**PERFORMANCE ANALYSIS OF WEB-APPLICATIONS: MONOLITHIC VS CLOUD NATIVE**

**Introduction**

* 1. Background
  2. Problem statement
  3. Project Goals
  4. Motivation
  5. Limitations
  6. Report Structure

**Background:**

The first web server in the world was deployed at CERN in Geneva, Switzerland in 1990 and three years later, there were 500 known web servers [source]. Suffice to say they were all monolithic as the idea of SOA (Service Oriented Architecture) became popular around late 1990s according to [source] and Microservice in 2010s after the blog by Lewis and Fowler[source]. There monolithic architecture meant they were developed in a single code base with all the services tightly coupled and none of the services could deployed independently. And at the time, they were all deployed on premise. Finally, in 2023, according to [source], there were more than a billion websites, suffice to say number of web servers that host these websites are at least in millions.

This exponential explosion was facilitated by Amazon’s launch of AWS with Elastic Compute Cloud (EC2) service followed soon by Google’s Google App engine (2008) and Microsoft’s Azure (2009). The could services transformed the landscape of web-server deployment. After this point, business, individual, or organizations no longer needed to host their software on premise. More importantly, they could make whatever services they wanted to provide to end-users at very low initial investment.

From here on, the demand for better and faster service continue to rise, leading to the innovation of new type of application called cloud native. They are built using microservice architecture and take full advantage of cloud computing. In microservice architecture, applications are broken down into multiple services and are developed independently. Unlike in monolithic applications, these services can be deployed independently as well. Netflix was one of the first companies to move away from its monolithic architecture in 2009. However, at this point, the concept of microservice was not even known. In fact, it was in 2011 when the term was first used and was official announced at the 33rd Degree Conference in Krakow in 2012. And finally in 2014, the architecture was popularized by Lewis and Fowlers’s blog [source].

**Problem statement:**

[Some of the sentences in the following paragraph are one to one copy from the source. Remember to present it in a different way.]

Most of the world’s web applications are monolithic in nature. They don’t take full advantage of the cloud computing. More companies would benefit from moving toward more cloud native applications.

**Project goals:**

[Some of the sentences in the following paragraph are one to one copy from the source. Remember to present it in a different way.]

Monolithic applications were industry standard for a web application from 1960s until recently. They are still perfect for small to medium size applications that don’t have large user base. However, internet is present everywhere these days. The number of users for an application can reach from thousands to millions to hundreds of millions. On top of that, technologies and market keep on changing regularly. With these new factors, monolithic applications have started to hit bottle necks. Updating and adding new features to increasingly changing market presents a problem especially if the monolithic application is already big. It might introduce a new bug, it may take a long time to introduce a new feature, it might take a long time to test give the size of the application, on top of that a single undetected bug in production can take down the entire application. And how are you to know if you user size grows in the future. It took Netflix two years to change their monolithic applications to cloud native. Should they have stared with microservice application straight away. But what if the Netflix as an application failed if they had started as cloud native applications.

An alternative that appeases these problems is of course cloud native applications with microservice architecture. You can introduce features quickly, you decrease cognitive overhead, and multiple teams can work on the same application simultaneously give the principle of loose coupling. They can scale horizontally as well as vertically much easily monolithic applications. This sounds like exactly what we needed to solve the problem of monolithic architecture. Yet, it has its flaws. Initial design stage takes longer, is just too complicated for small to medium applications with small user base, is costly on short term.

Given these complex properties of each type of application type, to a normal software developer, it isn’t exactly obvious what is the best choice. Individuals and companies spend hours trying to understand what is best for their specific needs.

**Project Goals:**

[The following paragraph is a brain shit]

With this project, I aim to clarify what each type of application provides along with their limitations along with their performance capabilities along with technologies that best suit each of the feature in the applications. [Here insert the method that you are going to use performance test it]. I plan to develop same application using monolithic architecture and another use microservice architecture. One will be deployed in traditional style while other will using cloud native approach. These two applications will be built using exact same programming languages as to remove the idea that performance boost might have been due the underlying programming language.

With this project, I aim to clarify the performance abilities of monolithic applications and cloud-native applications. To do this I will be developing and deploying same application twice, one using monolithic architecture and other using microservice architecture. Application with monolithic architecture will be deployed on a virtual machine in the cloud as to make the application monolithic and application with microservices architecture will be deployed on containers in the cloud as to make it a cloud-native application. The application will be developed using same technologies to make the comparison as fair as possible. Finally, to clarify the performance abilities of the application, I plan to perform load test on them using Jmeter with varying number of threads.

**Motivation:**

It is already known that application with large number of users or large application generally benefit from being cloud native. This seems to be the reason why most tech giants have been transitioning toward cloud native applications.

But do all the applications need to be cloud-native because of its benefits and its ability to perform better. To me it isn’t exactly obvious given the complexity of building cloud native application. I plan to load test my web applications in various scenarios in an attempt mimic different type of request given by different web application. This should give a better picture if an application needs to be cloud native or will function totally fine being a monolithic.

answer the question if an application of different nature benefit from being cloud native or it isn’t necessary for them to cloud native and remove the initial time taken to make decision for the architecture.

