

带你认识储存&数据库

作者：徐子霖

时间：2023.05.23

参考：字节第五届后端青训营

资源：<https://bytedance.feishu.cn/file/boxcn7QgwoNh57Tjg6HmxKMhMHf>

课程目录

- 1. 经典案例
- 2. 存储 & 数据库简介
- 3. 主流产品剖析
- 4. 新技术演进

1. 经典案例

数据的产生

某天，小明同学下载了新的app并且注册了账号

用户名	密码	密码提示问题
小明	helloworld	coding

就这样数据产生了，向app的后端服务器飞奔而去

数据的流动

一条用户的注册数据 --> 后端服务器 --> 数据库系统（才算持久化下来） --> 其他系统（不展开说）

为什么要持久化呢？

我们需要带有记忆的，如果知识在内存时，那么每次用户都要注册。

数据的持久化

- 校验数据的合法性
 - 小明是否已经存在
- 修改内存
 - 用高效的数据结构组织数据
 - 形成一个准备可以持久化的结果
- 写入存储介质
 - 以寿命&性能友好的方式写入硬件

潜在的问题

- 数据库怎么保证数据不丢？
- 数据库怎么处理多人同时修改的问题？
- 为什么用数据库，除了数据库还能存到别的存储系统吗？

- 答案是可以的
- 数据库只能处理结构化数据吗?
 - 答案是不一定
- 有哪些操作数据库的方式，要用什么编程语言？

2. 存储&数据库简介

什么是存储系统？

- 一个提供了读写、控制类接口，能够安全有效的把数据持久化的软件，就可以成为存储系统

其实就是用户和介质，但是可能也要和内存打交道，设计更好的性能。也有可能关注到网络编程。

系统特点

- 作为后端软件的底座，**性能敏感**
 - 很多后端架构里面，都是持久化、有状态的服务，那就要设计大量频繁并发的服务，所以我们需要超高设计
- 存储系统代码，既简单又复杂
 - 对性能要求高，在I/O（读写）路径上代码一定不能很复杂或者很多分支，搞得系统性能很差
 - 在非I/O路径上，考虑很多异常情况，比如硬件损坏
- 存储系统软件架构，容易受**硬件影响**
 - 存储系统往下，是直接和硬件打交道，比如磁盘。只要硬件发生变革，那么我们软件也需要做一定的变革，甚至不惜推倒重来。

