# 漫谈 SSD 和应用实践

作者:zolker(杨尚刚)

一．什么是 SSD

SSD 全称 Solid State Disk，就是大家平时提到的固态硬盘，是一种完全由电子元器件组成的持久化存储设备，这是和传统机械硬盘(Hard Disk Drive，简称 HDD)的重要区别。SSD 和机械硬盘相比，有着更高的 IOPS，更高的带宽，更低的访问 latency，还兼容目前主流的 HDD 接口，比如 SATA SAS 等。现在市场上的 SSD 主要都是基于 NAND Flash 的。而同样作为持久化存储，SSD 和 HDD 相比有哪些优势和劣势呢

| | SSD | HDD |
|---|---|---|
| 性能 | 平均响应时间 100us 左右 | 平均响应时间 12ms 左右 |
| 可靠性 | 主要是电子元件，可靠性更高 | 主要机械组件，高速旋转读写数据，相对更复杂 |
| 功耗(W) | 5 | 16 |
| IOPS | 70000+ | 200 |
| MTBF(万小时) | 200 | 160 |
| 成本(每 GB) | 1$ | 0.1$ |

二．SSD 主要组件和类型

1) SSD 主要组件

①. 控制器芯片

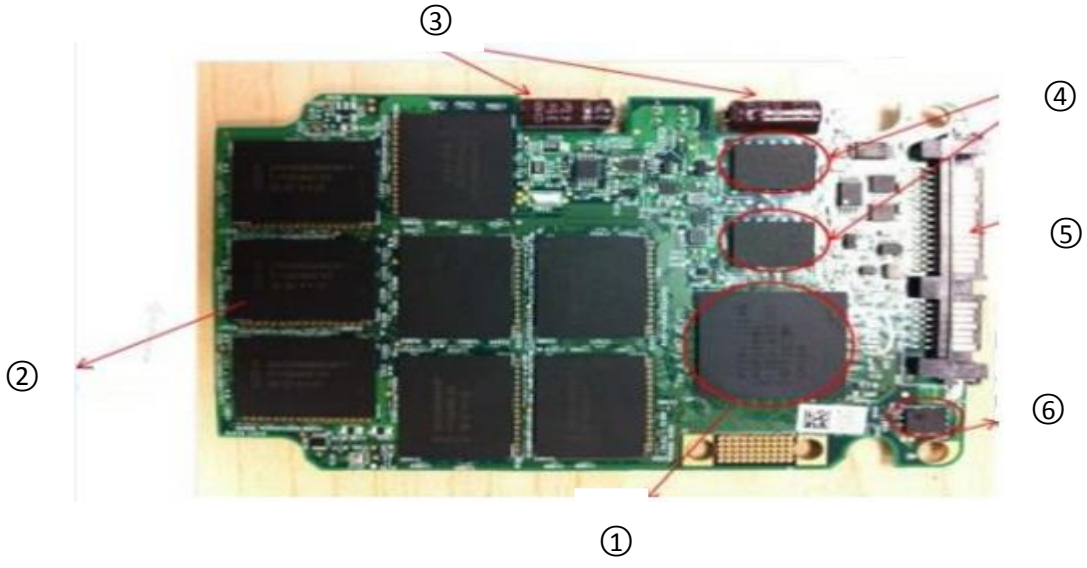
②. NAND 芯片

③. 电容，掉电保护，保证数据安全性

④. DRAM，缓存元数据

⑤. 对外接口，图中是 SATA 接口

⑥. NOR Flash，引导 SSD



2) SSD 类型

