新硬件环境下日志模块的设计与演进

朱阅岸
YY Research LAB

内容大纲

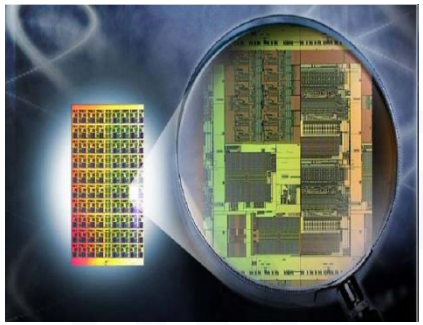
1. 现代处理器及新型存储的发展
2. 传统的数据库系统日志模块设计
3. 面向新型硬件的日志子系统设计
4. 总结

内容大纲

1. 现代处理器及新型存储的发展
2. 传统的数据库系统日志模块设计
3. 面向新型硬件的日志子系统设计
4. 总结

现代处理器

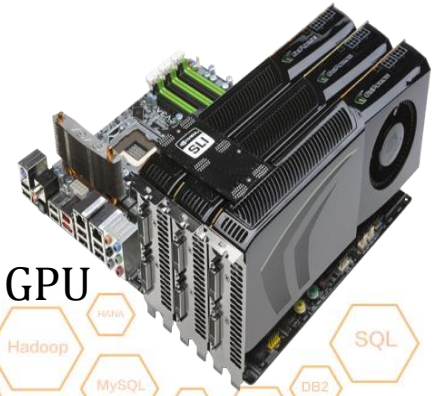
- 能耗与制造工艺的原因使得处理器制造商不再追求高频率，而是转向片上多处理器技术
- many-core的概念开始流行



Intel Haswell Ex
144 Cores

Xeon Phi
72 cores

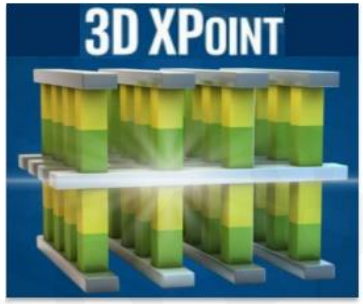
nVIDIA GeForce GPU
216 cores



Triple GeForce GPU
648 cores

新型存储设备

- 非易失性内存开始从实验室走向工业界，例如Intel傲腾
- 具有磁盘的持久存储特性与接近内存的访问速度
- 主要特性是非易失、低延迟、高密度和读写不对称

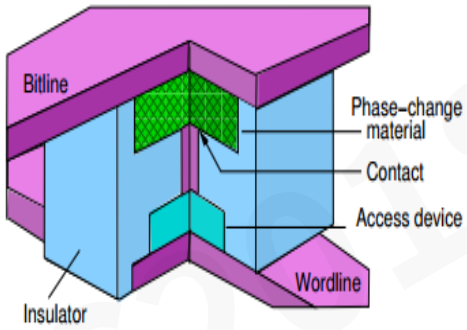


Non-Volatile Memory 走向工业界

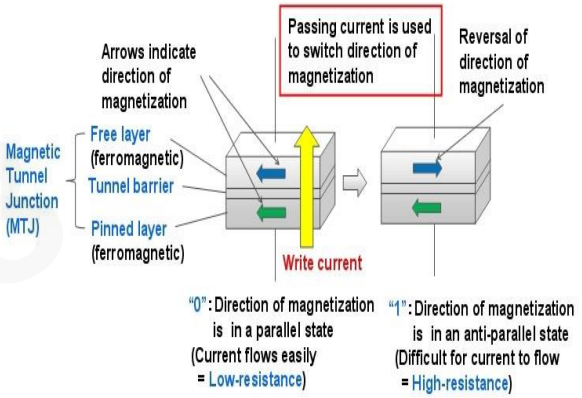
Non-Volatile Memory兼具磁盘与内存特征

新型存储设备原理简介

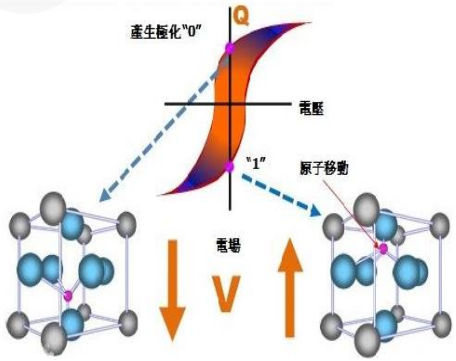
- 实现非易失性内存的技术主要有如下几种：
  - a) 相变存储器：材料可以在结晶状态与非结晶状态转变
  - b) 自旋磁矩：改变两层磁性材料磁矩方向
  - c) 铁电材料：材料所形成的电荷高低，二元状态
  - d) 忆阻器：是一种有记忆功能的非线性电阻



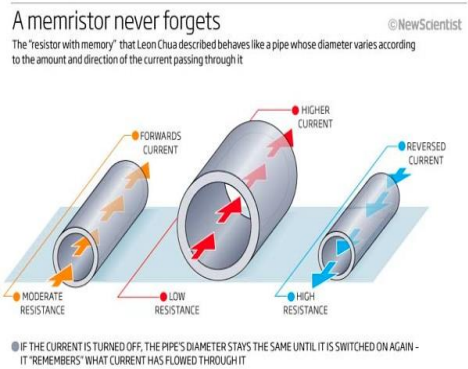
1. 相变技术(phase change memory)



