GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

搭建大规模可扩展的系统

大纲

分布式系统
数据库系统
经典架构
设计原则：CAP理论
一致性介绍
关系型数据库
ACID vs. BASE
Sharding 分片
NoSQL数据库
Cassandra
实时系统：Kafka, Storm

What is Scalability

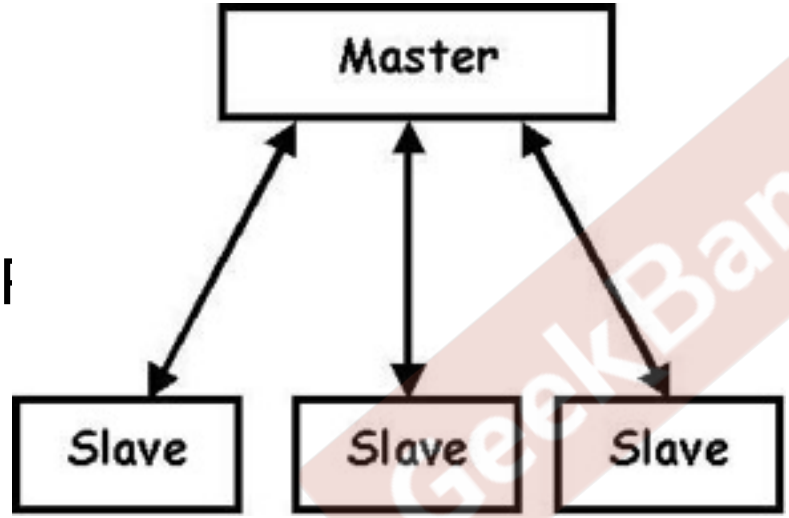
“The Scalability is measure of number of users it can effectively support at the same time without degrading the defined performance”

Has limits – E.g. “With two load balanced capacity it should support 1000 concurrent users with average response time of 3 seconds”

“Performance is what an individual user experiences; Scalability is how many users get to experience it TOGETHER”

Distributed System

Classic Master/Slave



如何检测一台机器是否宕机?

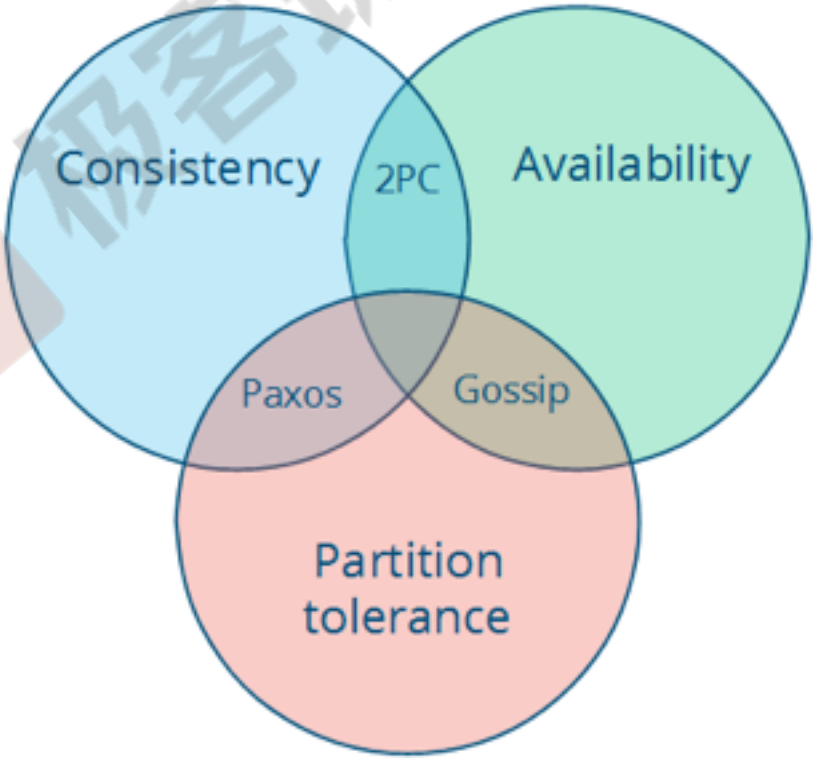
CAP Theorem

Consistency means that each client always has the same view of the data.

Availability means that all clients can always read and write.

Partition tolerance means that the system works well across physical network partitions.

“Pick any two.
You can’t have all three.”