2.1 存储器层级结构

Computer Memory Hierarchy

金字塔尖：容量极小，超高性能访问

金字塔底部：容量大，读写速度非常慢，访问方式不友好



想法：在我们的中间层，能不能存在一个坚固，持久化的新型存储器呢？

- Persistent Memory

数据怎么从应用到存储介质呢

- 如下链路图

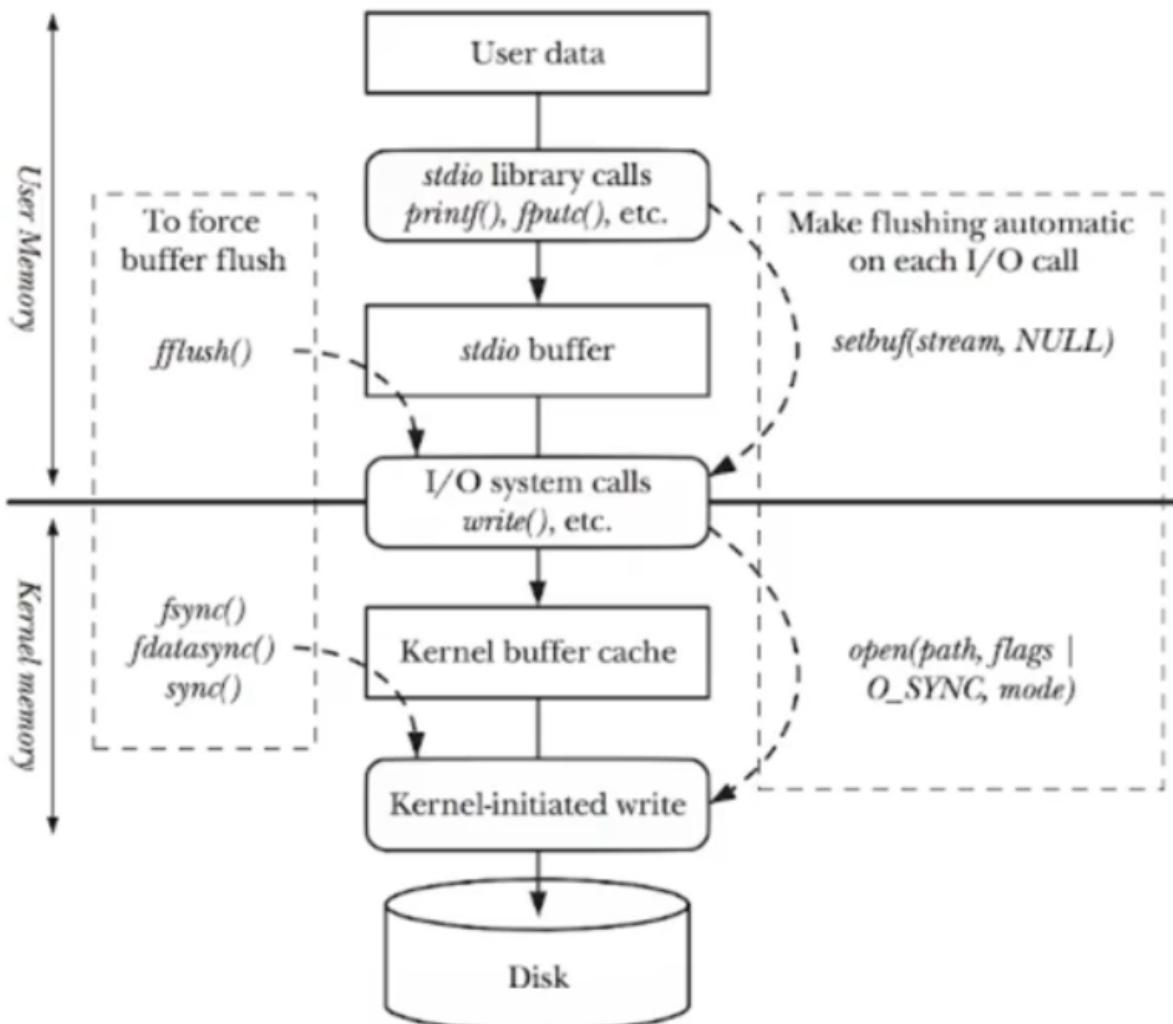


Figure 13-1: Summary of I/O buffering

@稀土掘金技术社区

缓存很重要，贯穿整个存储体系

- 存储介质访问有可能不是特别友好，比如有的硬件需要一个byte一个byte写入，但是软件不会这样写
- 软件去实现cache就很重要，帮助我们和硬件友好的方式去和硬件交互。
- 软件也是分层的，还能跨软件层

拷贝 是非常昂贵的

- 消耗CPU的操作

Disk 代指任何硬件， 需要抽象统一的接入层

RAID技术

Q： 单机存储系统怎么做到高性能，高性价比，高可靠性

A： RAID

RAIDO

- 多款磁盘简单组合
- 数据条带化储存，提高磁盘带宽
- 没有额外的容错设计
 - 没有对数据做冗余，备份。只是写进去会被切割

RAID1

- 一款磁盘对应一块额外镜像盘（克隆）
- 真实空间利用率50%
- 容错能力强

RAID 0 + 1

- 结合了0 和 1
- 真实空间利用率百分之50
- 容错能力强，写入带宽好

难道数据库和存储系统不一样么？

- 关系型数据库
- 非关系型数据库

2.2 数据库 - 概览

关系 (relation) 是什么？

- 关系 = 集合= 任意元素组成的若干有序偶对反应了事物间的关系

关系代数 = 对关系做运算的抽象查询语言

- 交， 并， 笛卡尔积。。。。。

SQL = 一种DSL = 方便人类阅读的关系代数表达式

- 去操作关系时用到的特定语言
- 人说人话，狗说狗话

2.2 数据库 - 关系型数据库的特点

关系型数据库是存储系统，但是在存储之外，又发展出其他能力

- 结构化数据友好
- 支持事务（ACID）
- 支持复杂查询语言

2.2 数据库 - 非关系型数据库特点

非关系型数据库也是存储系统，但是一般不要求严格的结构化

- 半结构化数据友好
- 可能支持事务
- 可能支持复杂查询语言

2.3 数据库 vs 经典存储 - 结构化数据管理

一条用户注册数据

```
{
  "user_name": "xiaoming",
  "password": "helloworld",
  "password_hint": "coding",
  ....
}
```

写入关系型数据库，以**表形式**管理，很简单自然

use_name	password
xiaoming	helloworld

写入文件，自行定义管理结构



第一个 **4 byte** 是这条数据的长度， 第二个就是描述第一个字段多长， 第四个就是描述**password**有多长

- 很痛苦，无时无刻都在和byte打交道，做byte级别运算

2.3 数据库 vs 经典存储 - 事务能力

ACID

- atomicity：事务内操作要么全做，要么不做
- consistency：事务执行前后，数据状态是一致的
 - 转账：A的账上有1k，B没钱。A给B转500，事务结束后。A账户有500，B有500
 - 不一致：A给B转钱，B没收到。数据库状态不一致了
- isolation：可以隔离多个并发事务，避免影响
 - 也取决于隔离级别
- durability：事务一旦提交成功，数据保证持久性

经典存储没有事务这种概念

2.3 数据库 vs 经典存储 - 复杂的查询能力

Q : 写入数据之后，想做很复杂的查询怎么办？

Example : 请查询出名字以xiao开头，且密码提示问题小于10个字的人，并按性别分组统计人数

```

Select gender, count(*) from user
  where user_name like 'xiao%'
    and len(password_hint) < 10
  group by gender;

```

VS

```

for each data {
  if (user_name ..... && password_hint .....){
    mark in list
  }
}
for each in marked_list{
  if (gender == .....){}
}

```

@稀土掘金技术社区 30

2.4 数据库的使用方式

Everything is Domain Specific Language → maybe SQL

以SQL为例，要操作数据时，支持以下操作：

- Insert
- Update
- Select
- Delete
- Where子句
- GroupBy
- OrderBy

要对数据定义做修改时，支持以下操作：

- Create user
- Create database
- Create table
- Alter table
-

@稀土掘金技术社区

3. 主流产品剖析

- 3.1 单机存储
- 3.2 分布式存储
- 3.3 单机关系型数据库
- 3.4 单机非关系型数据库
- 3.5 分布式数据库

3.1 单机存储 -- 概览

单机存储 = 单个计算机节点上的存储软件系统，一般不涉及网络交互

3.1 单机存储 — 本地文件系统

Linux经典哲学：一切皆文件

文件系统的管理单元：文件

文件系统接口：文件系统繁多，如Ext2/3/4, sysfs, rootfs等，但都遵循VFS的统一抽象接口

Linux文件系统的两大数据结构：**Index Node & Directory Entry**

Index Node



记录文件元数据，如id、大小、权限、磁盘位置等

inode是一个文件的唯一标识，会被存储到磁盘上

inode的总数在格式化文件系统时就固定了

Directory Entry



记录文件名、inode指针，层级关系(parent)等

dentry是内存结构，与inode的关系是N:1(hardlink的实现)