2.自旋磁矩



3. 铁电材质



4. 忆阻器

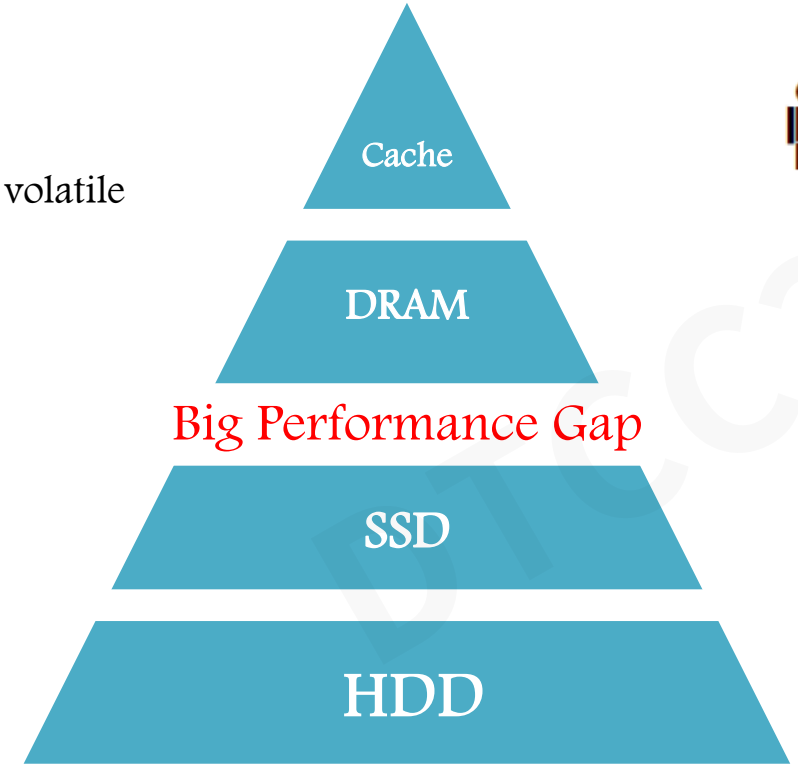
相关参数

	DRAM	NAND Flash	PCM
页面大小 读延迟 写延迟 带宽 擦除延迟	64B ~100ns ~100ns ~GB/s per die N/A	4KB ~25 µs ~500 µs 5-40 MB/s per die 2 ms	64B ~100ns ~1 µs 50-100 MB/s per die N/A
寿命	∞	$10^4 - 10^5$	$10^6 - 10^8$
读耗能 写耗能 空闲耗能	0.8 J/GB 1.2 J/GB ~100 mW/GB	1.5 J/GB [28] 17.5 J/GB [28] 1–10 mW/GB	1 J/GB 6 J/GB ~1 mW/GB
密度	1×	4×	2 – 4×

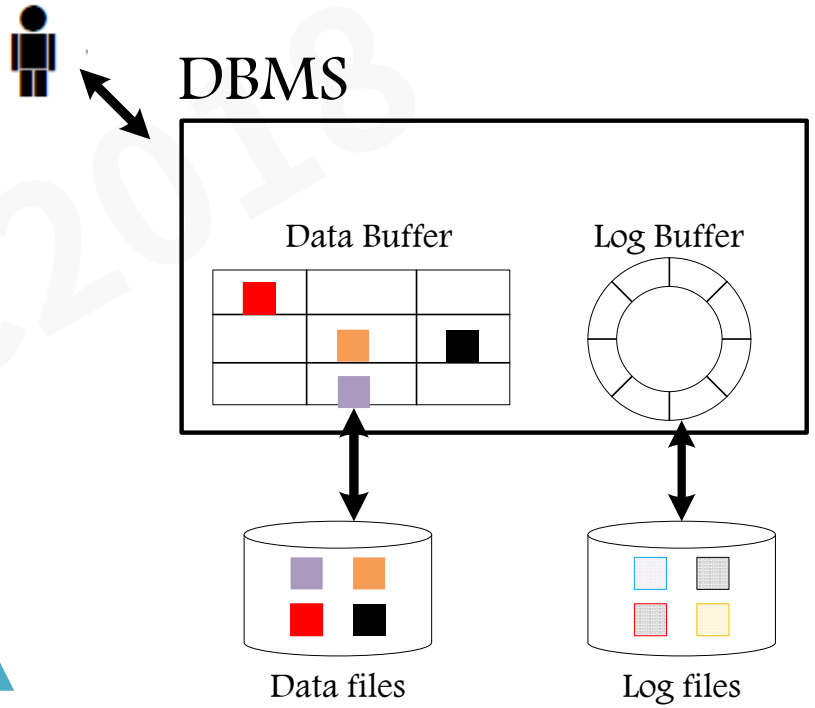
ref: 存储技术发展对数据管理研究的影响 李战怀

底层硬件决定上层软件设计

Fast, but volatile

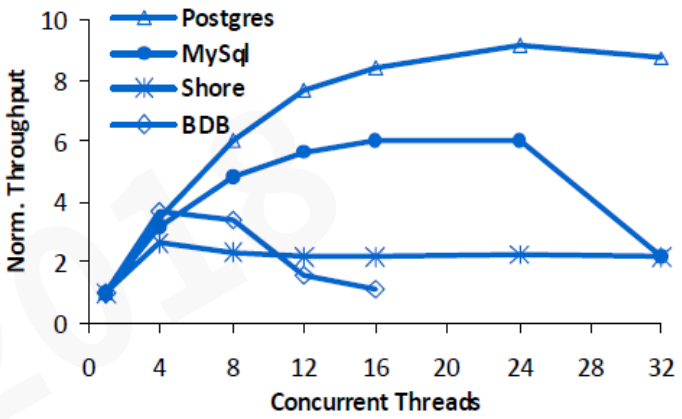


Durable, but slow

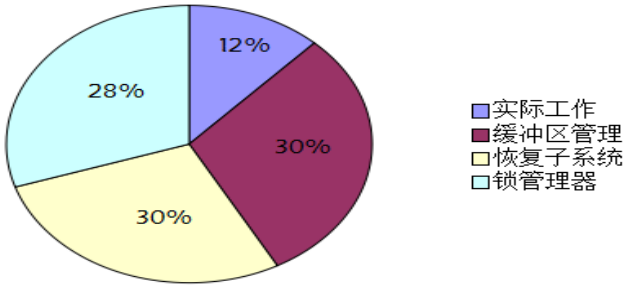


底层硬件决定上层软件设计

- 如今数据库系统落后的设计观念与现代硬件的矛盾日益突出
- 主要原因关系数据系统的研发始于上世纪70年代初，那时缓慢的外存设备以及有限并发的处理器决定了系统设计



