Microservices for an Enhanced User Experience

Colton Earls

Bagley College of Engineering, Mississippi State University

CSE 4233: Software Architecture and Design Paradigms

Dr. Tanmay Bhowmik

November 20, 2022

Microservices for an Enhanced User Experience

With the rise of social media platforms, following the popularization of smart phones, microservices have become an integral part of user interfaces and should be implemented to enhance the user experience. According to Diguer (2020), “Microservices promise quicker and easier software changes compared to traditional monolithic architectures by modularizing complex applications. Developers then compose applications from the resulting interchangeable, upgradable, and scalable parts”. A microservice design strategy will allow for a more interactive user experience and would enable more complex features within our applications. In addition to more complex applications, microservices enable greater scalability, downtime, code maintenance, and more autonomy in the deployment of software. Breaking away from past monolithic design architectures, through the implementation of a microservice paradigm, will allow our company to develop more modern systems to compete in the age of dynamic web applications.

Early Monolithic Implementations

Microservices as we know them today were not possible in the early days of computing. Standalone applications using a monolithic architecture were, and occasionally still are, often designed to complete a process in a single unified platform. Monolithic architecture was very prevalent in the early computers and remained popular into the early 2000s. Monolithic web applications first became popularized with the dawn of laptops in the 1990s and became more sophisticated and interactive when Netscape Communications released JavaScript in 1995. Chat rooms, message boards, and informative web applications were commonplace in this time under traditional monolithic architectures. These systems were usually self-contained with many of the

elements and data stores being tightly interconnected and co-dependent. Often systems such as this had very basic User Interfaces (UI) and were not very dynamic.

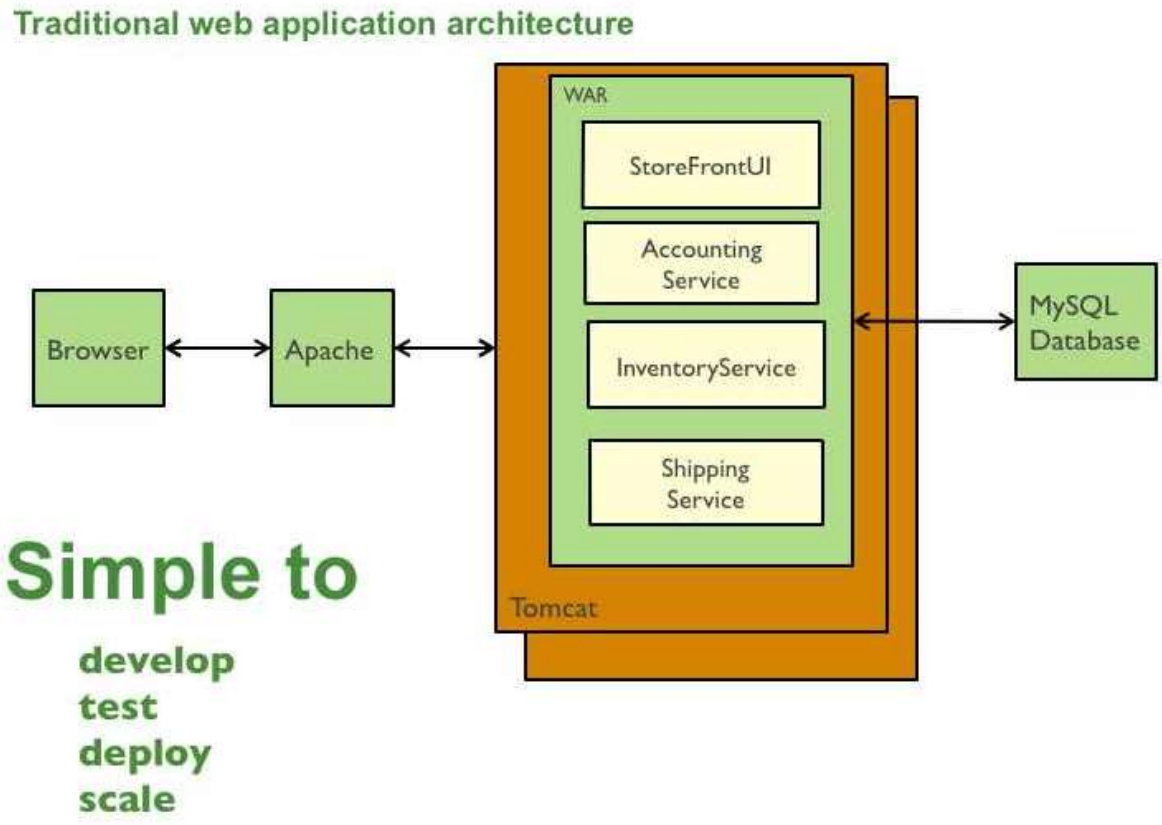
With the popularization of HTML 5 and client-side JavaScript, architecture began to evolve and become more dynamic. As technology continued to evolve in the early 2000s, business needs demanded more interactive software architecture to improve user experiences as well as revenue. Primary stakeholders, shareholders, and cooperate leadership soon began seeking paradigms which could run parallel to the increasing popularity of social media sites. This drive continues into the modern day as microservices are more prevalent than ever. According to one 2020 study, out of 1502 respondents, 54% of respondents described the adoption of microservices within their organization as being “mostly successful”. Another 92% of respondents reportedly had “at least some success”. Furthermore, a sizeable 61% of respondents noted that their organizations “have been using microservices for a year or more” (O’Reilly, 2020).