@稀土掘金技术社区

一个inode对应到一个文件。一个文件夹下面有一个a文件，看另一个文件夹也有，他们的inode不一样。

D Entry：不会被持久化存储

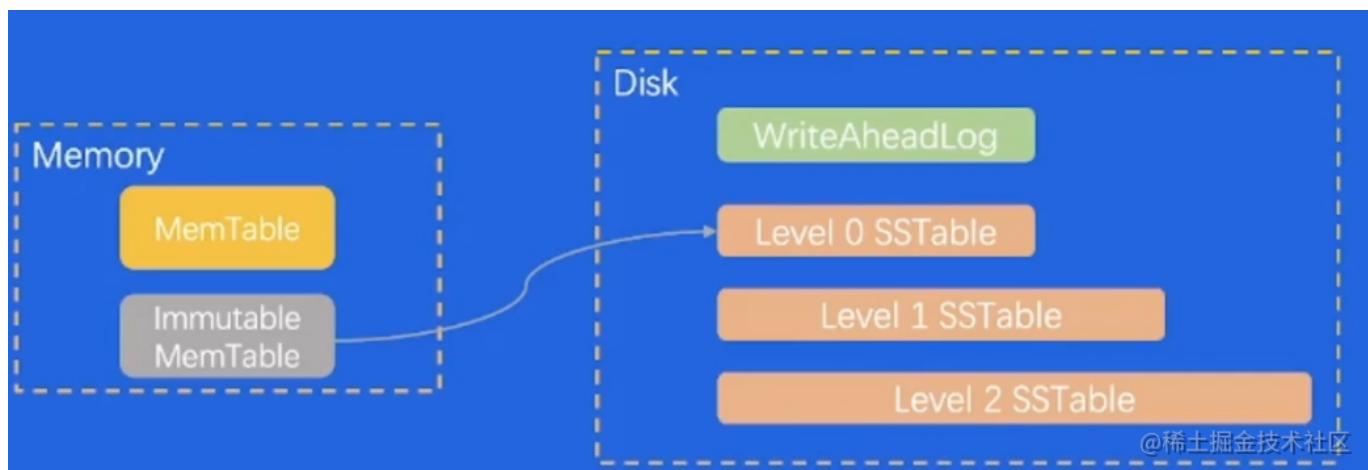
3.1 单机存储 - key value存储

世间一切皆key-value -- key是你身份证，value是你的内涵

常见使用方式：put, get

参见数据结构：LSM-tree，牺牲读性能，追求写入性能

拳头产品： RocksDB



3.2 分布式存储 -- 概览

分布式存储 = 在单机存储基础上实现了分布式协议，涉及大量网络交互

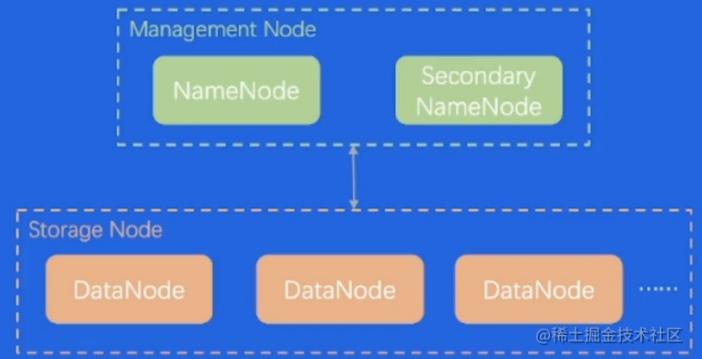
3.2 分布式存储 — HDFS

HDFS：堪称大数据时代的基石

时代背景：专用的高级硬件很贵，同时数据存海量很大，要求超高吞吐

HDFS核心特点：

- 支持海量数据存储
- 高容错性
- 弱POSIX语义
- 使用普通x86服务器，性价比高



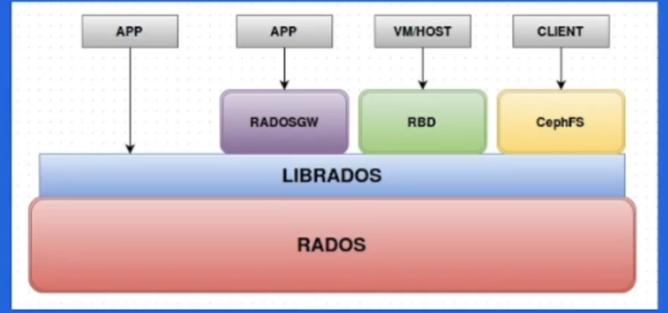
- 用一些便宜的硬件
- 在这个之上，发明了Hadoop

3.2 分布式存储 — Ceph

Ceph：开源分布式存储系统里的「万金油」

Ceph的核心特点：

- 一套系统支持对象接口、块接口、文件接口，但是一切皆对象
- 数据写入采用主备复制模型
- 数据分布模型采用CRUSH算法



3.3 单机数据库

单个计算机节点上的数据库系统 事务在单机内执行，也可能通过网络交互实现分布式事务

3.3 单机数据库 — 关系型数据库

商业产品Oracle称王，开源产品MySQL & PostgreSQL称霸



关系型数据库的通用组件：

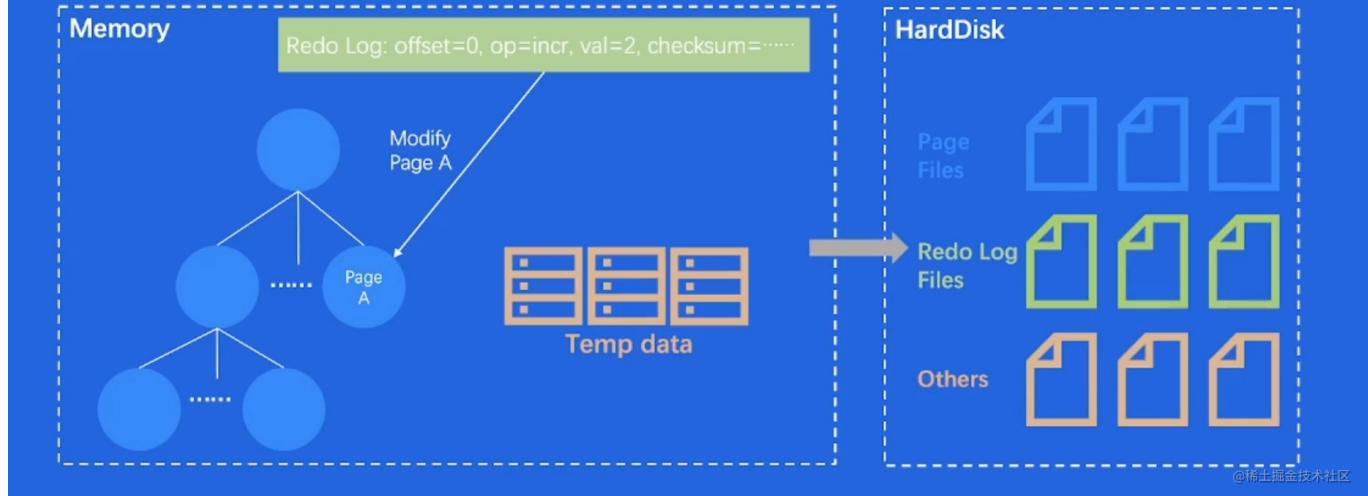
- Query Engine —— 负责解析query，生成查询计划
- Txn Manager —— 负责事务并发管理
- Lock Manager —— 负责锁相关的策略
- Storage Engine —— 负责组织内存/磁盘数据结构
- Replication —— 负责主备同步

关键内存数据结构：B-Tree、B+-Tree、LRU List等

关键磁盘数据结构：WriteAheadLog（RedoLog）、Page

@稀土掘金技术社区

3.3 单机数据库 — 关系型数据库



