存储系统若干问题

刘绍辉

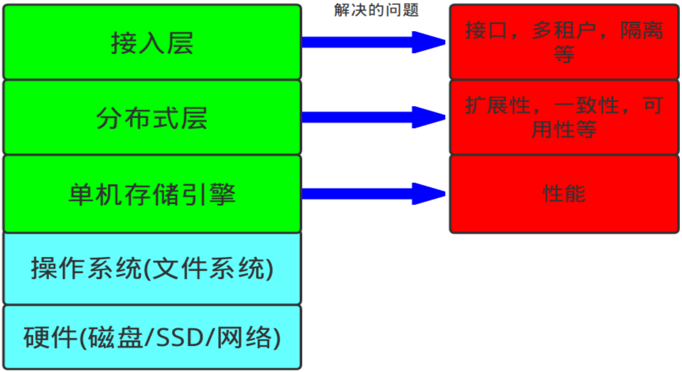
小米云存储组

May 20, 2016

目标

- 介绍存储系统基本层次和结构
- 核心问题取舍和实现方法

存储系统



解决的问题

接入层

接口，多租户，隔离等

分布式层

扩展性，一致性，可用性等

单机存储引擎

性能

操作系统(文件系统)

硬件(磁盘/SSD/网络)

提纲

1. 硬件
2. 操作系统
3. 单机存储引擎
4. 分布式层
5. 接入层

硬件

- 机械硬盘
- 固态硬盘(SSD)
- 内存(RAM is the new disk)
- 万兆网卡和网络拓扑

Traditional hard disk drive

Solid state hard drive

重要的数字

Numbers that every computer engineer should know

- Mutex lock/unlock: 17 ns
- Main memory reference: 100 ns
- Compress 1K bytes with Zippy: 10,000 ns
- SSD random read: 16 us
- Hard Disk seek: 8 ms
- Round trip within same datacenter: 300 us
- Round trip across zones in same region: 800 us
- Read 1 MB sequentially from memory: 12us
- Read 1 MB sequentially from SSD: 200 us
- Read 1 MB sequentially from disk: 2ms

http://www.eecs.berkeley.edu/ rcs/research/interactive_latency.html

测试工具

- 磁盘：fio
- 网络：ping/iperf
- mysql工具: pt-mysql-summary

提纲

1. 硬件

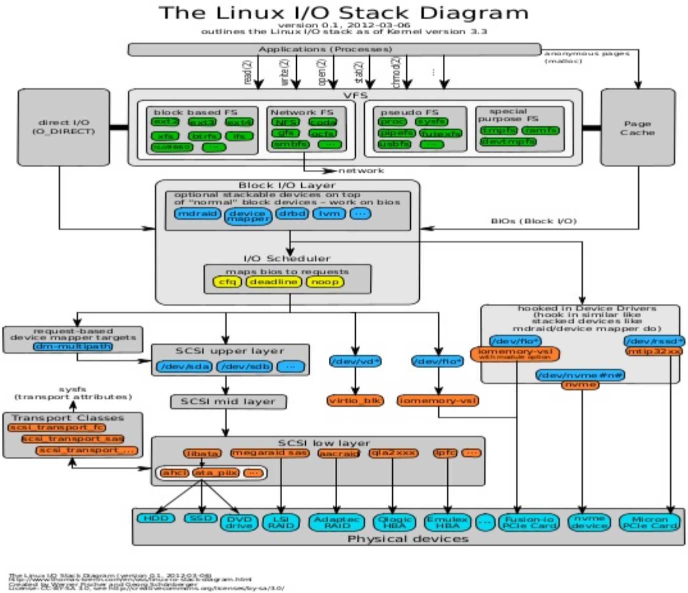
2. 操作系统

3. 单机存储引擎

4. 分布式层

5. 接入层

文件系统



The Linux I/O Stack Diagram

version 0.1, 2012-03-06
outlines the Linux I/O stack as of Kernel version 3.3

The Linux I/O Stack Diagram (version 0.1, 2012-03-06)
http://www.thomas-krenn.com/en/oss/linux-io-stack-diagram.html
Created by Werner Fischer and Georg Schönberger
License: CC-BY-SA 3.0, see http://creativecommons.org/licenses/by-sa/3.0/

数据持久性

- write: Only write data to OS page cache
- sync: flush all dirty buffers to disk
- fsync: flush all blocks that belong to a specific open file to disk
- fdatasync: flush all data blocks(no inode block) that belong to a specific open file to disk

