

D2 TERMINAL

手淘移动存储内核新实践

刘韩松

阿里巴巴 高级技术专家



目录

01

移动端存储体系概况

- Flash存储的原理与特性
- Flash Translation Layer (FTL)
- 文件系统针对Flash存储的设计与优化 (F2FS)

02

移动端存储模型及读写放大分析

- b-tree: SQLite
- LSM: LevelDB
- 文档: Realm

03

手淘移动存储内核的一些实践

- Disk Layout
- 冷热分区
- 写放大抑制
- 事务设计
- 内存压缩
- 新式哈希表的应用

第十七届

D2 终端技术大会

2022.12.17-12.18

前端 & 客户端

合作伙伴

阿里云开发者社区
ALIBABA CLOUD DEVELOPER COMMUNITY

 大淘宝技术
TAO TECHNOLOGY


 老司机技术
SWIFT OLDDRIVER



segmentfault 思否



 稀土掘金

 钉钉开发者

InfoQ 极客传媒

 T Salon

YOUKU 优酷



*排名不分先后顺序

扫码进入D2官网

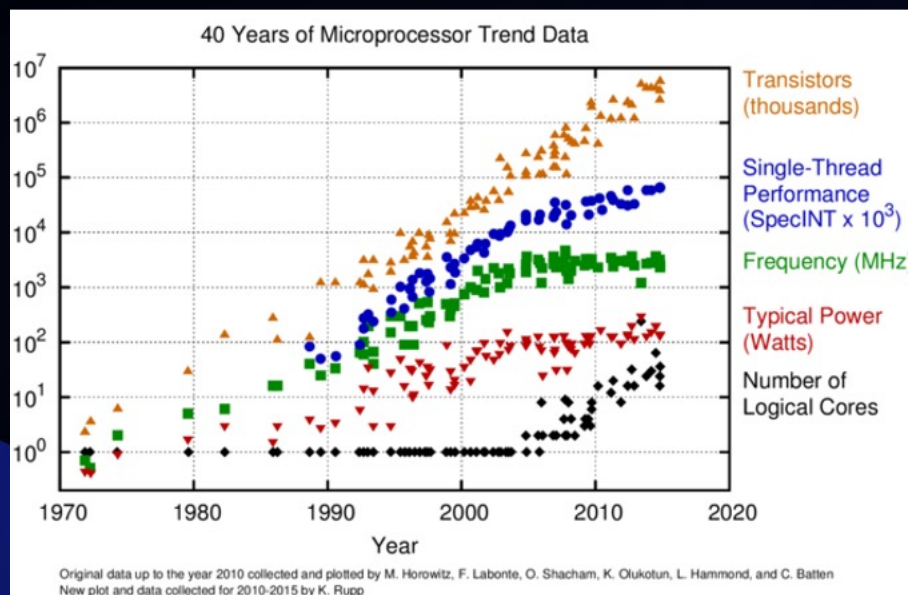
1 移动端存储体系概况

Flash存储的原理与特性

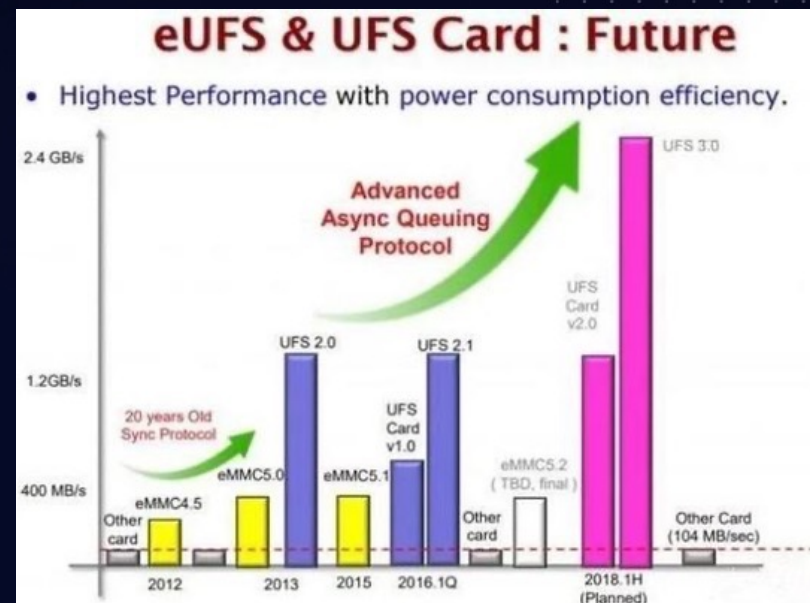
Flash Translation Layer(FTL)

