Curious Case of LinkedIn DB

Mukul Kumar Chaundhyan | Nilutpol Borah
Ritesh Sinha | Vijender Singh | Yogesh Kumar

LinkedIn – How NO SQL helped in overcoming performance barriers.

About Company

Headquartered in Silicon Valley and Founded in 2003, **LinkedIn** is the world's largest professional network with more than 756 Mn professionals.

A diversified business model with revenue coming from Talent Solutions, Marketing Solutions, Sales Solutions and Premium subscription products

Executive Summary

LinkedIn, as we all know, is a *professional social network,* a.k.a. the official signing page of 756mn members, primarily professionals, which includes executives of all the fortune 500 companies at the very least.

The platform hosts a range of ***businesses processes*** revolving around the ecosystem of professionals, including Social networking, Campaign management, Learning, Sales Insight and Talent insight.

What we as a consumer enjoy from the outside, in terms of seamless exchange of data shared by the professionals and read through various sources by their peers and prospective employers, is just the tip of the iceberg. What hides underneath it is a series of challenges that LinkedIn and its database team have faced over the years whilst evolving from the traditional RDBMS system to something more appropriate for the ever-growing network. The *challenges* include - dynamic schema requirements, the ability to shard the data given its growing volume, data centre failover, bulk data ingestion, growing analytics requirement and massive cost as the data multiplied exponentially, to name a few.

Over the years, LinkedIn has overcome these dynamic challenges by migrating its database system from RDBMS to NO SQL database to cater to the volume, variety and velocity while maintaining a balance on elements of consistency, availability, and partition tolerance.

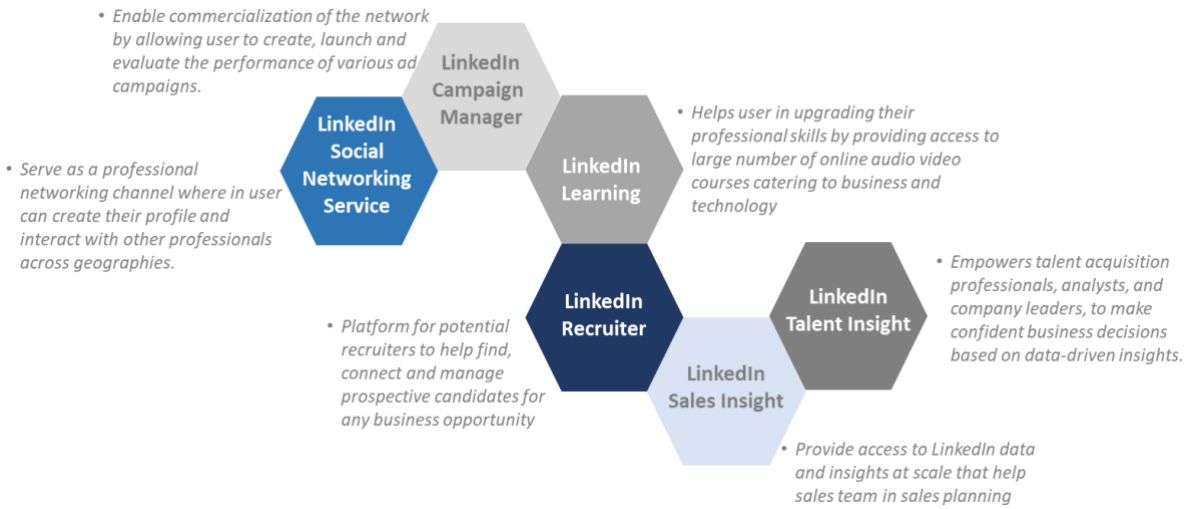
They first migrated to Voldemort no SQL key-value type database, which was more focused on AP. Later they moved to Espresso, no SQL document type database, which was more inclined towards CP. And now we have their latest in-house no SQL graph development, 'the liquid', which takes consistency and analytics of 2nd & 3rd degree connections to all-new levels.

So, in a nutshell, we see the successful evolution of the social network behemoth, which was possible through the sheer up-gradation of its backend database technology, with improved efficiency elements around CAP whilst dealing with the 3 V's of big data. And today, on account of the strong foundation built on the most advanced database and data analytics, it has solved the business problems of the professionals by providing him with an array of products that addresses their needs and generates a steady and growing revenue stream that capitalises on tapping the entire ecosystem.

Curious Case of LinkedIn DB

Mukul Kumar Chaundhyan | Nilutpol Borah
Ritesh Sinha | Vijender Singh | Yogesh Kumar

Key Business Processes



Key Challenges

Schema Evolution	With increasing user network and the volume of data shared by them, the backend schema needs to be evolved continuously. As a new feature comes up there is a continuous need to expand the existing tables to accommodate the same.
Scalability	RDBMS does not provide the functionality of Sharding the data onto multiple nodes implicitly, which would cause concern for any company with a massive amount of data.
Datacenter Failover	Earlier, when LinkedIn was using the RDBMS, they were operating in a master-slave modal. So, whenever they want to do a failover, it requires downtime to make the failover happen.
Bulk data Ingestion	LinkedIn is producing and consuming a large amount of data that could be used from a range of use cases such as feeds to analytics. With the legacy RDBMS, LinkedIn was not able to leverage the power of analytics in offline mode.
Cost	RDMS is always costly, as it requires a fixed set of or predefined infrastructure. Internet companies usually deal with loads of data, which would demand a heavy initial investment.

