**ZFS(Zettabyte File System)**

Geleneksel birim yönetimi ve dosya sistemi katmanlarını bir araya getiren ZFS,”copy on write” mekanizmasını kullanır.Klasik dosya sistemleri ve RAID dizilerinden çok daha farklıdır.Bu karmaşık yapıda bilinmesi gerekenler zpool,vdev ve devices’tır.

Ayrıca bu formatın sürekli gelişmesi için OpenZFS projesi başlatıldı.Unix tabanlı (Solaris,OpenSolaris) ve Unix benzeri FreeBSD,Linux dağıtımlarında kullanılabiliyor.

**ZPOOL :** En üstünde ki yapıdır.Her bir zpool içerisinde başka aygıtlar içeren vdev’ler bulundurur.Bir cihazda fiziksel olarak birden fazla zpool bulunur fakat her biri birbirinden kesinlikle farklıdır.Bir vdev’i birden fazla zpool paylaşamaz.

-İçerisinde “vdev” olmayan zpoollar kaybolur.

-ZFS dosya sisteminde zpool klasik bir RAID0 uygulaması değil, kompleks dağıtım mekanizmasının değişebileceği bir JBOD’dur.

-ZFS’de genellikle yazılan veriler diskteki boş alana göre halihazırdaki vdev’ler arasından dağıtılarak teoride tüm vdev’ler dolu hale gelir.Eğer herhangi bir vdev diğerine göre çok daha yoğun ve kritikse,atlanarak geçici süreliğine başkalarına yazma işlemi yapılabilir.

-Modern zpool yapılarında CACHE veya LOG vdev’lerine ait olan kayıplardan kurtulmak mümkün fakat küçük bir güç bozulması veya kesilmesi ile sistem hatasında LOG vdev hata alacağı için LOG vdev kaynaklı veri kaybı olabilir.

**VDEV :** Zpool’ları oluşturan sanal aygıtlara denir.Çoğu depolama amaçlı kullanılsa da CACHE , LOG , SPECİAL şekillerinde de kullanımı mevcuttur.

-ZFS dosya sisteminde vdev’lerin her biri 5 adet yapıya müsait.Bunlar tek bir cihaz,RAIDz1,RAIDz2,RAIDz3 ve aynalama(mirror) yapısıdır.

-RAIDz1,RAIDz2,RAIDz3 “diagonal parity RAID” olarak isimlendirilen özel yapılardır.Çeşitli eşlikler halinde bölünmüş diskler yerine RAIDz vdev’ler arasında bu eşlik bloklarını vdev’ler arasında yarı eşit şekilde dağıtır.

-RAIDz yapısı sahip olduğu eşlik blokları kadar disk kaybedebilir.Disk kaybı sonucunda zpool’da kaybolacaktır.

-Aynalama(Mirror) amaçlı kullanılan sanal aygıtlarda, her mirror bir vdev üzerinde,her blok ise vdev’de ki cihazda barındırılır.Mirror herhangi sayıda aygıt barındırabilir.

-Tek aygıta bağlı vdev tavsiye edilmez.Problem çıkması halinde özel veya depolama amaçlı kullanılan tüm zpool’lar kaybolur.

-Vdevler’ler cihazlardan,zpoollar ise vdev’lerden oluşur.

**DEVICE :** Zpool ve vdev’lere kıyasla daha anlaşılabilirdir.Rastgele blok erişimli aygıtlardır.

-HDD ve SSD’ler vdev’lerin yapı taşı olarak kullanılan blok aygıtlardır.

-Rastgele erişime izin verilen bir cihaz,tanımlayıcı olan herşeye erişim sağlayabilmektedir.Bu yüzden RAID dizileri ile ayrı aygıtlar şeklinde kullanılmalıdır.

-Hotspare amaçlı kullanılan cihazlar tek vdev’e ati değil,tüm havuza aittir.

**HOTSPARE**: RAID yapılarında disklerden birinin arızalanması durumunda bu diskin yerine geçecek olan diski ifade eder.

-Hotspare ile bozulan bir diskin yerine geçiş yapılması,RAID sistemlerinde “Rebuilding” olarak adlandırılırken,ZFS dosya sisteminde “resilvering” olarak adlandırılır.

