Cassandra 架构简介

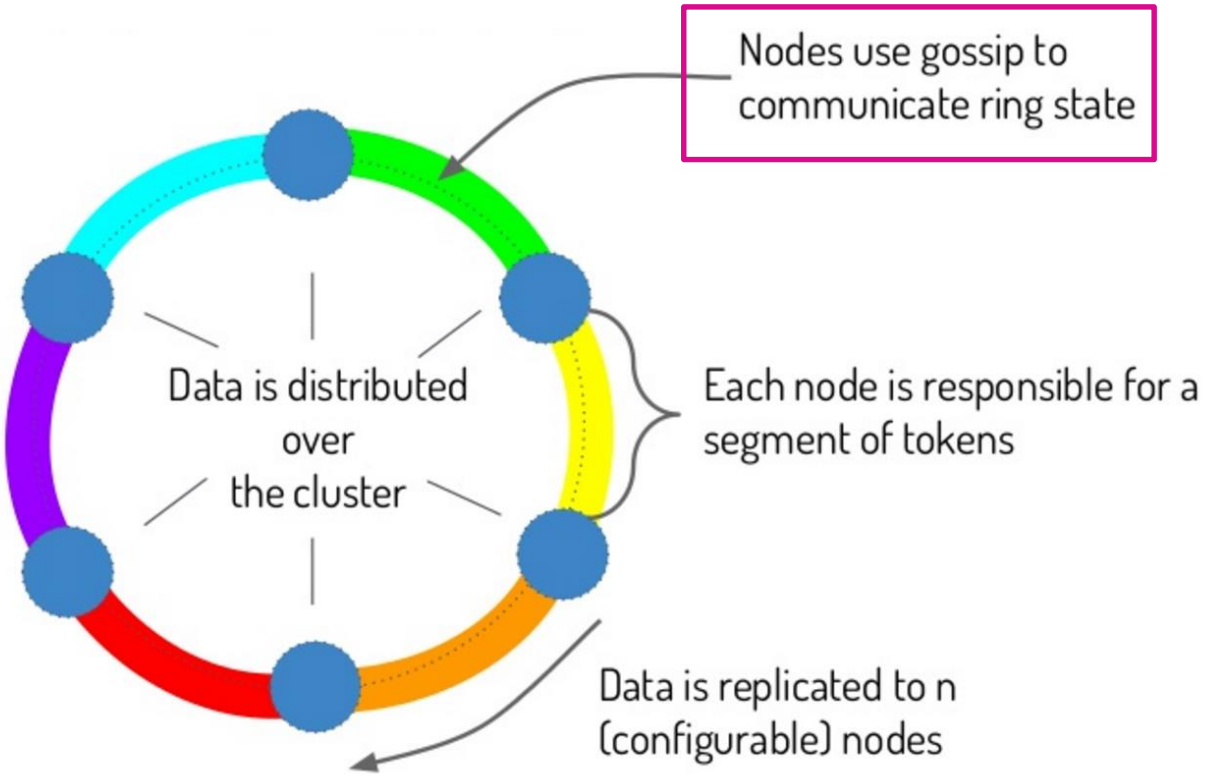
Team: Consumer SkyAid

问题：数据应该放到哪个节点？

几种架构:

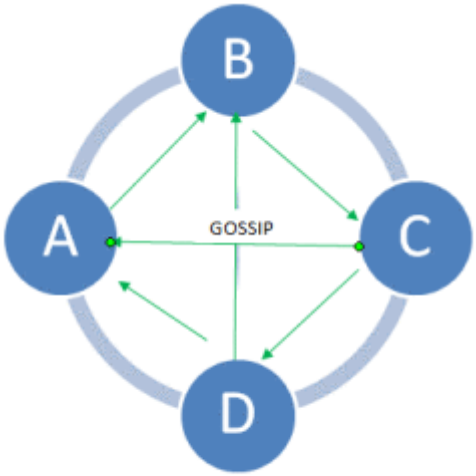
- Master-Slave
  - Example: Hbase
  - Master是大管家，认识所有slave，负责协调
- 主从复制 / 读写分离
  - Example: MongoDB / MySQL
  - Master 写， 然后复制到slave。 从slave上读
- All nodes equal
  - Example: Cassandra
  - 大家互相认识

