When Are Microservice Architectures Beneficial?

William Profit and Nicolás D’Cotta – 21st February, 2022 *(last modified 24th February, 2022)*

***Abstract*—We show there are several variations of microservices architectures and (mostly drawing from articles and personal experience) highlight possible motivations to adopt some of them; while also reflecting on the price to pay for such adoptions and on some of their pitfalls. We conclude that the trade-off is in terms of availability, team structure, and cost.**

I. A Company’s Options

Microservices is a widely used umbrella term to refer to a platform architecture that is not monolithic and not serverless. Beyond this, definitions remain vague, but in general they refer to having multiple servers that communicate over the network and are not necessarily owned by the same teams [1].

When developing a new platform, a company has more options that just choosing ‘whether to adopt a microservices architecture’ – according to Sam Newman, it is not a ‘yes or no’ decision [2]. In particular, it can use **a single monolith**, a **modularised monolith** (where modules are developed separately with stable binary APIs and compiled into the same executable), a **coarse grained distribution** (services are not necessarily small, teams manage a handful services at most) or a **fine-grained distribution** (new services are spun up whenever new functionality needs to be added); with even more variations in-between (should services share libraries?).

II. The benefits of microservices

Turning to a microservice architecture can have several advantages that could benefit a company. From a management perspective, it can become difficult for large companies to coordinate many engineers working on a single large monolith. Microservices lend themselves well to giving individual teams one or few services to develop and maintain which decouples teams from each other thus reducing friction [3]. They also allow for finer-grained independent deployability as services can be rolled out to production individually, more frequently, and more easily rolled back.

From a technical perspective, microservices are beneficial when high redundancy and failure tolerance are needed as it allows microservices to fail without bringing down the entire system and new instances of the microservice can be quickly spun up. This amounts to reducing *blast radius* when something goes wrong: if the services that serve a web frontend and process payments do not depend on each other, we can keep processing payments even if we are unable to serve a specific web page [4].

III. Blockchain.com: Distributing to Scale up

Blockchain.com uses a coarse-grained architecture where some of the microservices can be very big (such as the retail API gateway) – these large microservices are called *core* services and are further split into modules. This results in a hybrid of microservices with a couple monoliths. When adding a new feature, depending on its scope and what data it needs, it is either built into a microservice or simply inside an existing core service. If, further down the line, it is decided it needs to grow further, it can get moved out of a core service into a new microservice.

This is often the case when a feature becomes large enough to deserve its own engineer owners. This happened with a ‘crypto brokerage’ service which got split off from a core service [5]. In this example, the main motivation for seeking further distribution was to achieve more fine-grained deployability (brokerage was getting releases more often than the core, and we wanted to be able to roll back one without rolling back the other) and to partition the retail team (which had grown too big) to have a new separate brokerage team.

IV. The Price of Further Distributing

A problem that arises when distributing is strong coupling between services because of library dependencies. Even when APIs are well defined (say, via gRPC), when performing a change it is crucial it does not modify that API: otherwise one must update every one of the API clients and deploy those services in sync, effectively missing out on the advantages of distributed architectures (resulting in a ‘distributed monolith’). The only solution to this is problem is well-defined boundaries within services, which are hard to determine before building a product and very expensive to redraw once the services are in production.

The costs of maintaining consistency are also exacerbated with regards to the data management as it is now split between different microservices. Operations that in a monolith might have been simple, such as joining two tables, can now be much more costly to run as services must reconcile data from different sources.

Moving towards a microservices architecture is not simple and has many pitfalls. If services are too coupled, it can tend towards a distributed monolith architecture where one service going down triggers a domino effect which brings down the entire system, much like a monolith.

V. Conclusion

Microservices are not to be viewed as a single option but rather as a wide spectrum between the monolith and the fully microserviced architecture. They can answer needs for availability, scalability, and team partitioning; but at the cost of consistency and simplicity – companies should ask themselves whether it’s needed at all and if so how much they are willing to pay.

References

[1] Amazon AWS. "What are microservices?" (Feb. 2022), [Online]. Available: https://aws.amazon.com/microservices/ (visited on 02/10/2022).

[2] S. Newman and M. Fowler, *When to use microservices (and when not to!)* Remark by Sam Newman (minute 6:45), Sep. 2020. [Online]. Available: https://www.youtube.com/watch?v=GBTdnfD6s5Q (visited on 02/10/2022).

[3] Amazon AWS. "Service team pattern." (2022), [Online]. Available: https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-decomposing-monoliths/service-per-team.html (visited on 02/14/2022).

[4] Jeremy H, Martin Fowler, Sam Newman. "When to use microservices: Sam newman and martin fowler share their knowledge, Help isolate the "blast radius" of service failures." (2021), [Online]. Available: https://blog.dreamfactory.com/when-to-use-microservices-sam-newman-and-martin-fowler-share-their-knowledge/ (visited on 02/14/2022).

[5] Anonymous for Review, *Brokerage service separated from main retail core service*, Personal Communication, 2021.