Real world examples of CAP

Amazon’s Dynamo chooses availability over consistency. Dynamo implements eventual consistency where data become consistent over time
Google’s BigTable chooses consistency over availability

Consistentcy, Partition Tolerance (CP)

- Big Table
- Hbase

Availability, Partition Tolerance (AP)

- DynamoDB

Database System

Relational DBMS

Transactions in traditional system have to have the following properties

Earlier Systems were designed for **ACID** properties

A – Atomic

C – Consistent

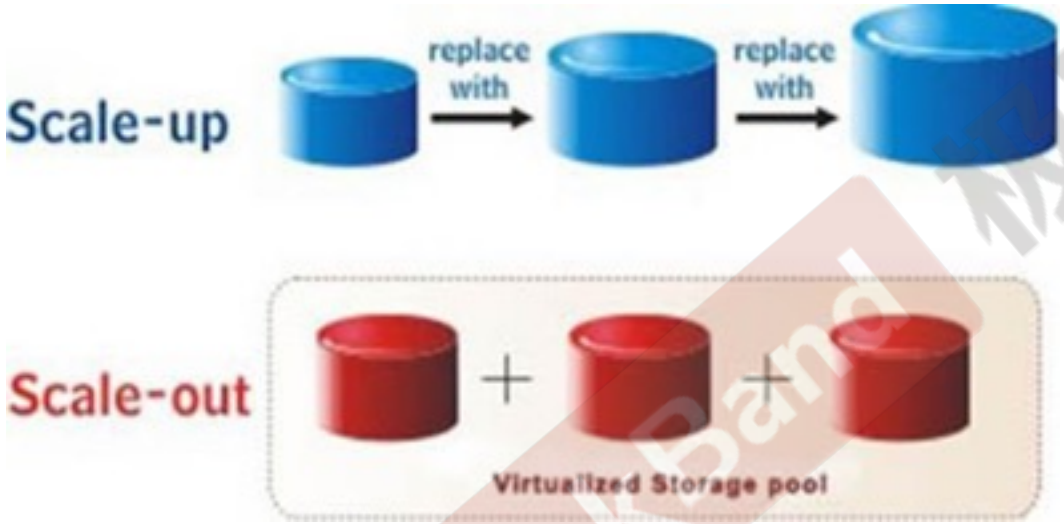
I – Isolated

D - Durable

ACID vs. BASE

“Basically Available, Soft state, Eventually consistent

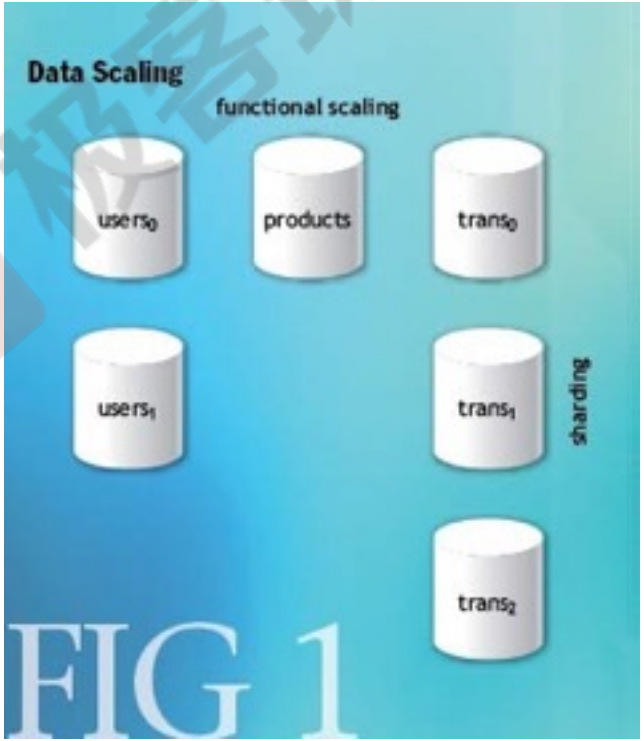
应用拆分



数据库拆分

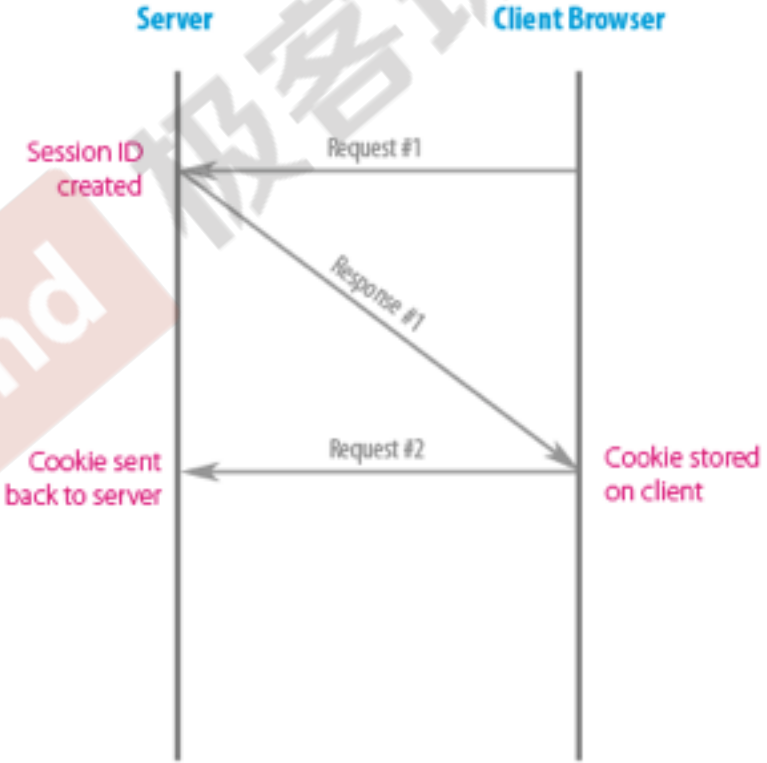
Horizontal Scaling
(Sharding)

