AARHUS UNIVERSITET

Microservices and DevOps

Scalable Microservices

Scalable Storage through Sharding

Henrik Bærbak Christensen

Wikipedia

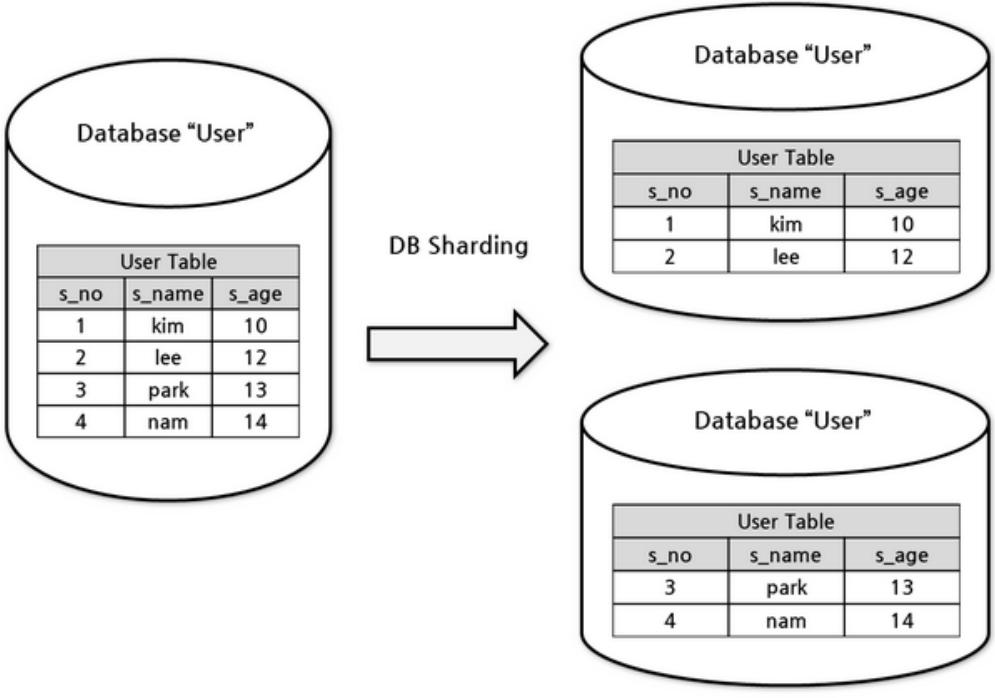
Database architecture [edit]

Horizontal partitioning is a database design principle whereby *rows* of a database table are held separately, rather than being split into columns (which is what normalization and vertical partitioning do, to differing extents). Each partition forms part of a **shard**, which may in turn be located on a separate database server or physical location.

There are numerous advantages to the horizontal partitioning approach. Since the tables are divided and distributed into multiple servers, the total number of rows in each table in each database is reduced. This reduces index size, which generally improves search performance. A database shard can be placed on separate hardware, and multiple shards can be placed on multiple machines. This enables a distribution of the database over a large number of machines, which means that the load can be spread out over multiple machines, greatly improving performance. In addition, if the database shard is based on some real-world segmentation of the data (e.g., European customers v. American customers) then it may be possible to infer the appropriate shard membership easily and automatically, and query only the relevant shard.[2] Disadvantages include :

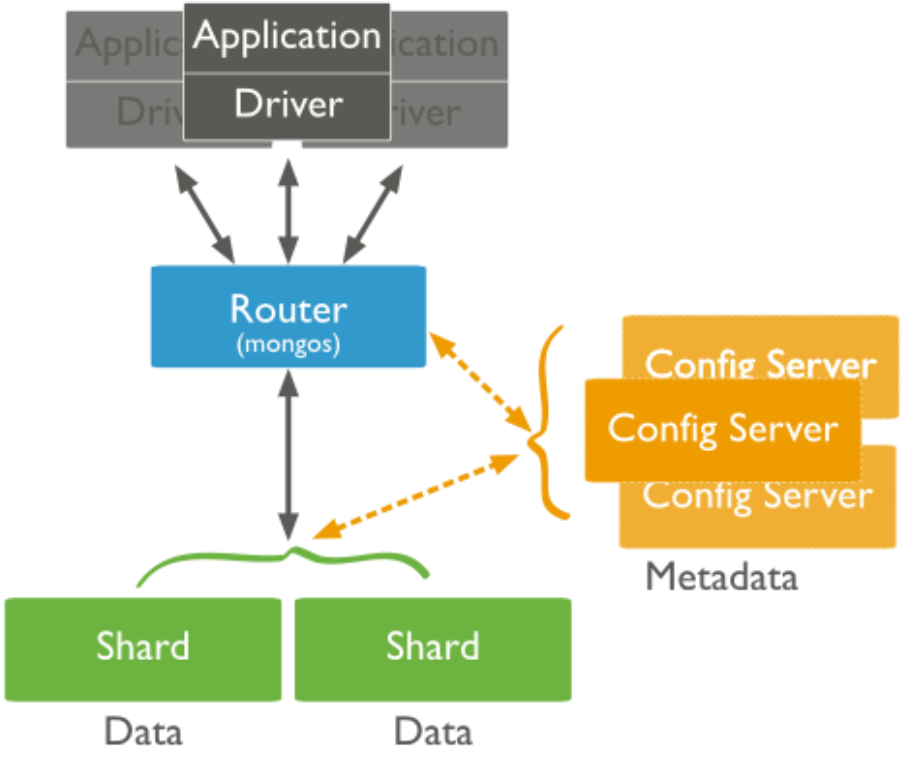
- A heavier reliance on the interconnect between servers[*citation needed*]
- Increased latency when querying, especially where more than one shard must be searched.[*citation needed*]
  - Data or indexes are often only sharded one way, so that some searches are optimal, and others are slow or impossible.[*clarification needed*]
- Issues of consistency and durability due to the more complex failure modes of a set of servers, which often result in systems making no guarantees about cross-shard consistency or durability.[*citation needed*]

So…


Database "User"
User Table
s_no s_name s_age
1 kim 10
2 lee 12
3 park 13
4 nam 14
DB Sharding
Database "User"
User Table
s_no s_name s_age
1 kim 10
2 lee 12
Database "User"
User Table
s_no s_name s_age
3 park 13
4 nam 14

MongoDB Sharding

- Shard Key
  - Doc field, determine which shard the doc will reside on
- Production
  - Each shard = a replica set!



AARHUS UNIVERSITET

Redis Sharding

- Is part of the Clustering mechanism…

- So – we have already been there…

AARHUS UNIVERSITET

Sharding

- Every key is part of a **hash slot**
  - Redis has exactly 16384 hash slots
    - Every key is mapped to one of these hash slots
- Every node (master) is responsible for a subset, e.g.
  - Node A contains hash slots from 0 to 5500.
  - Node B contains hash slots from 5501 to 11000.
  - Node C contains hash slots from 11001 to 16383.
- If we add two more nodes (masters), we just move the relevant hash slots to the new masters…
  - Uhum, transactions need to cover only keys in the same slot ☹
    - So there is a mechanism to guaranty that… anyway…