文件系统针对Flash存储的设计与优化

CPU及存储标准的性能趋势



<https://github.com/karlrupp/microprocessor-trend-data>



<https://www.jedec.org/standards-documents/focus/flash/universal-flash-storage-ufs>

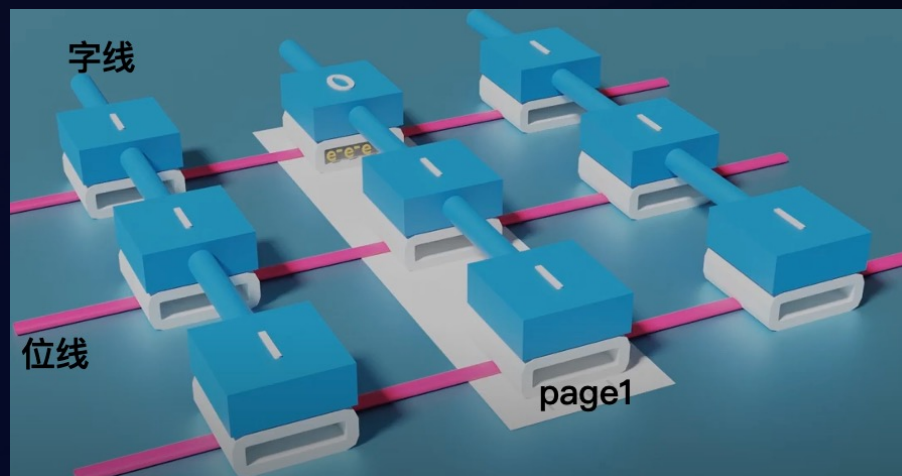
Flash存储原理

Flash存储特性

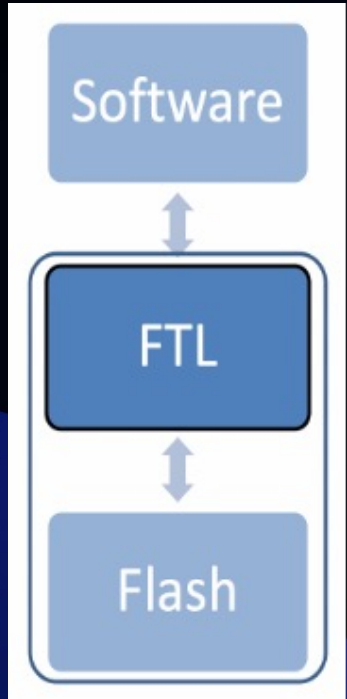
- 不对称的性能 -> 读性能最好，写性能次之，擦除性能最差
- 读写基本单元是page -> 潜在的读写放大
- Page写之前需要先擦除 -> “In-place-update” 的成本高
- 擦除的基本单元是block，block内部按顺序写入
- 块擦除有寿命限制

文件系统in-place-update

- Copy “整个block” 数据到内存
- 内存中修改数据
- 擦除Block
- 写入数据
- 成本太高！！



FTL(Flash Translation Layer)



文件系统

- 面向Flash逻辑地址读写

FTL

- 逻辑地址与物理地址映射
- 磨损平衡

Flash

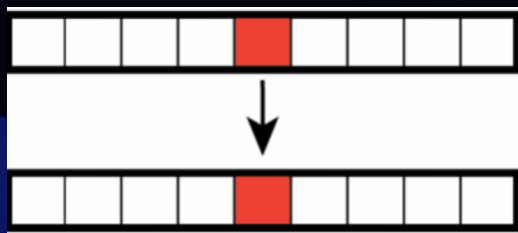
- 擦/写Flash单元
- 顺序写入Flash Page
- 坏块处理

FTL内部Block映射表结构

Block	Erased	Erase Count	Valid Page Count	Sequence Number	Bad Block Indicator
0	False	3	15	5	False
1	True	7	0	-	False
2	False	0	4	9	False

