ZFS 分層架構設計

目次

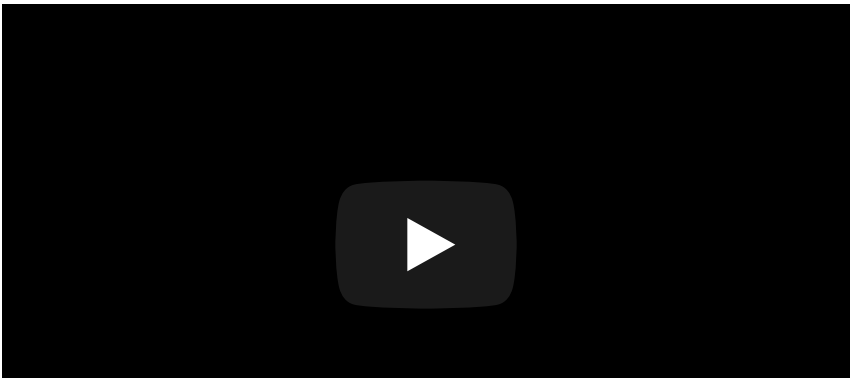
Contents

- L2ARC
- TOL
- DMU
- ZAP
- DSL
- ZIL
- ZVOL
- ZPL

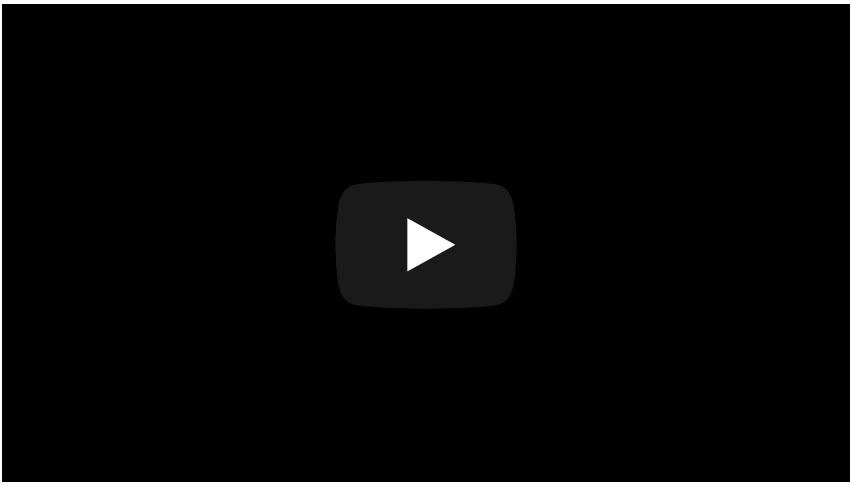
ZFS 在設計之初源自於 Sun 內部多次重寫 UFS 的嘗試，背負了重構 Solaris 諸多內核子系統的重任，從而不同於 Linux 的文件系統只負責文件系統的功能而把其餘功能（比如內存髒頁管理， IO調度）交給內核更底層的子系統， ZFS 的整體設計更層次化並更獨立，很多部分可能和 Linux 內核已有的子系統有功能重疊。

