MIGRATION: MONOLITHIC TO MICROSERVICES - BACKGROUND

Monolithic architecture as we can imagine, a monolith which is build in a single block structure. As in software engineering term, monolithic architecture is where a system is build in a single block structure as depicted in ***Figure 1.0***. Everything from user interface, APIs and database of multiple services are bounded together. In a good side, development phase may be faster. However, it will bring a lot of complication as the system started to grow bigger along with huge amount of concurrent and active users. Maintaining a large scale monolithic system can be very costly due to the fact that services are bound together and will require a lot of testing against services that shared same data.

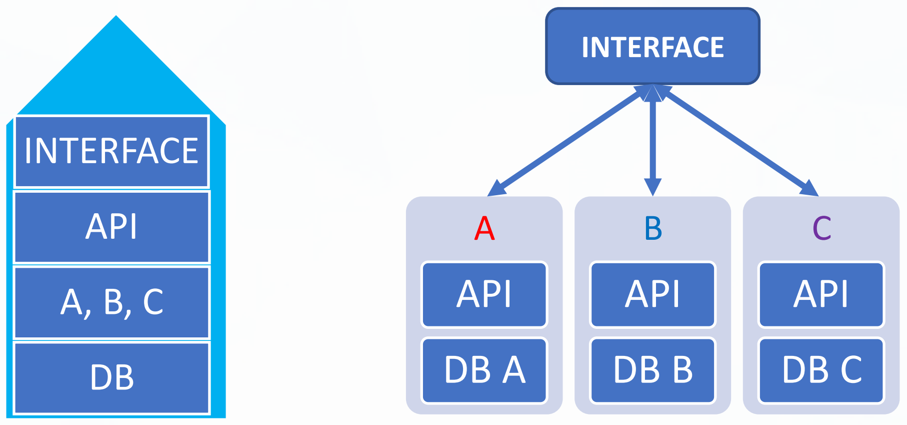

Figure 1.0 – Monolithic Architecture (Left) and **Microservices Architecture (Right)**

In solving the issues, **Microservices architecture** will be more efficient and provide better scalability to handle more users. ***Table 1.0*** below shows some of the differences between **Monolithic architecture** and **Microservices architecture**.

ASPECT	MONOLITHIC	MICROSERVICES
Development Speed	Faster development time but most likely not systematic.	Slower but systematic development time.
Maintenance Cost	Harder to maintain as the system grow in scale.	Easier to maintain even when the system grow in scale.
Bugs Fixing	Bugs may appear more frequently and takes time to track and be fixed.	Tracking bugs can sometimes be difficult.
Task Management	Tasks delegation may be a bit messy especially during development phase due to sharing database and dependencies issue.	Tasks delegation among teams can be easily managed for each service.
Downtime Effectiveness	If the system receive a fatal error, most likely the whole system will be at downtime.	If any of the services receive fatal error, other services may still working as usual.
Bottleneck Risk for Huge Active Users	High possibility of bottleneck to occur if concurrent users are not properly handled.	Bottleneck will not be a risk since different users may only subscribe to some services.

Table 1.0 – Differences between Monolithic and Microservices

POSSIBLE PROBLEMS

When comparing monolithic architecture with microservices architecture, microservices will provide more advantages for large scale system with many users. However, we cannot deny that nothing is perfect, so does Microservices. ***Table 1.1*** below shows the cons for microservices-based system.

CONS / DISADVANTAGE	EXPLANATION
Higher Development Cost	• Since microservices-based system will be divided into many micro services, thus require multiple development team to lead each services which eventually will increase the development cost.
Longer Development Time	• The time required to achieve the objectives will be longer since every services need to be constructed without any communication problem with each other service.
Higher Deployment Cost	• A large scale microservices-based system might require more servers to host each services.
Network Bandwidth Usage	• Each service will communicate with another service by request and this will increase the network bandwidth usage.

Table 1.1 – Cons / Disadvantages of Microservices-Based System

THE STEPS

In order to migrate a monolithic system to microservices system, software engineers must know where to start and what to be conducted.

a) Disintegrate services into individual services.
Existing monolithic system must be first identified and disintegrated into individual services as shown in ***Figure 1.1***.

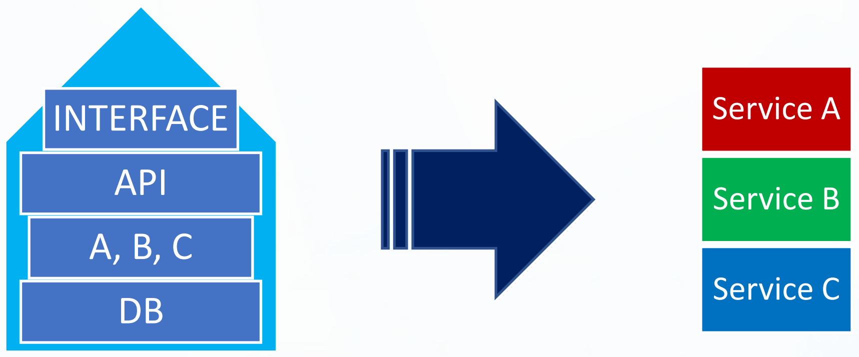

Figure 1.1 – Disintegration of Services

b) Identify each services data flows and relationships.