按照接口分类，目前使用的 SSD 主要有以下四种

①. SATA

企业级使用最多的产品，相对 PCIe 有成本优势和更好的稳定性

②. PCIe

在企业级使用也比较多，相对 SATA 接口有着更好的性能和更低的 Latency。

③. SAS

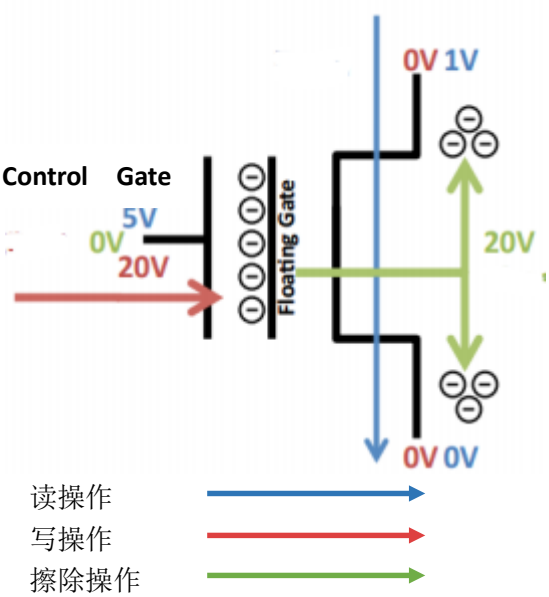
企业级使用的不多，介于 SATA 和 PCIe 中间的位置，性价比并不高。

④. mSATA & NGFF(M.2)

主要面向消费级产品，以后也会有用于企业级做为系统 boot 盘

四.NAND 芯片

这一部分我们重点分析一下 NAND 芯片的原理。NAND 是存储最终数据的介质，NAND 实际上是一种 EEPOM（加电可擦除可编程 ROM）,最基本的组成单位称作 cell，而 cell 是一种类似 MOSFET(金属-氧化层 半导体场效晶体管)的电子元件。主要的部分是 Control Gate(控制闸，简称 CG)和 Floating Gate（浮动闸，简称 FG）。




CG 的作用就是通过加不同电压来对 FG 进行充放电，改变 cell 存储的 bit

FG 是一个与周围绝缘的氧化物层，电子可以在 FG 里长久保存而不会轻易泄露，这也是 NAND 存储数据非易失的原因。

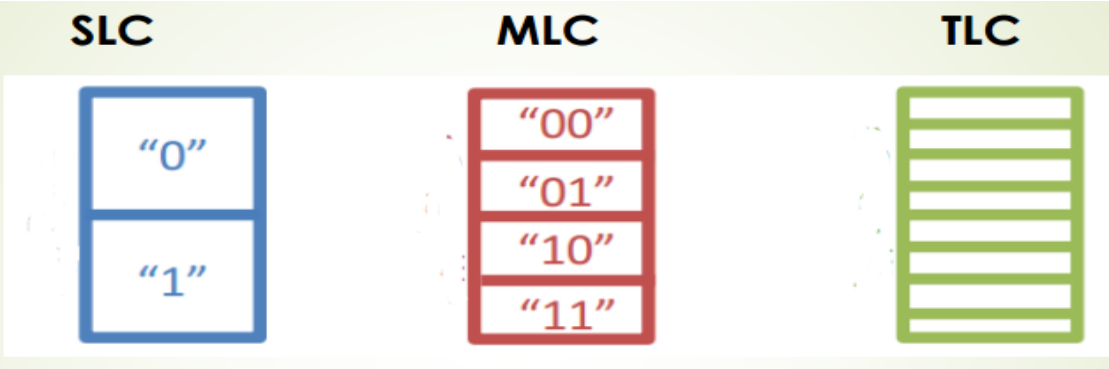
上图也介绍了针对 cell 的三种操作，分别是读操作 写操作和擦除操作，先假设每个 cell 只存储一个 bit