**DATASET :** Veri kümesi herhangi bir başka klasör gibi gözükür.

-Her yerleşik dosya sisteminde olduğu gibi ZFS dosya sisteminin de kendine özgü bir özelliği vardır.Bu özelliklerden biri ise “Dataset” lere koyulan kotadır.

-“zfs set quota=100G poolname/datasetname şeklinde kullanılır.

-“pool/parent/child” şeklinde bir yol izlenir.

-Zvols : Blok aygıtı.Kayıt boyutu sınırı yoktur.

**BLOCKS :** ZFS sisteminde metadata da dahil olmak üzere tüm veriler bloklarda saklanır.Recordsize özelliği ile her veri kümesi için boyut sınırı koyulabilir.Önceden veri yazılmış blokların boyutunu değiştiremeyiz.Standat kayıt boyutu 128 KİB’dır.(4K ile 1M arasında ayarlanabilir.)

-Birden fazla dosya tek bir blokta bulunamaz.

**SECTORS :** Fiziksel olarak yazılan ve okunan en küçük aygıt/birimidir.Çoğu dosya sisteminde standart olarak 512 bayt kullanılsa da ZFS dosya sisteminde “ashift” denilen sektörlerin boyutunu ayarlamak mümkündür.

Ashift : Temelde bir sektörün boyutunu gösteren ashift “Binary” üs olarak kullanılır.(Ashift 9 ise 2 üzeri 9 yani 512 bayt yapar.)

-Yeni bir vdev eklenirken blok yeni blok cihazlar teorik olarak ayrıntıları görüntülenir ve edinilen bilgilere göre “ashift” atanır.

-Ashift standart olarak 9’dur ama tavsiye edilen değer 12 veya 13’dür.

**Copy-On-Write yapısı :** Standart bir dosya sisteminde dosyanın yerini değiştirmek için istenilen işlem yerine getirilir.Fakat ZFS’ de bu işlem gerçekti dese de gerçekte bu işlem yapılmaz.

-Copy onWrite işleminin sağlanması için öncelikle değişen bloğun yeni bir versiyonu yazılır.Daha sonrasında eski blokla ilgili olan bağlantının kaldırılması için metadata güncellenir ve yeni yazılan blok bağlanır.

-İşlem esnasında güç kesintisi olur ise eski versiyona sahip oluruz.

-Yalnızca dosya sisteminde değil disk yönetimi seviyesinde de bulunur.Bu şekilde hazırlanan RAID sisteminde sistem bir çöküntüye uğrasa da yeniden başlatma veya yazım aşamasında ki bozukluğun ZFS’i etkilemediği görünüyor.

**ZIL-ZFS Intent Log :** ZFS’de iki adet yazma operatörü bulunur ve iş yükü bakımından asenkrondur.Dosya sistemi bunları bir araya getirip toplu bir şekilde işler.Veri parçalanması en alt düzeye indirilir.

-ZFS’de veriler hemen normal depolamaya kayıt edilmez.İlk etapta “ZFS Intent Log” adı verilen bir depolama alanına işlenir.Daha sonrasında asenkron yazma işlemleri ile beraber TGX(İşlem grupları) halinde depolamaya aktarılır.

-Aktarım halinde herhangi bir hata dolayısıyla işlem sekteye uğrar ise bu durumda ZIL üzerinde ki veriler pool içeriye aktarımı esnasında okunacak,TXG üzerindekiler ise ana depolama birimine aktarılıp;bağlantı kesilecektir.

-Ayrıca Secondary LOG cihazı ile ZIL’i depolamada ki vdev’lerde tutmak yerine ayrı bir vdev oluşturup onun üzerinde tutar.

**Snapshots :** Canlı bir dosya sisteminde verilerin kayıtlarının bulunduğu bir işaretçi ağaç vardır.SS işlemi sonucunda bu ağacın bir kopyasını daha almış oluruz.Salt okunurdur.

-Cow sisteminde ki çalışma mantığına göre düşünürsek;Eğer ki SS eski bloklardan birine yönelim sağlar ise değişmez öyle kalır fakat eski bloklar ortadan kalkmadığı için boş alan olmayacaktır.

**Replication :** Snapshots işlemi sonrasında kayıtların tutulduğu işaret ağacıdır.”zfs send” komutu ile gönderilen isteği,ağaçla birlikte kayıtları da alarak kullanır.