Microservice Implementation and Functionality

Traditionally web applications were designed as standalone monolithic systems. This means that all runtime components, elements, and connections are integrated into a single system. A simple example of such a system can be seen below in Figure 1. Older monolithic architectures such as this are not easily scaled or modified, and maintenance can be difficult without all documentation created at the construction of the system. The potential increased maintenance, compounded by decreased reliability and scalability, can increase the costs associated with implementing such an architecture.

Figure 1

Traditional Web Application Architecture

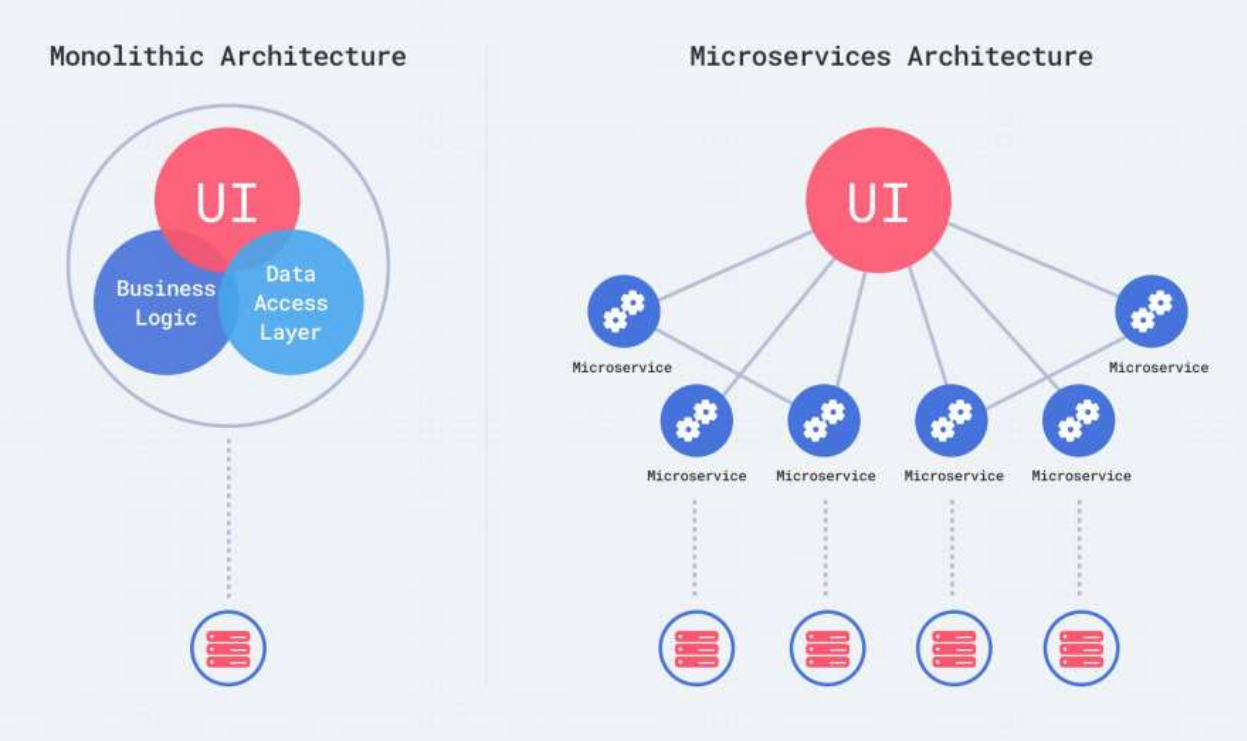


According to Nemer (2019), "Microservices are a way of breaking large software projects into loosely coupled modules, which communicate with each other through simple Application Programming Interfaces (APIs)". Microservices are independent blocks of functionality within a system that may do different thing and may not be dependent on each other to function normally. Web applications can be composed of many scalable and interchangeable applications. A web application may have one microservice that fetches the weather and displays this to the user's UI. Another microservice may fetch recent message notifications to display alongside the weather microservice. These microservices are self-contained blocks of

functionality that compartmentalize functionality of a site to the benefit of both user and investor stakeholders. Figure 2 illustrates the concept of compartmentalization of services within a software system as compared to traditional monolithic architecture.

Figure 2

Microservices vs. Monolithic Architecture (Barashkov, 2018)



The popular web application Instagram may be used as an example. Many different elements of the site are microservices integrated into the different sections of the site. If the Instagram messaging service is stopped because of a critical error or bug, the site and many of its other services will remain functioning. Maintenance of the messaging service would not require a large team and would be reasonably straightforward. Three distinct Instagram pages can be seen below. The first image shows the home page with a few different independent services. A user can send/receive messages, view video reels from followed accounts, scroll through posts

from followed accounts, and interact with each post seen within. None of the Instagram UI subunits depend on one another and may be updated, maintained, or removed entirely with ease.

Figure 3

Redesigned Instagram Pages (Carman, 2020)