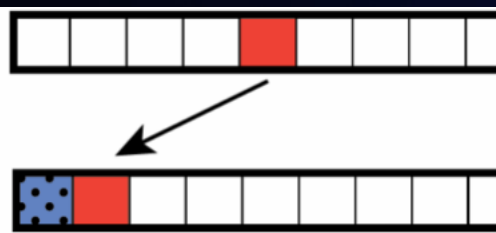
Out-place-update

文件系统视角: In-place-update

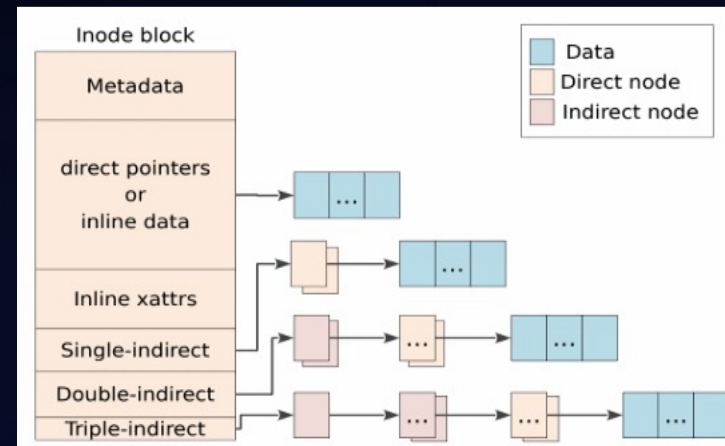


Hard Drive

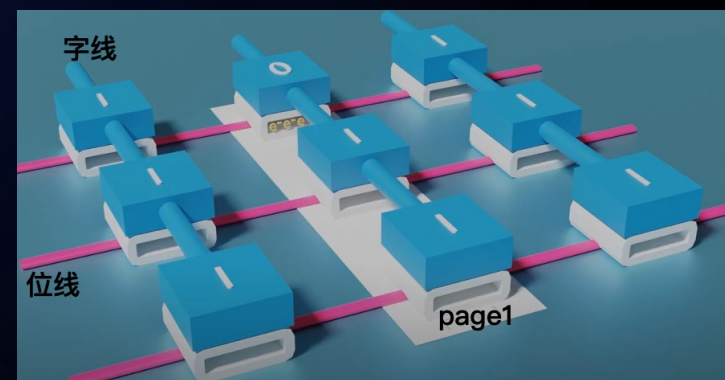
文件系统视角: out-place-update



Flash



Flash Translation Layer



文件系统

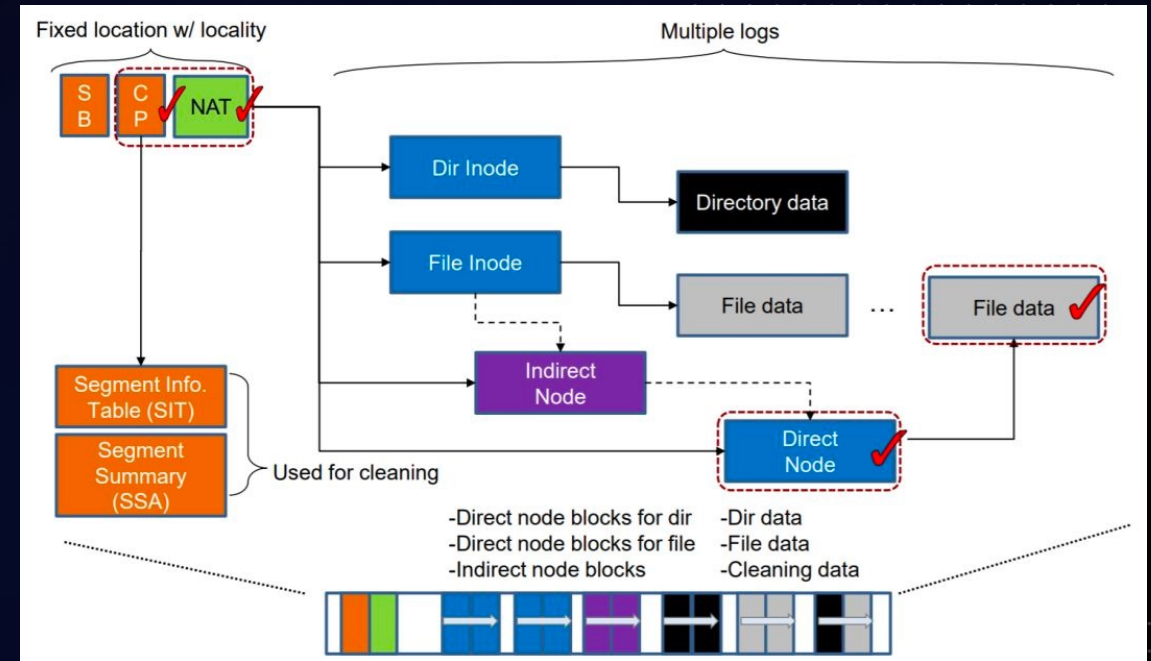
FTL

Flash

F2FS: Log-structured File System