Cassandra 架构



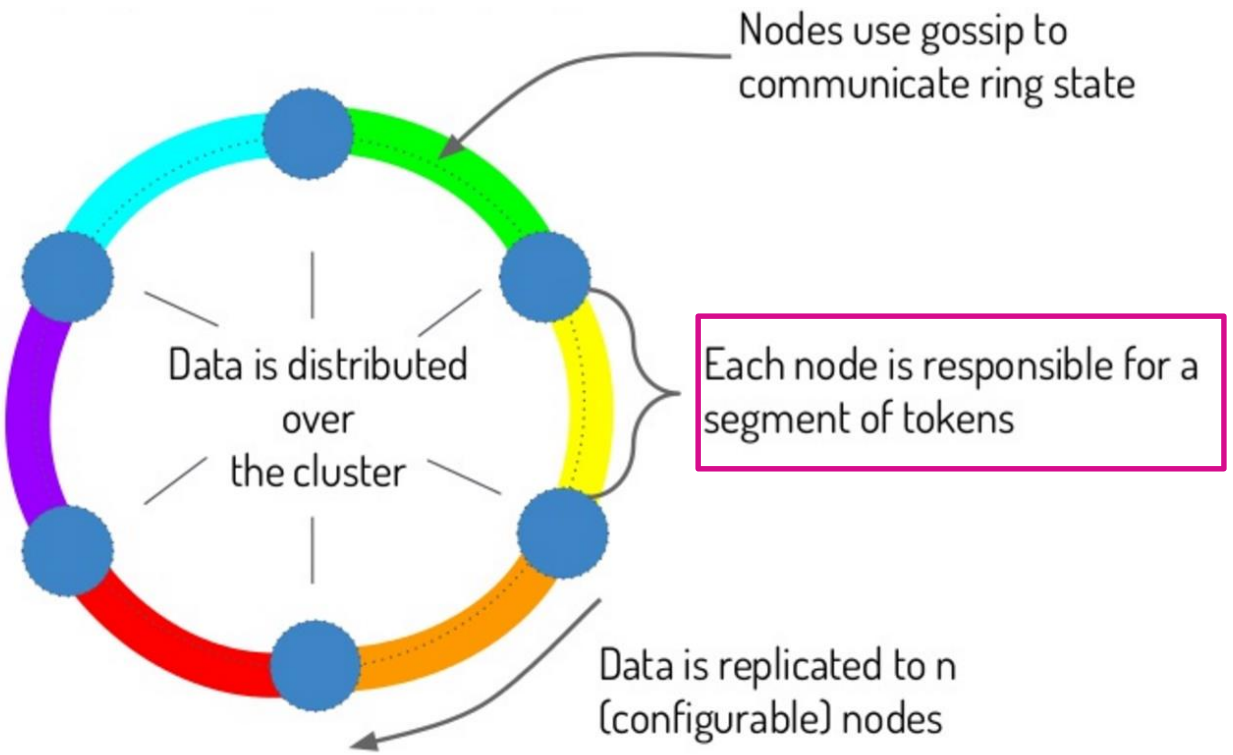
Gossip（流言蜚语）

- 作用：
  - 节点之间互相交换信息的通信协议
  - 每个节点都有其他所有节点的消息
    - Live or not / token range / …
    - 多集群？ 目前不太清楚。。。
- P2P
  - 每秒最多3个节点