Functional Scaling
(Scale out)



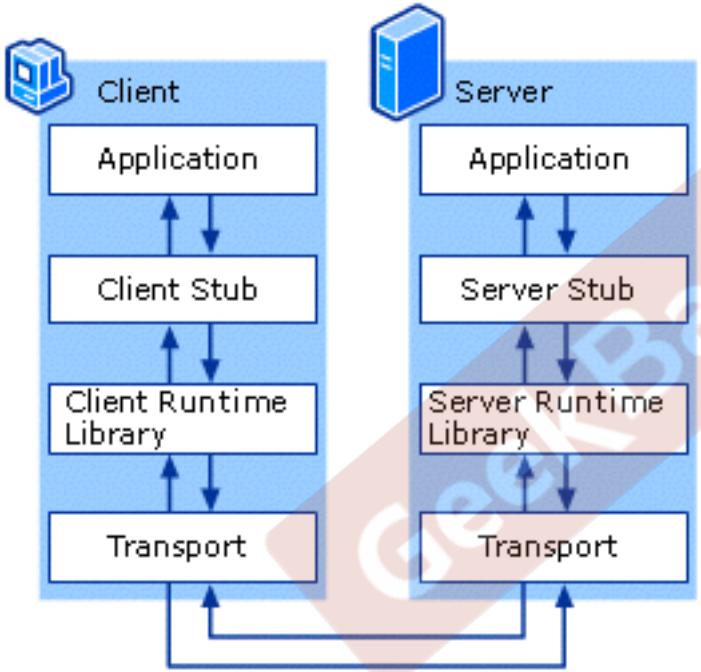
Stateless

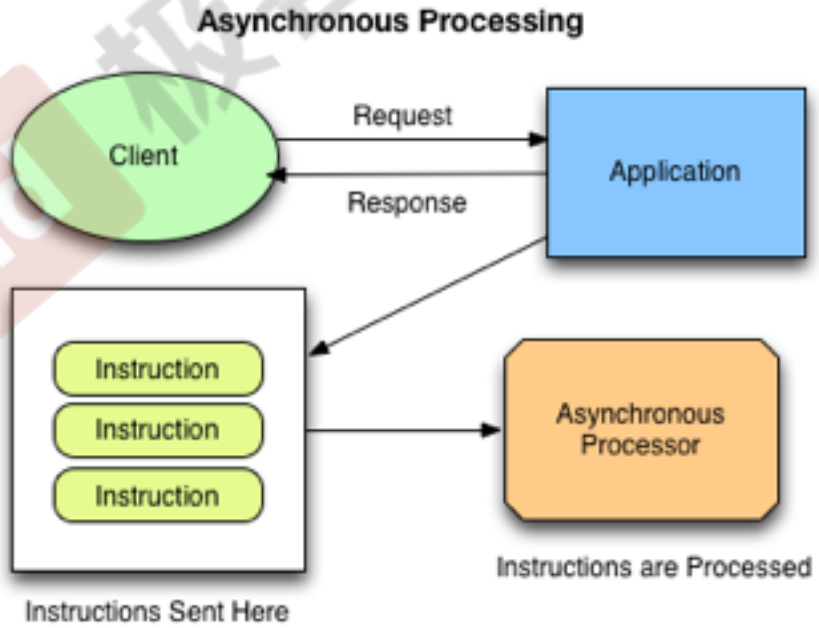
Session Vs. Cookie



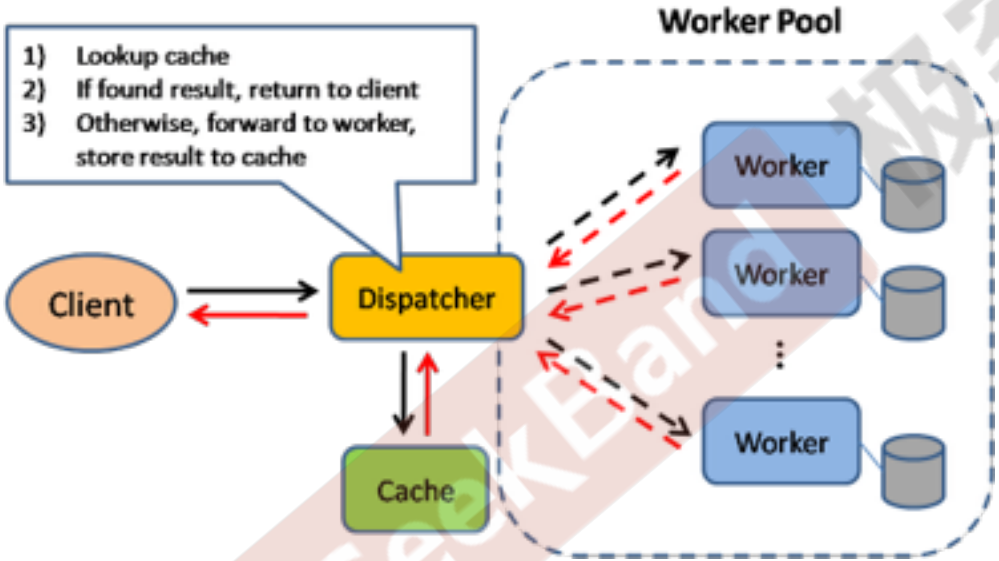
异步通信

RPC


Client
Application
Client Stub
Client Runtime Library
Transport
Server
Application
Server Stub
Server Runtime Library
Transport