Flash友好文件系统设计

- 文件系统元数据(inode)放到相邻Block中 -> 高频数据的局部性
- 文件系统按Block清理(FTL's GC Unit) -> 针对Flash擦除性能差
- 写放大抑制(减少indirect node传导) -> 减少写放大
- 日志模式写入(Multi-head logging) -> 减少out-place-update



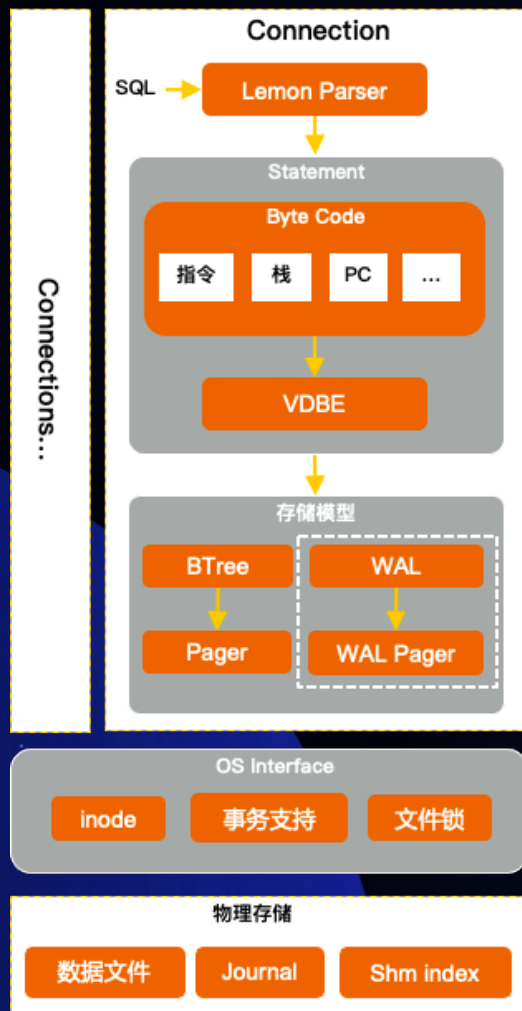
<https://www.usenix.org/conference/fast15/technical-sessions/presentation/lee>

