Thesis Title

Thesis Subtitle

Amir Rabiee

Master of Science

Electrical Engineering and Computer Science
Royal Institute of Technology
1-1-2018

Abstract

Abstract goes here

Abstrakt

Abstrakt

Acknowledgements

I want to thank...

Contents

Chapter 1

Introduction

The world of applications and the data being generated and transfered to and from them are constantly evolving in a fast paced manner. The amount of data generated over the Internet and the applications running on it are estimated for 2020 to hit over 40 trillion gigabytes [1].

The applications handling this data has requirements that needs to be met such as having high reliability and high availability. Because of the high demands for such requirements a natural outcome of this is to build a distributed system that can support these applications whether they be a load balancing system or a game application or anything in between. [2, p. 1].

The evolving of distributed systems stems from the relative cheap hardware commodity that one has been able to utilize to build networks of computers communicating together for a specific purpose. These systems are in comparison to the *client-server* architecture constructed of different applications running on multiple separated machines. Because of the inherit nature of having multiple machines coordinating together for a common task, an innately advantage for them is that distributed systems are more scalable, reliable and faster when architectured correctly in comparison to a *client-server* model. The advantages with these systems comes with a cost, as designing, building and debugging distributed systems are more complicated than systems running on only one machine [2, p. 2].

In order to assess the scalable part of a distributed system, appropriate measures have to be taken, to easily meet the need when the communication between different machines and their applications are put to the

test. To help facilitate the problems that can occur when an application on one machine tries to communicate with a different application on another machine one can use **message queues** and **message brokers** [3, p. 2].

The message brokers works in symbiosis with the message queues to help deliver messages sent from different destinations and route them according to the correct route [4, p. 326].

1.1 Background

The distributed systems that are developed to meet the requirements of having high availability and reliability are built upon some abstract message form being sent from one process located on one machine to another process on a different machine. Therefore an inherit attribute of a distributed system is that it needs an architecture or system that can distribute messages in order to achieve higher scalability and reliability.

1.1.1 Message oriented middleware

The architecture used, that strives to fulfill the requirements, is called *message-oriented middleware* (MOM), this middleware is built on the basis of an asynchronous interaction model which enables users to not be blocked after sending a message instead they continue processing with their execution [5, p. 4]. More importantly a message oriented middleware is used as a basis for distributed communication between processes without having to adapt the source system to the destination system. This architecture is illustrated in Figure 1.1 where the apps in this context refers to the various systems using the MOM.

A central concept and structure when using a message oriented middleware is the usage of **message queues**, these queues are used to store messages on the MOM platform. The systems using the MOM platform will in turn use the queues to send and receive messages through them. These queues have many configurable attributes, which include the name, size, the sorting algorithm for the queue and so forth [5, p. 7].

In order to efficiently distribute the messages to different queues one has to resort to a **message broker**, these message brokers can be seen as an architectural pattern according to [6, p. 288], the broker is responsible for translating message data formats between applications which

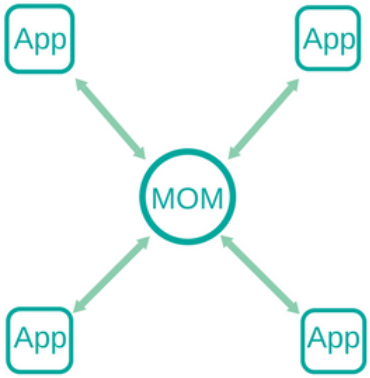

Figure 1.1: Structure of message oriented middleware.

abstracts the routing from one destination to another.

An important attribute of message brokers is their usage of message queue protocols, there are many different types of protocols that can be used such as **Streaming Text Oriented Messaging Protocol** (STOMP), **Extensible Messaging and Presence Protocol** (XMPP), **Message Queueing Telemetry Transport** (MQTT) and **OpenWire** and more [7, p.3].

The unique capabilities of the MOM-model comes from the messaging models that it uses, there are mainly two messaging models to be found, **point-to-point** and **publish/subscribe**.
The point-to-point model provides a communication link between the producer and consumer with the usage of a message queue, the consuming clients processes messages from the queue with the requirement of having only one receiver consuming the message albeit not being a strict requirement [5, p. 9], these messages are delivered **exactly once**.
In the publish/subscribe model, the producer produces a message to a specific topic, the consumers interested in these messages will subscribe to the topic, these messages are routed by a publish/subscribe engine.