用户想要update我们数据，其实就是操作我们的page

redo log：保证操作不会丢失 others: 存我们的临时数据文件

3.4 单机数据库 -- 非关系型数据库

3.4 单机数据库 — 非关系型数据库

MongoDB、Redis、Elasticsearch三足鼎立

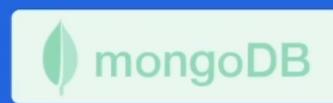
Rank				DBMS	Database Model	Score		
Apr 2022	Mar 2022	Apr 2021	Apr 2022			Apr 2022	Mar 2022	Apr 2021
1.	1.	1.	Oracle	Relational, Multi-model	Relational, Multi-model	1254.82	+3.50	-20.10
2.	2.	2.	MySQL	Relational, Multi-model	Relational, Multi-model	1204.16	+5.93	-16.53
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	Relational, Multi-model	938.46	+4.67	-69.51
4.	4.	4.	PostgreSQL	Relational, Multi-model	Relational, Multi-model	614.46	-2.47	+60.94
5.	5.	5.	MongoDB	Document, Multi-model	Document, Multi-model	483.38	-2.28	+13.41
6.	6.	7.	Redis	Key-value, Multi-model	Key-value, Multi-model	177.61	+0.85	+21.72
7.	8.	8.	Elasticsearch	Search engine, Multi-model	Search engine, Multi-model	160.83	+0.89	+8.66
8.	7.	6.	IBM Db2	Relational, Multi-model	Relational, Multi-model	160.46	-1.69	+2.68
9.	9.	10.	Microsoft Access	Relational	Relational	142.78	+7.36	+26.05
10.	10.	9.	SQLite	Relational	Relational	132.80	+0.62	+7.74

Reference: [4]

@稀土掘金技术社区

- elastic search : 基于文档，对文档操作
- mongoDB: 灵活
- redis: 数据结构丰富 (没有尝试去支持DSL)

3.4 单机数据库 — 非关系型数据库



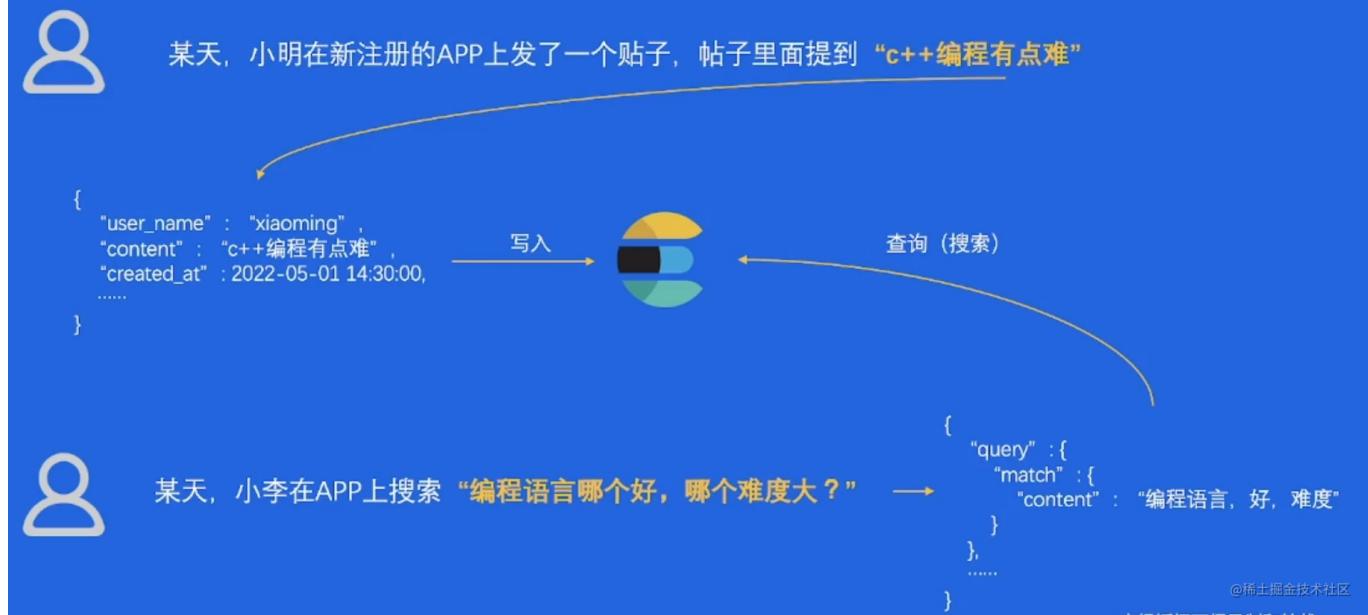
- 面向「文档」存储
- 文档可序列化成JSON，支持嵌套
- 存在「index」， index = 文档的集合
- 存储和构建索引能力依赖Lucene引擎
- 实现了大量搜索数据结构 & 算法
- 支持RESTFUL API，也支持弱SQL交互

- 面向「文档」存储
- 文档可序列化成JSON/BSON，支持嵌套
- 存在「collection」， collection = 文档的集合
- 存储和构建索引能力依赖wiredTiger引擎
- 4.0后开始支持事务（多文档、跨分片多文档等）
- 常用client/SDK交互，可通过插件转译支持弱SQL

- 数据结构丰富 (hash表、set、zset、list)
- C语言实现，超高性能
- 主要基于内存，但支持AOF/RDB持久化
- 常用redis-cli/多语言SDK交互

@稀土掘金技术社区

3.4 单机数据库 — Elasticsearch使用案例



跟RDVMS相比，ES天然能做模糊搜索，还能自动算出关联程度

3.5 从单机到分布式数据库

4. 新技术演进

- 软件架构变更
 - bypass OS kernel
- AI增强
 - 智能存储格式转换
- 新硬件革命
 - 存储介质变更
 - 计算单元变更
 - 网络硬件变更