Microservice Effectiveness Evaluated and Compared

Microservices are superior to older monolithic architectures for applications in the modern day in a few key areas. Applications under microservice architecture are much more reliable than monolithic architecture as their elements are isolated and not as co-dependent. In monolithic applications a detailed understanding of the software may be required for maintenance and code reuse is not common. Co-dependency of the many elements of a monolithic system would cause

vastly different behavior as opposed to microservice architecture upon encountering an error. For example: if a key element in a monolithic system encounters a fault it may affect other services. This is a significant con of the monolithic paradigm as elements are not isolated and contained within its own operating space. Microservices, as discussed previously, do not suffer this drawback. Additionally, a microservice may break but will not have far reaching consequences which increases reliability. A web application built using microservices offer vastly increased maintenance, scalability, upgradability, and interchangeability. Investor stakeholders desire growth at minimal maintenance and maintenance costs remain important factor in software development and for-profit organizations. One study found that on average the maintenance of an application can make up as much as 90% of the total budget of a system (Kushnir, 2021).

The popular microservice architectural style is not without its drawbacks. The Application Programming Interface (API) of the host platform must be consistent and thorough in order for the services within to remain stable. If the API changes it can break elements within one of the isolated microservices. The development of a reliable API can have added costs for the creation of microservice architecture. The added development time and costs reveals a disconnect between the desires of users and organizational leadership. Users typically prefer the modular and dynamic UIs offered by microservice applications, but organizational leadership may be dissuaded by the overhead of designing and maintaining such a system. It should be noted that Monolithic architecture doesn't suffer the longer, more complex development cycles as microservice architecture, making it a more attractive choice for smaller organizations who may not have the resources to host a DevOps team to maintain the many services. Additionally, Monolithic applications may be produced on existing frameworks such as Laravel or Ruby on Rails which makes this architectural style even more attractive. These are just a few of the