PEER TO PEER DISTRIBUTION MODEL OF CASSANDRA

Cassandra 架构

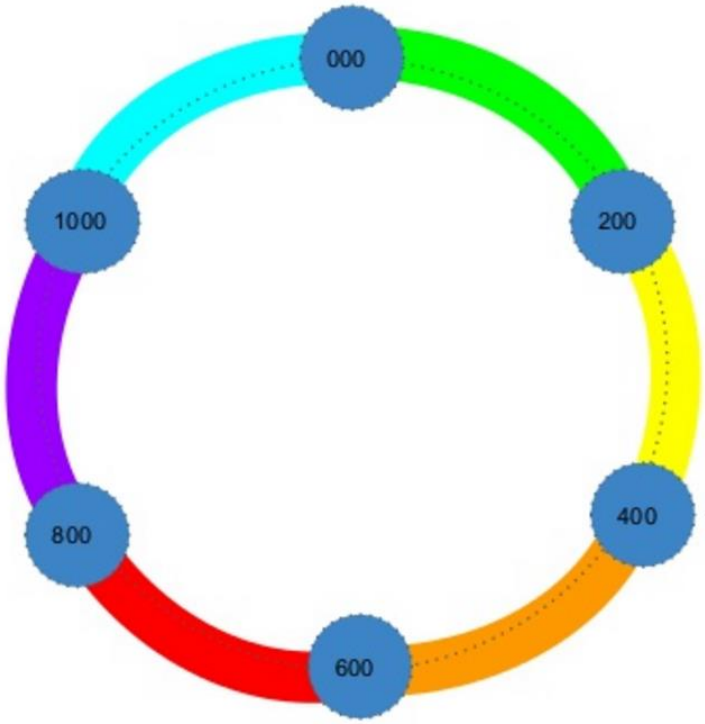


Token (令牌)

- 作用：
  - Row key / Partition key
  - Cassandra通过计算token来确定数据所应该在的位置
- Murmur3Hash
  - 大多数NoSQL都使用这个算法来计算hash

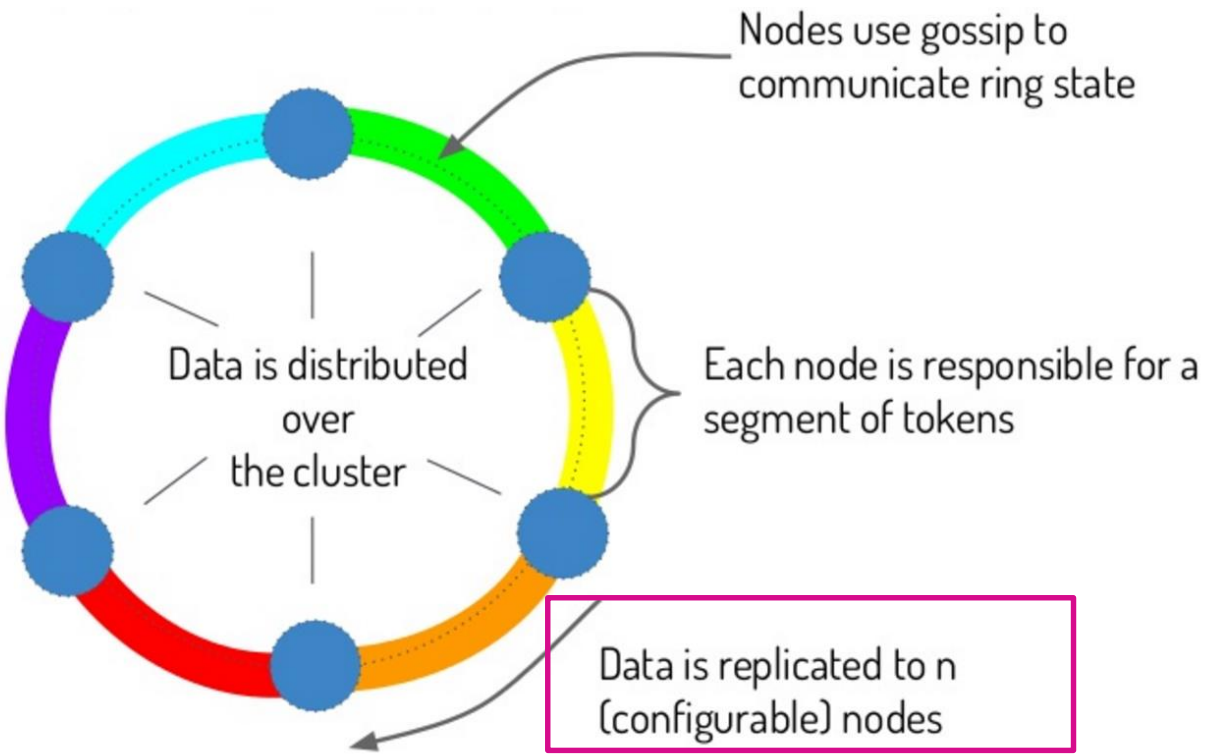
举例：

maasg	1441709070	working on my presentation



Murmur3Hash(“maasg”) = 451

Cassandra 架构



Replica strategy (复制策略)