Asynchronous Processing
Client
Request
Response
Application
Instruction
Instruction
Instruction
Asynchronous Processor
Instructions Sent Here
Instructions are Processed

有效利用Cache



Consistency & Replication

In order to increase reliability against failures data has to be replicated across multiple servers.
The problem with replicas is the need to keep the data consistent

一致性介绍

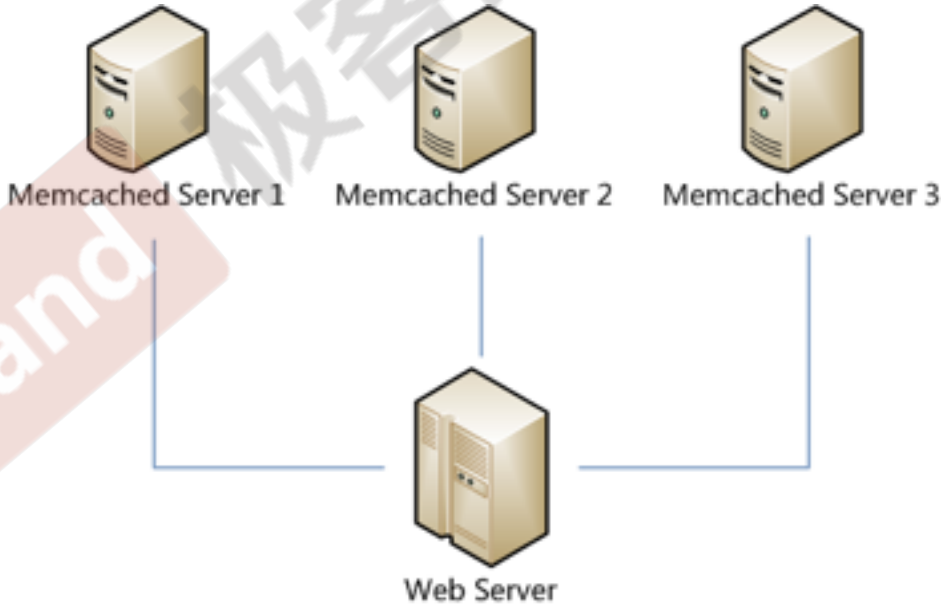
强一致性

弱一致性

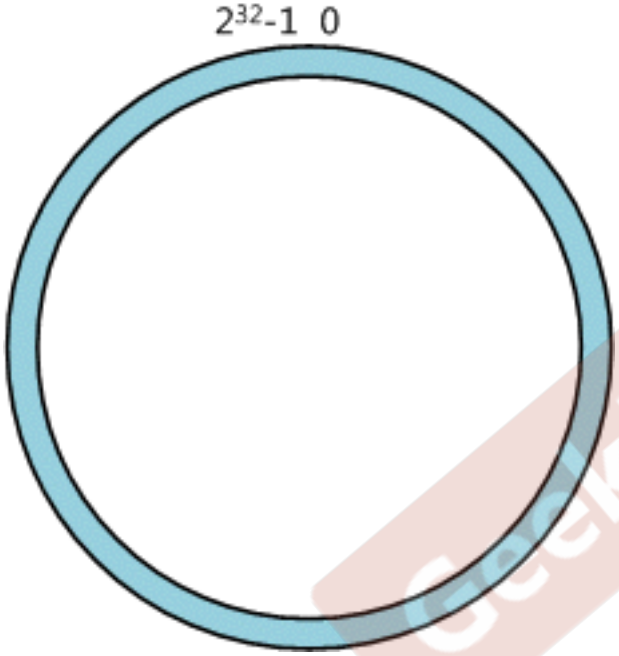
最终一致性

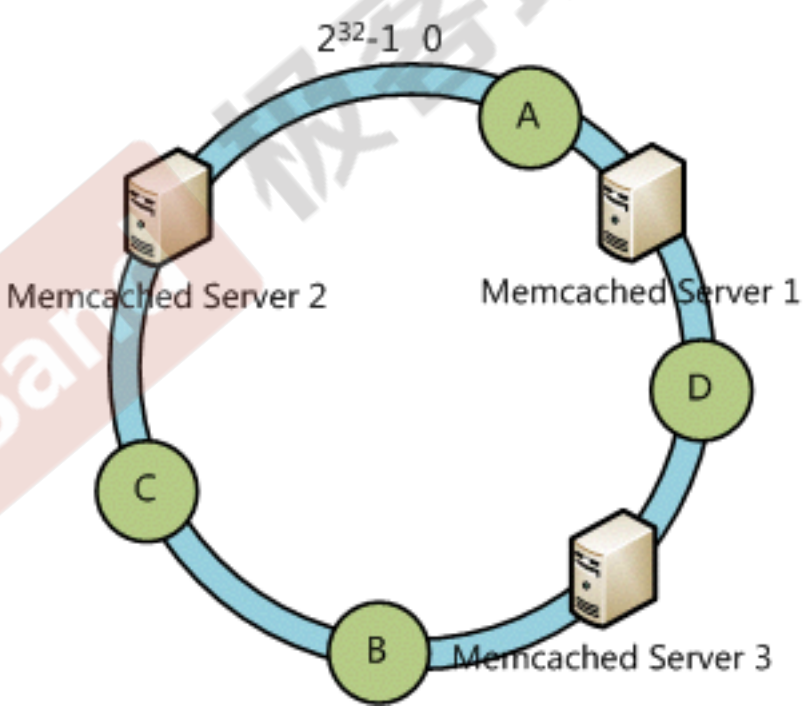
Consistent Hashing

Distributed Hash Table

