# **CIS 5100 Term Research Project**

**Microservices**

**Aakkriti Rupal CIN: 401989650**

**Mounica Durga CIN: 401991171**

**Abstract**

In recent years, Micro web services or Microservices have gained enormous popularity in the commercial application development domain. Debuted at a conference of software architectures in 2011, microservices is defined as an architectural approach that builds a type of application with small independent components linked together in a centralized management system. This kind of approach is utilized to improve the software scalability and flexibility in the development and deployment of service-based applications.

We chose this topic because of its valuable features and benefits in application development. It has been handling their large operations. In the current scenario, as the applications are becoming increasingly complex, architecting the entire application from start to end is becoming impossible and error prone. This indicates that microservices are the future for application development as they provide the flexibility of developing individual interconnected components and adding more features per the business needs.

**Monolithic Architecture**

**Definition**: Monolithic architecture is building one large single unit of codes with multiple modules. These modules are tightly interconnected to each other.

**Overview:** The traditional approach to build an application is to design it in layers. An application comprises of three layers, presentation layer, service layer and persistence layer. In monolithic architecture, as the name suggests, all the three layers are unified under one (mono) codebase as a single file for multiple functionalities. Further, the inner structure of an enterprise application is designed in three parts:

* **A database/ presentation layer:** Database is a structured collection of information or data files for better management and use.
* **A client-side user interface/ service layer:** Consists of HTML pages and/ or Java Script running in a browser
* **A server-side application/ persistence layer:** The server-side application handles HTTP requests, executes domain-specific logic, retrieves and updates data from the database, and populates the HTML views to be sent to the browser.

The drawback with the monolithic architecture is that these three layers are non-interactive, and their components do not communicate via API. These exposures an inflexibility of this architecture with respect to upgrades or modifications to an application. Since the components are indivisible, any change will lead to designing from scratch. *Dhanik, M. (2021, March 19)*

**Microservices Architecture**

**Definition:** Microservices architecture structures an application as a collection of services. In this architectural style, an application is designed such that its multiple components are single purpose and provides self-sufficient services that can be deployed independently.

**Overview:** The 3- tier application discussed in monolithic application is considered outdated. The primary reason is that the applications become large and extremely complex to make frequent changes. Secondly, it requires maintenance at all the three layers of hardware and software that might be inefficient for the businesses. Unlike monolithic framework, the independent modules/components are not tightly coupled which reduces the internal module dependency. Therefore, an upgrade or modification in a specific functionality is easier as it does not affect the entire application. This is considered as one of the major reasons for considering microservices. *Gnatyk, R. (2018, October 3)*

Please find the differences between monolithic and microservices architecture in a tabular form below *Kharenko, A. (2018, April 10):*

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **Category** | **Monolithic** | **Microservices** |
| 1 | Architecture Components | Monolithic applications have code built as one large code base consisting of 3-tier, 4 tier or 5-tier architecture with each tier responsible for implementing a specific section of the application (database, service, presentation etc.) | Microservices applications consist of multiple but independent components each designed to implement a specific functionality of the application (login, authentication, API etc.) |
| 2 | Scalability | Development and maintenance of monolithic architecture applications become exponentially complex as it is scaled to include more features/support more users | Development and maintenance of microservices architecture applications become linearly complex as it is scaled to include more features/support more users |
| 3 | Data Storage | All tiers of the applications share a common data storage repository (Database/persistent layer) | Each module of the microservices applications works independently and has their own database/persistent layer |
| 4 | Application Deployment and DevOps | Monolithic applications require building/compiling of all tiers of applications by Integrated Development Environment (IDE) and hence leads to an increase in deployment time and DevOps effort | Microservice application modules are built independent of each other and often in parallel. This leads to shortening the deployment time and less DevOps effort |
| 5 | Application Component Coupling | Each tier in the monolithic application is dependent on other tiers for its functioning and is therefore tightly coupled. This leads to an increased effort in changing any tier of the application when adding more features and changing language, framework, or technology | A module in a microservices applications can be changed without affecting other modules if the integration parameters (API input and output) are kept the same |