(a) 不同并发下，各个系统的扩展性



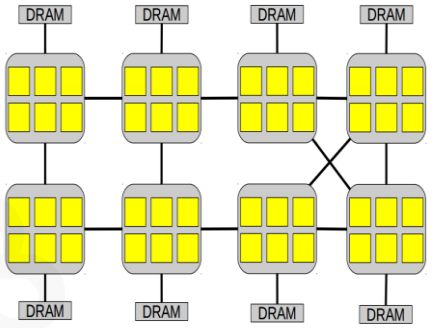
(b) 典型的DBMS事务执行时间耗费

ref: Shore-MT: A Scalable Storage Manager for the Multicore Era sigmod 2009

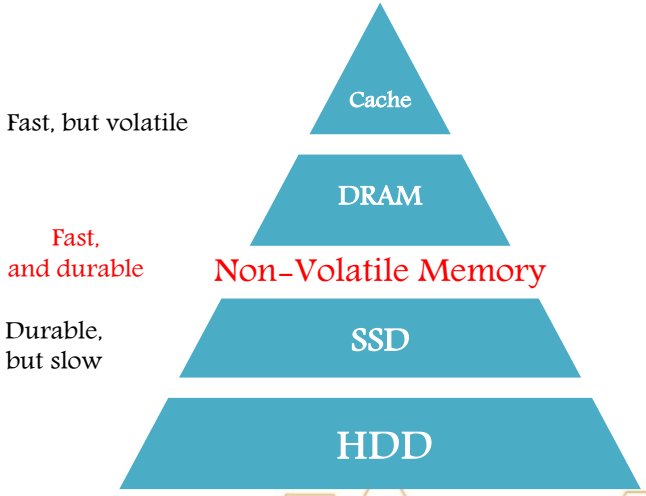
ref: OLTP Through the Looking Glass, and What We Found There sigmod 2008

小结

1. 受制造工艺与能耗的影响，CPU向多核、众核方向发展
2. 非易失性内存开始从实验室走向工业界。这类存储介质兼具内存的访问速度与磁盘的持久性存储
3. 上层软件必须适配底层硬件的发展才能获得最佳硬件红利



