Traditional Ways To Solve Scalability Problems With RDBMS

Abhishek Tiwari

Citation: *A. Tiwari*, ”Traditional Ways To Solve Scalability Problems With RDBMS”, Abhishek Tiwari, 2012. doi:10.59350/yq9sh-56257

Published on: October 05, 2012

Notes plus thoughts from my recent read Cassandra: The Definitive Guide. Common ways to solve scalability bottleneck with relational databases,

Throw More/better Hardware (memory And Cpu)

- Vertical scaling
- Faster disks (SSD vs RAID)

Move To A Database Cluster

- With master-slave configuration:
  - Master is now single point of failure.
  - Update slave as master during failure.
  - Separate read from writes - dedicated slaves for reads.
  - Load balance the read requests.
  - Replication latency and delay.
  - Master and slaves out of sync.
- With multi-master configuration:
  - Synchronous replication:
    - Each transaction to be applied to all the master.
    - Performance issues and blocking if one master goes down.
  - Asynchronous replication:
    - Each transaction is buffered in master (“deferred transaction queue”) and then pushed periodically to other masters.
    - Higher performance but consistency issues.
    - Avoid conflicts-duplicate keys, auto-increments.
    - Row based replication to avoid data drift.
    - Throw additional boxes in a database cluster.
    - Data replication and consistency issues during regular usage due to latency or during failover scenarios.

Work On Improving Indexes And Query Optimisation

- Reducing and reorganising joins.
- Removing resource-intensive features.

Employ Caching Layer

- Consistency issues due to gap in updates in the cache and updates in the database.
- Write though vs write back
  - Write through- data is written both cache and database. The total write time is the time to write to the cache plus the time to write the database.
  - Write back - no write time delay. Data is initially written to the cache, only when the cache is full or required is the data written to the database. Possible to lose data if cache fails.

Do Some De-normalization

- Duplication of data.
- Deviation from philosophy of 5 normal forms and basic principals of relations data modelling.

Introduce Sharding Into Your Relational Architecture

- Sharding can be simply defined as a "shared-nothing" partitioning in which there is no centralised or shared state.
- Common shard structures:
  - Feature-based shard or functional segmentation
  - Key-based sharding
  - Lookup table
- Shared-nothing architecture has no central controller and no notion of master/slave. All nodes are same.