①. 读操作，在 CG 上加 5V 的电压，然后根据读到电压值去判断这个 cell 存储的是 0 还是 1

②. 写操作，在 CG 上加 20V 电压，电子被 FG 捕获，相当于一个充电过程。一般写之后还会进行读校验，判断写入是否成功，正常写入完应该是 0

③. 擦除操作，加电压的方向和写操作正好是相反的，作用也是相反的，相当于对 FG 的一个放电过程，擦除完成 cell 代表的是 1

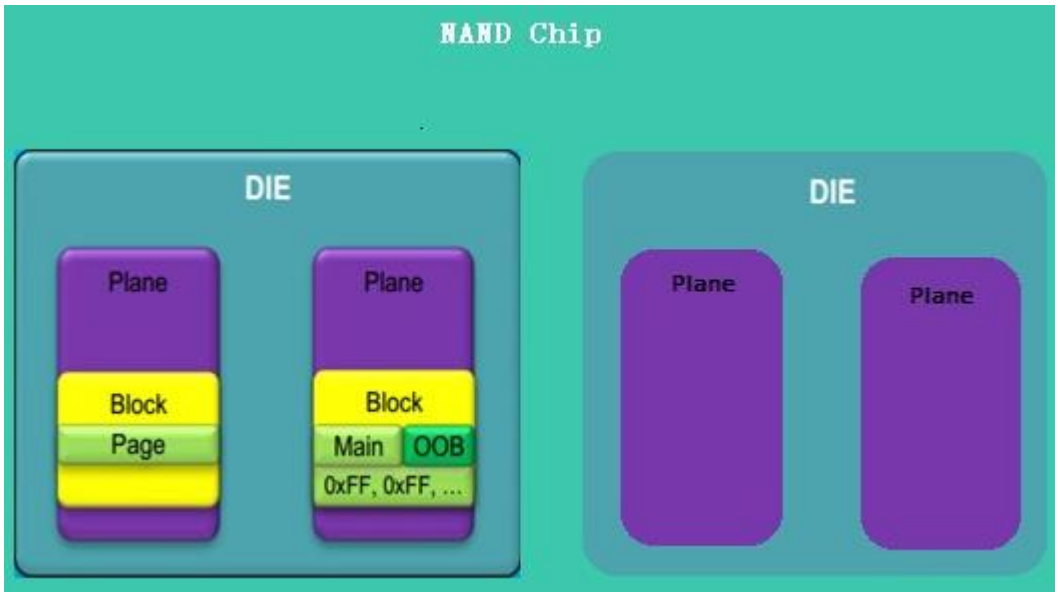
按照每个 cell 存储的 bit，目前可以分成三类 NAND:




SLC，每个 cell 存储一个 bit，擦除次数为 100000 次
MLC，每个 cell 存储两个 bit，擦除次数为 5000 次，还有 eMlC 支持最大擦写次数为 30000 次
TLC，每个 cell 存储三个 bit，擦除次数为 1000 次
我们对这三种。类型做一个横向对比

| 类型 | 性能 | 耐久度 | 价格 | 使用场景 |
|---|---|---|---|---|
| SLC |  |  |  | 对性能和寿命要求非常苛刻的场景，比如军工银行等 |
| MLC |  |  |  | 企业级普遍采用，性能和寿命能满足需求就可以，更关心价格 |
| TLC |  |  |  | 用于消费级和企业级冷数据存储 |

我们再了解一下一个 NAND 芯片是怎么组成的




Cell 组成 Page，Page 组成 Block，Block 组成 Plane，Plane 组成 Die，多个 Die 最终封装成我们之前看到的 NAND 芯片。
从上面组成中我们重点分析一下 Page 和 Block
Page 主要由两部分组成，Main 和 OOB 两部分，Main 主要是存储实际写的数据，而 OOB 主

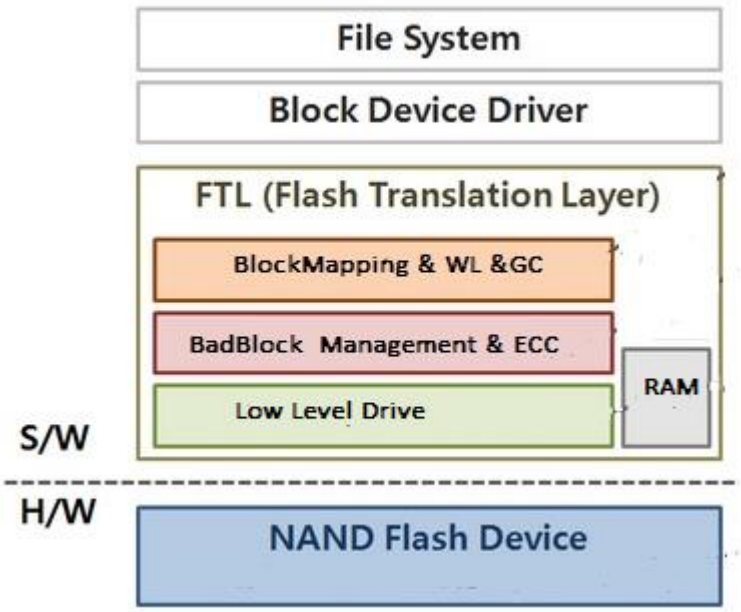
要存储一些元数据信息和 ECC 信息，Page 在出厂时一般是 0xFF，也就是全部擦出过一遍。Block 是由一组 Page 来组成。