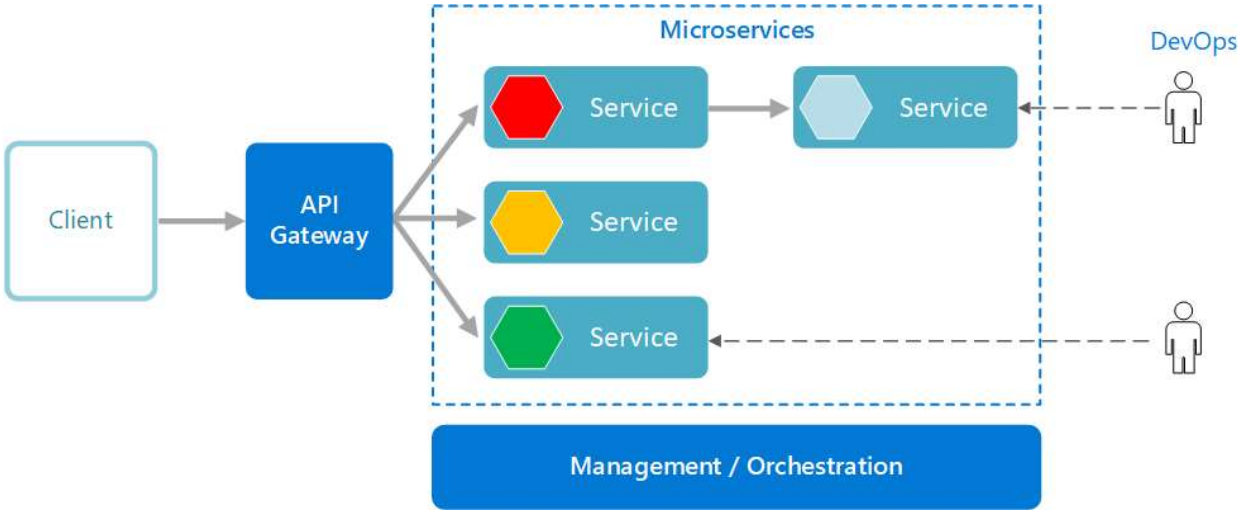
tradeoffs that must be considered and the pros and cons of each must be considered according to the needs of an organization.

Implementation of a Microservice Architectural System

Implementation of a microservice platform will not only enhance user experiences but will also enable a competitive foothold in the dynamic and ever-changing internet of things. Solid hosting infrastructure, security, maintenance, and expert software developers are required to build the base API before microservices can ever be implemented. In addition to a more costly development cycle, deployment may be more challenging as coordination between the microservices must be carefully done. While not a problem for larger organizations, smaller companies find themselves at risk of running over budget when developing a microservice architecture. Careful coordination and planning are required to build a successful and competitive system. In Figure 4 a top-down diagram of the API/Microservice architecture can be seen. It outlines the interactions between clients, DevOps, management, and API can be seen.

Figure 4

Top-Down Microservice Architecture (Microsoft, 2022)



To create the platform for hosting the many different services, a sophisticated and all-encompassing architecture must first be laid out. Boundaries between the concerns of the different services must first be established to determine the developmental distribution. An initial team of developers must be hired to create the underlying monolithic API. This may include (but is not limited to) installing servers, configuring databases, and drafting the primary architectural plans. Once the API reaches a level of sophistication that encompasses all the microservices' concerns, DevOps teams can be assigned to each individual service. Each team will utilize the API to build services to craft the desired functionality. Upon completion the new independent microservice may be hosted on the Monolithic platform application. In the case of a fault each service must be equipped to handle the fault in such a way that the integrity of the monolith and other services remain unaffected. This compartmentalizes the many different services and allows the application to be very reliable. This reliability is one of the greatest advantages of the microservice paradigm.

Past and Future Use of the Microservice Architecture Paradigm

In the modern day the microservice architectural style remains a favored construction style for many applications. It remains suitable for larger organizations with the resources to develop, update, and maintain highly versatile applications. Web applications using this architecture often exhibit greater reliability, performance, scalability, and scope when compared to other architectural styles such as monolithic. Our company should fund the research and development of a firm monolithic foundation to provide the best possible user experience with the Microservice Architectural Style.

References

Loukides, M. Swoyer, S. (2020, July 15). Microservices Adoption in 2020. https://www.oreilly.com/radar/microservices-adoption-in-2020/

Nemer, J. (2019, November 13). Advantages and Disadvantages of Microservices Architecture. https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/

Diguer, S. (2020, June 1). Microservices Advantages and Disadvantages: Everything You Need to Know. https://solace.com/blog/microservices-advantages-and-disadvantages/

Barashkov, A. (2018, December 4). Microservices vs. Monolithic Architecture. https://dev.to/alex_barashkov/microservices-vs-monolith-architecture-4l1m

Carman, A. (2020, November 12). Instagram redesigns its home screen for the first time in years, adding Reels and Shop tabs / New priorities for the company. https://www.theverge.com/2020/11/12/21561099/instagram-reels-shopping-home-screen-tab-update

Kushnir, A. (2021, October 11). Software Maintenance Cost: What Is It and Why Is It So Important? https://bambooagile.eu/insights/software-maintenance-costs/#:~:text=Maintenance%20costs%20can%20make%20up,the%20groups%20of%20services%20involved.

Microsoft Corporation. (2022). Microservice architecture style. https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices