45

Veri Bütünlüğü ve Koruma

Şimdiye kadar incelediğimiz dosya sistemlerinde bulunan temel ilerlemelerin ötesinde, bir dizi özellik üzerinde çalışmaya değer. Bu bölümde, bir kez daha güvenilirliğe odaklanıyoruz (daha önce RAID bölümünde depolama sistemi güvenilirliğini incelemiştik). Özellikle, modern depolama cihazlarının güvenilmez doğası göz önüne alındığında, bir dosya sistemi veya depolama sistemi verilerin güvenli olmasını nasıl sağlamalıdır?

Bu genel alan olarak adlandırılır**veri bütünlüğü**veya**veri koruması**. Bu nedenle, şimdi depolama sisteminize koyduğunuz verilerin, depolama sistemi size geri döndüğünde aynı olmasını sağlamak için kullanılan teknikleri inceleyeceğiz.

CRUX: HÇOKTÖEEMİNİYETDATABENDÜRÜSTLÜK

Sistemler, depolamaya yazılan verilerin pro-
dokundu mu? Hangi teknikler gereklidir? Bu tür teknikler, hem düşük alan hem de zaman giderleriyle nasıl verimli hale getirilebilir?

45.1 Disk Hatası Modları

RAID ile ilgili bölümde öğrendiğiniz gibi, diskler mükemmel değildir ve (bazen) arızalanabilir. İlk RAID sistemlerinde, arıza modeli oldukça basitti: ya tüm disk çalışıyor ya da tamamen arızalanıyor ve böyle bir arızanın tespiti çok kolay. Bu**başarısız durdurma**disk arızası modeli, RAID oluşturmayı nispeten basit hale getirir [S90].

Öğrenmediğiniz şey, modern disklerin sergilediği diğer tüm arıza modları hakkındadır. Spesifik olarak, Bairavasundaram ve ark. ayrıntılı olarak incelendiğinde [B+07, B+08], modern diskler bazen çoğunlukla çalışıyor gibi görünecek, ancak bir veya daha fazla bloğa başarıyla erişmede sorun yaşayacak. Spesifik olarak, iki tür tek blok hatası yaygındır ve dikkate alınmaya değerdir:**gizli sektör hataları**(**LSE'ler**) ve**yolsuzluğu engelle**. Şimdi her birini daha ayrıntılı olarak tartışacağız.

	Ucuz	pahalı
LSE'ler	%9.40	%1.40
Yolsuzluk	%0,50	%0,05

Şekil 45.1:**LSE'lerin Sıklığı ve Blok Yolsuzluğu**

LSE'ler, bir disk sektörü (veya sektör grubu) bir şekilde hasar gördüğünde ortaya çıkar. Örneğin, disk kafası herhangi bir nedenle yüzeye değerse (bir **kafa çarpması**, normal çalışma sırasında olmaması gereken bir şey), yüzeye zarar vererek bitleri okunamaz hale getirebilir. Kozmik ışınlar da yanlış içeriklere yol açan bitleri çevirebilir. Neyse ki, disk için**hata düzeltme kodları**(**ECC**) sürücü tarafından bir bloktaki disk üzerindeki bitlerin iyi olup olmadığını belirlemek ve bazı durumlarda bunları düzeltmek için kullanılır; iyi değillerse ve sürücüde hatayı düzeltmek için yeterli bilgi yoksa, bunları okumak için bir istek gönderildiğinde disk bir hata döndürür.

Bir disk bloğunun dönüştüğü durumlar da vardır.**yozlaşmış**diskin kendisi tarafından algılanamayacak bir şekilde. Örneğin, buggy disk üretici yazılımı bir bloğu yanlış konuma yazabilir; böyle bir durumda, disk ECC'si blok içeriğinin iyi olduğunu gösterir, ancak müşterinin bakış açısına göre, daha sonra erişildiğinde yanlış blok döndürülür. Benzer şekilde, bir blok hatalı bir veri yolu üzerinden ana bilgisayardan diske aktarıldığında bozulabilir; ortaya çıkan bozuk veriler diskte saklanır, ancak müşterinin istediği bu değildir. Bu tür hatalar özellikle sinsidir çünkü **sessiz hatalar**; disk, hatalı verileri döndürürken sorunla ilgili herhangi bir belirti vermez.

Prabhakaran ve ark. disk arızasının bu daha modern görüşünü şu şekilde tanımlar:**başarısız-kısmi**disk arızası modeli [P+05]. Bu görüşe göre, diskler yine de tamamen başarısız olabilir (geleneksel arıza durdurma modelinde olduğu gibi); ancak, diskler görünüşte çalışıyor olabilir ve bir veya daha fazla bloğa erişilemez hale gelebilir (örn. LSE'ler) veya yanlış içerikleri tutabilir (örn. bozulma). Bu nedenle, görünüşte çalışan bir diske erişirken, arada bir belirli bir bloğu okumaya veya yazmaya çalışırken bir hata (sessiz olmayan bir kısmi hata) döndürebilir ve arada bir sadece yanlış verileri döndürebilir. (sessiz bir kısmi hata).