移动端存储模型概况

B-Tree

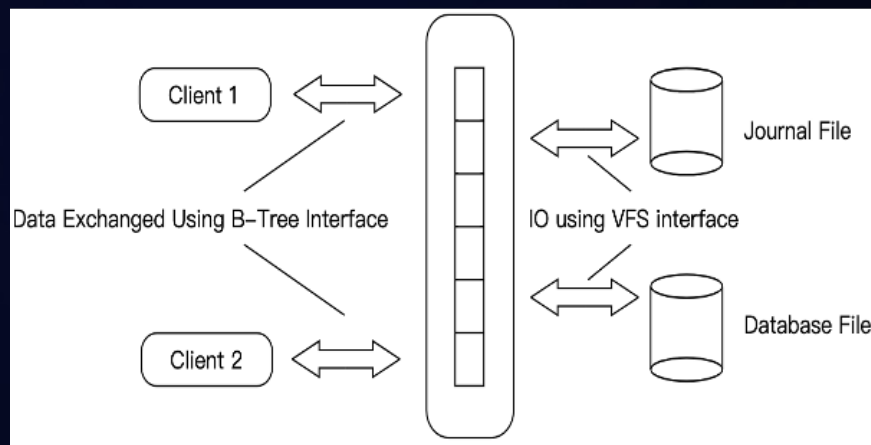
LSM

B-Tree

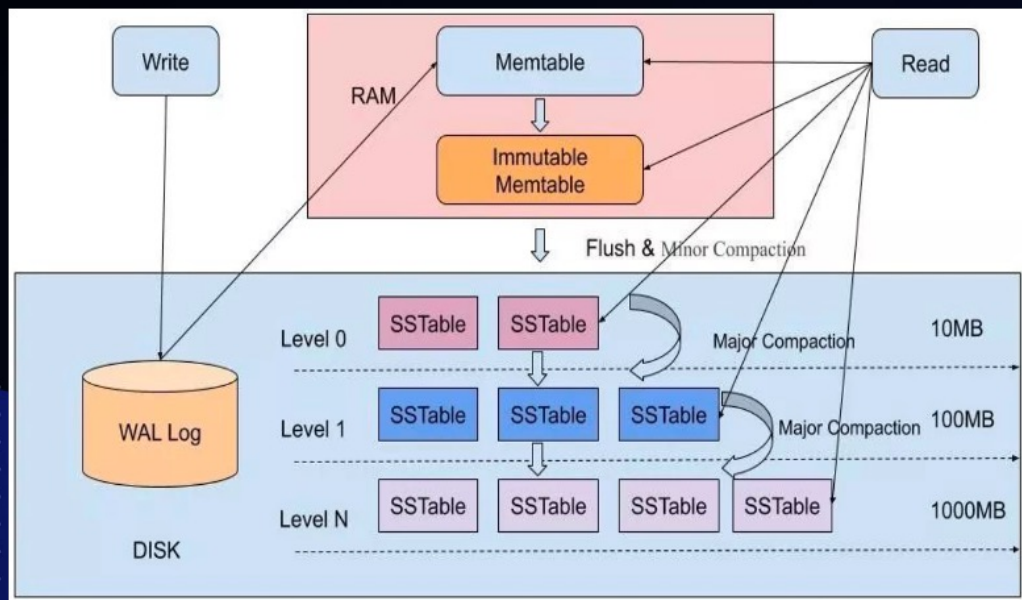


SQLite读写放大分析

- B-Tree大量随机读写，无法避免out-place-update的额外开销
- sqlite的journal log也是写放大产生的一个重要因素



LSM-Tree



LSDB读写性能分析

- 所有写操作均由顺序写实现（LSM）
- Compact产生的写放大对磁盘寿命会有一定的影响

3 手淘存储内核的实践

- Disk Layout
- 冷热分区
- 写放大抑制
- 事务设计
- 内存压缩
- 新式哈希表的应用

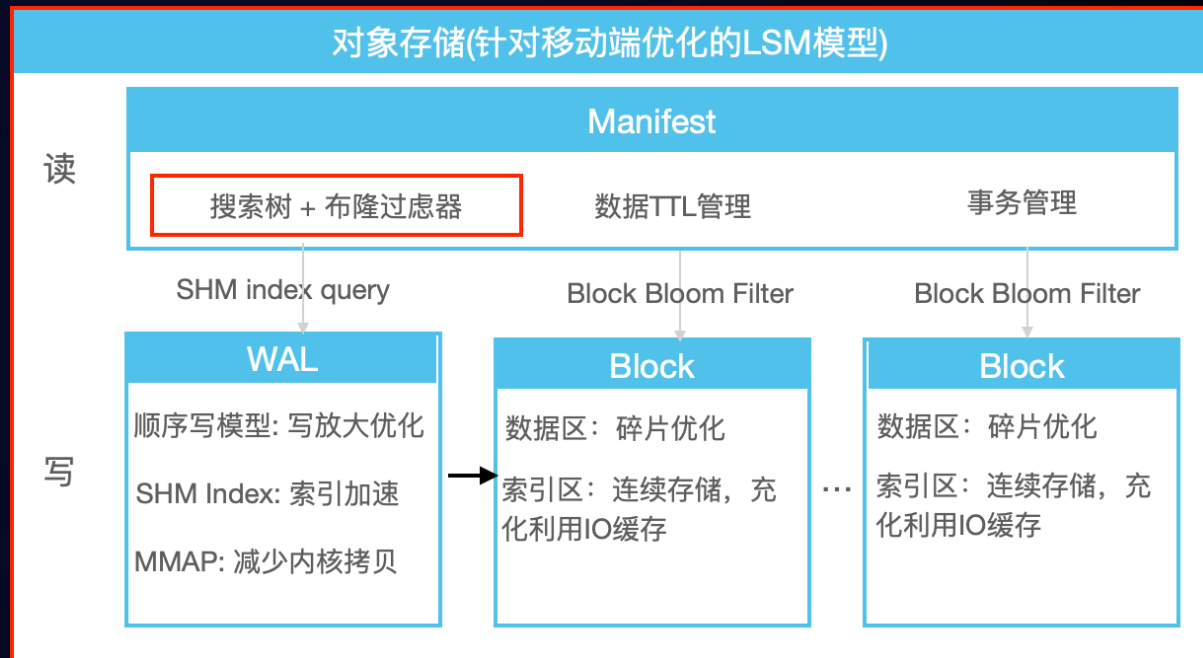
手淘针对移动端优化的LSM模型

手淘有较多IO密集场景，IO表现对用户体验非常关键

- 启动、信息流等场景大量小图片IO
- 游戏加载过程大量 web 资源 IO

关键feature

- **Flash友好**：所有写操作均由顺序写实现（LSM）
- **抑制碎片化**：数据紧密排列，避免碎片化带来的读写放大
- **抑制写放大**：只支持一级Compact。Compact只在后台闲时进行。