-Arka arkaya 2 kere “zfs send” komutu kullanılır ise aynı dataset’ten iki adet olacaktır.Bu sorunun ortadan kalkması için “zfs send –i poolname/dataname@1 poolname/dataname@2” kullanımı ile iki işaretçi ağaç karşılaştırılır ve sadece @2 ‘de bulunanlar yeni referans olur.

**Inline Compression(Sıkıştırma) :** CoW yapısı sıkıştırma işlemini de olumlu yönde değiştirir.Diğer dosya sistemlerinde sıkıştırma işlemi sırasında boyut işleminde çok fazla sorun çıkıyordu.Fakat ZFS dosya sisteminde bu sorun ortadan kalkmıştır.

-ZFS üzerinden Inline Compression ayarı varsayılan olarak kapalı haldedir.Günümüzde ZLE,Gzip(1-9) veya LZD4 algoritmalarına da destek verilir.

-LZ4 : Oldukça hızlı bir şekilde sıkıştırma ve açma işlemi gerçekleştirir.Çok düşük sistemlerde bile perfomanslı bir şekilde çalışır.

-GZIP : Linux/Unix işletim sistemi kullanıcılarının çok sık kullandığı bir sıkıştırma algoritmasıdır.1’den 9 a kadar seviyeleri vardır.Sıkıştırma oranı arttıkça CPU kullanımı oranı da artar.

-LZJB : ZFS’nin kullandığı sıkıştırma algoritmasıdır.Kullanımdan kaldırılmış olup kullanılması önerilmez.

-ZLE(Zero Level Encoding) : Bu algoritma da normal veriler olduğu gibi kalır,büyük sıfır dizileri ise algoritma yardımıyla sıkıştırılır.MP4,JPEG’leri görmezden gelir.

-Bu algoritmalar baz alındığında LZ4 sıkıştırması diğer algoritmalara göre daha performanslıdır.

**ARC(Adaptive Replacment Cache) :** En son okunan bloklara ait kopyaları RAM üzerinde saklamayıp kendi önbellekleme yöntemiyle saklama yapan tek dosya sistemidir.

-ARC mekanizmasına ait kullanılan belleğe ihtiyaç duyulursa “mallocate()” çağrısı başarısız sonuçlanabilir.

-Windows,BSD,Linux,Mac OS’da buna dahil oalrak LRU Algoritması aracılığıyla sayfa önbelleği mekanizmasını kullanırlar.

-LRU : Bellek yöneticisinin yeni bir sayfaya yer açmak için,hangi sayfayı dışarı alacağına karar veren algoritmadır.

**Lightweight File System :** Her kullanıcı için boş kota yoktur ancak dosya sistemi başına kota vardır böylece aynı dosya sisteminde ki klasörler yerine her kullanıcı için dosya sistemleri oluşturulur.

**Kapasite :** 128 bit dosya sistemidir.Bu sebeple 64 bit sistemlere göre çok daha fazla kapasiteye sahiptir.

Teorik olarak incelediğimiz de;

Bir klasördeki giriş sayısı : 248

Bir dosyadaki özellik sayısı : 248

Maks. Dosya boyutu olarak bayt : 264

Zpool içindeki cihaz sayısı : 264

Bir sist. Maks. Zpool sayısı : 264

Zpool içindeki maks. Dosya sistemi sayısı : 264

Bazı zpool maks. Boyutu olarak bayt : 278

**İşletim Sistemi Uyumluluğu :** Solaris işletim sistemine dahil edilmiştir.Fakat sistemin koduna serbestçe ulaşılabildiği için FreeBSD gibi platformlarda da görülebilir.

**Dosya Yapısı :** Zpools adı verilen sanal depolama havuzlarının üzerine oluşturulur.

-Otomatik olarak büyür veya küçülür.

-Havuzdaki tüm depolama alanları paylaşılır.

**Diğer Dosya Sistemleri İle Arasındaki Farklar:**

-Tek bir fiziksel cihazdan yapılandırılma yapılmıştır.

-Havuz Konsepti sayesinde belirli sayıda ki cihazlarla sınırlı değildir.

-Veri bozulmasını algılar ve veri bütünlüğünü korur.

-128 Bit olması depolama alanının çok daha fazla olmasını sağlar.

-CoW sistemi sayesinde işaretçiyi kolaylaştıran birden çok işleme aynı kaynağa erişim sağlar.