并且还有很重要的一点就是，Page 是 SSD 最小的读写单位，而 Block 是最小的擦除单位，这就造成了所谓的写入放大问题(Write Amplification)。SSD 按照 Block 去擦除，这样做的原因一方面是由于底层电路设计限制，另一方面原因是为了 SSD 本身寿命和响应时间考虑。

五．SSD 控制器

这部分重点介绍 SSD 控制器的功能，SSD 控制器最核心的是一个 Flash Translation Layer(Flash 转换层，简称 FTL)，大部分核心控制逻辑都是在是由这个层面完成。

我们先看一下 FTL 在整个存储体系中的地位和层次




主要介绍以下几个功能:

1) Block Mapping,逻辑块到物理块映射
2) Wear leveling ，磨损调度均衡
3) Garbage collection，垃圾回收
4) Bad block management，坏块管理
5) ECC 校验

1. Block Mapping

顾名思义从名字上看，这个功能就是做一个 Block 的映射功能，相当于是为了兼容 HDD，SSD 是没有扇区的概念的，但是通过块映射我们就可以在 SSD 上进行各种在 HDD 上的操作。

这一部分存储了底层物理设备存储的元数据，如果块映射数据损坏，数据就相当于丢了。后面的磨损调度和垃圾回收等都依赖块映射，还是非常重要的。

这一部分功能对 SSD 性能影响也非常大，基本没有厂商披露这方面信息，但是我们还是从一些学术论文里了解一些内容。

目前主流的映射方式有三种:

1) 基于 Page 映射，这种方式一个逻辑页可以映射到任何一个物理页。映射粒度最小，性能最好，也最耗费内存
2) 基于 Block 映射，这种方式一个逻辑页可以映射到任何一个特定的 Block。映射

粒度最大，性能最差，不过也最节省内存

3) Page+Block 混合映射，这种方式性价比介于上面两种方式之间，映射逻辑页更为复杂，本文就不详细描述了

2. Wear leveling

之前我们提到了组成 SSD 的 cell 的寿命是有限的，比如 MLC 在 3000 次，而如果你像传统硬盘一样一直在擦写一个扇区可能影响并不大，但是对于 SSD 来说却是致命的。为了提升 SSD 的寿命和写入速度，在 FTL 层引入了 Wear leveling 算法，这个算法的一个最重要的目的就是保证 SSD 所有的 cell 的擦除次数尽可能统一，最终大大提升了 SSD 的可用性和寿命。

我们可以先做个对比，假设 SSD 上有一个文件占用了 100 个 block，整个 SSD 包含 8196 个 block，我们如果每十分钟更新一次整个文件，每个 cell 最大擦写次数是 3000 次，更新过程需要的 200 个 block。

首先是在没有磨损均衡情况下

SSD 的寿命：

$$\frac{5000\text{ 次 }*200}{100*24}=416\text{ 天}$$

而如果有磨损均衡算法情况下 SSD 寿命

$$\frac{5000\text{ 次 }*8096}{100*24}=16866\text{ 天}\approx 46\text{ 年}$$

从这个简单的例子可以看出磨损调度算法的重要性。

目前磨损调度有下面两种：

| | 优点 | 缺点 |
|---|---|---|
| 动态磨损 | 速度快，逻辑简单，对性能影响小 | 提升寿命效果非常差 |
| 静态磨损 | 最大化 SSD 寿命，磨损算法更健壮 | 控制器逻辑复杂，对性能有影响，更高的功耗 |

3. Garbage collection

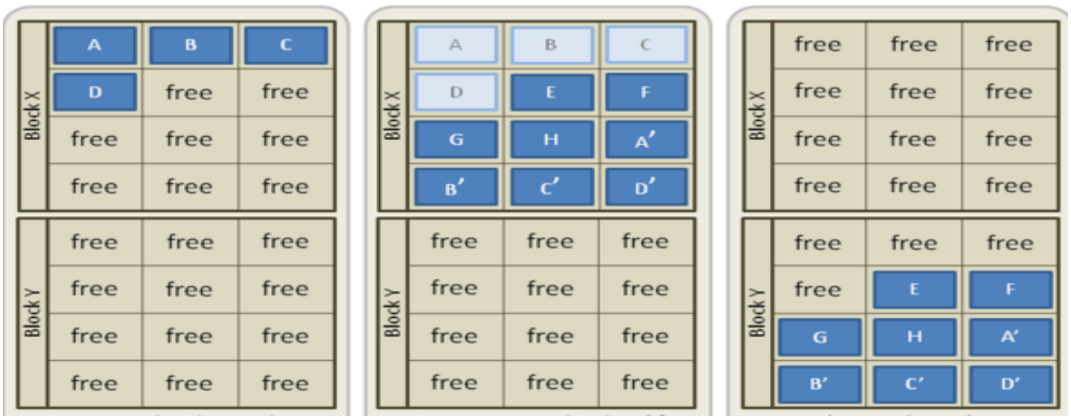
垃圾回收指的是回收已经失效的数据页，当然最终回收的时候还是以 block 为单位的。为什么 SSD 需要垃圾回收，而传统硬盘不需要呢。主要有两点原因构成：不支持 In Place Update 和擦除单位最小是 block。