**Risks and challenges during transition from monolithic to microservices**

* **Monitoring for errors:** A microservices architecture will have more granular components per the business needs that will constantly communicate. Building a distributed system is complex and when more services are interacting, more failure points are created.
* **Error or failure tracing:** With more distributed logic and data generating several logs, the effort to trace for error or to find the root cause to debug the error in case of a failure will be tedious.
* **Dealing with operational complexity:** Deploying small units is good from a development point of view but is very demanding to manage and monitor the small services. Since microservices have independent components that are not centralized unlike monolithic architecture, the operations overhead increase significantly.
* **Testing challenges:** Like operations, the testing part will also be more demanding in microservices architecture due to multiple services, complex integration, and interdependency. The testing team will have to prepare mock services for each unit. The team also must be well versed and have complete knowledge about the different services to conduct efficient testing.
* **Team Communications:** With the implementation of microservices architecture, the interconnected services, effective communication to the development team for any change in API logic is required. This is because it is difficult to determine all the modules that will be affected due to a change in the API exposed by the microservice. Each module owner needs to determine the changes required in the API calls from their modules due to this change/upgrade*. Colocation Data Centers, Cloud and Network. (n.d.). Retrieved December 20, 2021, Kurmi, A. (2020, January 4).*

**Features of Microservices:**

Microservices applications are built on the principle of reducing dependencies across different modules of the applications. This enables using an agile methodology to add, change or remove features from the applications. The key features of a microservices applications are as below:

* **Modular, loosely coupled components** – All modules/components of the microservices applications work independently of each other. Each module contains its own business logic, storage layer, and optional presentation layer. Each module exposes multiple APIs using which other modules can interact with each other. Hence, a change in the business logic, underlying storage/DB schema, or feature addition in one module does not affect other modules in the application If the API is used for integration remains the same. This enables loose coupling between different application modules/components.
* **Application deployment independence** – Since each module in a microservices application is independent of each other, they need not be deployed on the same server/data center. Based on the throughput (# of concurrent users’ module is expected to support), it can be deployed across different data centers or in cloud. Microservice modules catered to service users from a certain geographic region can be deployed in data centers that are present in the same geographic location to reduce network latency. Also, microservice modules that have increased user activity during a certain part of the day (e.g., social media users being more active during the day), month (e.g., financial systems busier during end of the month/quarter) can be deployed in cloud to increase server capacity and throughput as needed.
* **Application upgrade** – Upgrades to microservices applications are limited to the modules that need to be updated. The addition of new features will require the addition of new modules and will not affect existing modules. Similarly, modifications of existing features will be limited to changing the modules that support the feature to be modified. Removal of application features is as simple as deprecating the application modules supporting that application feature. This reduces the complexity involved in adding, updating, and removing application features. It also results in faster testing cycles and reduced DevOps effort for deployment/application upgrades.
* **Integration with other microservices modules** – Integration of one microservice with other microservices occurs using APIs. Each microservice will expose a set of APIs with defined input and output parameters. Any module that requires integration with other modules will utilize the APIs exposed and communicate using the defined input and output parameters. This provides flexibility to variable types/objects that can be used to interact with different modules of microservices applications.

**Examples of different business Migration from monolithic to microservices**

Microservices is a style of structuring an application as loosely coupled services.Each service is a component of the system which can be developed, maintained, scaled individually. Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other services. This helps a company in resilience, high scalability, faster time to market and smooth maintenance

As shown in the below image, each service can run its own unique process and communicate autonomously without having to rely on the other microservices or the application. Well-designed and implemented microservices help address the issues such as low delivery process, system scalability, defects detection, verification difficulties by splitting up the large app into the tiniest components which are divided by specific domains and communicate with each other via APIs. Examples of the leading tech companies that use microservices are Netflix, Amazon, Uber, eBay, Sound Cloud, Coca-Cola, Etsy, Spotify, Twitter, Target. We will discuss dew in detail

**Netflix**

Netflix is one of the first companies to have successfully migrated from a traditional [monolithic to cloud-based microservices architecture](https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/). Netflix’s subscribers spend over 165 million hours of watching over 4,000 films and 47,000 episodes daily accounting for over 15% of the world’s Internet bandwidth capacity. Netflix has been among the best online subscription-based video streaming services in the world for many years, so it really is not surprising that other companies followed Netflix’s lead. *The story of Netflix and microservices*. *GeeksforGeeks. (2020, May 17)*

**Problem Statement:** There was a service outage in its own data centers shutting the entire DVD renting services down for three days in August 2008.

**Solution/ Steps taken:** After the outage for 3 days, Netflix realized that it needs a more reliable infrastructure with no single point of failure. Therefore, it has made two important decisions: 1) Migrating the IT infrastructure from its data centers to a public cloud and 2) Replacing monolithic programs with small manageable software components by microservices architecture. Today Netflix has over 1000 microservices, each managing a separate part of the site.