Bu tür hataların her ikisi de biraz nadirdir, ancak ne kadar nadirdir? Şekil 45.1, iki Bairavasundaram çalışmasından [B+07,B+08] bazı bulguları özetlemektedir.

Şekil, çalışma boyunca (yaklaşık 3 yıl, 1,5 milyondan fazla disk sürücüsü) en az bir LSE sergileyen veya blok bozulması sergileyen sürücülerin yüzdesini gösterir. Şekil, sonuçları "ucuz" sürücüler (genellikle SATA sürücüler) ve "pahalı" sürücüler (genellikle SCSI veya Fiber Kanal) olarak alt gruplara ayırır. Gördüğünüz gibi, daha iyi diskler satın almak her iki sorun türünün sıklığını azaltırken (yaklaşık bir büyüklük sırasına göre), yine de depolama sisteminizde bunları nasıl ele alacağınızı dikkatlice düşünmeniz gerekecek kadar sık meydana geliyor.

LSE'ler hakkında bazı ek bulgular şunlardır:

- Birden fazla LSE'ye sahip maliyetli disklerin, daha ucuz diskler kadar ek hatalar geliştirmesi olasıdır.
- Çoğu sürücü için yıllık hata oranı ikinci yılda artar
- LSE'lerin sayısı disk boyutuyla birlikte artar
- LSE içeren çoğu diskte 50'den az
- LSE içeren disklerin ek LSE geliştirme olasılığı daha yüksektir
- Önemli miktarda uzamsal ve zamansal yerellik mevcuttur.
- Disk temizleme yararlıdır (çoğu LSE bu şekilde bulunur)

Yolsuzlukla ilgili bazı bulgular:

- Bozulma olasılığı, aynı sürücü sınıfındaki farklı sürücü modellerinde büyük farklılıklar gösterir.
- Yaş etkileri modeller arasında farklıdır
- İş yükü ve disk boyutunun yolsuzluk üzerinde çok az etkisi vardır
- Bozuk disklerin çoğunda yalnızca birkaç bozulma vardır
- Bozulma, bir disk içinde veya RAID'deki diskler arasında bağımsız değildir
- Mekansal yerellik ve bazı zamansal yerellik vardır.
- LSE'ler ile zayıf bir korelasyon vardır.

Bu başarısızlıklar hakkında daha fazla bilgi edinmek için orijinal belgeleri [B+07,B+08] okumalısınız. Ancak umarım ana nokta açık olmalıdır: gerçekten güvenilir bir depolama sistemi oluşturmak istiyorsanız, hem LSE'leri tespit edip bunlardan kurtarma hem de bozulmayı engelleme makinelerini dahil etmelisiniz.

45.2 Gizli Sektör Hatalarını Ele Alma

Bu iki yeni kısmi disk arızası modu göz önüne alındığında, şimdi onlar hakkında ne yapabileceğimizi görmeye çalışmalıyız. İlk olarak, ikisinden daha kolay olanı, yani gizli sektör hatalarını ele alalım.

CRUX: HÇOKTÖHANDLELBİR ÇADIRSEKTOREÇÖZÜMLER
Bir depolama sistemi gizli sektör hatalarını nasıl ele almalıdır? Bu tür bir kısmi arızanın üstesinden gelmek için ne kadar fazladan makineye ihtiyaç vardır?

Gizli sektör hatalarının (tanım gereği) kolayca algılanabildikleri için ele alınması oldukça basittir. Bir depolama sistemi bir bloğa erişmeye çalıştığında ve disk bir hata döndürdüğünde, depolama sistemi doğru verileri döndürmek için sahip olduğu artıklık mekanizmasını kullanmalıdır. Örneğin, yansıtılmış bir RAID'de, sistem alternatif kopyaya erişmelidir; pariteye dayalı bir RAID-4 veya RAID-5 sisteminde, sistem, parite grubundaki diğer bloklardan bloğu yeniden oluşturmalıdır. Böylece, LSE'ler gibi kolayca tespit edilen problemler, standart fazlalık mekanizmaları yoluyla kolayca düzeltilir.

LSE'lerin artan yaygınlığı, yıllar içinde RAID tasarımlarını etkilemiştir. RAID-4/5 sistemlerinde özellikle ilginç bir sorun, hem tam disk arızaları hem de LSE'ler art arda meydana geldiğinde ortaya çıkar. Spesifik olarak, tüm disk arızalandığında, RAID**yeniden inşa etmek**eşlik grubundaki diğer tüm diskleri okuyarak ve eksik değerleri yeniden hesaplayarak diski (örneğin, etkin bir yedeğe) Yeniden oluşturma sırasında diğer disklerden herhangi birinde bir LSE ile karşılaşılırsa, bir sorunumuz vardır: yeniden yapılandırma başarıyla tamamlanamaz.

