

Hybride Speicherarchitekturen mit ZFS

Transparente Nutzung von SSDs

Franz Haberhauer
Sun Microsystems GmbH
Zettachring 10a
70567 Stuttgart
Deutschland
<franz.haberhauer@sun.com>

1 Abstract

Seit kurzem sind Solid State Disks (SSDs) auf der Basis von Flash-Speicher erhältlich, die sich von ihrer Zuverlässigkeit her für den Einsatz in der Unternehmens-IT eignen. Durch die Nutzung von SSDs in einer zusätzlicher Schicht in der Speicherhierarchie zwischen Hauptspeicher und klassischen Plattenlaufwerken lassen sich Speichersysteme bauen, die sehr performant und zugleich kostengünstig sind. Dazu ist allerdings auch eine entsprechende Unterstützung durch das Betriebssystem erforderlich. Solaris ZFS erlaubt eine transparente Nutzung von SSDs einerseits zur Beschleunigung von synchronen Schreiboperationen durch die Nutzung für den Intentlog und andererseits um Leseoperationen in Form einer schnellen zusätzlichen Cache-Ebene zu beschleunigen. Während die Kapazität konventioneller Platten in den letzten Jahren enorm anwuchs, ist die Bandbreite für Direktzugriffe (IOPS) nur unwesentlich gestiegen. SSDs sind nun in der Lage hier eine enorme Bandbreite bereitzustellen. ZFS ermöglicht hybride Speicherarchitekturen, die die Kapazität konventioneller Laufwerke mit dem Performance-Potential von SSDs kombinieren.

Nachfolgend wird zunächst die SSD-Technologie vorgestellt. Anschliessend wird auf die Nutzung in Verbindung mit ZFS eingegangen und aufgezeigt, wie mit Open Source Technologien leistungsfähige hybride Speicherarchitekturen aufgebaut werden können, die auf General Purpose Hardware das Konzept von Open Storage realisieren. Abschliessend wird der Einsatz in Anwendungsszenarien analysiert.

2 Solid State Disks (SSDs)

Flash-Speicher sind nichtflüchtige Halbleiterspeicher, die elektrisch gelöscht und neu beschrieben werden können. Dabei kommen unterschiedliche Technologien zum Einsatz, die sich in einem breiten Spektrum beim den Leistungswerten wie auch im Preis von SSDs widerspiegeln: Es gibt Flash-Speicher in NOR- und NAND-Technologie, wobei NAND-Flash kompakter und daher mit größeren Kapazitäten verfügbar und zudem bei gleicher Kapazität kostengünstiger ist. Neben SLC-Flash (Single Level Cell) wird auch günstigeres MLC-Flash (Multi Level Cell) angeboten, bei dem nicht ein Bit sondern zwei bis vier in einer Zelle gespeichert werden. MLC-Flash ist dadurch deutlich günstiger, dafür aber langsamer beschreibbar und defektanfälliger, daher wird MLC-Flash insbesondere für den Massenmarkt verwendet [1].

Einzelne Flash-Zellen überstehen nur eine beschränkte Zahl von Löschkzyklen, eine typische Zahl sind 100.000 Zyklen, wobei diese im letzten Jahr signifikant weiter gesteigert werden konnte. Um diese Schwäche auszugleichen, verteilen Flash-Controller Schreibzugriffe auf logische dieselben Stellen physisch auf unterschiedliche Zellen. Über dieses sogenannte Wear-Leveling differenzieren sich die Hersteller von Flash-Controllern. Darüber hinaus werden Reserve-Zellen oder -Blöcke über die Nennkapazität hinaus vorgehalten, die abgenutzte Zellen ersetzen können. Desweiteren muss vor dem Neubeschreiben immer ein ganzer Block gelöscht werden, der je nach Typ 64 oder 128KB umfassen und was 1ms dauern kann. Auch hier können Controller durch das Bereitstellen von Pools mit leeren Blöcken und eine geeignete Abbildung Abhilfe schaffen.