這篇筆記試圖從 ZFS 的早期開發歷程開始，記錄一下 ZFS 分層架構中各個子系統之間的分工。 也有幾段 OpenZFS Summit 視頻佐以記錄那段歷史。

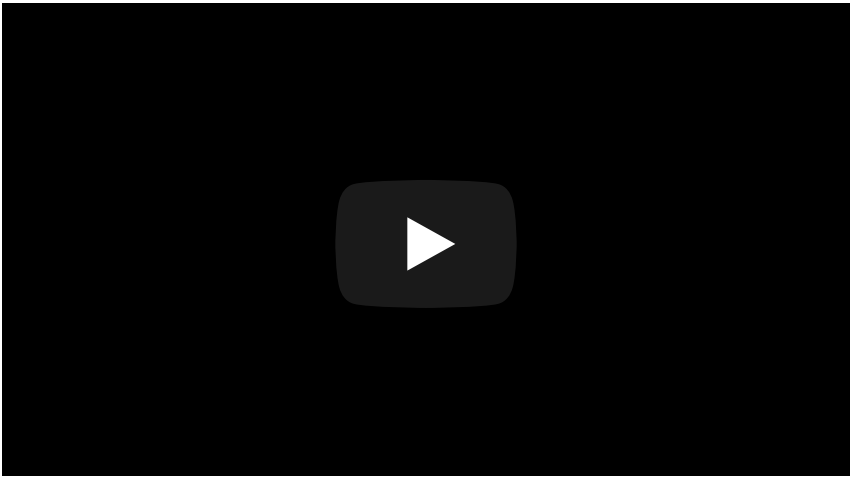
The Birth of ZFS by Jeff Bonwick



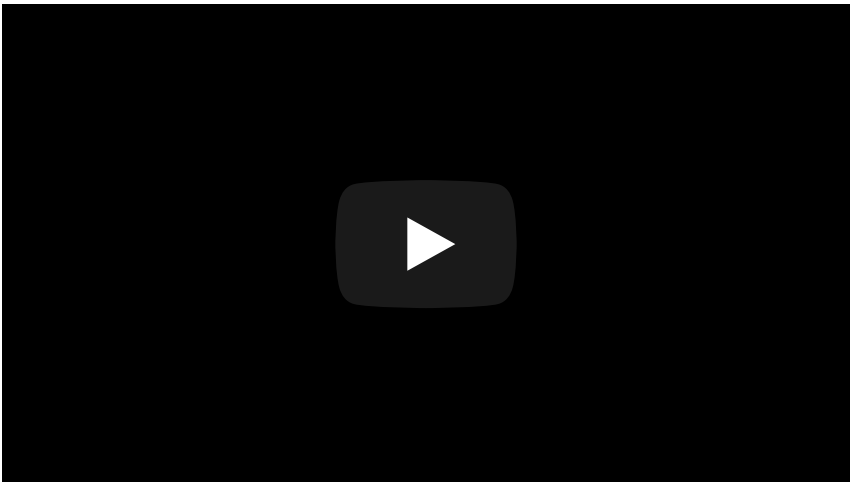
Story Time (Q&A) with Matt and Jeff



ZFS First Mount by Mark Shellenbaum



ZFS past & future by Mark Maybee



早期架構

早期 ZFS 在開發時大體可以分爲上下三層，分別是 ZPL， DMU 和 SPA ，這三層分別由三組人負責。

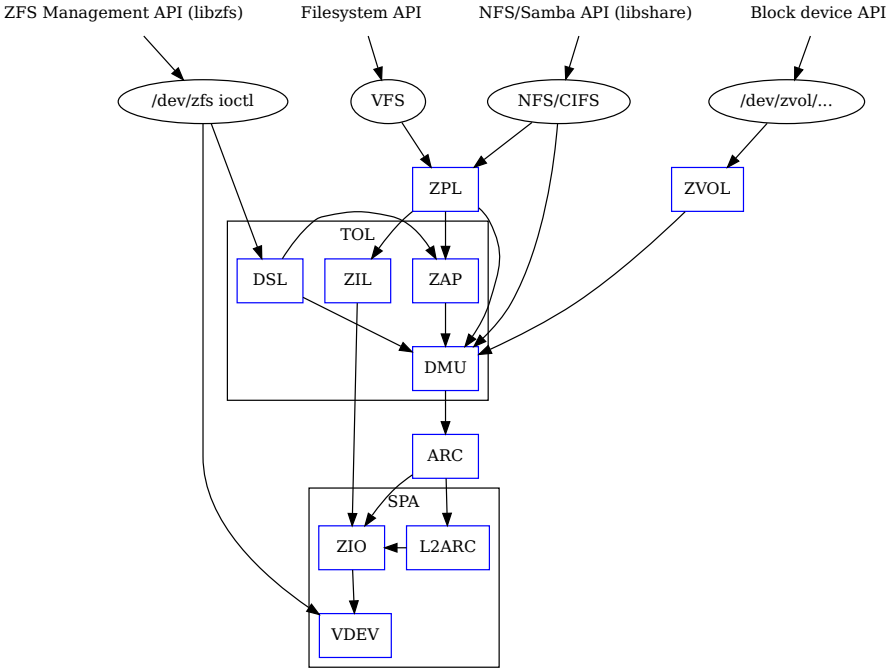
最初在 Sun 內部帶領 ZFS 開發的是 Jeff Bonwick ，他首先有了對 ZFS 整體架構的構思，然後遊說 Sun 高層，親自組建起了 ZFS 開發團隊，招募了當時剛從大學畢業的 Matt Ahrens 。作爲和 Sun 高層談妥的條件，Jeff 也必須負責 Solaris 整體的 Storage & Filesystem