An intuitive metaphor that can be used to understand the usage of the MOM-model is to see it as a post terminal where the message brokers

are the persons delivering the posts to the right post code area which in this case are the message queues.

1.2 Problem

With a plethora of different message brokers available to help with implementing a message oriented middleware, choosing one is a multifacetted question that has many different aspects that need to be taken in consideration. Every message broker has their own design and implementation goals and can therefore be used for different purposes and situations. An overview of some message brokers and the protocols supported can be seen in Table 1.1.

Table 1.1: Message brokers and their supported protocols

Message broker	**Protocols supported**
Apache Kafka	Uses own protocol over TCP
RabbitMQ	AMQP, STOMP, MQTT,
ActiveMQ	AMQP,XMPP, MQTT, OpenWire

1.2.1 84codes, CloudKarafka, CloudAMQP

The message brokers found in Table 1.1 are widely used and can be deployed on different server architectures and platforms [8], the company **84codes** offers two different services called **CloudKarafka** and **CloudAMQP** [9][10]. CloudAMQP is a RabbitMQ-focused implementation and CloudKarafka is a Apache Kafka solution, these two services are hosted on different cloud platforms such as Amazon Web Services, Google Cloud, Microsoft Azure etc.

These two platforms as a service needs to be tested on their enqueueing performance in order to get a deeper understanding of the underlying mechanisms behind the enqueueing decisions when sending a message from a producer to a consumer. With this in mind the main problem adressed by this thesis work is to perform testing of both Apache Kafka and RabbitMQ as message brokers on Amazon Web Services to be able to answer the question and quantify which situations to use RabbitMQ and Kafka on a cloud platform such as AWS?

1.3 Purpose

Because of the intricacy of the different message brokers and their corresponding protocols they use it is a relative difficulty in grasping both the fine grained differences between them as well as the coarse grained. This thesis discusses and shows an overview of the available messaging middleware solutions and aims to validate and verify the enqueueing performance for two message brokers.

Furthermore the two different message brokers RabbitMQ and Apache Kafka is a debatable topic on deciding which one to use [11], this thesis work will try to shed a light on this subject, as well as focus on testing the two platform-as-a-service (PaaS) CloudKarafka and CloudAMQP on Amazon Web Services, primarily the enqueuing performance of messages.

The thesis work will present the designed testing experiments and the results of them, in order to visualize the performance of the two message brokers running on the cloud platform Amazon Web Service.

1.4 Goal

The goals of this project is presented below:

- Design experiments for testing the enqueueing performance for Apache Kafka and RabbitMQ.
- Compare the results from each experiment and discuss the findings with regards to the the cloud platform and the respective message broker.
- Evaluate with a 95th percentile for sending and processing messages.
- Evaluate the results with a statistical linear model.
- Use the project as reference material when analyzing Apache Kafka and RabbitMQ for their enqueuing performance.

These goals presented lays the foundation for this thesis work and the results derived from the goals can be further used as a reference point for when to use RabbitMQ over Kafka and vice versa.

1.4.1 Benefits, Ethics and Sustainability

With the amount of data being generated in the present day which can be found in many enterprises one has to think of the ethical issues and aspects that can arise with processing and storing this much information and what the possibilities are with extracting valuable data from this content.

This thesis work is not focused on the data itself that is being stored, sent and processed, but rather the infrastructures which utilizes the communication passages of messages being sent from one producer to a consumer. Nonetheless the above mentioned aspects are important to discuss, because from these massive data-sets being generated one can mine and extract patterns and human behaviour [12], which in turn can be used to target more aggressive advertisements to different consumer groups.

Moreover another important aspect to be brought up is the privacy and security of peoples personal information being stored which can be exposed and used for malicious intent [13], this will be remedied to a degree with the introduction to the new changes in the General Data Protection Registration that comes into effect May 2018 [14] .