Heute sind Enterprise Solid State Disks mit Formfaktoren von 2.5 oder 3.5 Zoll und SATA-II Interface verfügbar mit Kapazitäten zwischen 16 GB und 128 GB und teilweise zusätzlichem DRAM Cache, der bei Spannungsverlust über Kondensatoren noch mit Strom versorgt um die Daten auf den Flash-Speicher auszuschreiben. Während manche Laufwerke für den Massenmarkt beim sequentiellen Zugriff und beim Schreiben im Vergleich zu schnellen konventionellen Platten Platten schlecht abschneiden[1], sind Enterprise Solid State Disks mit 250 MB/s sequentieller Lese- und 150MB/s Schreib-Performance sowie zum Beispiel 30.000 Lese-IOPS und 7.000 schreibenden IOPS erhältlich mit einer Latenz von unter 100 us. Diese Raten – sowohl die absoluten Zahlen, wie auch das Verhältnis zwischen den IOPS beim Lesen und Schreiben variieren stark je nach Produkt – und damit auch die Preise für ein Laufwerk, die in der Größenordnung von 1.000 bis 10.000 Euro liegen. Damit ist die Auswahl eines konkreten SSD-Produkts für einen spezifischen Einsatzzweck nicht ganz so offensichtlich wie bei traditionellen Plattenlaufwerken, bei denen sich die Leistungswerte etwa von Platten mit 10.000 rpm nicht so drastisch unterscheiden. Da es aufwändig ist, SSDs mit mit hohen absoluten bzw. im Verhältnis zum Lesen hohen Schreibraten zu bauen, liegen solche Platten am oberen Ende des Preisspektrum. Bei SSDs mit sehr unterschiedlichen maximalen Schreib- und Leseraten ist darauf zu achten, daß eine Platte unter Umständen mit der maximalen Schreibrate ausgereizt werden und nicht parallel dazu gelesen werden kann. SSDs benötigen lediglich um die 2 Watt für den Betrieb, haben dazu einen breiteren thermischen Betriebsbereich und sind weniger anfällig gegen Vibrationen. Die MBTF liegt mit 2 Millionen Stunden höher als die von Plattenlaufwerken (1,2 Millionen Stunden). [2]

In den letzten Jahrzehnten konnte zwar die Kapazität traditioneller Festplatten-Laufwerke dramatisch gesteigert werden – von 2 GB im Jahr 1992 über ca. 100 GB 2001 auf heute 2 TB. Die Zugriffsraten (gemessen in IO-Operationen pro Sekunde IOPS) sind allerdings nicht entsprechend gewachsen, sondern liegen weiterhin bei 100-500 IOPS. Auch mit Latenz liegt immer noch bei einigen Millisekunden.

Insbesondere die Plattenkonfigurationen transaktionsverarbeitender Systeme, aber auch vieler anderer IT-Systeme müssen primär auf die Anforderungen für Zugriffsraten ausgelegt werden – die Kapazitätsanforderungen sind damit dann in der Regel übererfüllt.

Betrachtet man die Kosten je IOPS, sind SSDs bereits heute sehr attraktiv: Festplatten liegen in der Größenordnung von \$1,25-2,50/IOPS während Enterprise SSDs bei \$0,02-0,20/IOPS liegen. Allerdings sind die Kosten bezogen auf die Kapazität meist noch zu hoch für ausschließlich auf SSDs basierende Lösungen.

Hier bieten sich hybride Architekturen an, die SSDs als einen neue Cache-Ebene in der Speicherhierarchie nutzen.