Team，於是他也從 Solaris 的 Storage Team 抽調了 UFS 部分的負責人 Mark Shellenbaum 和 Mark Maybee 來開發 ZFS。而如今 Jeff 成立了獨立公司繼續開拓服務器存儲領域， Matt 是 OpenZFS 項目的負責人，兩位 Mark 則留在了 Sun/Oracle 成爲了 Oracle ZFS 分支的維護者。

在開發早期，作爲分工， Jeff 負責 ZFS 設計中最底層的 SPA，提供多個存儲設備組成的存儲池抽象； Matt 負責 ZFS 設計中最至關重要的 DMU 引擎，在塊設備基礎上提供具有事務語義的對象存儲； 而兩位 Mark 負責 ZFS 設計中直接面向用戶的 ZPL，在 DMU 基礎上提供完整 POSIX 文件系統語義。

子系統整體架構

首先 ZFS 整體架構如下圖，其中圓圈是 ZFS 給內核層的外部接口，方框是 ZFS 內部子系統：


ZFS Management API (libzfs)
Filesystem API
NFS/Samba API (libshare)
Block device API
/dev/zfs ioctl
VFS
NFS/CIFS
/dev/zvol/...
ZPL
ZVOL
TOL
DSL
ZIL
ZAP
DMU
ARC
SPA
ZIO
L2ARC
VDEV

接下來從底層往上介紹一下各個子系統的全稱和職能。

SPA

Storage Pool Allocator

從內核提供的多個塊設備中抽象出存儲池的子系統。 SPA 進一步分爲 ZIO 和 VDEV 兩大部分。

SPA 對 DMU 提供的接口不同於傳統的塊設備接口，完全利用了 CoW FS 對寫入位置不敏感的特點。 傳統的塊設備接口通常是寫入時指定一個寫入地址，把緩衝區寫到磁盤指定的位置上，而 DMU 可以讓 SPA 做兩種操作：

1. `write` ， DMU 交給 SPA 一個數據塊的內存指針， SPA 負責找設備寫入這個數據塊，然後返回給 DMU 一個 block pointer 。
2. `read` ，DMU 交給 SPA 一個 block pointer ，SPA 讀取設備並返回給 DMU 完整的數據塊。

VDEV

Virtual DEVice

作用相當於 Linux 內核的 Device Mapper 層或者 FreeBSD GEOM 層，提供 Stripe/Mirror/RAIDZ 之類的多設備存儲池管理和抽象。 ZFS 中的 vdev 形成一個樹狀結構，在樹的底層是從內核提供的物理設備， 其上是虛擬的塊設備。每個虛擬塊設備對上對下都是塊設備接口，除了底層的物理設備之外，位於中間層的 vdev 需要負責地址映射、容量轉換等計算過程。

ZIO

ZFS I/O

作用相當於內核的 IO scheduler 和 pagecache write back 機制。 ZIO 內部使用流水線和事件驅動機制，避免讓上層的 ZFS 線程阻塞等待在 IO 操作上。 ZIO 把一個上層的寫請求轉換成多個寫操作，負責把這些寫操作合併到 transaction group 提交事務組。 ZIO 也負責將讀寫請求按同步還是異步分成不同的讀寫優先級並實施優先級調度， 在 OpenZFS 項目 wiki 頁有一篇描述 ZIO 調度 的細節。

除了調度之外， ZIO 層還負責在讀寫流水線中拆解和拼裝數據塊。上層 DMU 交給 SPA 的數據塊有固定大小， 目前默認是 128KiB ，pool 整體的參數可以調整塊大小在 8KiB 到 8MiB 之間。ZIO 拿到整塊大小的數據塊之後，在流水線中可以對數據塊做如下操作：

