchange type

Punctutation

for example name, durability, auto-delete, arguments etc. Durability allow

Grammar

messages to persist on disk in case of broker restarts. Auto-delete deletes the exchange, once all queue

Grammar

have finished using the exchange.

BINDINGS

Bindings are the rules that define how exchanges will route messages to the queue. To route all messages from exchange E to queue Q, Q has to be bound to Exchange E. Bindings use routing key as one of the criteria to route messages to a queue. However, routing key is optional and not always applicable.

Grammar

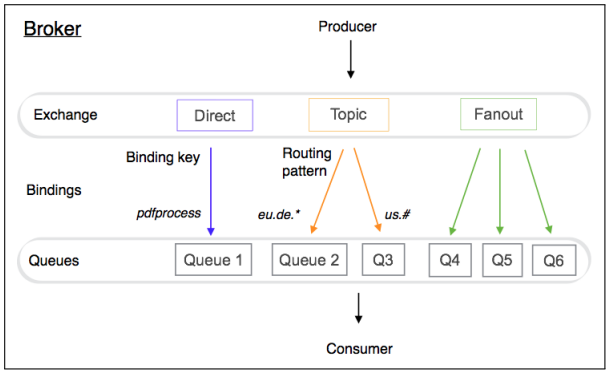

Fig. 2. Different type of exchanges supported by RabbitMQ [5]

Citation

CONSUMERS

Messages from message queue are eventually used or consumed by consumers. Consumers can use push or pull mechanism to consume these messages. Push API have messages delivered to the consumer whereas a Pull API is used to fetch messages from the queue.

MESSAGE ACKNOWLEDGEMENT

It has a built-in mechanism to send and receive acknowledgements.

It's not clear what "it" refers to here.

Producer can send messages and wait to acknowledgement in the response queue from consumer.

Grammar

Consumer can receive messages and post acknowledgement to response queue. Here request queue and request exchange

Grammar

can be used to send and receive messages. Whereas

Grammar

Don't start a sentence with "whereas."

response queue and response exchange can be used to send and receive acknowledgements. This mechanism makes RabbitMQ robust in case of failures.

CLUSTERING

[4]To

Citation

achieve high availability and making sure producers and consumers send and receive data without knowing about node failures, clustering was introduced. RabbitMQ follows OTP(open telecom platform) framework provided by erlang to achieve high availability. RabbitMQ by default doesn't replicate the content of queues i.e. all queues are stored on one node in the cluster. To achieve clustering rabbitmq

Term

keeps track of metadata for queue, exchange, binding and vhost. In case of cluster,

Not clear. What do you mean "in case of cluster?"

it only stores all information about the queue like metadata, state and contents on one node rather than all nodes in the cluster. However, it stores metadata and pointer to actual data on each node in the cluster. This is to optimize storage space and performance.

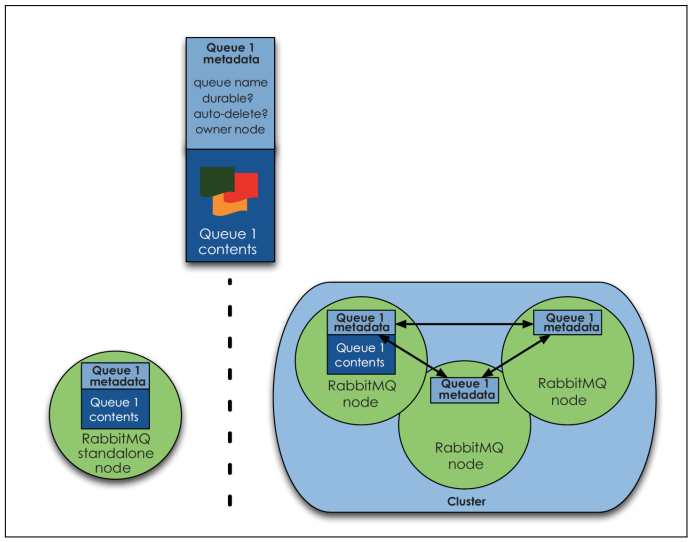

Fig. 3. Shows queue metadata for a cluster vs single node [4]

Citation

On the other hand exchanges are just bindings to the queues. So when you send a message to rabbitmq

Term

exchange, the channel checks the routing key of the message and compares it against the queue bindings. Further it sends the message to appropriate queue. Since,

Grammar

exchanges are just lookup tables for queue bindings, they are replicated across all nodes on the cluster.

It's better to explain exchanges like you have in this paragraph earlier in the paper, since they've been discussed quite a bit already.