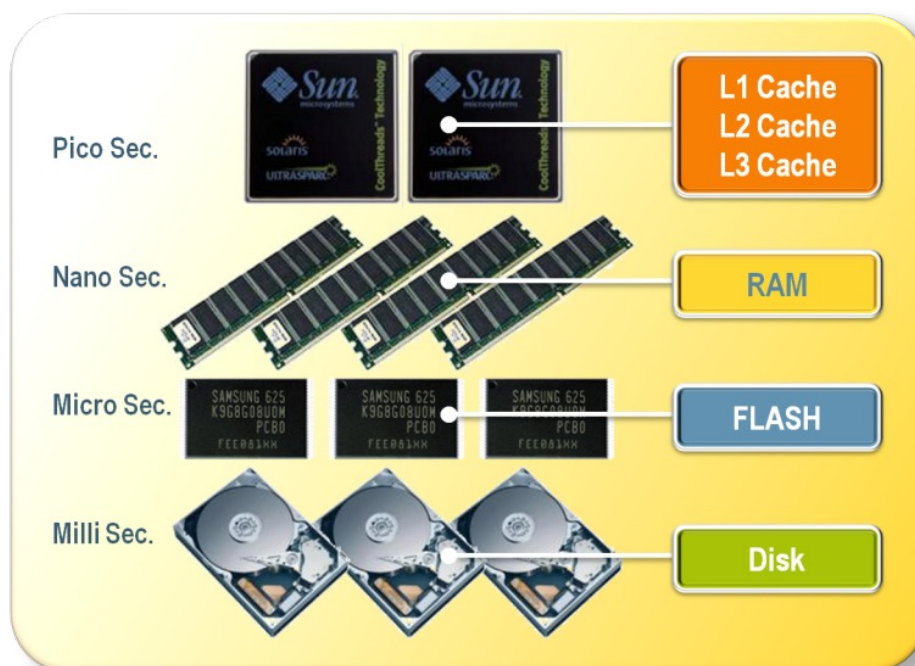


Abbildung 1: Die neue Speicherhierarchie

3 ZFS

Nicht nur bei den Speichermedien, auch bei den Dateisystemen gibt es eine Innovation, die im Markt ein sehr großes Interesse findet: ZFS, das in Solaris 10 und OpenSolaris integriert ist und weiterentwickelt wird [3], [4]. Mittlerweile wird es auch auf andere Plattformen insbesondere FreeBSD und MacOS X portiert. Die meisten heute gebräuchlichen lokalen Dateisysteme folgen Architekturprinzipien, die vor 20-40 Jahren von den Charakteristika der damals vorhandenen Plattensystemen geprägt wurden. ZFS bricht mit der Trennung von Dateisystem und Volumemanagement. Diese Integration eröffnet nicht nur neue funktionale Möglichkeiten sondern vereinfacht darüber hinaus die Administration. Platten werden in sogenannten Pools zusammengefasst, in denen jeweils ein oder auch mehrere Dateisysteme angelegt werden können. Dabei sind RAID-0, RAID-1 oder RAID-Z bzw. RAID-Z2 Konfigurationen möglich. Durch die Integration von Dateisystem und Volume-Management wurde mit RAID-Z eine sichere, gegen das "Write Hole"-Problem gefeierte Software-RAID-5 Implementierung möglich. Schreiboperationen werden hierbei immer als Full-Stripe-Writes ausgeführt. Durch die konsequente Verwendung von Prüfsummen können durch Software- oder Hardware-Probleme verfälschte Daten sicher identifiziert und die Datenintegrität sichergestellt werden. Viele dieser Fehler sind transient und können, falls die Daten redundant gespeichert sind, automatisch korrigiert werden, da ZFS bei einer Dateninkonsistenz gegebenenfalls von einem Spiegel die Daten auf der zweiten Spiegelhälfte lesen kann. Bei traditionellen Volumemanagern und Dateisystemen werden solche Fehler noch nicht einmal festgestellt und unter Umständen inkonsistente Daten an Anwendungen gegeben.

ZFS nutzt das Copy-on-Write- und das Transaktionsparadigma und stellt damit sicher, dass der Zustand auf der Platte immer konsistent ist, weswegen ein fsck grundsätzlich nicht nötig ist. Die Implementierung des atomaren Zustandsübergangs für das Transaktionsparadigma durch Ausschreiben eines sogenannten Überblocks als Wurzel der ZFS Baumstruktur erlaubt billig eine unbegrenzte Anzahl von billigen Snapshots – der alte Überblock wird nicht verworfen, sondern als Snapshot verfügbar gehalten. Snapshots können als sogenannte Clones auch geschrieben werden. Durch den Copy-on-Write-Mechanismus werden verstreute Schreiboperationen serialisiert, was sowohl den Leistungscharakteristika moderner Festplatten wie denen von SSDs entgegen kommt.