- SimpleStrategy：
  - 单数据中心

- NetworkTopologyStrategy
  - 多DC

举例：

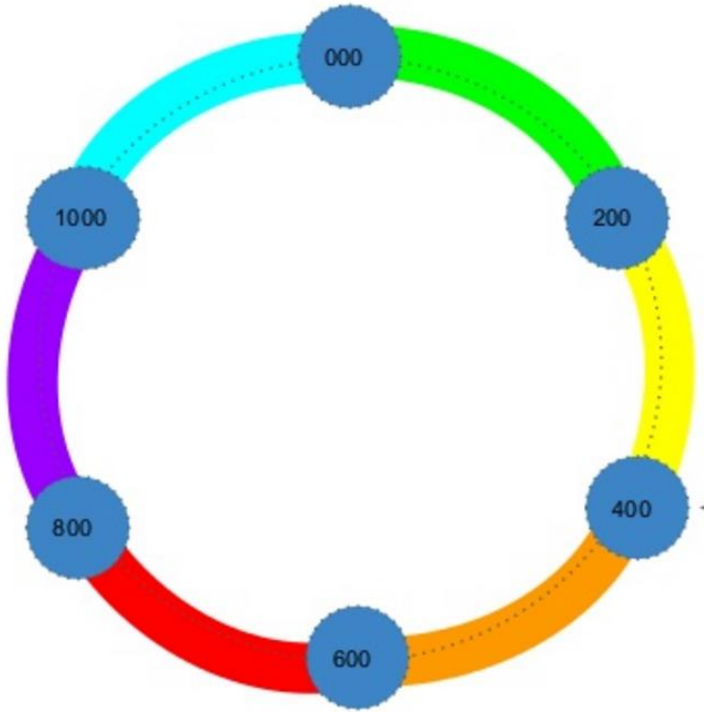
```
CREATE KEYSPACE Excelsior WITH REPLICATION =
{
    'class' : 'SimpleStrategy',
    'replication_factor' : 3
};
```

```
CREATE KEYSPACE Excelsior WITH REPLICATION =
{
    'class' :  'NetworkTopologyStrategy',
     'dc1' : 3,  'dc2' : 2
};
```