Bu sorunla mücadele etmek için bazı sistemler fazladan fazlalık derecesi ekler. Örneğin, NetApp'ın**RAID-DP**bir [C+04] yerine iki eşlik diskine eşdeğerdir. Yeniden oluşturma sırasında bir LSE keşfedildiğinde, ekstra eşlik, eksik bloğun yeniden yapılandırılmasına yardımcı olur. Her zaman olduğu gibi, her şerit için iki parite bloğunu korumanın daha maliyetli olması nedeniyle bir maliyeti vardır; ancak, NetApp'ın günlük yapılı yapısı**WAFL**fi Dosya sistemi çoğu durumda bu maliyeti azaltır [HLM94]. Kalan maliyet, ikinci eşlik bloğu için fazladan bir disk biçimindeki alandır.

45.3 Yolsuzluğu Tespit Etme: Sağlama Toplamı

Şimdi daha zorlu bir problem olan veri bozulması yoluyla sessiz arızaların üstesinden gelelim. Bozulma ortaya çıktığında ve disklerin kötü veri döndürmesine yol açtığında, kullanıcıların hatalı veri almasını nasıl önleyebiliriz?

CRUX: HÇOKTÖPREZERVDATAbenDÜRÜSTLÜKDESPITECARALIK

Bu tür arızaların sessiz doğası göz önüne alındığında, bir depolama sistemi ne yapabilir? yolsuzluğun ne zaman ortaya çıktığını tespit etmek için? Hangi tekniklere ihtiyaç var? Bunları verimli bir şekilde nasıl uygulayabiliriz?

Gizli sektör hatalarından farklı olarak,tespit etmeyolsuzluk önemli bir sorundur. Bir müşteri bir bloğun kötüye gittiğini nasıl anlayabilir? Belirli bir bloğun kötü olduğu bilindiğinde,kurtarmaöncekiyle aynıdır: bloğun başka bir kopyasına sahip olmanız gerekir (ve umarım bozuk değildir!). Bu nedenle, burada tespit tekniklerine odaklanıyoruz.

Modern depolama sistemleri tarafından veri bütünlüğünü korumak için kullanılan birincil mekanizma,**sağlama toplamı**. Bir sağlama toplamı, girdi olarak bir veri yığınını (diyelim ki 4 KB'lik bir blok) alan ve söz konusu veriler üzerinde bir işlev hesaplayarak veri içeriğinin (örneğin 4 veya 8 bayt) küçük bir özetini üreten bir işlevin sonucudur. Bu özet, sağlama toplamı olarak adlandırılır. Bu tür bir hesaplamanın amacı, sağlama toplamını verilerle birlikte depolayarak ve daha sonra verinin mevcut sağlama toplamının orijinal depolama değeriyle eşleştiğini daha sonraki erişimde onaylayarak bir sistemin verilerin bir şekilde bozulup bozulmadığını veya değiştirilip değiştirilmediğini algılamasını sağlamaktır.

TIP: TBURADA'sNöFREELUNCH

Bedava Öğle Yemeği Diye Bir Şey Yoktur veya kısaca TNSTAAFL, görünüşte bedava bir şey alırken aslında muhtemelen bunun için bir miktar bedel ödediğinizi ima eden eski bir Amerikan deyimidir.

BT. Yemek yiyenlerin, müşterileri çekmek umuduyla onlara bedava öğle yemeği reklamı yaptıkları eski günlerden geliyor; ancak içeri girdiğinizde, "bedava" öğle yemeğini elde etmek için bir veya daha fazla alkollü içecek satın almanız gerektiğini fark ettiniz. Tabii ki, bu aslında bir sorun olmayabilir, özellikle de alkolik olmaya hevesliyseniz (veya tipik bir lisans öğrencisiyseniz).

Genel Sağlama Toplamı İşlevleri

Sağlama toplamlarını hesaplamak için bir dizi farklı işlev kullanılır ve güçleri (yani, veri bütünlüğünü korumada ne kadar iyi oldukları) ve hızları (yani, ne kadar hızlı hesaplanabilecekleri) bakımından farklılık gösterir. Sistemlerde yaygın olan bir değiş tokuş burada ortaya çıkar: genellikle, ne kadar çok koruma alırsanız, o kadar pahalı olur. ücretsiz öğle yemeği diye bir şey yoktur.