我们先介绍另外一个相关的概念 Write amplification(写入放大)

$$\frac{\text{data written to the flash memory}}{\text{data written by the host}}=\text{write amplification}$$

上面的公式右边被称为写入放大因子，公式是放大因子计算方法。写入放大的原因也是我们之前提到的 SSD 不支持数据覆盖写。从 SSD 写入原理来看，写入放大因子一般是大于 1 的，目前企业级 SSD 的写入放大因子一般都在 1.1 左右。

下面我们看一下一个非常经典的垃圾回收流程图




1.首先在数据块 X 的四个可用页里写入 A-D

2.在数据块 X 写入 E-H,然后对 A-D 进行更新，然后把新数据写入到 A'-D',A-D 标记为无效

3.为了使 A-D 数据页可用，就只能把 E-H 和 A'-D'复制到数据块 Y，然后擦除数据块 X

现在垃圾回收一般都是在后台去执行，垃圾回收对读写影响还是会比较大的，所以要控制好这个速度，既要保证当前可用空闲块比例，也要保证尽可能不要影响写入能力。

提到垃圾回收，还涉及到另外两个概念:

1) Trim

Trim 是一个 ATA 指令。由操作系统发送给 SSD 主控制器，告诉它哪些数据占的地址是“无效”的，避免不必要的垃圾回收迁移数据。

2) Over-provisioning

OP 简单来说就是预留空间，SSD 出厂时候一般都有预留，出厂 OP 空间比例在 10%左右，用户也可以自己预留 OP。OP 可以用于磨损调度和垃圾回收，OP 空间越大，性能越好越稳定，寿命越久，写入放大比例越低。

上面的 Trim 和 OP 都是对垃圾回收效率和提高寿命有好处的。

4. Bad Block Management(坏块管理)和 ECC

关于坏块管理，先说一下坏块的来源，第一是出厂时候有一定的坏块比例，这个比例 MLC 一般控制在 5%，SLC 一般控制在 2%。第二是随着读写次数增加，导致 cell 电子泄漏，无法修复就变成坏块。控制器会维护一张坏块表，记录哪些数据块是不可用的。

ECC 算法是保证 SSD 数据可靠性和寿命的一个非常重要的特性。尤其在 NAND 制程越来越低的情况下，cell 之间的距离越来越小，读写都有可能造成对邻近 cell 的状态变化。

现在有以下几种 ECC 算法

1) Hamming 算法
2) Reed-Solomon 算法

应用非常广泛的 ECC 算法，之前用于 MLC

3) BCH
4) LDPC

在 NAND 制程降到十几纳米时候，需要更高的可靠性，有取代 BCH 的趋势

另外 SSD 不只是在从 NAND 读取数据时需要进行 ECC 校验，在整条数据链路上都有 ECC 和 CRC 的校验，数据的安全性还是非常高的。

在 NAND 制程越来越小，而对性能和延时要求越来越高，设计一个高效且稳定的 ECC 算法还是非常有挑战的。

六．当前企业级 SSD 对比和选择

目前企业级 SSD 主要是两种类型，分别是 SATA SSD 和 PCIe SSD，大家应该对 SATA SSD 更熟悉一点。

下面这个表是 PCIe 和 SATA 的对比（数据参考目前主流产品）

| | 容量 | 4K随机读/写 | 每GB | 带宽 | 延时 |
|---|---|---|---|---|---|
| PCIe | 1TB+ | 70w+/20w+ | 2-3$ | 8GB/s | 10-40 μs |
| SATA | <1TB | 7w/2w | 1-2$ | 6Gb/s | 50-100μs |