观测工具

- 磁盘：iostat/iotop
- 网络：iftop
- sysdig systemstap

吐血推荐：Red Hat Enterprise Linux Performance Tuning Guide

提纲

1. 硬件

2. 操作系统

3. 单机存储引擎

4. 分布式层

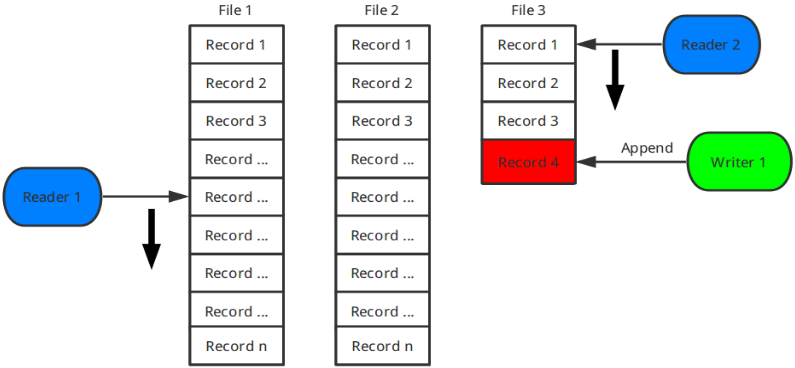
5. 接入层

单机存储引擎

- 文件: HDFS/Kafka datanode
- Hash Table (BitCask/Memcached/Redis)
- B+ Tree (MySQL innoDB engine)
- LSM Tree (LevelDB/RocksDB)

http://www.xaprb.com/blog/2015/04/02/state-of-the-storage-engine/

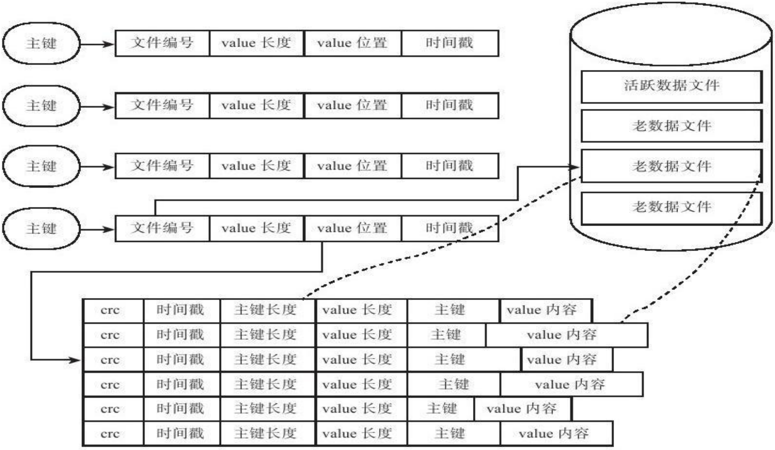
File: 顺序读写



HDFS/Kafka datanode

Hash Table: 随机读写

BitCask



主键	文件编号	value 长度	value 位置	时间戳
主键	文件编号	value 长度	value 位置	时间戳
主键	文件编号	value 长度	value 位置	时间戳
主键	文件编号	value 长度	value 位置	时间戳

活跃数据文件

老数据文件

老数据文件

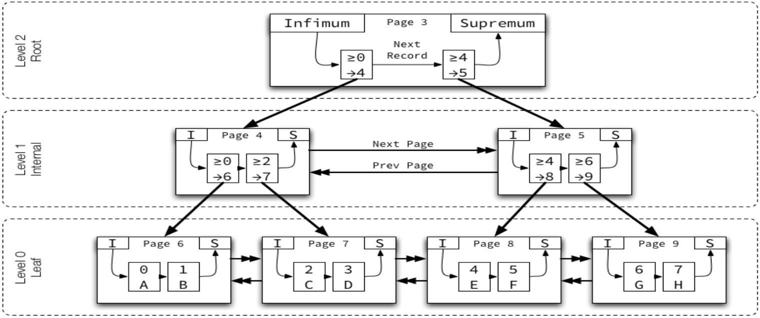
老数据文件

crc	时间戳	主键长度	value 长度	主键	value 内容
crc	时间戳	主键长度	value 长度	主键	value 内容
crc	时间戳	主键长度	value 长度	主键	value 内容
crc	时间戳	主键长度	value 长度	主键	value 内容
crc	时间戳	主键长度	value 长度	主键	value 内容
crc	时间戳	主键长度	value 长度	主键	value 内容

面向小文件

B+ Tree: Scan

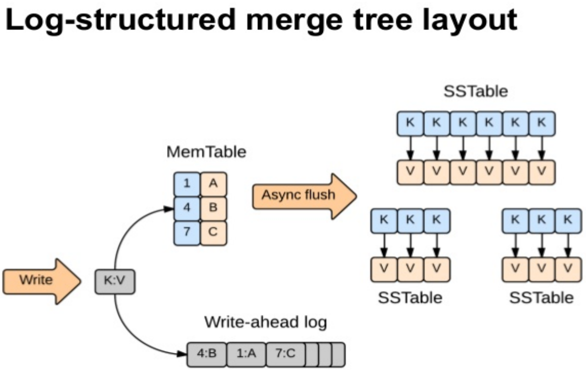
B+Tree Structure



Levels are numbered starting from 0 at the leaf pages, incrementing up the tree.
Pages on each level are doubly-linked with previous and next pointers in ascending order by key.
Records within a page are singly-linked with a next pointer in ascending order by key.
Infimum represents a value lower than any key on the page, and is always the first record in the singly-linked list of records.
Supremum represents a value higher than any key on the page, and is always the last record in the singly-linked list of records.
Non-leaf pages contain the minimum key of the child page and the child page number, called a "node pointer".

面向读

LSM Tree: Scan



Log-structured merge tree layout

面向写，问题：写放大，标记删除

单机存储引擎

为什么需要这么多不同存储引擎?
是否有一个引擎满足所有需求?

提纲

1. 硬件

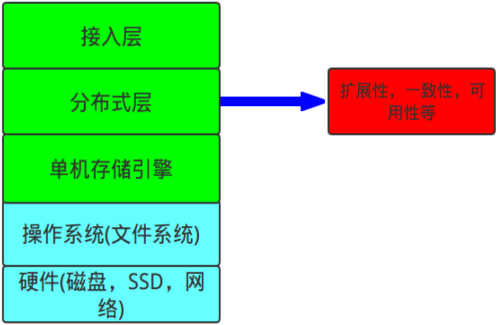
2. 操作系统

3. 单机存储引擎

4. 分布式层

5. 接入层

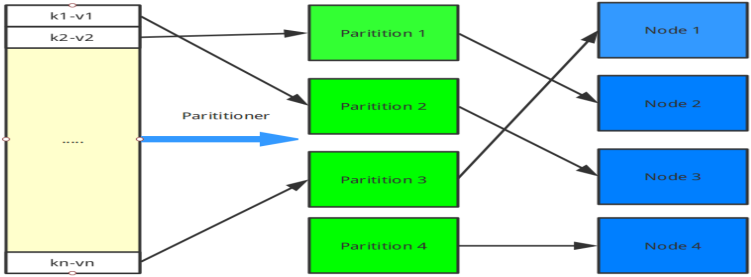
分布式层



扩展性

数据

- File : (Block1, Block2, ...)
- Table : (Range1, Range2, ...) / (Hash1, Hash2, ...)
- Queue : (parition1, parition2, ...)

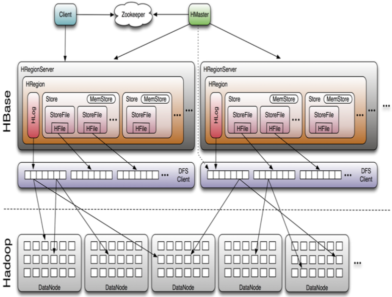


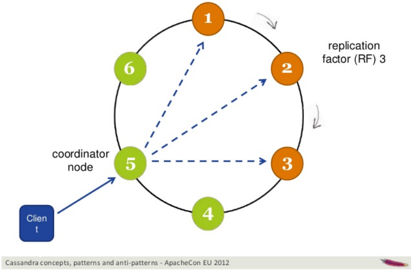
扩展性

元数据

- parition信息
- 路由信息等等

中心化(Master-Salve) vs 去中心化(P2P)





Cassandra concepts, patterns and anti-patterns - ApacheCon EU 2012

Master-Worker优势：结构简单，容易控制

问题：单点问题

一致性和可用性

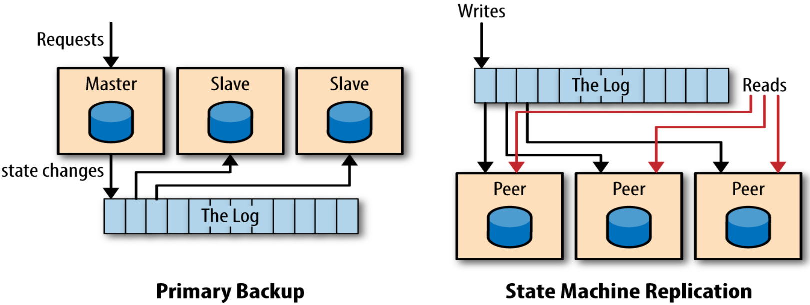
在保证强一致的情况下如何保证更高的可用性。

一致性分类：

- 强一致: 写成功返回之后马上可以读到最新结果
- 弱一致: 最终一致性/因果一致性/顺序一致性等等

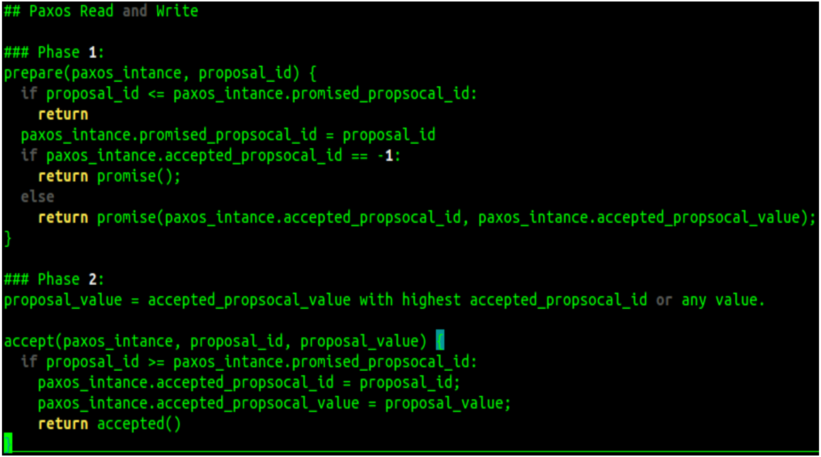
一致性和可用性

- Primary-backup协议(Zab协议)
- state machine replication(Paxos/Raft/Pacific)



Paxos

核心：尊重之前的决定，并帮忙他完成



```
## Paxos Read and Write

### Phase 1:
prepare(paxos_intance, proposal_id) {
  if proposal_id <= paxos_intance.promised_propsocal_id:
    return
  paxos_intance.promised_propsocal_id = proposal_id
  if paxos_intance.accepted_propsocal_id == -1:
    return promise();
  else
    return promise(paxos_intance.accepted_propsocal_id, paxos_intance.accepted_propsocal_value);
}

### Phase 2:
proposal_value = accepted_propsocal_value with highest accepted_propsocal_id or any value.

accept(paxos_intance, proposal_id, proposal_value) {
  if proposal_id >= paxos_intance.promised_propsocal_id:
    paxos_intance.accepted_propsocal_id = proposal_id;
    paxos_intance.accepted_propsocal_value = proposal_value;
    return accepted()
}
```

其他问题

- Master的高可用(Paxos协议/Zookeeper + HDFS)
- 跨机房备份和容灾
- 多机房服务
- 分布式事务

提纲

1. 硬件
2. 操作系统
3. 单机存储引擎
4. 分布式层
5. 接入层

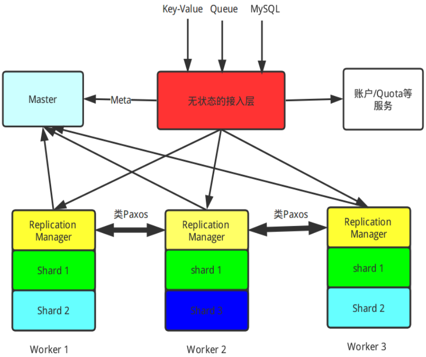
接入层

服务化

- 接口（给用户的承诺）：File/Queue/Key-Value/Table/SQL/Key-Objects
- rpc or rest
- 多租户：用户认证和权限检查
- 隔离：Quota限制
- 审计和计费

理想状态

- Master-Worker架构
- 水平扩展，Shard策略可配置
- 类Paxos副本复制协议
- 支持多种存储引擎
- 接口：File/Queue/Key-Value/Table/SQL/Key-Objects
- 多租户和隔离



参考

- MapReduce: Simplified Data Processing on Large Clusters
- The Google File System
- Bigtable: A Distributed Storage System for Structured Data
- Megastore: Providing Scalable, Highly Available Storage for Interactive Services
- Spanner: Google's Globally-Distributed Database
- F1 - The Fault-Tolerant Distributed RDBMS Supporting Google's Ad Business

开源项目

- TiDB：https://github.com/pingcap/tidb
- cockroachdb: https://github.com/cockroachdb/cockroach

图书

- <<大规模分布式存储系统：原理解析与架构实战>> 杨传辉
- <<Distributed systems: for fun and profit>> https://github.com/mixu/distsysbook