**LITERATURE REVIEW:**

**2.1 The advantages of monolithic architecture**

**2.2 Challenges in large monolithic applications**

**2.3 Benefits of cloud-native**

**2.4 The shortcomings of cloud-native**

**2.5 Scalability**

**2.6 Data persistence layer**

**2.7 Security in Cloud-Native and Monolithic applications**

**2.8Monitoring Cloud-Native and Monolithic applications**

**Monolithic Applications:**

* Monolithic applications are called monolithic because they are built using monolithic architecture, which is designed to deployed as a single, indivisible unit. Given all components are tightly coupled, they all run in single process space. This has its own problems and advantages which will be discussed down below. Traditionally, all monolithic applications were deployed in premise, but nowadays, they are hosted on cloud.

**Advantages of Monolithic Applications:**

* One of the main advantages of monolithic applications is that it simplifies development, management, and deployment processes. The application runs in single process space so it is lot easier to track the errors that occur during the development and production.

**Advantages of Monolithic applications:**

* Simple to develop and deploy
  + Given that whole application is built to be run on the same process. The process of developing and deploying is lot easier. The developers have to constantly test the applications functionality and ensure that they are what is required from the application, this is lot easier to do in monolithic applications because everything to do with the application can be invoked on single run.
* Ease of debugging and Testing
  + Testing is easy in monolithic because all you have to do is think about the single process. (I ACTUALLY DON’T KNOW WHY TESTING IS EASIER)
* Consistency:
  + Less duplication of the code. Easier to enforce coding practices. [FIND AN EXAMPLE]
* Less Operational Overhead
  + Things such as single database and tight coupling means that there is no need to design how different components will interact. Of course, this might lead to applications with badly structured code base.
* Small Monolithic applications are easy to maintain.
  + Small applications will not have complex design of how the different components will interact mainly because there are very few components. Tracking how they interact is much easier this way.

**Disadvantages of monolithic applications:**

* Scalability challenges
  + One of the main reasons, monolithic applications are going out of fashion is because the can only be scaled vertically, more about it later.
* Difficulty in Adopting New Technologies
  + An application has many parts each part might function differently and there might be better technologies to each part more efficiently and effectively. However, with monolithic you can only use on technology to write it.
* Increased Risk of Deployment
  + During re-deployment, sometimes it is likely that you tampered with a function that messed up the functionality of the other.
* Longer Development Time as the application grow
  + Developer need to spend more time understanding the application as a whole as slight change in code can affect other functionality.
* Limited Flexibility
  + Change to one part of the application might make it so that another part of the application needs to be updated as well as the application is tightly coupled.
* Continuous deployment becomes more challenging
  + Once again we come to the part where a single bug or change affect the whole applications.
* More challenging to make applications reliable and fault proof
  + A singular bug can take down the application.
* Team collaboration becomes more difficult as the code base grows
  + As team grows which happens if the application grows, it is very difficult to segregate the duties as the application is tightly coupled give that they run on the same process.

**Cloud Native Applications:**

* Cloud native applications are built using microservice architecture and deployed in many processes/servers. They started to become popular after the cloud computing became popular. Nowadays, if a large application is to be built almost all large companies choose to build cloud native applications

**Advantages of cloud-native applications:**

* Highly scalable
* High resilience and fault tolerance
* Faster time to market
* Cost efficiency
* Flexibility and Portability
* Improved developer Productivity
* Optimized for cloud

**Disadvantages of cloud-native applications:**

* Complex
* Skill Gap
* Security Concerns
* Increased Operational Overhead: Initially, the transition to cloud-native can increase operational overhead. Setting up CI/CD pipelines, orchestration platforms like Kubernetes, and monitoring tools takes significant effort and time.
* Cost Management Challenges: While cloud-native applications can be cost-effective due to their efficient use of resources, managing costs in the cloud can be complicated.
* Transition and Migration Challenges: Moving existing applications to a cloud-native architecture can be challenging and resource-intensive.

**Scalability**

* **Vertical Scaling**
* **Horizontal Scaling**
* **Scaling monolithic architecture**
* **Scaling cloud-native applications**

**Data Persistence Layer**

* **SQL**
  + **Transaction model**
* **Scaling SQL**
* **NoSQL**
  + **Transaction Model**
* **Scaling NoSQL**
* **NewSQL**

**Security in Cloud-Native and Monolithic applications**

**Monitoring Cloud-Native and Monolithic applications**

**History of performance review:**

**LITERATURE REVIEW:**

**PERFORMANCE ANALYSIS:**

There have been many studies for the performance analysis of Monolithic vs Microservices architectures for various scenarios.

In 2018, using Jmeter as the benchmarking tool, Al-Debagy and Martinek conducted a performance test on an application built using monolithic and microservice architecture. There applications were developed in JHipster using Sprint Boot (java) and Angular JS frameworks while Jmeter was installed remote client. There results were as follows. During load testing there was no significant difference in performance between the different architecture. Meanwhile, during concurrency testing, monolithic architecture showed a better performance in terms of throughput by 6 % on average. There results showed an obvious winner, the monolithic architecture. However, this might have due to the fact microservices application was deployed locally where its full potential can’t be utilized.

From: <https://arxiv.org/abs/1905.07997>

In 2020, [reference] assessed the performance of two same web applications, one developed using monolithic architecture while other using microservices. Monolithic application operated on a virtual server with KVM, and microservice architecture ran in containers. The study found that microservices architecture leverages CPU and packet transmission more efficiently than monolithic architecture, meaning microservice architecture resulted in better system performance than monolithic.

FROM:

https://www.researchgate.net/publication/343805251\_From\_Monolithic\_Systems\_to\_Microservices\_A\_Comparative\_Study\_of\_Performance