我们选择 SSD 需要考虑哪些因素呢

价格，容量，性能，可靠性，稳定性，IO 延时，可维护性，功耗

七.使用场景和实践

主要使用场景

1) 高 IOPS，尤其随机读写，低读写延迟，比如 MySQL Oracle 等关系数据库
2) 混合存储

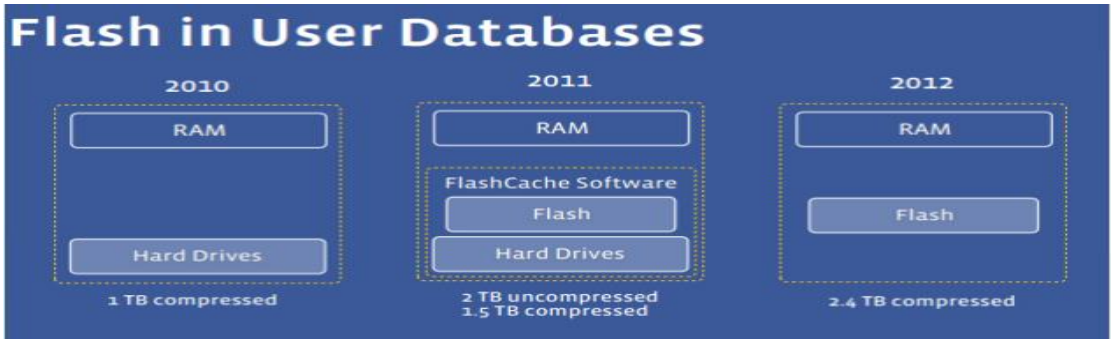
   把 SSD 和 HDD 通过软件或硬件方式虚拟成一个设备使用，比如 Facebook 的 flashcache。
3) 缓存和 NoSQL

   把 SSD 当做 DRAM 的二级存储，存储长尾或更冷的数据，虽然牺牲一点性能，但是带来的扩展性和成本的降低都是非常大的。这个使用的案例也是非常多的

我们先看一下 SSD 在互联网公司 MySQL 的使用实践

不得不说，SSD 解决了 MySQL 的随机 IO 瓶颈，否则不少人还是停留在不停的 Sharding，更重要的是响应时间的降低，这个不是 sharding 和堆机器能搞定的，硬件的性能天花板是突破不了的。

下图是 Facebook 的 MySQL 数据库硬件选型演变




2010 年全部使用机械硬盘，2011 年使用 Flash+HDD,也就是上面提到的 Flashcache 混合存储方案，2012 年开始全 Flash 化

为什么会有这个选型的变化呢，主要有下面几个原因吧

1. SSD 成本越来越低，这个是最重要的原因
2. 用户访问量的增长
3. 用户对于访问响应时间更高的要求。在现在这种互联网格局下，应用的用户体验也越来越重要。

关于 2011-2012 年，由混合存储变为全 flash，成本不一定会降低，但是为什么要这么做

呢。这个其实就要考虑混合存储的应用场景问题，混合存储比较适合做离线业务，如果在 OLTP 场景使用混合存储，场景的访问最好有明显的热点，使用的效果才会好，所以限制还是比较大。另外使用混合存储还是考虑两个问题，一是带来的存储架构复杂性，运维成本会增加，出问题的风险会增加，二是响应时间的不稳定性，使用混合存储能带来性能的提升过于单一，不具有通用性。

那我们在什么时候使用混合存储呢。如果用 SSD 和 HDD 来说，一是 SSD 的成本远高于 HDD 的时候，也就是在 2011-2012 年，现在明显不适合了，还有一个就是本身数据量大访问也相对集中，这时候也是可以考虑使用混合存储的，综合评估上面的风险点和带来的收益，牺牲一点通用性。

新浪微博核心数据库也有在使用 flashcache，解决了我们那个阶段的问题，整体还是利大于弊的，当然现在重新审视这种方案可能已经不那么适合。

这个过程可以对比一下国内大型互联网公司的数据库硬件变化，其实还是很相似点的。国内比较早的使用 SSD 实践应该是从阿里的去 IOE 开始的，这个话题大家应该都有过了解，目前来看去 IOE 之后也是非常成功的。可以说这个过程如果没有 SSD 的使用，去 IOE 还是很困难的。

