Aiming At Customer Delight

info@newtglobal.com

CONTINUOUS INTEGRATION & CONTINUOUS DELIVERY

BUILDING AN EFFICIENT MICROSERVICES ARCHITECTURE

AN INTRODUCTION

In the land of Micro Services the question of analytics, complexity of algorithms, schema reporting gets well defined with a resilient data model. The culture and design principles should embrace failure and faults, similar to anti-fragile systems. As the Data system do not change as often as the surrounding application stack, it is important to a design the data base system for long term. This is a classic example of NoSQL database overcoming relational database systems in the recent years. However, the CRUD operations are continued by the Database Administrators and Programmers on a day to day basis to make sure the updates are in place with an un-interrupted data flow. The update and delate action eventually result in potential data corruption or a software bug.

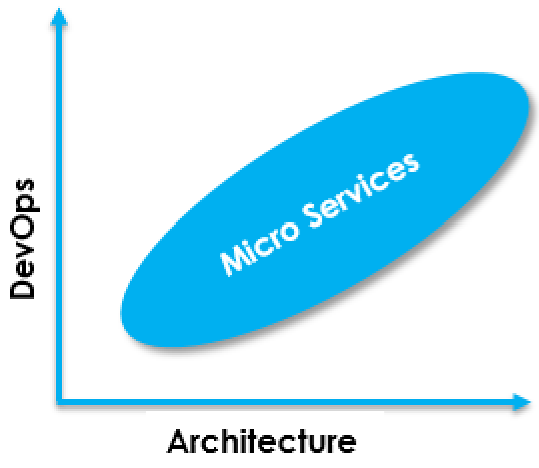
An immutable data store essentially eliminates the update and delete aspects of CRUD, allowing only the creation and reading of data records. How can we have a functional data store without the ability to update and delete data?

As explained by Mike Gancarz, Micro Services Architecture is the UNIX philosophy applied to distributed systems. The philosophy is quite simple –

"Do one thing and Do it well", further drilled to -

- Deep grained services to perform a single function.
- Implementation of automation of testing and deployment

  by individuals on a parallel track.
- Services that is simpler, flexible, robust, maintainable, and complete.



BUILDING MICRO SERVICES USING LAMBDA ARCHITECTURE

"Focus on business logic and not on infrastructure cost". At first glance, this seems like a major hurdle. Immutable deployments result in Disposable infrastructure. A Lambda is a function that takes an event and a context with the architecture that lends itself to a continuous delivery software development process. A change to a small part of the application only requires one or a small number of services to be rebuilt and redeployed that can be executed on a parallel track.

With Lambda Architecture, a timestamp is appended for each row in a database along with the id and other columns. Whenever there is an update or modification on any data row, the timestamp picks the recent append and throws the outcome. This method eradicates the process of the query running through each row of the database.

Consider the following example, with an immutable data store with Lamda Architecture

Initial scenario:

Notice that each fact is recorded with a time stamp. Each fact as recorded remains true for all time.

Customer ID	Customer Name	Preferred ID	Product	Time Stamp
1	John	2801	VOIP	1391123230
2	Robert	2804	Router	1391423650

After update:

Customer 2 now prefers VOLTE. An additional record is appended to reflect this new fact. The new record contains an updated time stamp.

Customer ID	Customer Name	Preferred ID	Product	Time Stamp
1	John	2801	VOIP	1391123230
2	Robert	2804	Router	1391423650
3	Robert	2981	VOLTE	1391423769

The benefit is that even if the real-time layer fails, no data will be lost. As long as incoming writes are being propagated to the batch storage layer, the results will eventually catch up when the next batch job runs. This provides greater resilience to the Data system with an option to de-normalize as needed for the real time and batch views. Under the Lambda Architecture, results from the batch layer are always eventually consistent. This is where most of the system complexity is isolated, by choosing the right critical path. The system can interact with the non-critical path using a feedback mechanism that can be defined as Strategies.

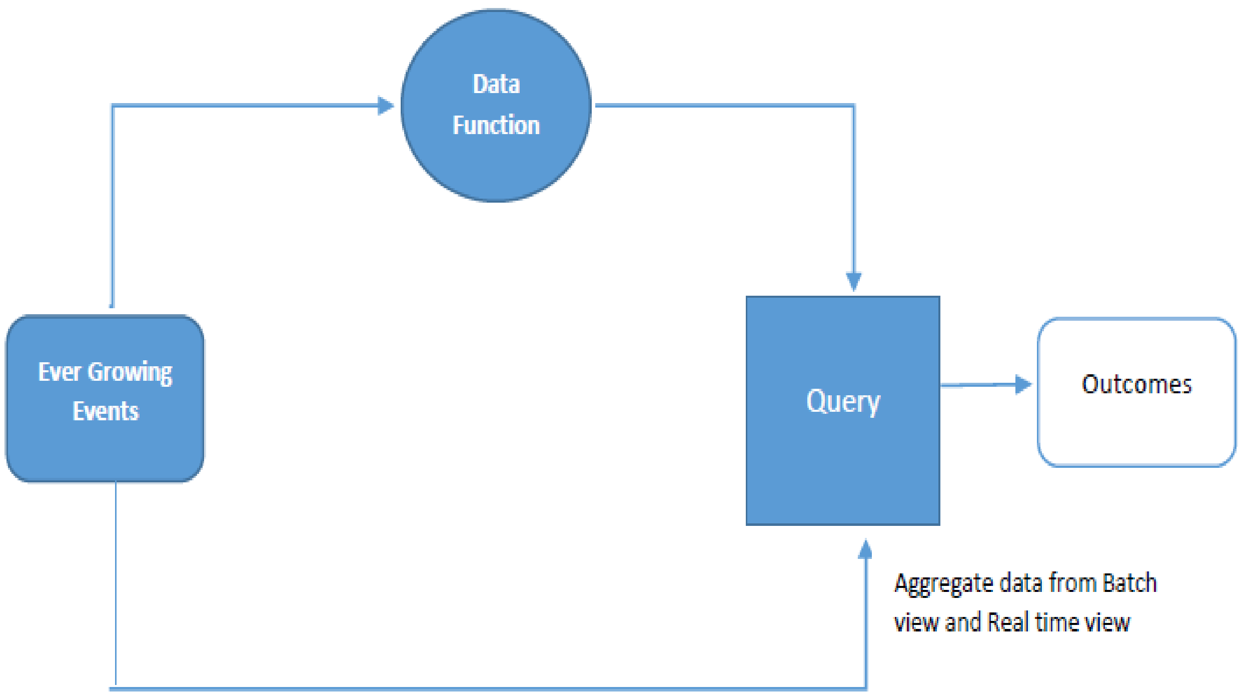


FIG.: LAMBDA ARCHITECTURE

info@newtglobal.com

MICRO SERVICES PERFORMANCE

The goal developing Micro Services in a Lambda Architecture is to bring simplicity, scalability, and performance to server side data. With an immutable data events, pre-built test automation has the maximum coverage of code and enables a healthy integration and component testing. Lowering network and data latencies yields better infrastructure as the application need keep changing.

The Lambda architecture results in –

- Better Data Resilience as the batch layer fares well over real time data that is vulnerable to errors and potential data flaw
- Better Conflation of queries and data, as the data can now be stored in a normalized manner and can be de-normalized as needed
- Better Scalability based on the Complexity of the application