图a. many-cores CPU



图b. NVM使内存与外存设备的界限不复存在

内容大纲

1 现代处理器及新型存储的发展

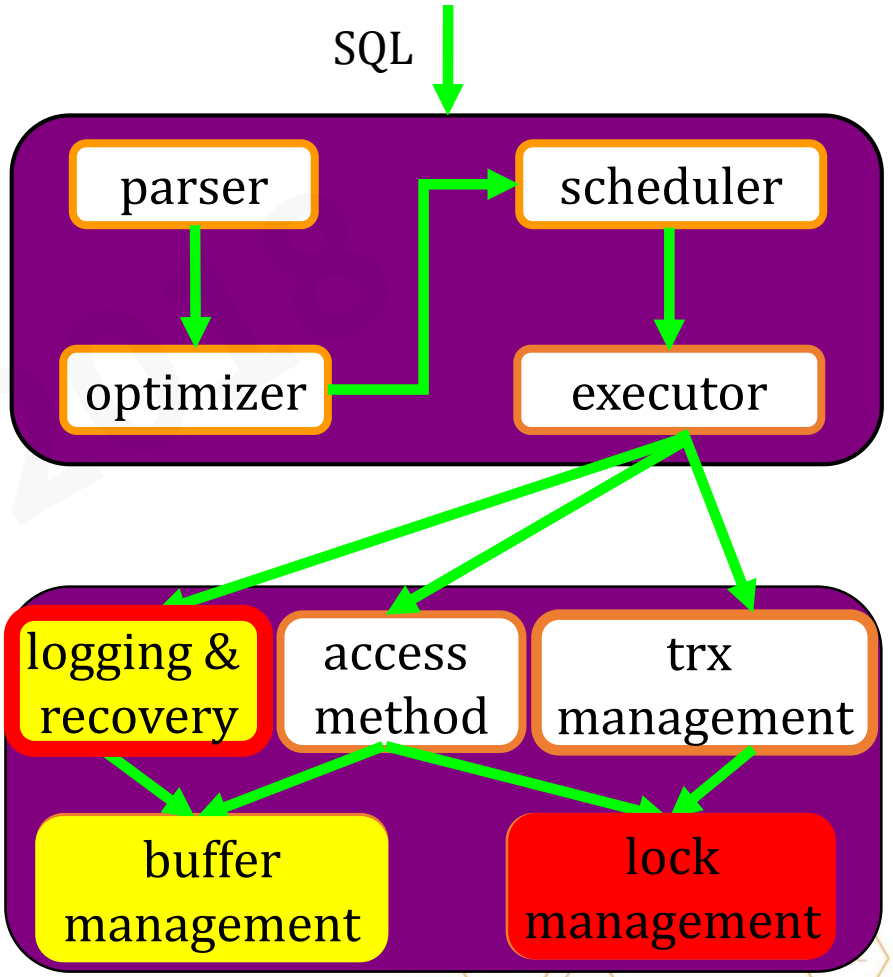
2 传统的数据库系统日志模块设计

3 面向新型硬件的日志子系统设计

4 总结

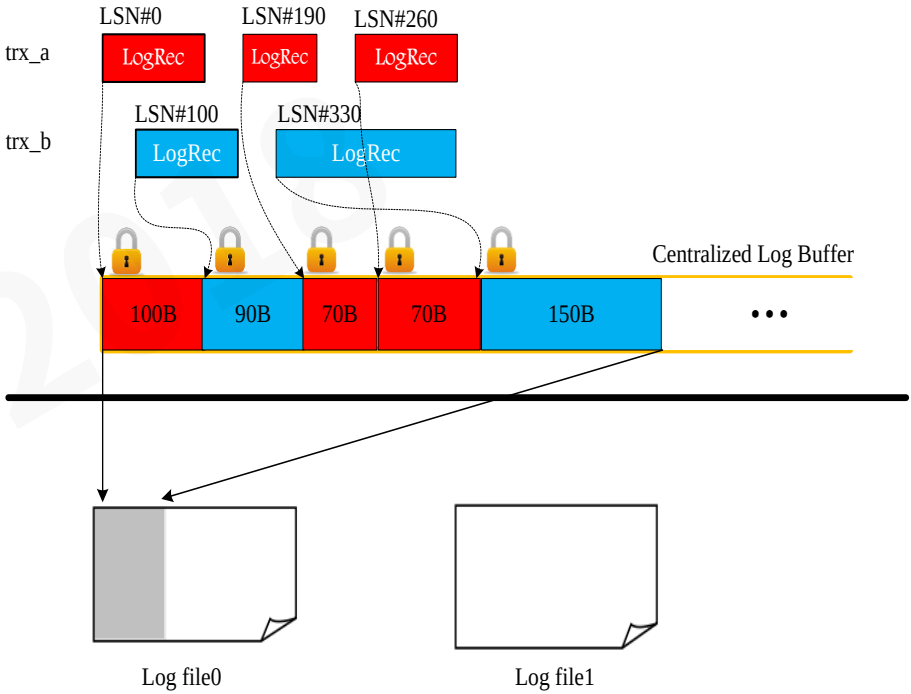
开源系统状况

- InnoDB的性能瓶颈目前主要集中在锁管理器与日志模块；PostgreSQL的性能瓶颈体现在缓冲区管理模块与日志模块
- 日志模块。数据库通常采用集中的WAL方法实现，引发热点问题



WAL实现上的问题

- 传统的日志提交过程主要有三个步骤：
  1. 首先获取写日志缓冲区上的排他锁
  2. 递增LSN，代理线程将事务日志记录拷贝到日志缓冲区相应位置
  3. 释放缓冲区上的锁



WAL的集中式实现引发热点问题

InnoDB 事务日志提交

当提交一个mini transaction时，需要将记录数据的更改日志提交到公共buffer中，并将对应的脏页加到flush list上。入口函数为mtr_t::commit，执行过程如下：

Step 1: mtr_t::Command::*prepare_write* 获取 log_sys->mutex

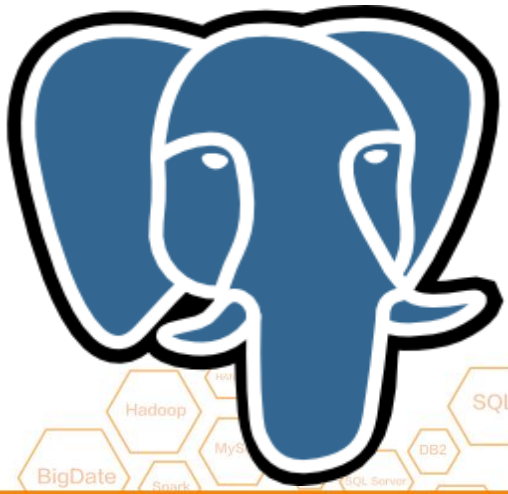
Step 2: mtr_t::Command::*finish_write* 将日志从mtr中拷贝到公共log buffer。这里需要考虑block空间不足的情况（持有log_sys->mutex)

Step 3: 若本次事务产生脏页，申请flush list mutex 随后释放log_sys->mutex

PostgreSQL 事务日志提交

```
XLogInsert ( ... )
{
    ...
    START_CRIT_SECTION( );
    /* Now wait to get insert lock */
    LWLockAcquire( WALInsertLock, LW_EXCLUSIVE );
        ...
        ...
        ...
    LWLockRelease(WALInsertLock);
    END_CRIT_SECTION( );
    ...
}
```

临界区代码长度约300行



小结

1. 很多数据库系统，特别是开源系统在临界区设计上比较粗糙，面临扩展性问题
2. 从目前的研究实践看来MySQL与PostgreSQL的性能瓶颈主要聚焦在日志模块
3. 在当前硬件环境下重构日志模块具有重要意义