Über ZFS können für Anwendungen transparent hybride Speicherlösungen mit konventionellen Festplatten und SSDs implementiert werden, wobei es sogar möglich ist, gezielt spezifische Typen von SSDs einzusetzen, um eine optimale Performance zur erzielen.

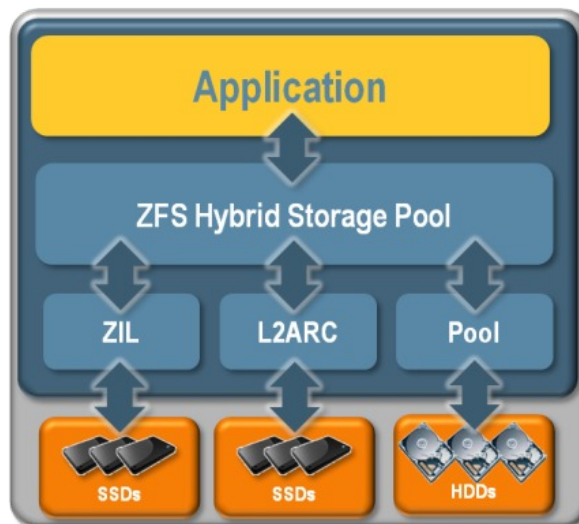


Abbildung 2: ZIL und L2ARC

SSDs können zur Beschleunigung von synchronen Schreiboperationen wie sie etwas von DBMS oder von NFS-Servern kommen genutzt werden. Hierfür wird der ZFS Intent Log (ZIL) auf eine SSD oder einen Pool von SSDs gelegt. Hierfür sollten SSD-Typen verwendet werden, die hohe Raten kleiner Schreiboperationen abarbeiten können.

ZFS verwendet den sogenannten Adaptive Replacement Cache (ARC) als primären Dateisystempuffer. Insbesondere um SSDs in der Speicherhierarchie effizient nutzen zu können wurde jetzt eine optionaler weiterer nachgelagerter Cache implementiert: der Level 2 ARC oder kurz L2ARC [5]. Ein SSD-basierter L2ARC kann deutlich größer sein als der ARC im RAM des Systems. Geeignete SSDs dafür sind solche mit möglichst großer Kapazität, die dabei Leseoperationen besonders effizient handhaben. Um diese unterschiedlichen Lastcharakteristika deutlich zu machen wird auch von "Writezilla" oder "Logzilla" für die schreib-optimierten SSDs für den ZIL und "Readzilla" für die lese-optimierten für den L2ARC gesprochen. Beim "Readzilla" ist zu beachten, dass von einer SSDs während des Schreibens nicht gelesen werden kann. Um Performance-Einbrüche hierdurch zu vermeiden sollten immer mindestens 2 SSDs für "Readzillas" verwendet werden. Die Algorithmen in diesen Komponente sind auf SSDs abgestimmt: so werden Hotspots vermieden und Speicherbereiche gleichmäßig geschrieben, um das Wear-Leveling zu unterstützen.

Während in OpenSolaris beide Funktionalitäten verfügbar sind, gibt es auf Solaris 10 ab Solaris 10/08 vorerst nur die Funktionalität für des separierten ZIL.

4 Open Storage

Betrachtet man das breite Spektrum an Storage-bezogenen Funktionalitäten, die OpenSolaris bietet [6], ist es naheliegend, dass sich damit auf der Basis von Standard-Hardware hervorragende Appliances bauen lassen.