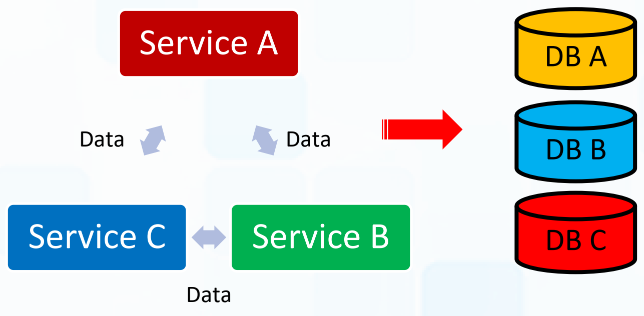

Figure 1.2 – Data Flows and Relationships

Once services has been disintegrated, databases for each services must be then identified. This require a precise data flows and relationships analysis among services outlined as depicted in ***Figure 1.2***.

c) Create API end points for each database.

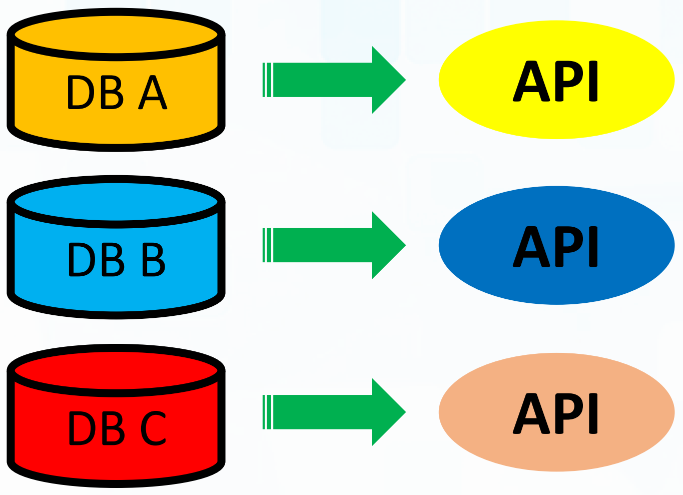

Figure 1.3 – API End Points

Figure 1.3 shows the illustration on APIs creation from databases. As databases schema would be different that in monolithic architecture, for every databases, new APIs end points will be created according to the each services business logic and their relationships.

d) Create or reuse user interfaces.

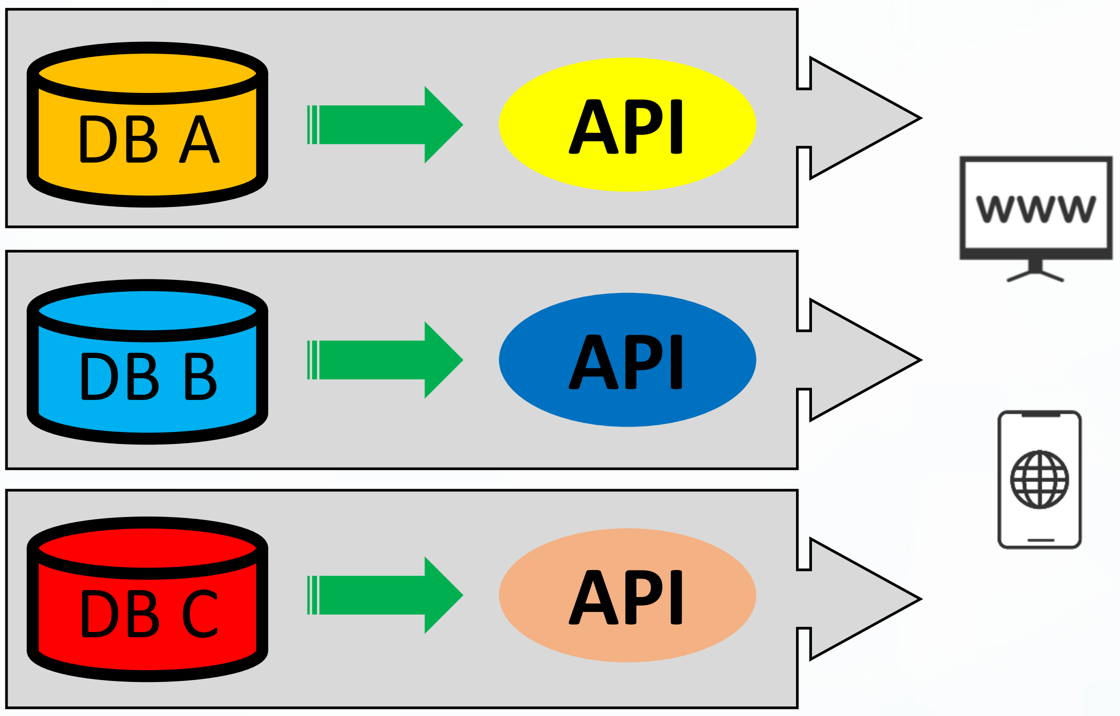

Figure 1.4 – Creating or Reusing Existing User Interfaces

User interfaces can be reused or be created only when necessary as microservices architecture only affect the back-end of the system as illustrated in ***Figure 1.4***. However some of the APIs' end points should be reconfigure. In the case where the decision is to convert the business logic into subscription based, then changes must be done to the user interface to fit the logics.

CONCLUSION

Migrating from monolithic architecture to microservices architecture will have the disadvantages and advantages. As we compared them together, we can see that microservices obviously provide better scalability as progression growth. Regardless, we still have to look for the possible issues that could occur.