内容大纲

1 现代处理器及新型存储的发展

2 传统的数据库系统日志模块设计

3 面向新型硬件的日志子系统设计

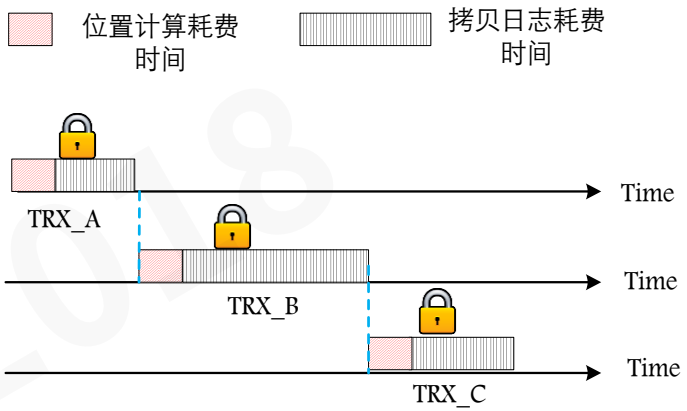
4 总结

大并发场景优化

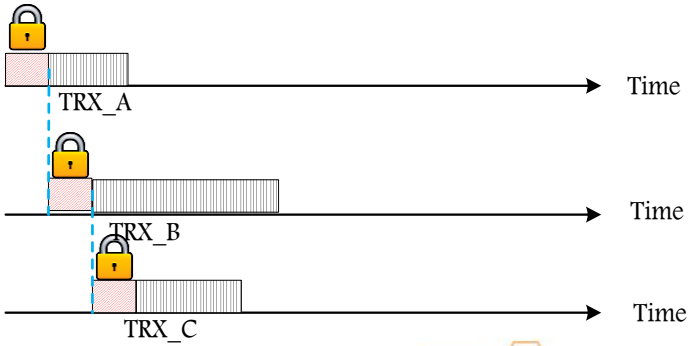
➢ 预留空间

1. 事务只需在临界区内简单计算日志记录所占长度。事务之间可以并行拷贝日志
2. 需要追踪日志记录填充情况。

   事务提交时刻需要确保之前的事务已经完成日志拷贝



图(a).日志拷贝串行



图(b).预留空间实现

大并发场景优化之一

- **Express Logging Ensuring Durable Atomicity**

1. 系统区分3类线程：工作线程、日志状态追踪线程以及日志刷盘线程
2. 写日志由3个步骤的流水线组成

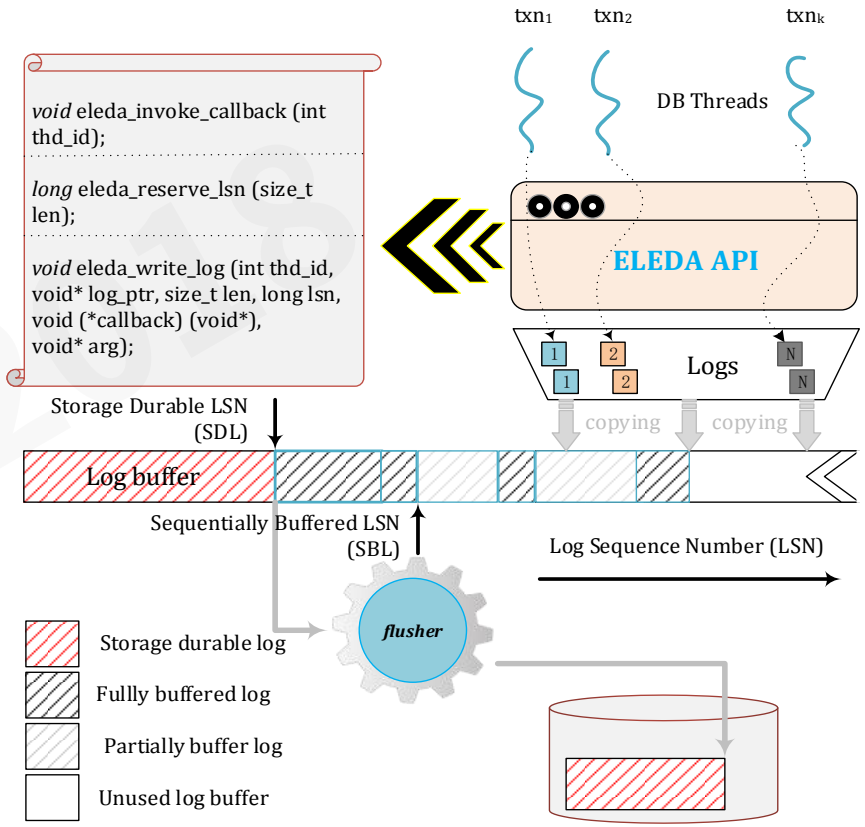


图. ELEDA总体设计

ref: Scalable Database Logging for Multicores vldb 2018

大并发场景优化之一

- **Express Logging Ensuring Durable Atomicity**

1. 正常工作线程:预留空间，维护两个list

   日志状态追踪线程：维护SBL (Sequential Buffered LSN)