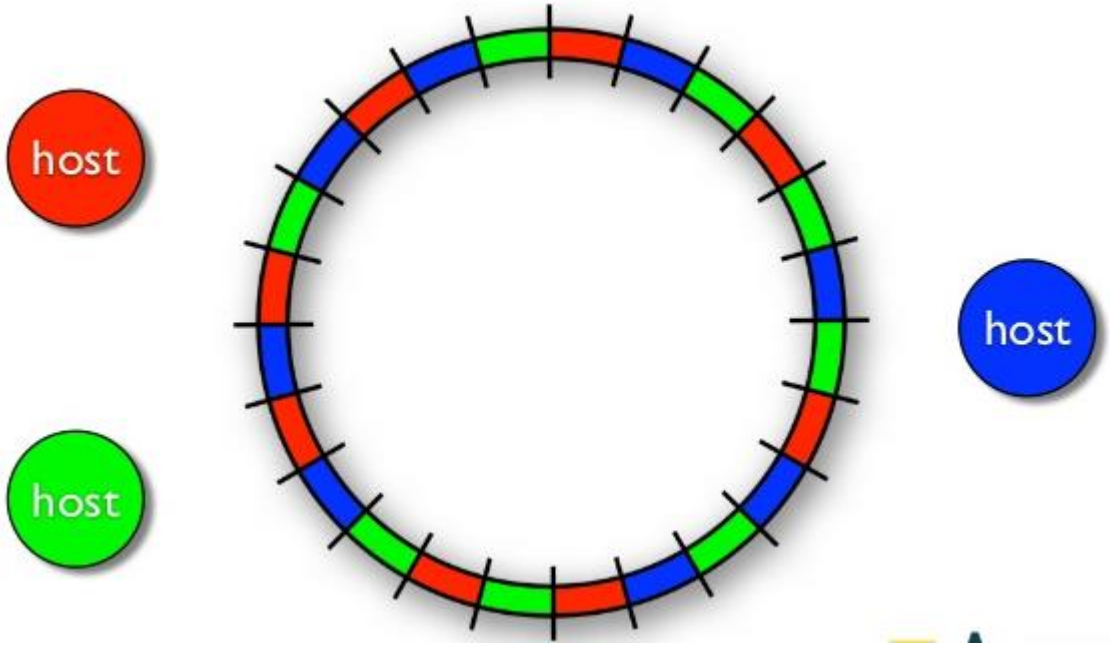
问题：有新节点加入的时候，数据流动？

数据依次流动么？

- 依次流动非常低效



Vnode（virtual nodes）



Vnode

- 每个节点拥有较多不连续的hash值范围

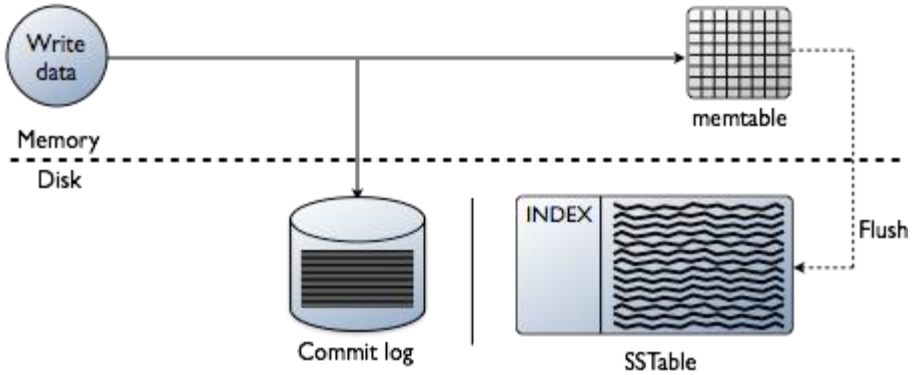
- 作用：
  - 进一步解决数据倾斜
  - 加速新节点的融入速度

- 配置: num_tokens: 256
  - 默认每个节点有256个vnode
  - 如果机器性能不足，可以减少vnode数量

问题：数据是如何写到磁盘的？

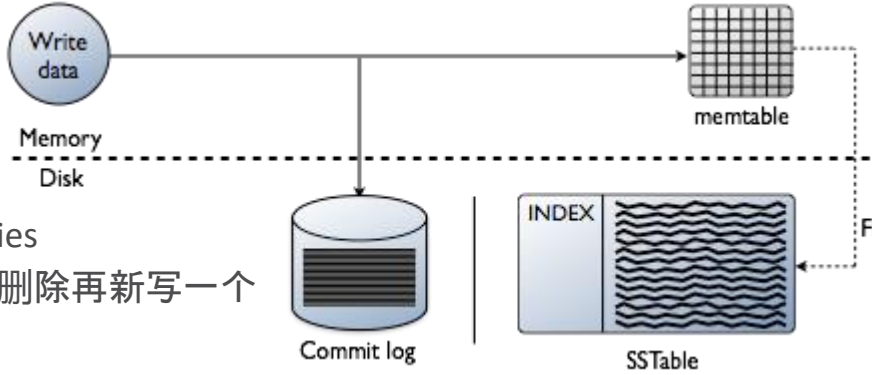
数据写入的过程

- Logging data in the commit log
- Writing data to the memtable
- Flushing data from the memtable
- Storing data on disk in SSTables

