Candace Ralston

6/24/2023

CS 470 Final Reflection

CS 470 Project Two
Presentation clralstc

I am glad that I was able to take this class. I learned a lot about serverless and cloud-development. I didn't even know that something like serverless existed. I am planning a business and this might be useful. For now, I am looking into jobs that I can use my degree and then start my business. I am thinking of maybe a frontend developer job would be go. I like developing the frontend of a website and show people what it is about. So, I am thinking maybe an App developer or a website designer.

Microservice is taking an application, chopping it into pieces, and running it as a collection of smaller parts. This is also designed to be scalable and fault-tolerant. To handle scaling, you can use load balancing and horizontal scaling. Load balancing is the process of distributing incoming network traffic across multiple servers. Horizontal scaling is the process of adding more instances of an application to handle the increased load. For handling any errors, you can use circuit breakers, retry patterns, and fallbacks. Circuit breakers are used to defect failures and prevent cascading failures by stopping requests to a service that is falling. Retry patterns are used to retry failed requests after a certain amount of time. Fallbacks are used to provide a default response when a service is unavailable.

For the cost of microservice you have to think of a few things. One of these is the scale of the update, to start small and then work your way up. Another one is the maintenance of the microservice. It increases the cost of the software system as the number of published API endpoints increase and is very hard to predict. Another one is distributed systems; this is where the system is using message passing to communicate. Another one is transactions where they rely on a relational database system. Still another one is time across the cluster where it distributed systems and they are different from shared memory software. Another one is foreign keys where they add some constraints to the cost of microservices development as they are not easily available. Another one is ad-hoc querying where it adds complexity to the infrastructure and creates a substantial amount of extra work as it requires writing extra code. Also, when it comes down to working with containers vs. serverless the serverless is a better cost option. Serverless is used by less then 1,000 users, especially if there are times during the week when the system is not used at all. Containers comes with baseline costs, you are paying for idle resources, especially for systems that are low to medium load.

The pros and cons to microservices are:

Pros	Cons
Easier Scaling up	Increased Complexity of Communication
Improved Fault Tolerance	Requires More Resources
Ease of Understanding of the Codebase	Global Testing and Debugging is Difficult
Scope for Experimenting	Not Practical for Small Applications
Independent Deployment	Relatively Complex Deployment

Elasticity and pay-for-service are two important factors that play a role in decision making for planned future growth in microservice. In elasticity for microservices it is important because it allows the system to handle increased workload by adding resources to the system. The pay-for-service is just as important because it allows customers to pay for only services they use.

References

Thorpe, S./Aug. 14, 2018/Scaling Microservices: The Challenges and Solutions - DZone

What are Microservices? | AWS (amazon.com)

Muzammil K/Sept. 25, 2020/The Cost of Development of Microservices Applications (aalpha.net)

Wittig, A./June 24, 2020/Containers vs. Serverless: Thoughts About Your Cloud Strategy | cloudonaut

Taylor, K./HitechNectar