With the usage of large cloud platforms and their appropriate message brokers that is used to send millions of messages between one destination to another, one has to think of the incumbent storage solutions for the data, which in this case has resulted in the building of large datacenters. These datacenters consumes massive amount of electricity, up to an amount of several megawatts [15], in comparison to a toaster that uses about 1 kilowatt.

The company 84codes and their services holds a greater standpoint on the ethical aspect because both CloudKarafka and CloudAMQP is used by companies all over the world for sending data and therefore the ultimate responsibility of security lays on 84codes and their strategies and design decisions to keep the data intact and not exposed to third-parties or other non-authorized parties.

The main benefitters of this thesis work are those standing at a crossroads of choosing a message broker for their platform or parties interested in a quantitative analysis focused on the enqueuing performance of two message brokers, Apache Kafka and RabbitMQ on a cloud platform such

as Amazon Web Services.

1.5 Methodology

The focus of this thesis work is to analyze and test the enqueuing performance of two different message brokers, and with that in mind, one has to think of the different methodologies and research methods that are to avail and that are the most suitable to conduct such a thesis work. The fundamental choosings of a research method is based on either a *quantitative* or *qualitative* method, both of these have different goals and can be roughly divided into either a numerical or non-numerical project [16, p. 3].

The quantitative research method focuses on having a problem or hypothesis that can be measured with statistics and validified as well as verified, a qualitative research on the other hand focuses on opinions and behaviours to reach a conclusion and to form a hypothesis.

For this project the most suitable research method is of quantitative form because of the experiments and tests to be run gathers numerical data.

Another important aspect to be chosen for the thesis work is the *philosophical assumption* that can be made and there are several school of thoughts to be considered. Firstly the *positivism* element relies on the independency between the observer and the system to be observed as well from the tools to be measured with. Another philosophical school is the *realistic*, which collects data from observed phenomenons and thereafter develop knowledge. Thirdly a *criticalism* element is the one which focuses on learning how users can affect different types of computer systems. [16, p. 4]

There are several others philosophical principles but for this project the most fitting ones was to use a combination of both the positivism as well as the realistic.

Moreover to continue with the experimental testing of the enqueuing performance of the different message brokers a research method that fits the requirements of the thesis work had to be chosen. There are mainly two divisions of research methods, a *experimental* or a *non-experimental*

research method.

Because of the nature of the thesis residing in experimenting with the correlation of variable changes and their relationship between one another in order to see how the message brokers becomes affected an obvious choosing would be a experimental research methodology. An *analytical* research method based on previous testing of both RabbitMQ and Apache Kafka is also showcased for more a comprehensive conclusion. [16, p. 4]

1.6 Delimitations

Explain the delimitations. These are all the things that could affect the study if they were examined and included in the degree project. Use references!

1.7 Outline

In text, describe what is presented in Chapters 2 and forward. Exclude the first chapter and references as well as appendix.

- **Chapter 2** shows a more in-depth technical background of Apache Kafka and RabbitMQ and their protocols.
- **Chapter 3** presents the research methodologies of the thesis work with the appropriate statistical tools.
- **Chapter 4** will present the different experiments and test cases with the goal in mind.
- **Chapter 5** will present and visualize the results from the experiments.
- **Chapter 6** will discuss the results and the conclusions that can be drawn from the thesis work.

Chapter 2

Theoretical background

In this chapter, a detailed description about background of the degree project is presented together with related work. Discuss what is found useful and what is less useful. Use valid arguments.

Explain what and how prior work / prior research will be applied on or used in the degree project /work (described in this thesis). Explain why and what is not used in the degree project and give valid reasons for rejecting the work/research.

Use references!

Never use subtitles after each other without text in between the sections.

If figures are used, write Figure 1 in text to refer to the figure.

THE FIGURE / PICTURE

Figure 1. Text

If tables are used, refer to the table by Table 1.

Table 1. Text for the table

THE TABLE

Figures and Tables are numbered independently.

Chapter 3

Methodologies

Describe the engineering-related contents (preferably with models) and the research methodology and methods that are used in the degree project.

3.1 Engineering-related and scientific content

Applying engineering-related and scientific skills; modeling, analyzing, developing, and evaluating engineering-related and scientific content; correct choice of methods based on problem formulation; consciousness of aspects relating to society and ethics (if applicable).

As mentioned earlier, give a theoretical description of methodologies and methods and how these are applied in the degree project.

Chapter 4

Work

Describe the degree project.

Chapter 5

Result

Describe the results of the degree project.

Chapter 6

Conclusion

Describe the conclusions (reflect on the whole introduction given in Chapter 1).

Discuss the positive effects and the drawbacks.

Describe the evaluation of the results of the degree project.

Describe valid future work.

Bibliography

[1] EMC Digital Universe with Research Analysis by IDC. *The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things*. 2014. URL: https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm.

[2] Brendan Burns. *Designing Distributed System. Patterns and Paradigms for Scalable, Reliable Services*. O'Reilly, 2017.

[3] Dotan Nahum Emrah Ayanoglu Yusuf Aytaş. *Mastering RabbitMQ. Master the art of developing message-based applications with RabbitMQ*. Packt Publishing Ltd, 2015.

[4] Gregor Hohpe. *Enterprise integration patterns : designing, building and deploying messaging solutions*. eng. The Addison-Wesley signature series. Boston: Addison-Wesley, 2004. ISBN: 0-321-20068-3.

[5] "Message-Oriented Middleware". In: *Middleware for Communications*. John Wiley Sons, Ltd, 2005, pp. 1–28. ISBN: 9780470862087. DOI: 10.1002/0470862084.ch1. URL: http://dx.doi.org/10.1002/0470862084.ch1.

[6] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0321200683.

[7] Vasileios Karagiannis et al. "A Survey on Application Layer Protocols for the Internet of Things". In: *Transaction on IoT and Cloud Computing* (2015). URL: https://pdfs.semanticscholar.org/ca6c/da8049b037a4a05d27d5be979767a5b802bd.pdf.

[8] Philippe Dobbelaere and Kyumars Sheykh Esmaili. "Kafka Versus RabbitMQ: A Comparative Study of Two Industry Reference Publish/Subscribe Implementations: Industry Paper". In: *Proceedings of the 11th ACM International Conference on Distributed and*

Event-based Systems. DEBS '17. Barcelona, Spain: ACM, 2017, pp. 227–238. ISBN: 978-1-4503-5065-5. DOI: 10.1145/3093742.3093908. URL: http://doi.acm.org.focus.lib.kth.se/10.1145/3093742.3093908.

[9] *CloudAMQP - RabbitMQ as a Service*. URL: https://www.cloudamqp.com/ (visited on 03/08/2018).

[10] *CloudKarafka - Apache Kafka Message streaming as a Service*. URL: https://www.cloudkarafka.com/ (visited on 03/08/2018).

[11] Pieter Humphrey. *Understanding When to use RabbitMQ or Apache Kafka*. 2017. URL: https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka (visited on 03/09/2018).

[12] Farshad Kooti et al. "Portrait of an Online Shopper: Understanding and Predicting Consumer Behavior". In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. WSDM '16. San Francisco, California, USA: ACM, 2016, pp. 205–214. ISBN: 978-1-4503-3716-8. DOI: 10.1145/2835776.2835831. URL: http://doi.acm.org.focus.lib.kth.se/10.1145/2835776.2835831.

[13] Ryuichi Yamamoto. "Large-scale Health Information Database and Privacy Protection." In: *Japan Medical Association journal : JMAJ* 59.2-3 (Sept. 2016), pp. 91–109. ISSN: 1346-8650. URL: http://www.ncbi.nlm.nih.gov/pubmed/28299244%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5333617.

[14] *Key Changes with the General Data Protection Regulation*. URL: https://www.eugdpr.org/key-changes.html (visited on 03/19/2018).

[15] Bill Weihl et al. "Sustainable Data Centers". In: *XRDS* 17.4 (June 2011), pp. 8–12. ISSN: 1528-4972. DOI: 10.1145/1961678.1961679. URL: http://doi.acm.org.focus.lib.kth.se/10.1145/1961678.1961679.

[16] Anne Håkansson. "Portal of Research Methods and Methodologies for Research Projects and Degree Projects". In: *Computer Engineering, and Applied Computing WORLDCOMP* (2013), pp. 22–25. URL: http://www.diva-portal.org%20http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-136960.

Appendix A

Appendix Title