NO SQL CONSIDERATIONS

Voldemort is a distributed KV store that is both persistent and fault-tolerant. Primary included in the stack to meet two requirements – faster read access and fault-tolerant write.

Espresso is a fault-tolerant and distributed NO SQL database with a hierarchical structure. It helps in low latency reads and writes and provides a timeline consistent change capture mechanism as required for nearline and offline data processing.

HOW NO SQL HELPED

Voldemort: Key features that helped in achieving the above objective includes:

- Data replicated over multiple servers as defined by users with a Consistent hashing mechanism to identify the correct data location.
- Server failure is handled using actual request with predefined SLA instead of simple PING
- Data is versioned using a vector clock mechanism that ensures the repair of data in case of server failure with the correct version
- No central point of failure as there is no master-slave relationship.

However, the system was not without drawbacks few of which include

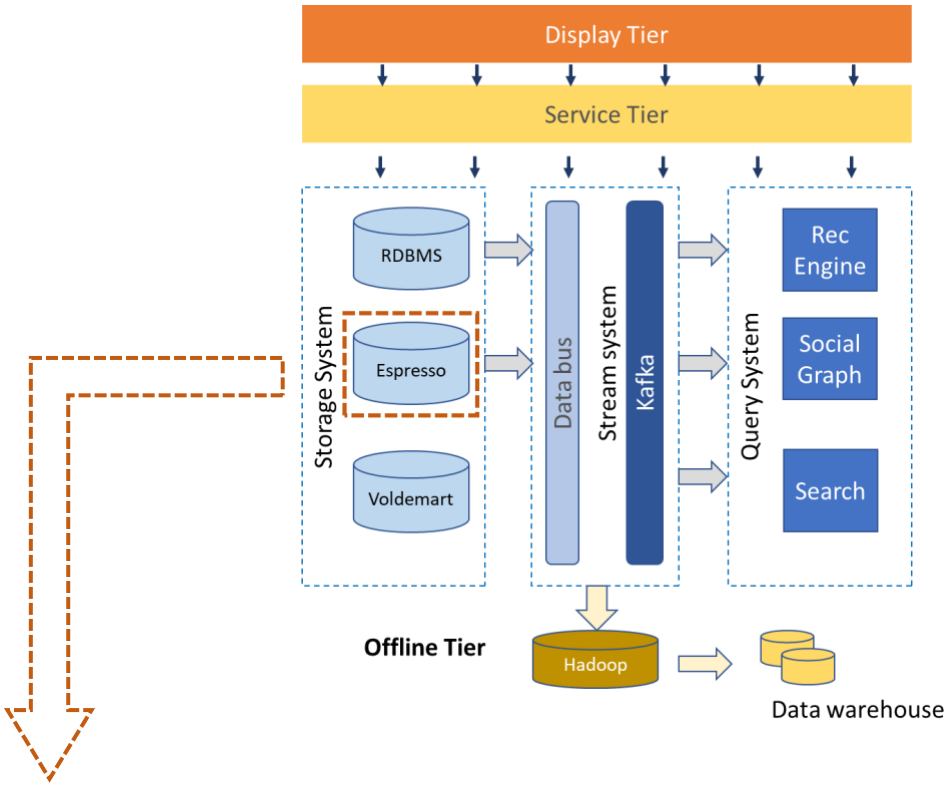
- Inability to handle complex queries owing to KV structure
- Absence of triggers and constraints
- Poor in detecting failure due to absence of master node

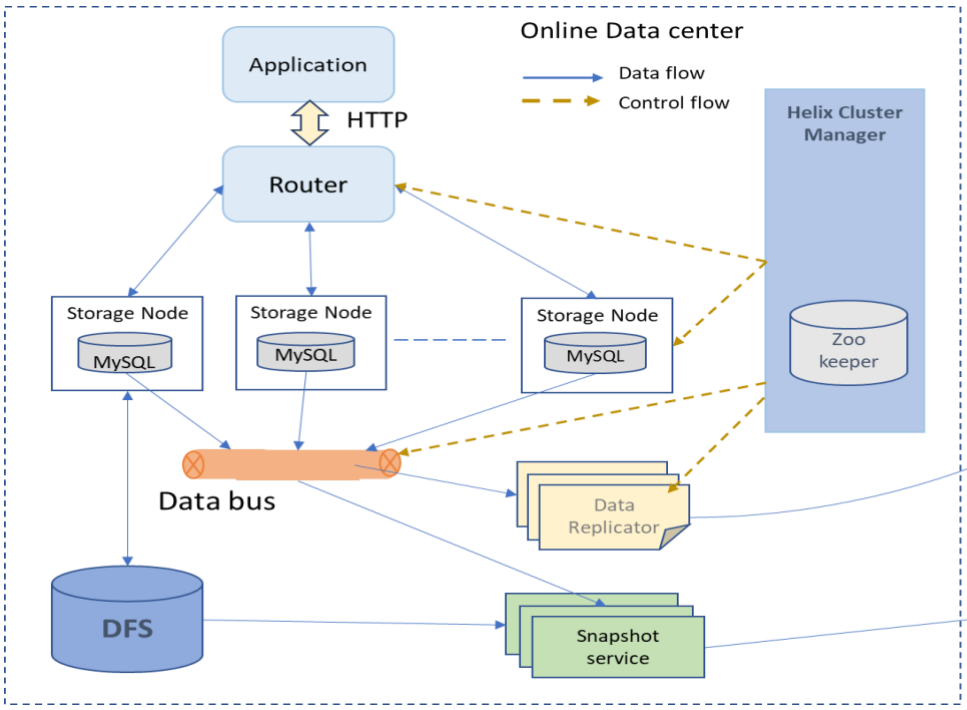
Espresso: Key components and features of Espresso that helped solve the problem include:

