Does moving from a monolith to a microservice architecture help with resilience?

Microservice architecture is componentized by breaking down into many small and independent services, which can be deployed independently through an automated deployment mechanism. Moving from a monolith to a microservice architecture can help with resilience.

Compared to monolithic architecture, microservice architecture can divide a complex application into simple, highly decoupled services, and these services communicate with each other through lightweight communication mechanisms, such as message queues and HTTP APIs[1]. Since these services are independent, when one of the services fails, the others are not affected. In contrast, in a monolithic architecture, the failure of one service affects the entire system. Microservices are highly decoupled and built for failure, which can be capable to cope with failure and outages[1]. Many large-scale websites and applications, such as Netflix, Amazon, have all evolved from monolithic to microservices architecture. One of the representative examples is the API team of Netflix company[2]. In order to enable members to continue watching movies and TV shows when server failures occur, the API team has restructured the API to enable graceful fallback mechanisms to kick in when a service dependency fails[2]. As a result, microservice architectures can eliminate an application's sensitivity to a single point of failure by dispersing functionality across multiple services, which allows applications to perform better, reduce downtime and scale as needed[3].

In addition, using Circuit Breaker pattern in the microservice architecture can handle faults to improve resilience. Circuit Breaker can accept failures and track those failures[4]. If the service is in the failed state, the circuit will send the error message without making a call. If the service is up, then breaker forwards the call to the needed service. The most famous tool for implementing this model is Hystrix from Netflix[4]. In this way, it does not cause the client to wait for internal server errors and help Netflix company provide a better user experience.

Through moving from a monolith to a microservice architecture can increase resilience in the application. Furthermore, developers can use some cloud design patterns in the microservices, such as Circuit Breaker[4], Compensating Transaction Pattern[4], Health Endpoint Monitoring Pattern[4], to improve availability and resilience of the application.

References

[1] Abbassi, **Reliability Is Not Enough-Building Resilient Applications in Microservices** `https://blog.giantswarm.io/reliability-not-enough-resilient-applications-containerized-microservices/` *Date Accessed: 1 March 2019*

[2] Schmaus, **Making The Netflix API More Resilient** `https://medium.com/netflix-techblog/making-the-netflix-api-more-resilient-a8ec62159c2d` *Date Accessed: 1 March 2019*

[3] MuleSoft, **Microservices vs Monolithic Architecture** `https://www.mulesoft.com/resources/api/microservices-vs-monolithic` *Date Accessed: 1 March 2019*

[4] Jegasingam, **Circuit Breaker for Microservices** `https://medium.com/@jegasingamjeyanthasingam/circuit-breaker-pattern-for-microservices-eb71569dc44d` *Date Accessed: 1 March 2019*