1. 用壓縮算法，壓縮/解壓數據塊。
2. 查詢 dedup table ，對數據塊去重。
3. 如果底層分配器不能分配完整的 128KiB （或別的大小），那麼嘗試分配多個小塊，多個用 512B 的指針間接塊連起多個小塊的 gang block 拼成一個大塊。

可見經過 ZIO 流水線之後，數據塊不再是統一大小，這使得 ZFS 用在 4K 對齊的磁盤或者 SSD 上有了一些新的挑戰。

ARC

Adaptive Replacement Cache

作用相當於 Linux/Solaris/FreeBSD 中傳統的 page/buffer cache 。 和傳統 pagecache 使用 LRU (Least Recently Used) 之類的算法剔除緩存頁不同，

ARC 算法試圖在 LRU 和 LFU(Least Frequently Used) 之間尋找平衡，從而複製大文件之類的線性大量 IO 操作不至於讓緩存失效率猛增。

不過 ZFS 採用它自有的 ARC 一個顯著缺點在於，不能和內核已有的 pagecache 機制相互配合，尤其在 系統內存壓力很大的情況下，內核與 ZFS 無關的其餘部分可能難以通知 ARC 釋放內存。所以 ARC 是 ZFS 消耗內存的大戶之一（另一個是可選的 dedup table），也是 ZFS 性能調優 的重中之重。

和很多傳言所說的不同， ARC 的內存壓力問題不僅在 Linux 內核會有，在 FreeBSD 和 Solaris/Illumos 上也是同樣，這個在 ZFS First Mount by Mark Shellenbaum 的問答環節 16:37 左右有提到 。其中 Mark Shellenbaum 提到 Matt 覺得讓 ARC 併入現有 pagecache 子系統的工作量太大，難以實現。

L2ARC

Level 2 Adaptive Replacement Cache

這是用 ARC 算法實現的二級緩存，存在於高速存儲設備上。通常 ZFS 可以配置一塊 SSD 作爲高速緩存， 減輕內存 ARC 的負擔並增加緩存命中率。

TOL

Transactional Object Layer

這一部分子系統在數據塊的基礎上提供一個事務性的對象語義層。最主要的部分是 DMU 層。

DMU

Data Management Unit

在塊的基礎上提供「對象」的抽象。每個「對象」可以是一個文件，或者是別的 ZFS 內部需要記錄的東西。 DMU 這個名字最初是 Jeff 想類比於操作系統中內存管理的 MMU(Memory Management Unit)， Jeff 希望 ZFS 中增加和刪除文件就像內存分配一樣簡單，增加和移除塊設備就像增加內存一樣簡單， 由 DMU 負責從存儲池中分配和釋放數據塊，對上提供事務性語義，管理員不需要管理文件存儲在什麼存儲設備上。 這裏事務性語義指對文件的修改要麼完全成功，要麼完全失敗，不會處於中間狀態，這靠 DMU 的 CoW 語義實現。

ZAP

ZFS Attribute Processor

在「對象」基礎上提供緊湊的 name/value 映射，從而文件夾內容、文件屬性之類的都是基於 ZAP 。

DSL

Dataset and Snapshot Layer，數據集和快照層。

ZIL

ZFS Intent Log，記錄兩次完整事務語義提交之間的 log，用來加速實現 fsync 之類的保證。

ZVOL

ZFS VOLume

有點像 loopback block device，暴露一個塊設備的接口，其上可以創建別的 FS。對 ZFS 而言實現 ZVOL 的意義在於它是比文件更簡單的接口所以一開始先實現的它，而且 早期 Solaris 沒有 sparse 文件的時候可以用它模擬很大的塊設備，測試 Solaris UFS 對 TB 級存儲的支持情況。

ZPL

ZFS Posix Layer，提供符合 POSIX 文件系統的語義，也就是包括文件、目錄這些抽象以及 inode 屬性、權限那些，對一個普通 FS 而言用戶直接接觸的部分。