MANAGEMENT

RebbitMQ

Term

provides all management using:

- web ui

  Needs to be capitalized: Web UI

- REST interface
- Rabbitmqctl command line utility

Web UI can be used by administrators to create user

Grammar

, monitor queues and exchanges, view statistics, add configurations etc. Similar functionality can be achieved by cli

Term

utility rabbitmqctl and REST API. Rabbitmqctl can be used to automate rabbitmq deployments and management. IT

Spelling

can also be used to write automated tests. REST Api can

Grammar

used for integrating with 3rd part

Term

UI and plugins. Using REST api

Term

you can monitor the number of connections, download or upload a configuration, list the nodes in the cluster, create or delete rabbitmq users, view or create virtual host, set permission for a user etc. You can also list all current APIs using following url: http://localhost:55672/api with some api documentation and explanations.

Don't list the URL. This is a small technical detail that is not in scope for a paper like this, and it is probably subject to configuration. What if I changed the port during configuration?

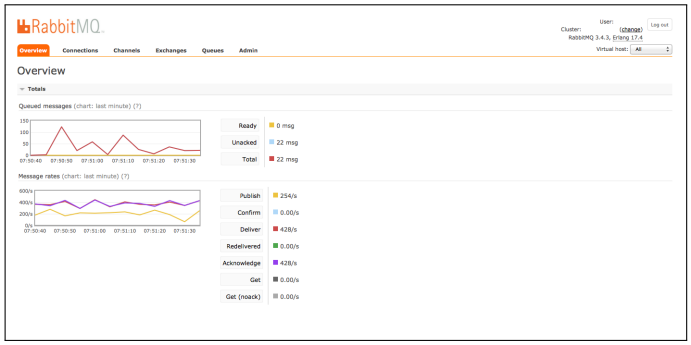

Fig. 4. RabbitMQ management UI showing messages queued and message data rates

[5]

Citation

LICENSING

RabbitMQ is licensed under Mozilla Public License(MPL) and GPL v2. [2]

You should briefly state what this means. What kind of usage does MPL allow?

USE CASES

RabbitMQ messaging can be useful in applications which require asynchronous messaging for example an application initiates a task by posting a message to RabbitMQ, it doesn't have to wait for the task to get completed. Rather it can periodically check the status of task.

Grammar

It can be useful to scale up as the data volume grows by adding additional nodes to RabbitMQ cluster. With distributed applications where applications run as micro-services in a container or virtual machine,

Grammar

RabbitMQ is useful to communicate and share data between different services. RabbitMQ can be managed separately with its management UI, CLI tool and REST api

Term

interface. This decouples the messaging layer from application

Grammar

and makes the overall design very

Avoid adverbs like "very" in a scientific paper like this.

robust.

CONCLUSION

RabbitMQ is an open source platform and provides a robust messaging platform for applications. It provides simple manageability decoupling with the application. RabbitMQ can scale well when application demand increases and can handle more data by adding more nodes to the cluster. Based on test

Grammar

[6]conducted on RabbitMQ with set of 4K(4096) and 16K(16384) messages, the performance of RabbitMQ on multi node cluster decreases in comparison with single node cluster to reach a threshold and then becomes stable. This decrease in performance can be primarily accounted due to replication between nodes in the cluster.These tests were conducted on combination of single publisher and single subscriber, multiple publishers and single subscriber, multiple publishers and multiple subscribers but there is no concrete conclusion to these test and numbers.

Why not? Where are the tests lacking, and what other tests would be helpful in better understanding RabbitMQ's performance?

ACKNOWLEDGEMENTS

Special thanks to Professor Gregor von Laszewski, Dimitar Nikolov and all associate instructors for all help and guidance related to latex and bibtex, scripts for building the project, quick and timely resolution to any technical issues faced. The paper is written during the course I524: Big Data and Open Source Software Projects, Spring 2017 at Indiana University Bloomington.

REFERENCES

[1] Pivotal, "RabbitMQ, clients and developer tools," Web Page, accessed: 2017-02-26. [Online]. Available: https://www.rabbitmq.com/devtools.html

[2] ——, "RabbitMQ, components," Web Page, accessed: 2017-02-02. [Online]. Available: https://www.rabbitmq.com/

[3] S. Vinoski, "Advanced message queuing protocol," *IEEE Internet Computing*, vol. 10, no. 6, 2006. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4012603

[4] A. Videla and J. J. Williams, *RabbitMQ in action*. Manning, 2012.

[5] Lovisa Johansson, "Rabbitmq for beginners - what is rabbitmq?" WEb Page, May 2015, accessed: 02-15-2017. [Online]. Available: https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

[6] B. Jones, S. Luxenberg, D. McGrath, P. Trampert, and J. Weldon, "Rabbitmq performance and scalability analysis," *project on CS*, vol. 4284, 2011. [Online]. Available: https://people.cs.vt.edu/butta/cs4284/spring2011/butta/RabbitMQPaper.pdf