It took more than two years for Netflix to achieve complete migration to the cloud. Not only did Netflix perfect the use of microservices but it also managed to open source many of the tools which were used to build it. The Netflix OSS (Open-Source Software Center) has a lot of tools and technologies which can be used by other companies to create microservice architecture on the cloud. *Nguyen, C. D. (2020, May 5).*

**Amazon**

Amazon: Starting as a small bookstore, Amazon has grown into the world’s largest e-commerce platform, selling books, magazines, music, DVDs, videos, electronics, computers, software, apparel and accessories, shoes, jewelry, tools and hardware, houseware, and more. Amazon initially was built as a two-tier monolithic app. Over time, as they started to grow, Amazon faced a pressing problem with their system’s scalability.

Problem statement: Amazon initially was built as a two-tier monolithic app. Over time, as they started to grow, Amazon faced a pressing problem with their system’s scalability. All the bottlenecks that come with a monolithic architecture started to appear, development and deployments were taking too long, vast databases were hard to manage, adding new features to the system became tough and risky, dealing with fluctuating website traffic became problematic. All this caused frequent outages and financial losses for Amazon. *Satnic, C. (2018, March 10)*

Solution: After frequent outages and financial losses, Amazon’s architects understood that the current path was limiting the company’s growth and that it was time to move toward a more flexible architecture. The idea was that every feature of the platform would be provided by a microservice and that all those services would interact with each other via APIs (application programming interfaces). Using the cloud for microservices helped Amazon automate operational processes and allowed them to scale services depending on traffic and current business needs. This also resulted in the adoption of a continuous delivery approach, which allowed Amazon to develop faster and deploy more flexibly. *HackFwd. (2021, December 17)*

**Uber**

Uber: Uber Technologies, Inc., commonly known as Uber, is an American mobility as a service provider based in San Francisco, with operations in over 900 metropolitan areas worldwide

Problem Statement: Uber, the most popular taxi-hailing app, started with a monolithic package to serve commuters in the city. Soon the service spread to other cities and countries. Problems began showing up when Uber wanted to launch new features, fix bugs, or upgrade their services to global levels. As new features were added, even bringing about a minor change required that the full application be redeployed.

Solution: A REST API connected the Uber drivers and passengers. Three adapters with embedded API served functions such as billing, payments, and chats. All features including a MySQL database were contained within the monolithic structure. They soon built microservices for the different functionalities such as management of trips and passengers. The services communicated with the outside world via API gateways. With this, each development has clear ownership which improved the speed and quality. When the individual teams concentrated on the services that required scaling, all the work related to scaling took place fast. There was no downtime when specific services were up for maintenance. The system fault tolerance became more reliable. *5 microservices examples: Amazon, Netflix, Uber, Spotify, and etsy: Sayone*

Microservices and the advantages that they can bring are very evident from the success stories of these large corporations. There are also many other companies like Spotify, Esty, Walmart, Target, Twitter achieved success after the migration from monolithic to microservices. Though the challenges were overwhelming, the results have brought important benefits to these businesses.

**Advantages of Microservices**

* **Improved Scalability:** The capacity of each microservice to run autonomously makes it easy to add, remove, update, and scale individual microservices. This can be done without disrupting the other microservices that comprise the application. As demand for an app increase, it is easier to scale using microservices. This also means scaling is faster and often more cost-efficient as well.
* **Microservices reduce downtime through fault isolation:** If a specific microservice fails, you can isolate that failure to that single service and prevent cascading failures that would cause the app to crash. This fault isolation means that your critical application can stay up and running even when one of its modules fails.
* **Programming Language and Technology Agnostic:** When creating a microservices-based application, developers can connect microservices programmed in any language. They can also connect microservices running on any platform. This offers more flexibility to use the programming languages and technologies that best fit the needs of the project and the skillsets of the team.
* **Better Data Security and Compliance:** Most developers use secure APIs to connect microservices. A secure API safeguards the data it processes by making sure it is only accessible to specifically authorized applications, users, and servers.
* **Real-time processing support:** The publish-subscribe framework at the core of a microservices architecture enables data processing in real-time to deliver direct output and insights.
* **Rapid growth facilitation: Due** to their modular architecture, microservices enable a prominent level of code and data reuse, making it faster and easier to deploy additional data-driven use cases and solutions for additional business value.
* **Team optimization:** Due to their focused functionality, microservices allow you to create optimized teams by selecting members based on their specific capabilities allowing them to fully focus on the scaling and availability requirements of their assigned service.
* **Highly maintainable**: well-structured, well-commented, easy to understand, easy to change, easy and fast to build*. QAT Global. (2021, October 28)*

