New name

- subtitle -

Project Report

Group: SW705E19

Aalborg University
Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg East, DK

Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East, DK
www.cs.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Title

Theme:
Theme here

Project Period:
Fall Semester 2019

Project Group:
SW705E19

Participant(s):
Andreas Stenshøj
Daniel Moesgaard Andersen
Frederik Valdemar Schrøder
Jens Petur Tróndarson
Rasmus Bundgaard Eduardsen
Mathias Møller Lybech

Supervisor(s):
Gabriela Montoya

Copies: 1

Page Numbers: 25

Date of Completion:
May 28, 2018

Abstract:

The abstract is right here

Contents

Abbreviations

Terms and abbreviations used in the report:

Term : Description

Todo list

Chapter 1

Introduction

Chapter 2

Problem analysis

2.1 Scalability

In this section we will look at what scalability in software is and how you can design a system to be scalable.

2.1.1 What is scalability

There are different types of scalability related to software design [1].

- **Scalability in performance** The system should be able to maintain a high performance even if there are many users. This can be facilitated by adding more capacity to the system's components to allow more parallel computing. However, performance does not scale linearly as there is a limit to how much speedup that can be achieved by using multiple processors according to *Amdahl's Law*.

- **Scalability in availability** According to the *CAP theorem*, consistency, availability and partition tolerance cannot all three be guaranteed in a distributed system. Usually in web applications that needs to be scalable, availability and partition tolerance are favored over consistency. The system will instead make use of *eventual consistency*, meaning that all the users will at some point be able to read the most reecent updates but not at the same time. This way of doing things will decrease synchronism in the application, but allows the system to be available to all users even during network partition, meaning that users may not read the most recently updated data, but it will still be available to them during network failure. As the network partition becomes resolved, consistency will be restored.

- **Scalability in maintenance** The software that is written must be continuously maintained and updated even after deployment of the system to make sure that the different components of the system are up to date and will continue running. Different kinds og platforms and tools can be used to facilitate

this by monitoring the application and ensuring that the system is operating properly.

- **Scalability in expenditure** This relates to managing the total cost of developing, maintaining and operating the system. It is important to not only think of the costs of developing different components versus reusing existing ones, but also look at what the costs of maintaining these components will be in the future. Another thing to consider is using industry standard technologies, as it will most likely also be it easier to hire experts on these in the future.

2.1.2 Designing a system for scalability

When designing systems to be scalable there are some principles that should be followed [1].

- Always design for having at least two of everything to avoid a single point of failure. This allows for more availability and performance, thus making the system easier to scale, but adds to the operational cost.
- Scale the system horizontally instead of vertically. Add new servers to the system instead of buying a new server with larger capacity each time to replace the old one when more capacity is needed. In theory, this should also be cheaper since the old servers will still be in use. But for a system with multiple servers to work, some sort of load balancing is required.
- Build the API first to avoid tailoring it to a specific type of frontend. The API should work as a service so that it can be used for any type of application such as a smartphone application or a web site.
- Caching the data that is being accessed can have significant performance benefits, since the system will not have to recompute the results that are being requested often.
- As mentioned in scalability in availability, the application does not need to provide the newest data available at all times if it is not necessary for the users to know. Relying on eventual consistency will lead to better availability, especially if the system has a large userbase.
- The system should be designed to make maintenance and automation easy so that after the software has been released it can be monitored and the different parts can be updated when needed.
- Write code to be able to run asynchronously rather than synchronously to allow tasks to be executed on different threads. This will increase the availability of the system.

- Avoid storing information about a component's state in the application servers. State information should only be kept in the components made for it, and in general it should be stored in as few places as possible. By doing it this way, no components and servers are dependent on each other and any server can handle any client request.

Chapter 3

Design

Chapter 4

Implementation

Chapter 5

Discussion

Chapter 6

Conclusion

Chapter 7

Appendix

Bibliography

[1] Elasticys. “Scalability Design Principles”. In: (Sept. 2015).

List of Figures

List of Tables

Listings