   日志刷盘线程：SDL→SBL

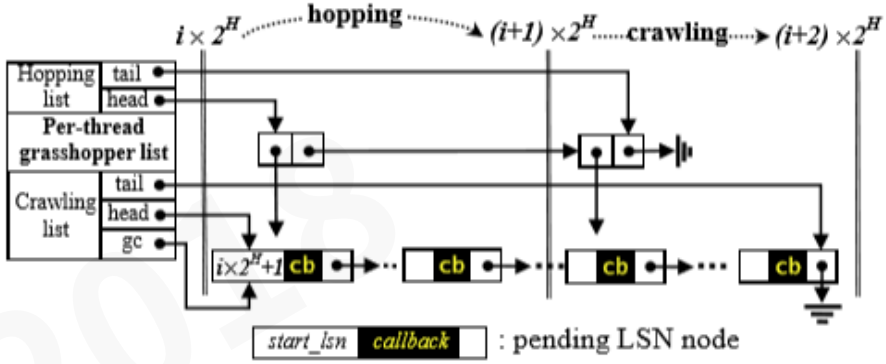


图. Per-thread list

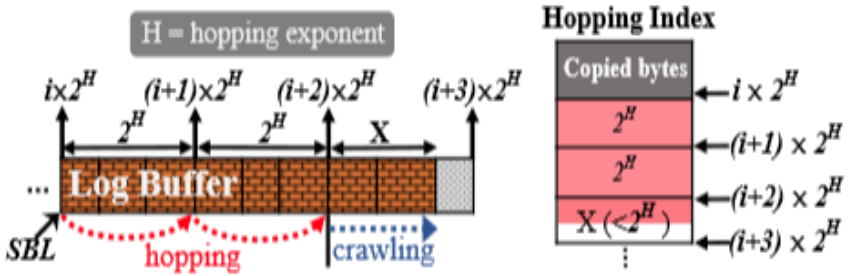


图. 日志空洞追踪

ref: Scalable Database Logging for Multicores vldb 2018

大并发场景优化之一

- **ELEDA performance**

1.测试环境： 36-Core Intel Xeon E5-2699 v3, 488G DRAM

2.MongoDB default storage Engine： WiredTiger

3.YCSB(R:0.5,W:0.5)以及TPC-C (32 warehouse)测试基准

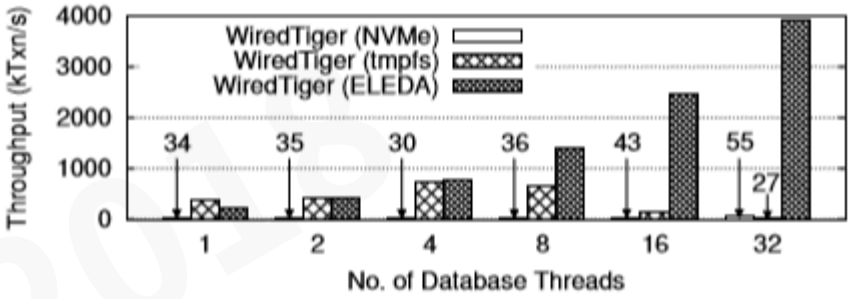


图. **YCSB** 测试结果

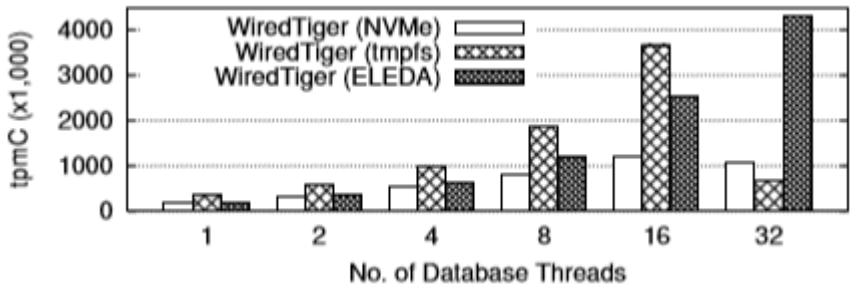


图. **TPC-C**测试结果

ref: Scalable Database Logging for Multicores vldb 201

大并发场景优化之二

- **PostgreSQL 9.4**

1.PostgreSQL9.4开始重构日志插入算法

2.将临界区代码缩短为只有5行，利用8把锁追踪日志空洞

3.在大并发场景下性能能够提升30%

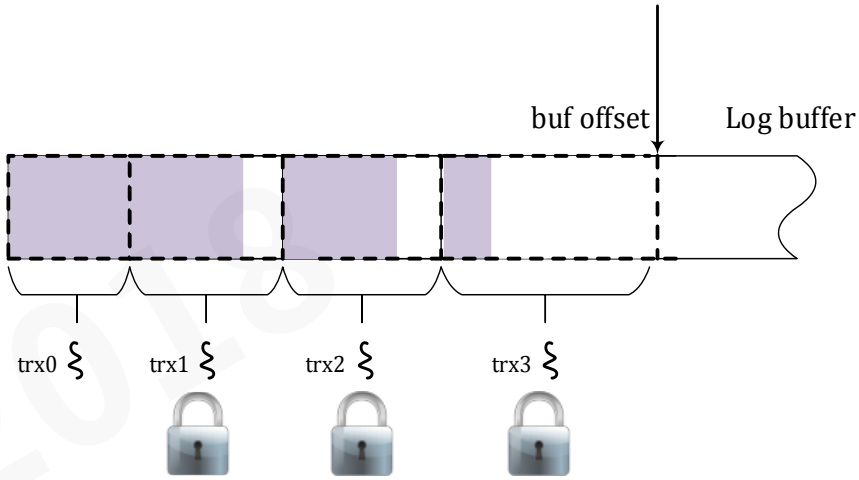


图. Pg9.4日志插入算法

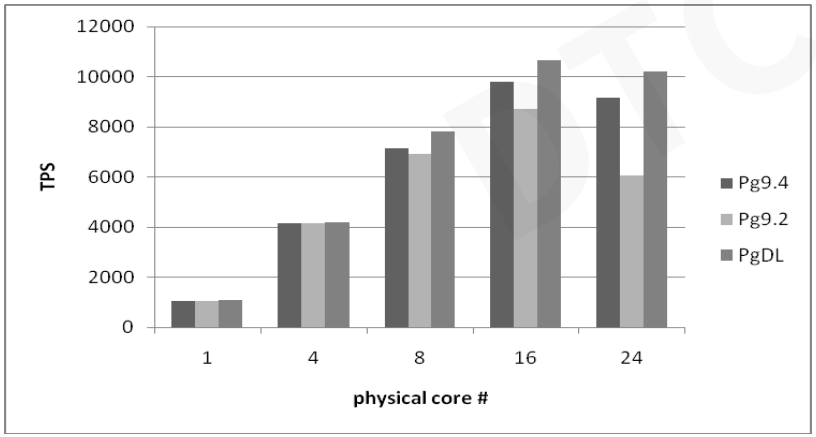


图.TPC-B 性能测试

```
Spin_lock;
Startbytepos          = Insert->CurrBytePos;
endbytepos            = startbytepos + size;
prevbytepos           = Insert->PrevBytePos;
Insert->CurrBytePos   = endbytepos;
Insert->PrevBytePos   = startbytepos;
Unlock;
```