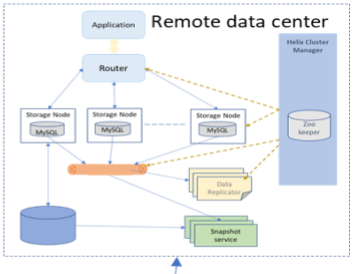
- **Hierarchal data structure** – The hierarchy is maintained in the order Database->Table->Collection->Document. Each data point is fully accessible with the fully specified key made up of partition key and subkey. The partition key helps in identifying the partition of the table where the document is stored, and the subkey finally identifies the document
- **Elastic & distributed**: Easy to scale horizontally with the help of sharding, and the tables are spread over multiple nodes in the N configurable clusters.
- **Storage nodes**: Data is stored as a partition in storage nodes, with each storage node hosting multiple partitions. The storage node is responsible for query processing and maintaining secondary indexes to search within documents. They also support transaction at the collection level and is responsible for maintaining consistency in the data.
- **Routers**: The router is responsible for sending the request to the correct storage node within the cluster based on the partition data that it holds. The routing data changes every time the cluster manager change the state of the partition
- **Fault-tolerant cluster management:** The whole system is divided into clusters managed by Apache Helix, and each cluster has multiple storage nodes. Each partition has one master and configurable slaves, and once the master goes down, a slave picks up a master role to fill the void. Data is replicated over multiple storage nodes to provide fault tolerance. Helix continuously evaluates the current state of the partitions and issues the transitions required to reach an ideal state.
- **High throughput Databus:** It helps implement a change capture mechanism by transporting transactions(e.g., “writes”) in the order of commit. It is primarily used to send events to downstream applications or during data centre replication using a Data replicator
- **Data Replicator:** It helps replicate data across remotely distributed Espresso clusters by consuming events in the change stream as published by databus. It periodically stores the state of replication in a zookeeper to survive any kind of node failure.
- **Data backup using Snapshot service:** It regularly backups data from the Espresso cluster to Hadoop to be integrated with other downstream data pipelines.

Mukul Kumar Chaundhyan | Nilutpol Borah
Ritesh Sinha | Vijender Singh | Yogesh Kumar

HOW DOES IT ALL FIT IN A STACK







Mukul Kumar Chaundhyan | Nilutpol Borah
Ritesh Sinha | Vijender Singh | Yogesh Kumar

Benefits Realized

Faster Read Write

With Voldemart the read write latency was brought down from over 400 ms to 10 ms while increasing the overall data store capacity

Detecting failure

Both Voldemart and Espresso provides flexibility to detect failure quickly with Espresso taking an edge with its master slave architecture within each cluster

High Scalability & Cost effective

The systems can be scaled horizontally using key sharding and provides a faster lookup of the storage node using effective hash mechanism. The relative cheaper infra ensures cost saving at the same time.

Flexible & Complex schema support

While Voldemart has a few limitations in storing/accessing complex data type, Espresso with its document store and effective search mechanism can support schema evolution and faster query processing.

Partition tolerant

Both the databases are AP system that ensures continuous availability of data even **when** one cluster goes down.

What Next?

The platform has seen tremendous growth in the network during the last decade. While they have successfully met all their data needs to date, the future demands much faster data processing in real-time. **LinkedIn** has therefore invested heavily in the next wave of NO SQL implementation – **Liquid, a graph-based database** and can meet all the future requirements. It is an excellent boost to the social networking scenarios where the load time was several hundred milliseconds, and with the help of the new graph DB, they can cut it short by a hundred milliseconds.

With the new DB, the platform expects to handle queries related to 2nd and 3rd level connections quite at ease even when an individual node has a degree of more than 100. It further brings substantially real-time processing capabilities that can generate helpful recommendations in a **reasonable** time frame.

References

- https://blog.linkedin.com/2009/04/01/project-voldemort-part-ii-how-it-works
- https://blog.linkedin.com/2009/03/20/project-voldemort-scaling-simple-storage-at-linkedin
- https://www.project-voldemort.com/voldemort/
- https://engineering.linkedin.com/espresso/introducing-espresso-linkedins-hot-new-distributed-document-store