Lecture Notes 3

Partitioning (Sharding)

- Split data over multiple nodes
- Partitioning improves performance

Replication

Consistency

- All nodes see same data at same time
- Consistency - there are multiple types
  - Strong - offers up to date data but at cost of high latency
  - Eventual - offers low latency but may reply to read requests with stale data since all nodes of the database may not have the updated data

Caching

- Data is written once and read many times - many systems have data with read-write ratio > 1
- Cache Consistency Problems
  - Write through, Write back
- Cache replacement policies

Caching in Distributed Systems

CAP Theorem

- Consistency, Availability, and Partition Tolerance
- Consistency - all nodes see same data at same time
- Availability - node failures do not prevent survivors from continuing to operate
- Partition Tolerance - the system contibues to operate despite message loss due to network failures

- CP Systems - Incorporates network partitions into their failure model and Strong Consistency, ie Twutter Manhattan Storage System
- AP Systems - Eventual consistency and highly available, ie most NoSQL databases, HBase

NoSQL

- NoSQL: Not Only SQL
  - HBase, Cassandra, MongoDB, CouchDB
  - Schema not predefined
  - For Horizontal Scaling

- Modern applications create massive volumes of semi-structured data
- Scale out architectures to reach the demand
- Traditional RDBMS were not designed to cope with the scale that faces modern applicatoins

RDBMS vs NoSQL

- BASE, not ACID
  - RDBMS (ACID) - Atomicity consistency Isolation Durability
  - NoSQL (BASE) - Basically Available Soft state Eventual consistency

- NoSQL: Schema-free, easy replication support, simple API, and High Availability (eventual consistent)
- What is missing in DoSQL?
  - No joins support in the Databse - Applications need to take care of joins
  - No complex transactions as in SQL
  - No constrains support (values could be null or empty)

- NoSQL is Highly Available (eventual consistend) and Scalable

Zookeeper

- Coordination in large scale distributed systems
  - locking service (paxos)
  - dynamic configuration management
  - group membership, etc.

- Zookeeper is already used in HBase, HDFS
- Design goals - High availability, fault tolerance, performance
- Runs on a collection of machines and is designed to be highly available, so applications can depend on it
- Zookeeper can help you avoid introducting single points of failure into your system, so you can build a

reliable application

Interesting Distributed System Design problems

- Tiny URL
- Google search
- Facebook Newsfeed
- Dropbox
- Twitter

Quiz on Friday

- There is an exam (1:10 to 7:45)
- 1 sentence answer, true false, 1 design (Interesting distributed system design questions)