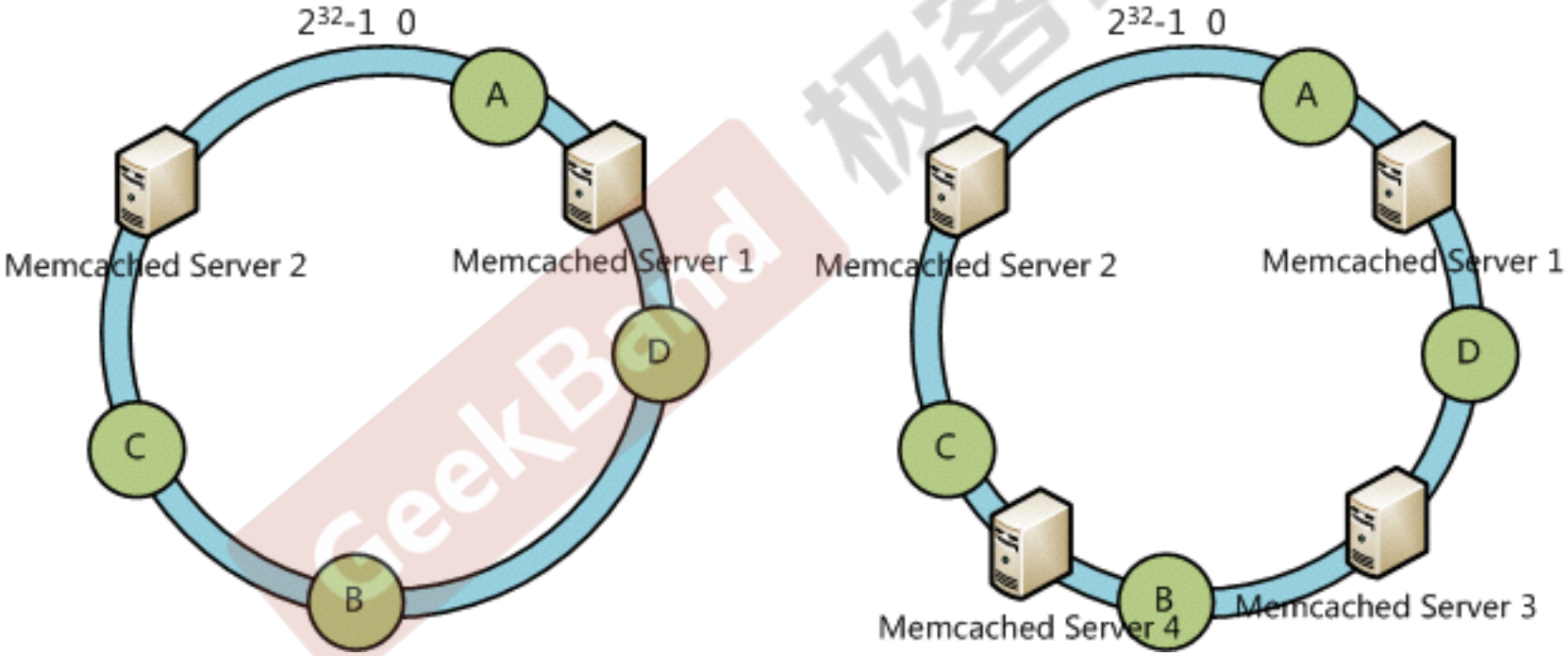


一致性hash算法

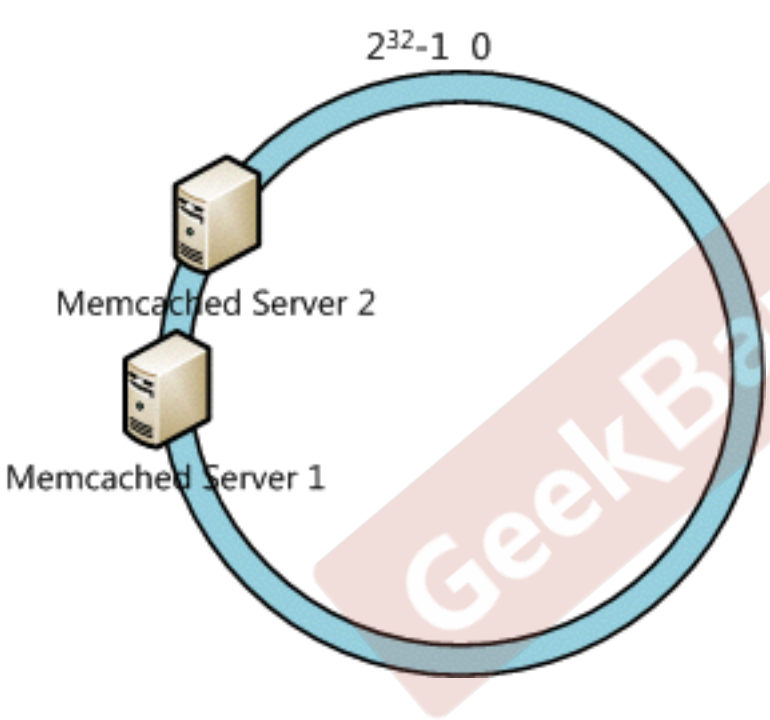


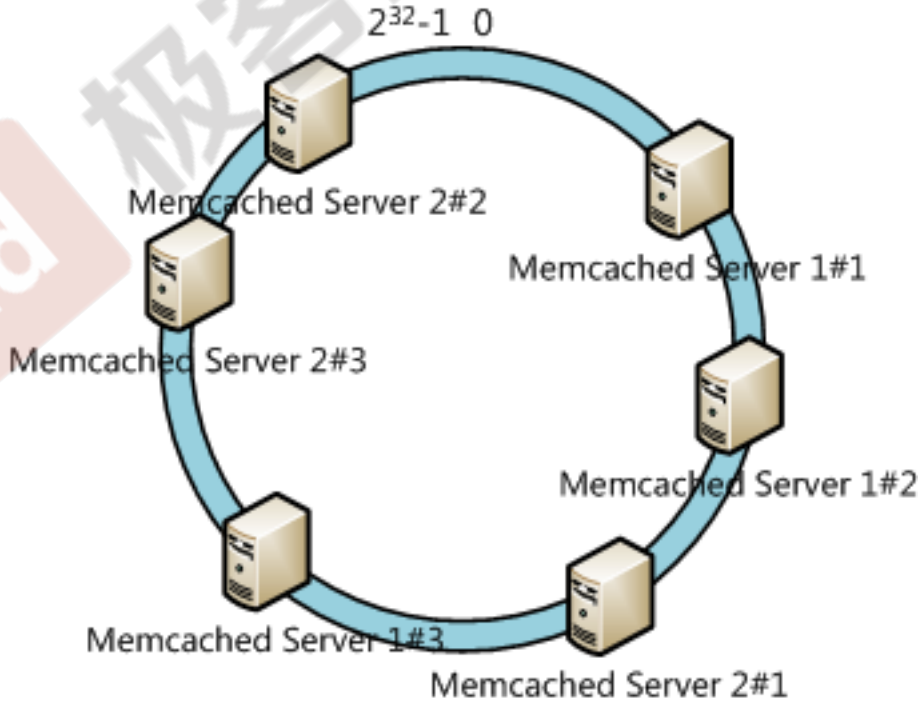


容错性和扩展性分析


2^{32}-1 0
A
Memcached Server 2
Memcached Server 1
D
C
B
2^{32}-1 0
A
Memcached Server 2
Memcached Server 1
D
C
B
Memcached Server 4
Memcached Server 3