图. 改进后代码复杂度

NVM上的日志管理器

➢并行写日志

1.将负载分散到N个不同的日志管理器上，提高写日志的并行性

2.需要解决日志空洞与确定日志回放顺序问题

3.Passive group commit技术

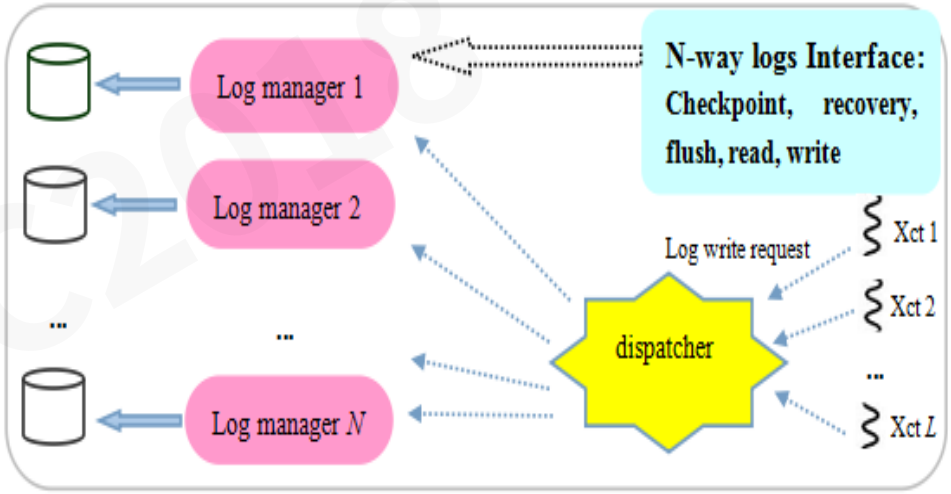


图. 多路日志管理器

ref: Scalable Logging through Emerging Non-Volatile Memory vldb 2014

NVM上的日志管理器

➢并行写日志

1.采用每个事务维护一个私有日志缓冲区的策略

2.两种日志持久化策略:*flush-on-insert*以及*flush-on-commit*

3.在当前的存储架构下，需要使用*Cflush*以及*mfence*等指令

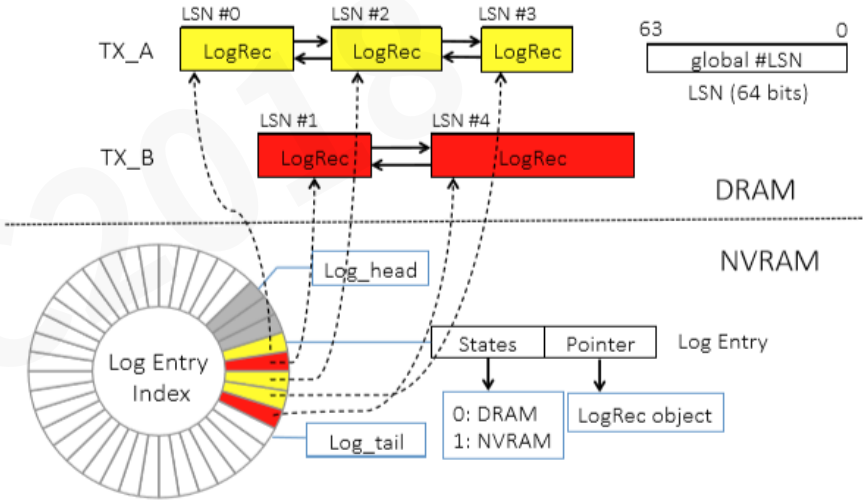


图. 完全分散的日志管理器

ref: NVRAM-aware Logging in Transaction Systems vldb 2014

NVM上的日志管理器

➢并行写日志

1.Redo空间按照页面进行划分，undo空间按照事务进行划分;并行执行，并行回放

2.TPC-C 性能提升1.2-1.6倍

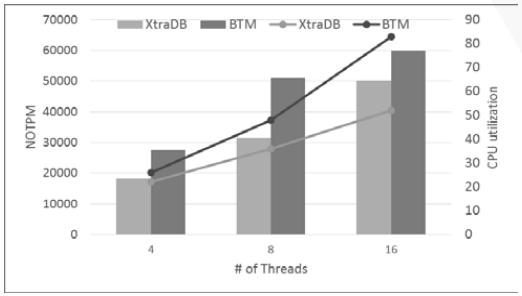


图. 性能比较

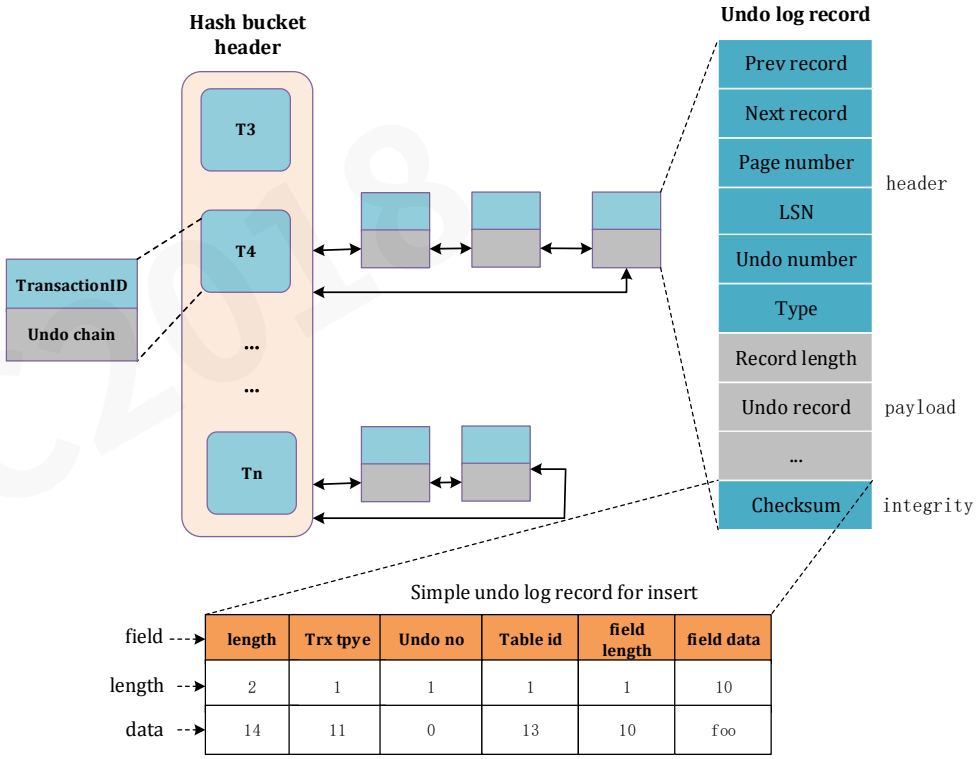


图. Undo日志空间划分

ref: BaSE(Byte addressable Storage Engine) Transaction Manager COMAD 2016

NVM上的日志管理器

➢新的实践

1.Write-behind logging.运行时只需追踪脏页，无需构造redo日志。事务提交时刻先写脏页，然后写日志

2. 统一数据块与日志块，完全避免写日志

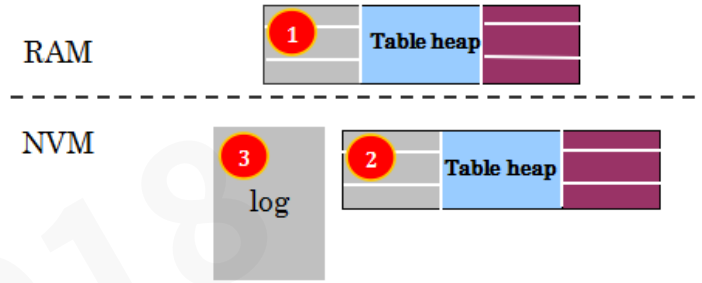


图. 写次序

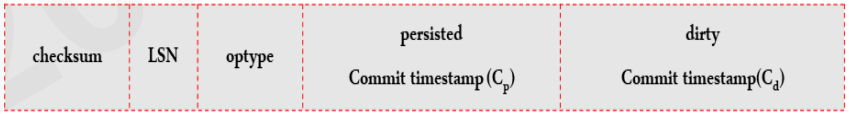


图. 日志格式

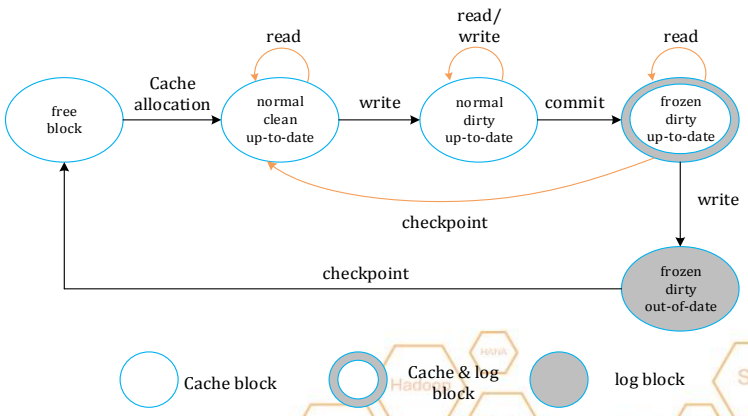


图. 统一数据块与日志块

ref: Write-behind logging vldb 2017