- **支持内存压缩**：通过内存压缩（Beringei算法）减少磁盘写入量
- **支持冷热分区**：“热”数据有更高的内存局部性和缓存优先级



Disk Layout

WAL(Write-Ahead-Log) Layout

- 提供顺序写模型：减少SSD写放大
- 数据紧密排列(非按页对齐)：减少读/写放大
- 数据同步使用fdatasync()：避免inode同步带来的写放大
- mmap用于减少内核数据copy

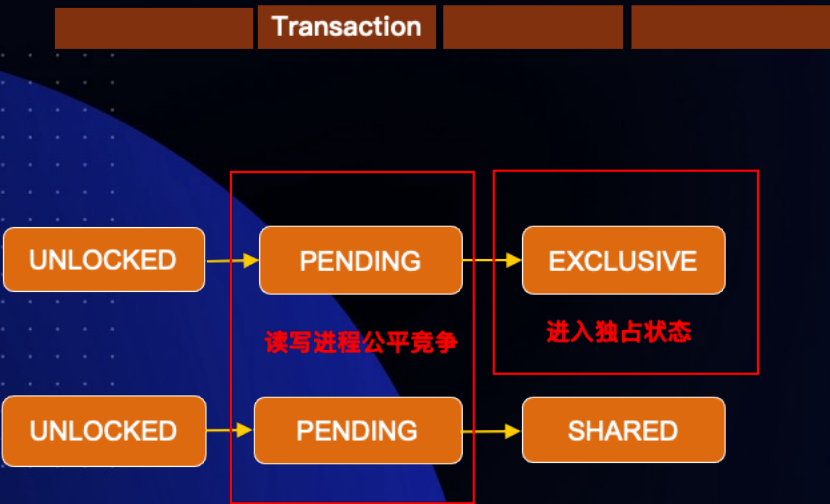
存储产品	手淘存储	LevelDB	SQLite
写放大系数	2倍	3倍	5倍(无索引)
有索引压缩	有(整型压缩比可以达到3:1)	无	无

WAL Disk Layout

Magic(u4)	Version No.(u4)	Header Size(u4)	Page Size(u4)	Wal Offset(u4)	
Reserved Header Space(112)					
	Timestamp(u8)	Data Size(u4)	Deletion Flag(u1)	Checksum(u4)	
Reserved Data Entry(14 bytes)					
Data Body					
...					