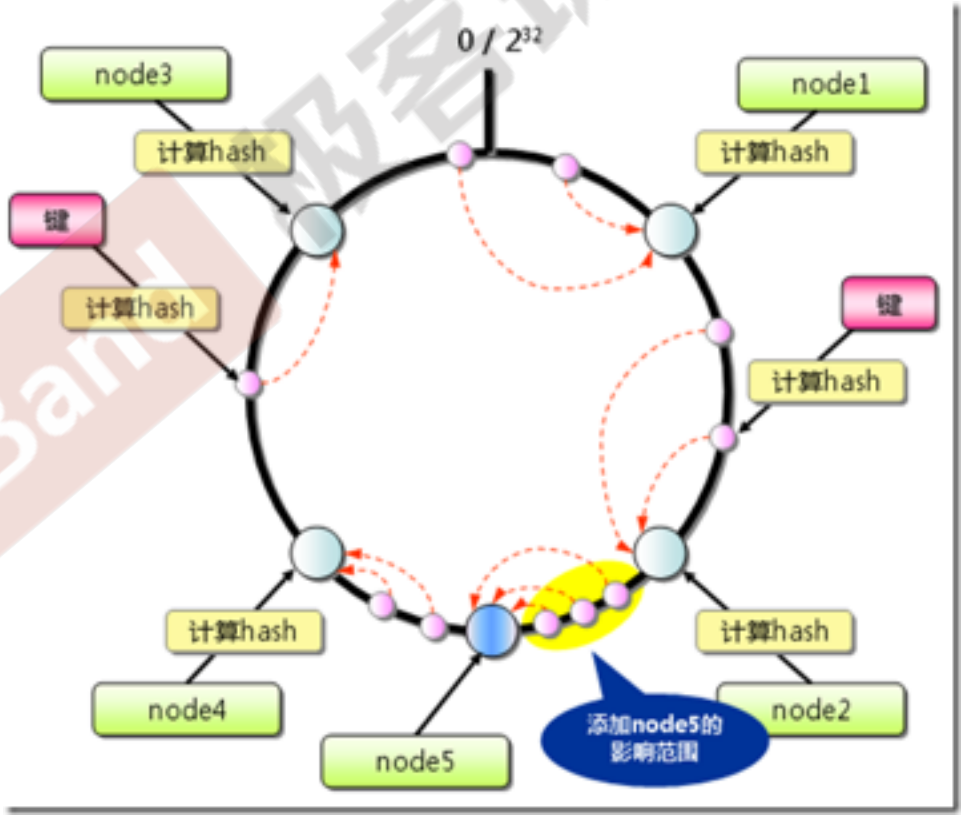
虚拟节点





一致性hash案例

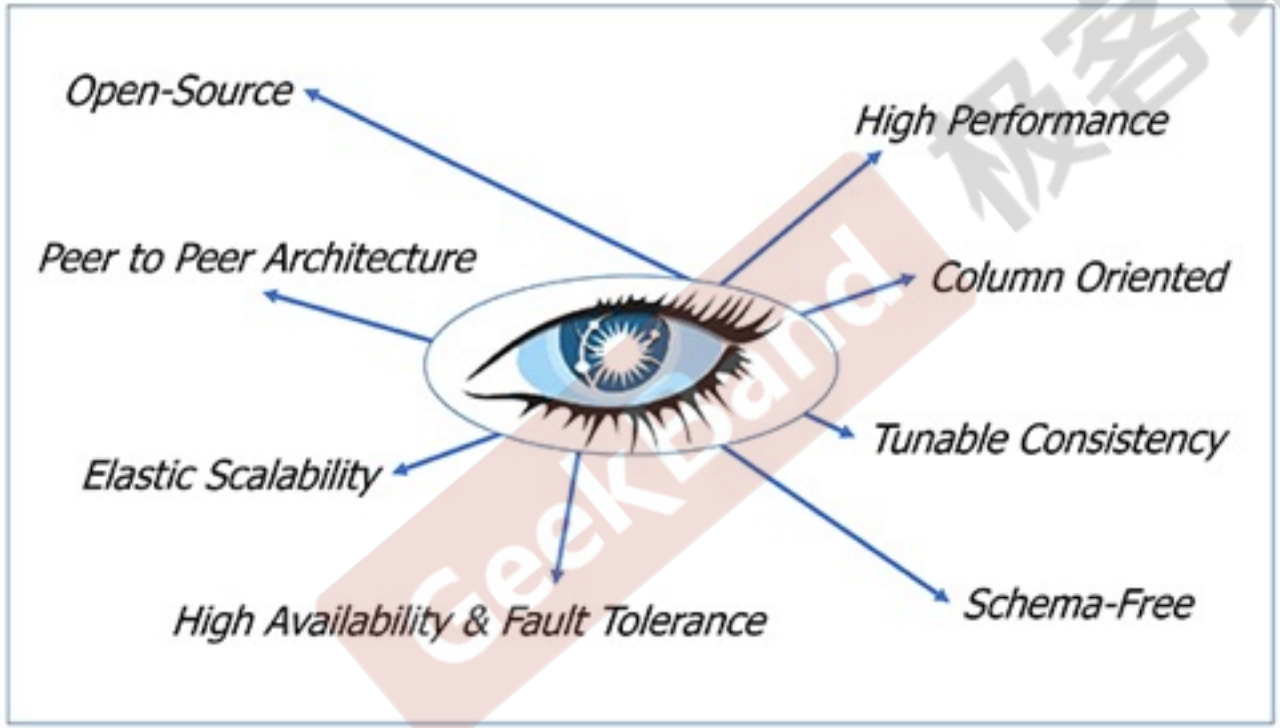
DB examples:
Cassandra
Amazon’s Dynamo
HBASE
MongoDB



NoSQL Databases

- Databases horizontally partitioned
- Simple queries based on gets() and sets()
- Access are made on key/value pairs
- Cannot do complex queries like joins
- Database can contain several hundred million records

Cassandra


Open-Source
High Performance
Peer to Peer Architecture
Column Oriented
Tunable Consistency
Elastic Scalability
High Availability & Fault Tolerance
Schema-Free