SSD 高随机读写性能天热的适合关系数据库场景，现在 SSD 在国内互联网公司已经非常普遍，各个公司都在使用 SSD 上积累了不少经验。

下面介绍几个对性能优化比较重要的点

关于 SSD 做 Raid 问题，早期一般使用 SATA SSD，考虑到稳定性和容量，一般会做 Raid5，牺牲一块盘的容量，不过随着 SSD 性能不断提升和单盘 SSD 容量增长，做 Raid 牺牲的容量也越来越多，Raid 卡吞吐量现在已经成为 SSD 的性能瓶颈，如何突破这个瓶颈发掘 SSD 的性能也是目前面临的一个非常重要的问题。其实当 SATA SSD 容量达到 TB 级时，是可以考虑通过 HBA 卡直连 SSD 方案，性能会有成倍的提升，不过对 SSD 稳定性要求比较高。选择 PCIe SSD 就可以绕过 Raid 卡这个瓶颈，通过 PCIe 总线直连 CPU，不存在 SATA SSD 做 Raid 的性能瓶颈。

关于 SATA SSD 和 PCIe SSD 选择上，如果从价格和性能去权衡，对价格更敏感的最好选择 SATA SSD，对性能和响应时间要求高的选择 PCIe。还有另外一个方面也要考虑到，就是稳定性问题，SATA SSD 的驱动是在内核的，而 PCIe 的驱动基本都是各个厂商各搞各的，在系统兼容性和稳定性相对会差一些。不过从长远来看，我还是更倾向选择 PCIe SSD 的，尤其在今年开始推广 NVMe，PCIe 的稳定性也会逐步完善。

在文件系统上，建议选择 xfs，关闭文件系统预读，IO 调度算法设置为 deadline 或 noop，禁用 Raid 卡预读。对于 MySQL 优化，把顺序读写的部分迁移到 HDD 上，比如 binlog redolog 等，采用更大的 redo 文件，使用 InnoDB 或 Tokudb 压缩，减少实际写入到 SSD 的数据，减少写入放大，使用更小的 InnoDB 数据页。

做好对线上 SSD 的监控，如果是 SATA SSD，一般都是支持 smart 特性的，可以用 smart 工具去做监控，里面重点监控磨损程度和读写数据量，这一点对于保证线上业务稳定性以及检测 SSD 产品的可靠性都是非常重要的。如果使用 PCIe，厂商一般也都有相应工具去支持来做重要性能参数的监控。

SSD 和 NoSQL 结合

这里要先说一下 SSD 在系统设计中的两种使用方式

1) SSD 做磁盘的扩展，提高磁盘速度，就是我们上面提到的混合存储
2) SSD 做内存的扩展，把内存扩展更大

这两种方式有一个共同点就是成本问题，就是在面对大数据量时候存储的成本问题

目前 SSD 在 NoSQL 领域使用主要采用第二种方式。SSD 在这种方式开始使用有两个原因

1) DRAM 和 SSD 相比成本还是会高很多
2) 数据中心网络的延时已经和 SSD 的访问延时接近，有必要尽可能把数据 local 到本地，提高数据传输效率和降低访问时间

按照这种理念设计的产品还是很多的，比如 Twitter 的 Fatcache，Facebook 的 RocksDB 和 McDipper，国内的 TSSD 、SSDB 等。基本原理是索引在内存，数据持久化在 SSD，每个产品实现上实现有区别的，一般也都会对写入进行合并、压缩等，以减小对 SSD 寿命的损耗。

在这种场景下使用 SSD，我更倾向于使用 PCIe SSD，提供和内存更接近的访问时间以及更高的读写性能。现在很多互联网公司已经在线上用类似产品，效果都还不错，并且在这些场景下的 SSD 尝试是非常有意义的。

虽然现在基于 SSD 的 NoSQL 产品很多，但是目前看还没有一个普及度高的产品，健壮性和稳定性都需要检验，选择时候都要慎重，数据的安全性和运维复杂度的提高是一个很大的挑战。

八．总结

现在 SSD 已经是一项成熟的技术，我们可以根据我们遇到的场景去选择尝试使用 SSD。当然 SSD 要合理使用，不要滥用，从软硬件整体去做优化，要尽可能去优化和发掘 SSD 的潜力。合适你的才是最好的，**One size doesn't fit all !**

**附联系方式**