ref: Unioning of the Buffer Cache and Journaling Layers with Non-volatile Memory FAST 13

小结

1. 传统的ARIES算法是面向磁盘而设计的，在大并发情况下日志模块容易成为系统瓶颈
2. 软件的设计需要适配底层硬件的发展。
3. 优化集中式日志管理器主要有三种方式：缩短临界区、并行化写日志操作、开辟新的理论实践

内容大纲

1. 现代处理器及新型存储的发展
2. 传统的数据库系统日志模块设计
3. 面向新型硬件的日志子系统设计
4. 总结

总结

1. 计算机硬件的发展为数据库理论的创新打开一扇新的大门
2. 多核与内存计算时代下，系统设计人员应当将更多的精力放在系统扩展性以及数据访问的局部性
3. NVM的出现使得系统设计人员可以将注意力完全地从I/O上移除，专注系统扩展性设计

这是最坏的时代，
这是最好的时代

THANKS


HANA
Hadoop
SQL
MySQL
DB2
BigDate
Spark
SQL Server
NewSQL
Oracle
MPP
NoSQL
OLAP

讲师申请
联系电话（微信号）：18612470168

关注“ITPUB”更多
技术干货等你来拿~

与百度外卖、京东、魅族等先后合作系列分享活动

微学堂

让学习更简单

微学堂是以ChinaUnix、ITPUB所组建的微信群为载体，定期邀请嘉宾对热点话题、技术难题、新产品发布等进行移动端的在线直播活动。

截至目前，累计举办活动期数60+，参与人次40000+。

ITPUB学院

ITPUB学院是盛拓传媒IT168企业事业部（ITPUB）旗下
企业级在线学习咨询平台
历经18年技术社区平台发展
汇聚5000万技术用户
紧随企业一线IT技术需求
打造全方式技术培训与技术咨询服务
提供包括企业应用方案培训咨询（包括企业内训）
个人实战技能培训（包括认证培训）
在内的全方位IT技术培训咨询服务

ITPUB学院讲师均来自于企业
一些工程师、架构师、技术经理和CTO
大会演讲专家1800+
社区版主和博客专家500+

培训特色

无限次免费播放
随时随地在线观看
碎片化时间集中学习
聚焦知识点详细解读
讲师在线答疑
强大的技术人脉圈

八大课程体系

基础架构设计与建设
大数据平台
应用架构设计与开发
系统运维与数据库
传统企业数字化转型
人工智能
区块链
移动开发与SEO

联系我们

联系人：黄老师
电　话：010-59127187
邮　箱：edu@itpub.net
网　址：edu.itpub.net
培训微信号：18500940168