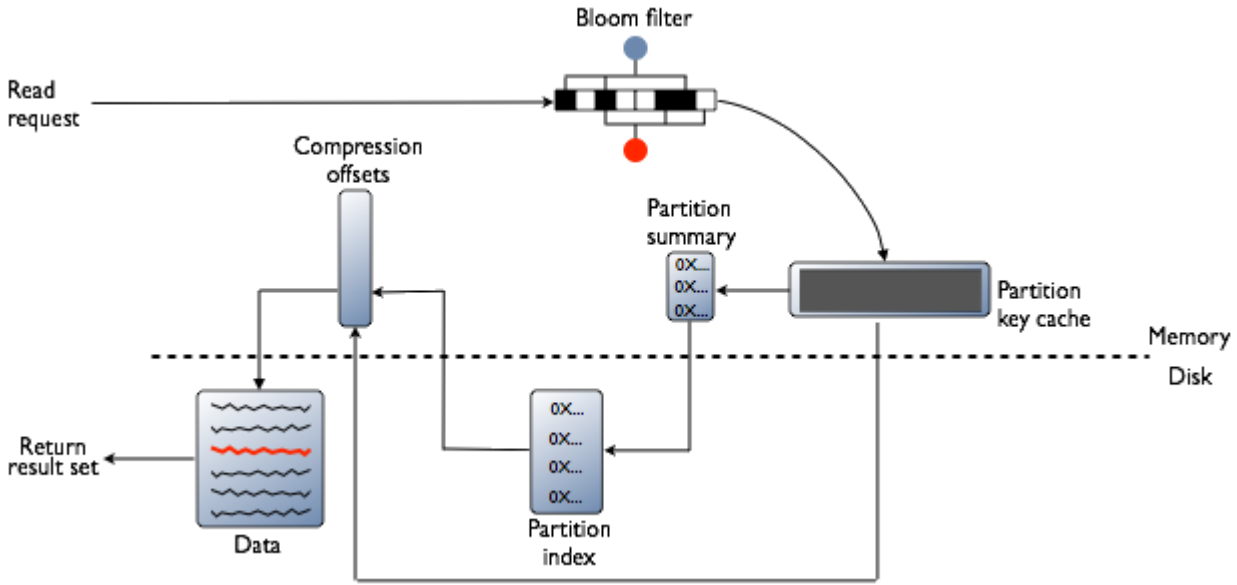


过程详解

- 首先会写入Memtable
  - Memtable：内存之中的数据结构
- 为了能够持久化与当机恢复，会同时写入CommitLog
  - 对应的配置：commitlog_directory
- 每当Memtable的数据达到一定条件时将数据Flush到SSTable
  - 条件在配置文件之中定义
    - memtable_heap_space_in_mb
    - memtable_offheap_space_in_mb
  - SSTable:
    - 真正存储到了硬盘之上：data_file_directories
    - SSTable是不可变的，每次会将SSTable完全删除再新写一个
- Flush之后，CommitLog会被自动删除



读数据的过程



其他过程::

- 更新数据的过程
- 删除数据的过程
- ooo

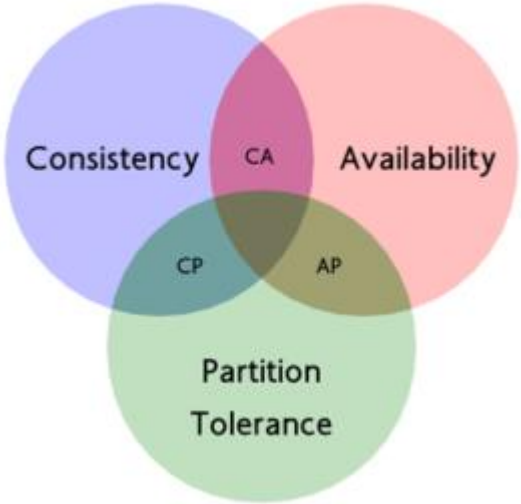
- 请阅读官网文档

存储引擎

- LSM Tree
  - MySQL 常用的是B-Tree
  - Wiki link
- No Read-Before-Write
- 普通消费者级别的SSD就能工作得很好了

数据一致性

- CAP 理论之中的AP数据库
  - 最终一致性



常见一致性配置

Level	Description	Usage
All	集群之中的所有节点	最高一致性 最低可用性
QUORUM	半数以上的节点，公式如下： quorum = (sum_of_replication_factors / 2) + 1	
ONE	1个节点	最低一致性 最高可用性
TWO / THREE	与ONE类似	
Local_*	表示local datacenter only	
ANY	写入Only，个人不推荐 表示：写入之后不管成功与否	