Final Year Project

Performance Analysis of Design Patterns in Microservice Architecture

Rajit Banerjee

Student ID: 18202817

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science with Data Science

Supervisor: Professor John Murphy

UCD School of Computer Science

University College Dublin

21st November 2021

Contents

Chapter 1: Project Specification

1.1 Problem Statement

Microservice architecture is a style of designing software systems to be highly maintainable, scalable, loosely-coupled and independently deployable. Moreover, each service is built to be self-contained and implement a single business capability. Design patterns in software engineering refer to any general, repeatable or reusable solution [1] to recurring problems faced during the software design process. The aim of this project is to analyse the performance of a number of microservice design patterns (based on metrics such as query response time, CPU/RAM usage, cost of hosting and packet loss rate), and evaluate their benefits and shortcomings depending on the business requirement and use case. A non-exhaustive list of design patterns that could be explored is as follows:

- API Gateway
- Chain of Responsibility
- Asynchronous Messaging
- Database or Shared Data
- Event Sourcing
- Command Query Responsibility Segregation (CQRS)
- Saga
- Circuit Breaker
- Strangler (Decomposition)
- Consumer-Driver Contract Test
- Externalise Configuration
- Aggregator
- Branch

For the aforementioned design patterns, sufficiently complex simulations will be designed for the performance engineering experiments. The project will also look at some common issues in microservices, and how they compare with traditional monolithic architectures.

1.2 Background

Microservices have gained traction in recent years with the rise of Agile software development and a DevOps [2] approach. As software engineers migrate from monoliths to microservices, it is important to make appropriate choices for system design and avoid "anti-patterns". Although no one design pattern can be called the "best", the performance of systems can be optimised by following design patterns suited to the use case, with the right configuration of hardware resources.

1.3 Related Work

Due to their popularity, microservices have been written about extensively in books like [3], [4], [5]. Articles such as [6], [7], [8], [9], [10] discuss the intricacies of microservice architecture as well as the trade-offs between various common design patterns. In [11], the performance problems inherent to microservices are explored, with evaluations performed using a custom-built prototyping suite. Akbulut and Perros [12] dive into the performance analysis aspect of microservices that is being proposed in this project, where they consider 3 different design patterns.

1.4 Datasets

Any data that is to be used or analysed in this project will be generated during the course of experiments. There are no dependencies on additional datasets.

1.5 Resources Required

A non-exhaustive list of resources is specified below, following preliminary needs assessment.

- Languages/Frameworks: Node.js + Express.js, React.js
- Tools: Git, Docker, Apache JMeter
- Database: MongoDB
- Compute: Linux server maintained by the UCD School of Computer Science

Chapter 2: Related Work and Ideas

The aim of this chapter is to provide the readers with a holistic view of microservices and performance evaluation, especially some of the important terminology and latest developments in the field. The discussion will consider several published works which will illustrate the how the topic has been previously explored, and why performance engineering is useful at all, especially for distributed systems such as microservices.

2.1 Microservices and the transition from monoliths

2.2 Software design patterns

In software engineering and related fields, **design patterns** are generally defined as reusable solutions to commonly occurring problems in software design. Although design patterns cannot be directly converted to code (like an algorithm described in pseudo-code), they provide a blueprint on how a problem can be approached in various situations. Unlike *algorithms*, design patterns are not meant to define any clear set of instructions to reach a target, but instead provide a high level description of an approach. The characteristic features and final result are laid out, however the actual implementation of the pattern is left up to the requirements of the business problem and use case. Every "useful" design pattern should describe the following aspects: the intent and motivation, the proposed solution, the appropriate scenarios where the solution is applicable, known consequences and possible unknowns, as well as some examples and implementation suggestions.

Over half a decade of software engineering experience has taught developers that it is indeed rare to come across a hurdle that hasn't been crossed before in some shape or form. Most obstacles and day-to-day decisions would have been tackled previously by another developer, thanks to which the idea of *best practices* has been formed over the years. Such solutions are accepted as superior, as they save time, are adequately efficient, and don't have many unknown side effects.

The most widely known literature on the topic is the 1994 textbook [13] by the Gamma, Helm, Johnson and Vlissides (Gang of Four), which is considered as the milestone work that initiated the concept of software design patterns. The authors, inspired by Christopher Alexander's definition of patterns in urban design [14], describe 23 classic patterns that fall under 3 main categories: *creational, structural*, and *behavioral* patterns.

In recent times, design patterns have had a tendency of coming across as somewhat controversial, primarily due to a lack of understanding about their purpose. In this regard, it is important for developers to note that in the end, design patterns are merely guidelines and not hard-and-fast rules that must not be broken. The one-size-fits-all model does not apply to a field as vast as software design, and design patterns should be treated as what they are: incredibly useful *tools* that when applied in an appropriate manner can speed up the development process manyfold.

2.3 Performance evaluation of distributed systems

- https://en.wikipedia.org/wiki/Performance_engineering - Describe performance engineering in few paragraphs - definitions, goals, etc. - Mention Chaos Engineering/Chaos Monkey at Netflix

2.4 Issues and challenges with microservices

2.5 Common design patterns in microservice architecture

2.6 Performance evaluation of microservice design patterns

2.7 Summary

Bibliography

[1] Wikipedia. "Software design pattern," [Online]. Available: https://en.wikipedia.org/wiki/Software_design_pattern (visited on 26th Oct. 2021).

[2] Amazon Web Services. "What is DevOps?" [Online]. Available: https://aws.amazon.com/devops/what-is-devops/ (visited on 26th Oct. 2021).

[3] C. Richardson, *Microservices Patterns: With Examples in Java*. Manning Publications, Oct. 2018, Book.

[4] M. Kleppmann, *Designing Data-Intensive Applications*. O'Reilly Media, Mar. 2017, Book.

[5] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Dec. 2014, Book.

[6] M. Kamaruzzaman. "Effective microservices: 10 best practices," [Online]. Available: https://t.co/ZM78yg19OR?amp=1 (visited on 26th Oct. 2021).

[7] M. Kamaruzzaman. "Microservice architecture and its 10 most important design patterns," [Online]. Available: https://towardsdatascience.com/microservice-architecture-and-its-10-most-important-design-patterns-824952d7fa41 (visited on 26th Oct. 2021).

[8] S. Kappagantula. "Everything you need to know about microservices design patterns," [Online]. Available: https://www.edureka.co/blog/microservices-design-patterns (visited on 26th Oct. 2021).

[9] M. Udantha. "Design patterns for microservices," [Online]. Available: https://dzone.com/articles/design-patterns-for-microservices-1 (visited on 26th Oct. 2021).

[10] J. Lewis and M. Fowler. "Microservices." (25th Mar. 2014), [Online]. Available: https://martinfowler.com/articles/microservices.html (visited on 26th Oct. 2021).

[11] K. Cully, "Performance problems inherent to microservices with independent communication and resiliency configuration," M.S. thesis, University College Dublin, Ireland, Mar. 2020.

[12] A. Akbulut and H. G. Perros, "Performance Analysis of Microservice Design Patterns," *IEEE Internet Computing*, vol. 23, no. 6, pp. 19–27, 2019. DOI: 10.1109/MIC.2019.2951094.

[13] E. Gamma, R. Helm, R. Johnson and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Addison-Wesley Professional, 1994, ISBN: 0201633612.

[14] C. Alexander, S. Ishikawa and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press, 1977, ISBN: 0195019199.