Cassandra Write Data Flows

Single Region, Multiple Availability Zone

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk

If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

SSTable disk writes and compactions occur asynchronously

NETFLIX

分布式系统设计模式1

1.系统失败是很平常的事情：每年有1-5%的硬盘会报废，服务器每年会平均宕机两次，报废几率在2-4%几率。

2.将一个大而复杂系统切分为多个服务：而且服务之间依赖尽可能的少，这样有助于测试，部署和小团队独立开发。例子：一个google的搜索会依赖100多个服务。ike：需要一套机制来确保服务的fault-tolerant，不能让一个服务的成败影响全局。

3.需要有Protocol Description Language：比如protocol buffers。ike：这样能降低通信方面的代码量。

4.有能力在开发之前，根据系统的设计来预测性能：在最下面有一些重要的数字。

5.在设计系统方面，不要想做的很全面，而是需要抓住重点。

6.为了增量做设计，但不为无限做设计：比如：要为5-50倍的增量做设计，但超过1000倍了，就需要重写和重新设计了。

7.使用备份请求来降低延迟：比如一个处理需要涉及1000台机器，通过备份请求这个机制来避免这个处理被一台慢机器延误。ike：这个机制非常适合MapReduce。

分布式系统设计模式2

8.使用范围来分布数据，而不是Hash：因为这样在语义上比较简单，并且容易控制。ike：在大多数情况下语义比性能更重要，不要为了20%的情况hardcode。

9.灵活的系统，根据需求来伸缩：并且当需求上来的时候，关闭部分特性，比如：关闭拼写检查。

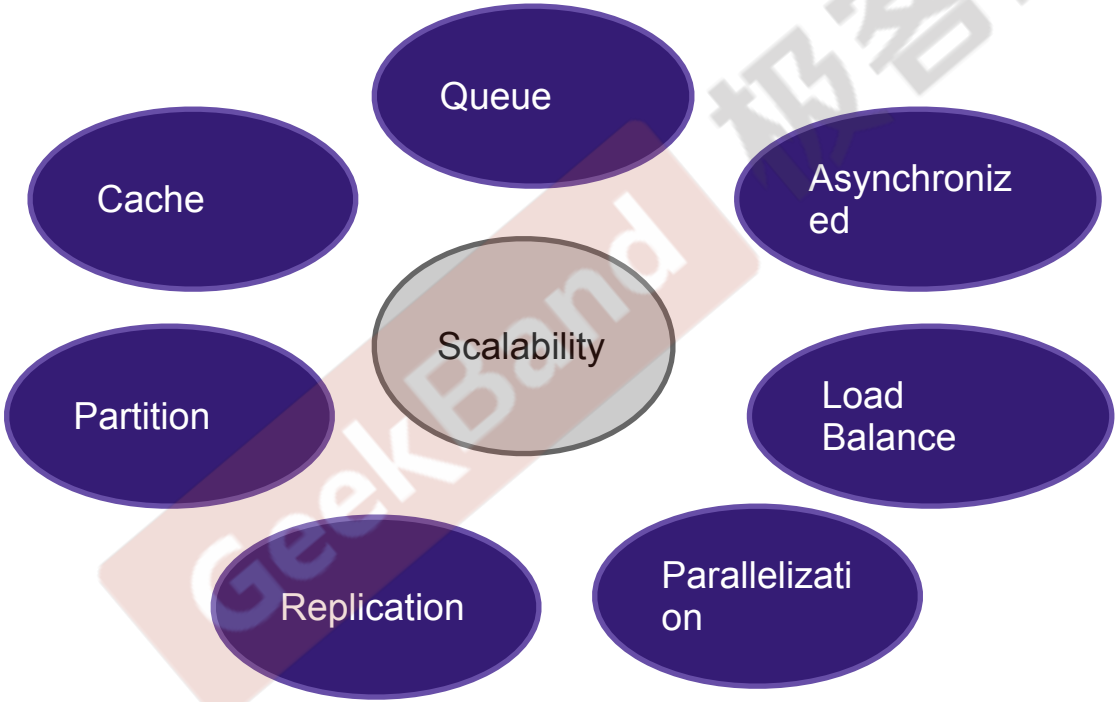
10.一个接口，多个实现。

11.加入足够的观察和调式钩子（hook）。

12.1000台服务器只需单一Master：通过Master节点来统一指挥全部的行动，但客户端和Master节点的交互很少，以免Master节点Crash，优点是，在语义上面非常清晰，但伸缩性不是非常强，一般最多只能支持上千个节点。

13.在一台机器上运行多个单位的服务：当一台机器宕机时，能缩短相应的恢复时间，并且支持细粒度的负载均衡，比如在BigTable中，一个Tablet服务器会运行多个Tablet。

Scalability Principle



Reference

- Brewer's CAP Theorem
- NOSQL Patterns
- Amazon's Dynamo
- Massive Data Mining
- Scalability for Dummies - Part 1: Clones
- The Underlying Technology of Messages
- An Unorthodox Approach To Database Design : The Coming Of The Shard
- facebook design question 总结
- How do I learn building scalable systems like Twitter, FB, and LinkedIn?
- What have been Facebook's greatest technical accomplishments?

Other tricks

Checkpointing
Quorum Protocol
Gossip Protocol
Lead Election (Paxos Algo)
Byzantine Failures
Vector Clocks

System Load on Scaling