**Are microservices the future?**

It is no secret that with the success Google, Microsoft, Netflix and many other companies have had with Microservices, every engineer and organization want to follow in their footstep and break down their monolithic application into smaller microservices. Just because microservices worked for Google, Microsoft and Netflix does not mean they will work for a much smaller company building much smaller software.

The question is, when should an organization seriously consider moving their technology to utilize microservices?

To answer this question, we must look into what advantages a microservice architecture and distributed systems bring to the table.

According to my research, microservices are a fantastic way to scale software to millions of users. For example, suppose the software is hit with a heavy load on the authentication service. In that case, the system will detect that and spawn a few more servers and services to accommodate this heavy demand, and once it is over, those services and servers can be killed off to save on energy, processing power, and cost. On the other hand, with monolithic applications, if the same thing occurs, the software engineers have no option but to create new servers that would host the entire software, which can be large and a waste of processing power and server space.

The other advantage of microservices is if one of the services goes down, the additional services won't be affected and will continue working normally. That is not always the case, though, because for instance, if some service relies on the downed services, then this service would not function properly without the downed service.

Another advantage is debuggability. Since each microservice functions separately, if something goes wrong in one of the services, it is easy to pin it down. However, again this is not a hard rule since this depends heavily on how the software was architected. Distributed systems are known for being extremely hard to debug because an issue can be spread over multiple computers rather than a single computer.

One disadvantage of microservices is speed. Microservices are known for being slower than monolithic applications. The reason for this is when a monolithic application requires a particular piece of information, it can fetch it and process it fairly quickly. In a microservice architecture, the application has to wait for one or more services to finish their processing before it can respond to the user request.

In conclusion, microservices have many advantages but also have their fair share of disadvantages. At the end of the day, just like any decision in computer science, a trade-off must be made. Do you, as a software engineer, care about speed and ease of debuggability or do you care more about the scale and serving as many users as possible? Problems at Microsoft and Google are not the same problems a typical organization would face. So it would be best if you did not blindly do what big tech companies do.

References:

Fridman, L. (2020) Java was created because of pointer bugs in C/C++ | James Gosling and Lex Fridman, YouTube. Available at: https://youtu.be/RrMptmNYkSw?t=636 (Accessed: December 12, 2022).

Renjith GRenjith G 4 et al. (1958) What is difference between monolithic and micro kernel?, Stack Overflow. Available at: https://stackoverflow.com/questions/4537850/what-is-difference-between-monolithic-and-micro-kernel (Accessed: December 12, 2022).