事务设计

- 支持WAL锁、库锁两种粒度
- 引入PENDING Lock，防事务饿死设计



存储产品	手淘存储	LevelDB	SQLite	Realm
支持事务	支持	无	支持	支持
锁粒度	WAL锁/锁	无	库锁	库锁
防饿死设计	有	无	有	有

稀疏文件

- 稀疏文件方案快速初始化WAL，与ftruncate()相比，WAL初始化时间(Y67): 100ms+ -> 5ms
- DataEntry固定大小，只加载基础类型到内存

存储产品	ProtoDB	LevelDB	SQLite	Realm
Log首次加载 耗时(Y67)	5ms	50ms	50ms	60ms
Log恢复时间 (4M , Y67)	5 – 10ms	50ms	80ms	80ms

冷热分区

- 高频访问、密集访问的数据存放于热数据区
- 有更好的内存局部性和缓存优先级



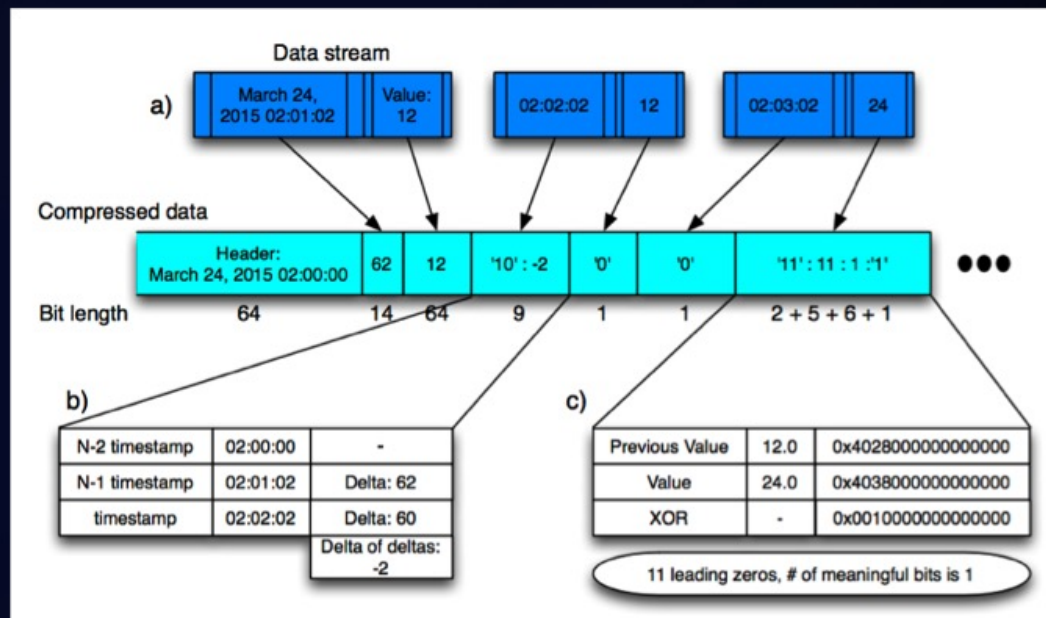
场景\命中率	热区(WAL)	WARM区	COLD区
图片库	73%	21%	6%
配置文件	65%	23%	12%

内存压缩

Beringei 算法简介

- 时间戳：Delta-of-Time压缩
- 浮点数：XOR压缩
- 内存压缩率超过60%，平均一个float只需要1.5个字节

Beringei 算法示意



优化效果及数据分析

WAL(Write-Ahead-Log) Layout

- 写性能
- 随机读性能
- 顺序读(遍历)



移动端存储可以有哪些演进方向？

1, offload to hardware: new cpu instructions 2, offload to kernel: new syscalls

- **SSE4.2 2008** <https://en.wikipedia.org/wiki/SSE4#SSE4.2>

- Intel introduced SSE4.2 STTNI (String and Text New Instructions)

- **AES-NI 2010**

https://en.wikipedia.org/wiki/AES_instruction_set

- AES-NI (or the Intel Advanced Encryption Standard New Instructions; AES-NI) was the first major implementation

- **SHA 2013**

https://en.wikipedia.org/wiki/Intel_SHA_extensions

- Intel SHA Extensions are a set of extensions to the x86 instruction set architecture which support hardware acceleration of Secure Hash Algorithm (SHA) family.

- **sendfile** 1998 FreeBSD 3.0, 1999 Linux 2.2

- **recvmsg** 2010 Linux 2.6.33, 2011 Linux 3.0 (UDP/QUIC)

- **kTLS** 2015 FreeBSD, 2017 Linux 4.13 send, Linux 4.17 receive



**扫码回复「D2」
获取第十七届 D2 演讲 PDF 材料**

后续也将推送 D2 会后技术文章，敬请关注！！

THANK FOR YOUR WATCH

感谢大家观看