**Conclusion**

The study and detailed research above cover the features of Microservices architecture applications, its advantages and challenges, examples of organization that are utilizing this architecture, and comparison to monolithic architecture applications. While it comes with its own complexity of application management and planning during the initial implementation phase, it is offset by the scalability, flexibility, and reliability that it offers eventually. With the growing complexity of business applications and the need for scalability and resiliency, microservices architecture is becoming increasingly popular and widely adopted. The study and research provided us with valuable insights into this emerging and disruptive technology.

**References**

*Nguyen, C. D. (2020, May 5). A design analysis of cloud-based microservices architecture at Netflix. Medium. Retrieved December 19, 2021, from* [*https://medium.com/swlh/a-design-analysis-of-cloud-based-microservices-architecture-at-netflix-98836b2da45f*](https://medium.com/swlh/a-design-analysis-of-cloud-based-microservices-architecture-at-netflix-98836b2da45f)

*The story of netflix and microservices. GeeksforGeeks. (2020, May 17). Retrieved December 19, 2021, from* [*https://www.geeksforgeeks.org/the-story-of-netflix-and-microservices/*](https://www.geeksforgeeks.org/the-story-of-netflix-and-microservices/)

*HackFwd. (2021, December 17). Werner Vogels - Amazon and the Lean Cloud. Vimeo. Retrieved December 19, 2021, from* [*https://vimeo.com/29719577*](https://vimeo.com/29719577)

*Satnic, C. (2018, March 10). Amazon, microservices and the birth of AWS cloud computing. LinkedIn. Retrieved December 19, 2021, from* [*https://www.linkedin.com/pulse/amazon-microservices-birth-aws-cloud-computing-cristian-satnic/*](https://www.linkedin.com/pulse/amazon-microservices-birth-aws-cloud-computing-cristian-satnic/)

*5 microservices examples: Amazon, Netflix, Uber, Spotify and etsy: Sayone. Software Development Company in USA with Top Software Services. (n.d.). Retrieved December 19, 2021, from* [*https://www.sayonetech.com/blog/5-microservices-examples-amazon-netflix-uber-spotify-and-etsy/*](https://www.sayonetech.com/blog/5-microservices-examples-amazon-netflix-uber-spotify-and-etsy/)

*15 benefits of microservices you need to know about + 23 disadvantages. QAT Global. (2021, October 28). Retrieved December 19, 2021, from* [*https://www.qat.com/15-benefits-microservices/*](https://www.qat.com/15-benefits-microservices/)

*Dhanik, M. (2021, March 19). Microservices vs monolithic architecture : Is deploying faster your priority. SourceFuse. Retrieved December 20, 2021, from* [*https://www.sourcefuse.com/blog/microservices-vs-monolithic-architecture-is-deploying-faster-and-scalable-applications-your-priority/*](https://www.sourcefuse.com/blog/microservices-vs-monolithic-architecture-is-deploying-faster-and-scalable-applications-your-priority/)

*Gnatyk, R. (2018, October 3). Microservices vs Monolith: Which architecture is the best choice for your business? Microservices vs Monolith: which architecture is the best choice? Retrieved December 20, 2021, from* [*https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/*](https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/)

*Kharenko, A. (2018, April 10). Monolithic vs. Microservices Architecture. Medium. Retrieved December 20, 2021, from* [*https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59?gi=6a49999a4d2*](https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59?gi=6a49999a4d2)

*Challenges in transitioning to microservices. Enterprise IT Made Easy. Colocation Data Centers, Cloud and Network. (n.d.). Retrieved December 20, 2021, from* [*https://www.datacenters.com/news/challenges-in-transitioning-to-microservices*](https://www.datacenters.com/news/challenges-in-transitioning-to-microservices)

*Kurmi, A. (2020, January 4). What are the challenges in microservices architecture? Medium. Retrieved December 20, 2021, from* [*https://medium.com/microservices-architecture/what-are-the-challenges-in-microservices-architecture-2ee9149cfc4e*](https://medium.com/microservices-architecture/what-are-the-challenges-in-microservices-architecture-2ee9149cfc4e)