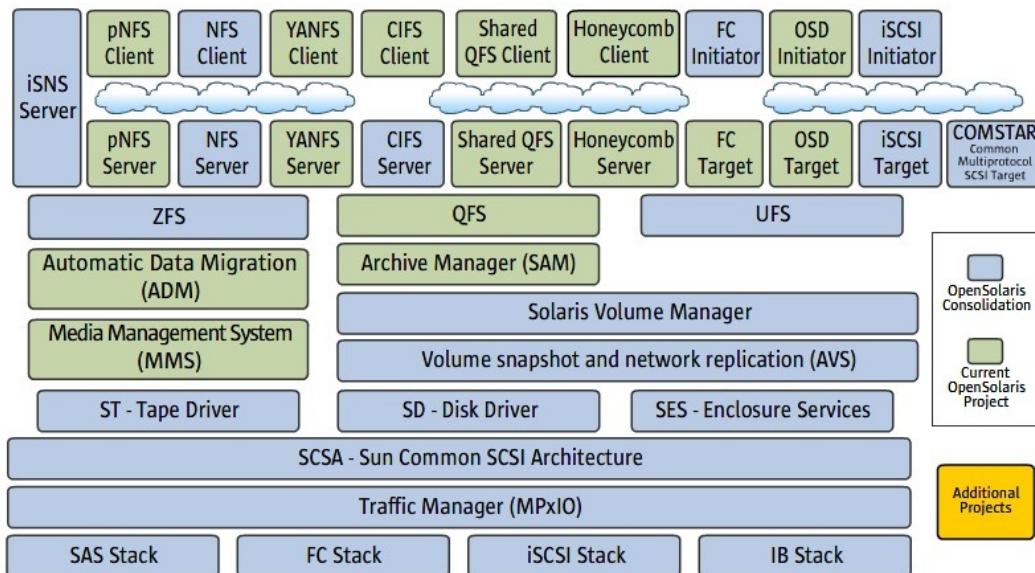


Abbildung 3: Storage-Projekte in OpenSolaris

Sun Microsystems hat Ende 2008 mit der neuen Produktlinie "Sun Storage 7000 Unified Storage System" [7] diese Idee umgesetzt. Es sind drei Systeme verfügbar, wobei in den beiden größeren SSDs konfiguriert werden können. Beim mittleren System als "Logzilla" - für 45 x 500 GB 7200 RPM SATA Disks ein 18 GB Write Flash Accelerator (schreiboptimierte SSD spezifiziert mit 16.000 schreibende IOPS) und für 44 x 1 TB 7200 RPM SATA Disks zwei 18 GB Write Flash Accelerator. Beim grossen System können für bis zu 576TB Platten bis zu 600GB (6 x 100GB) Read Flash Accelerator (leseoptimierte SSDs, spezifiziert mit 100MB/s Durchsatz sowie 15.000 IOPS beim Lesen und 1.500 IOPS beim Schreiben) konfiguriert werden. Dieses System kann auch in einer Hochverfügbarkeitskonfiguration mit zwei Cluster-Knoten betrieben werden. Hier befinden sich die Write Flash Accelerator ("Logzilla") im Speichersystem, das zwischen den Knoten geschwenkt wird. Die Read Flash Accelerator ("Readzilla") für den L2ARC in den jeweiligen Knoten, da über den L2ARC Metadaten im RAM der Systeme gehalten werden, die im Fehlerfall nicht übernommen werden.

Diese Appliances verfügen über sehr leistungsfähige, auf DTrace[8] basierende Performance-Analyse Werkzeuge. in der folgenden Grafik ist schön zu sehen, wie durch Aktivierung des "Logzilla" die Latenz der NFS-Operationen von 3-9 ms auf unter 1 ms fällt und wie Rate der NFS-Operationen drastisch ansteigt.

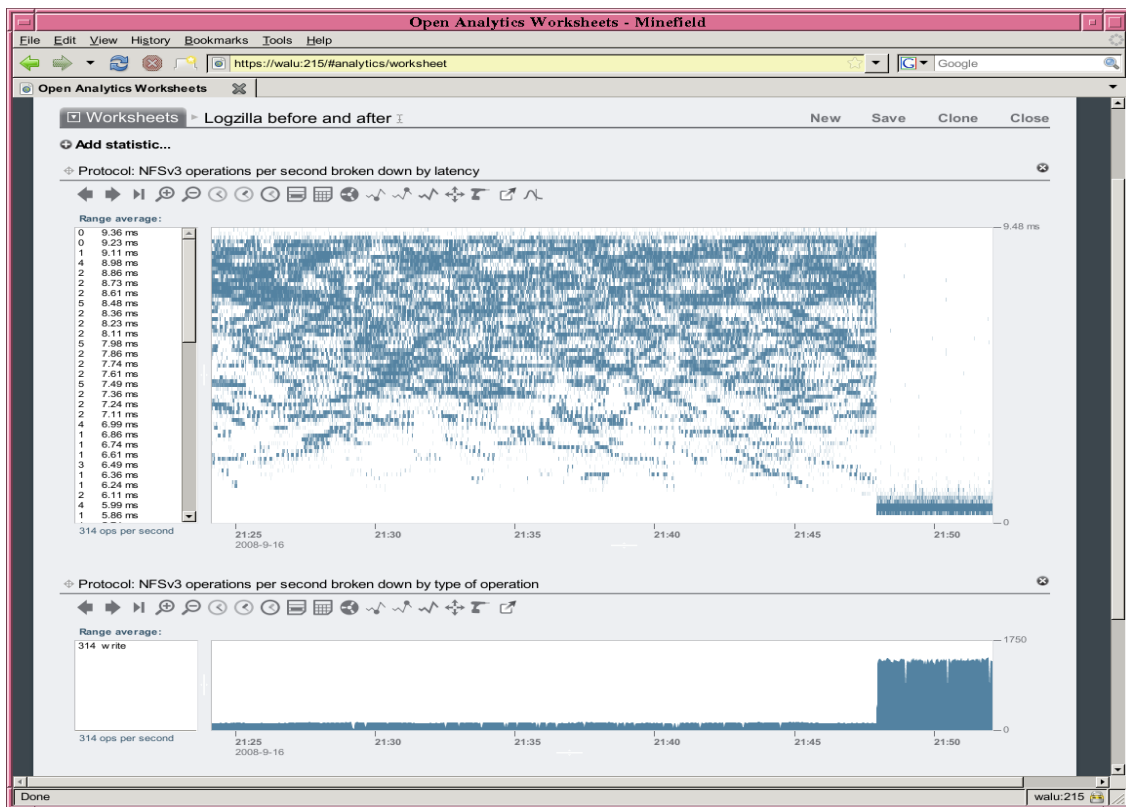


Abbildung 4: Auswirkung des Logzilla auf Latenz und Rate von NFSv3-Operationen

Weiterführende Beispiele für den L2ARC auf SSDs finden sich im Web [9].

5 Zusammenfassung

Dieser Artikel hat gezeigt, wie durch hybride Speicherarchitekturen signifikante Performance-Gewinne erzielt werden können. Solaris ZFS liefert die Infrastruktur für den transparenten Einsatz von Solid State Disks. Eine optimale Leistung kann durch spezifisch für ihren Einsatz ausgewählte SSDs erzielt werden. Sun Microsystems verwendet in der neuen Produktlinie „Sun Storage 7000 Unified Storage System“ solche SSDs. Hier wurden auf der Basis offener Komponenten - Open Solaris und „General-Purpose“- Hardware – leistungsstarke und preislich attraktive NAS-Appliances gebaut, die sich auch durch eine einfache graphische Management-Oberfläche auszeichnet. Insbesondere die Monitoring-Möglichkeiten („Analytics“) auf der Basis von Solaris DTrace sind extrem feingranular und weitreichend und eignen sich sehr gut, um die Einfluss von SSDs auf die NAS-Performance zu beobachten und zu analysieren.

References

- [1] Boi Feddern, Benjamin Benz: Überflieger. Solid State Disks für den Massenmarkt, *c't*, 2008, 21, S. 122-127
- [2] Roger Bitar: Deploying Hybrid Storage Pools With Sun Flash Technology and the Solaris ZFS File System, Sun Blueprint, 2008 <http://wikis.sun.com/display/BluePrints/Deploying+Hybrid+Storage+Pools+With+Flash+Technology+and+the+Solaris+ZFS+File+System>
- [3] Franz Haberhauer: ZFS, *Informatik-Spektrum*, Dezember 2006, S. 451 – 456
- [4] http://www.sun.com/software/solaris/data_management.jsp
- [5] Brendan Gregg: ZFS L2ARC <http://blogs.sun.com/brendan/entry/test>
- [6] <http://opensolaris.org/os/community/storage>
- [7] http://www.sun.com/storage/disk_systems/unified_storage/
- [8] <http://www.sun.com/solaris/observability.jsp>
- [9] http://blogs.sun.com/brendan/entry/l2arc_screenshots