Bazı kullanımların özel veya (XOR) tabanlı basit bir sağlama toplamı işlevi. XOR tabanlı sağlama toplamlarında, sağlama toplamı, sağlama toplamı yapılan veri bloğunun her bir parçasının XOR'lanmasıyla hesaplanır, böylece tüm bloğun XOR'unu temsil eden tek bir değer üretilir.

Bunu daha somut hale getirmek için, 16 baytlık bir blok üzerinde 4 baytlık bir sağlama toplamı hesapladığımızı hayal edin (bu blok, gerçekten bir disk sektörü veya bloğu olamayacak kadar küçük, ancak örnek teşkil edecek). Onaltılı 16 veri baytı şöyle görünür:

```
365e c4cd ba14 8a92 ecef 2c3a 40be f666
```

Onları ikili olarak görüntülersek, aşağıdakileri elde ederiz:

```
0011 0110 0101 1110    1100 0100 1100 1101
1011 1010 0001 0100    1000 1010 1001 0010
1110 1100 1110 1111    0010 1100 0011 1010
0100 0000 1011 1110    1111 0110 0110 0110
```

Verileri satır başına 4 baytlık gruplar halinde sıraladığımız için, ortaya çıkan sağlama toplamının ne olacağını görmek kolaydır: son sağlama toplamı değerini elde etmek için her sütun üzerinde bir XOR gerçekleştirin:

```
0010 0000 0001 1011    1001 0100 0000 0011
```

Sonuç, onaltılık olarak 0x201b9403'tür.

XOR makul bir sağlama toplamıdır ancak sınırlamaları vardır. Örneğin, her sağlama toplamı biriminde aynı konumdaki iki bit değişirse, sağlama toplamı bozulmayı algılamaz. Bu nedenle, insanlar diğer sağlama toplamı fonksiyonlarını araştırmışlardır.

Diğer bir temel sağlama toplamı işlevi toplamadır. Bu yaklaşımın hızlı olma avantajı vardır; hesaplamak, taşmayı göz ardı ederek, verilerin her bir parçası üzerinde 2'ye tümleyen toplama işlemi gerçekleştirmeyi gerektirir. Verilerdeki birçok değişikliği algılayabilir, ancak örneğin veriler kaydırıldığında iyi değildir.

Biraz daha karmaşık bir algoritma olarak bilinir**Fletcher sağlama toplamı** , (tahmin edebileceğiniz gibi) mucit John G. Fletcher [F82] için adlandırılmıştır. Hesaplaması oldukça basittir ve iki kontrol baytının hesaplanmasını içerir,s1 ves2.Özellikle, bir blok varsayalımDbaytlardan oluşurd1 ... dn; s1aşağıdaki gibi tanımlanır:s1 = (s1 +d_i)mod255 (her şey üzerinden hesaplanmış d_i);s2 sırayla:s2 = (s2 +s1)mod255 (her şeyden önce tekrard_i) [F04]. Fletcher sağlama toplamı neredeyse CRC kadar güçlüdür (aşağıya bakın), tüm tek bitlik, çift bitlik hataları ve birçok patlama hatasını [F04] tespit eder.

Yaygın olarak kullanılan son bir sağlama toplamı,**döngüsel artıklık kontrolü**(**CRC**). Bir veri bloğu üzerinden sağlama toplamını hesaplamak istediğinizi varsayalım.D.Tek yaptığın tedavi etmekDsanki büyük bir ikili sayıymış gibi (sonuçta sadece bir bit dizisidir) ve üzerinde anlaşmaya varılan bir değere bölün (k).Bu bölümün geri kalanı, CRC'nin değeridir. Görünüşe göre, bu ikili modulo işlemi oldukça verimli bir şekilde uygulanabilir ve bu nedenle CRC'nin ağ oluşturmadaki popülaritesi de artar. Daha fazla ayrıntı için başka bir yere bakın [M13].

Kullanılan yöntem ne olursa olsun, mükemmel bir sağlama toplamının olmadığı açık olmalıdır: aynı içeriklere sahip iki veri bloğunun aynı sağlama toplamlarına sahip olması mümkündür;**çarpışma**. Bu gerçek sezgisel olmalıdır: Sonuçta, bir sağlama toplamı hesaplamak, büyük bir şeyi (örneğin, 4 KB) almak ve çok daha küçük (örneğin, 4 veya 8 bayt) bir özet üretmektir. İyi bir sağlama toplamı işlevi seçerken, hesaplaması kolay kalırken çarpışma olasılığını en aza indiren bir işlev bulmaya çalışıyoruz.

Sağlama Düzeni

Artık bir sağlama toplamını nasıl hesaplayacağınızı biraz anladığınıza göre, şimdi bir depolama sisteminde sağlama toplamlarının nasıl kullanılacağını analiz edelim. Ele almamız gereken ilk soru, sağlama toplamının düzenidir, yani sağlama toplamları diskte nasıl saklanmalıdır?

En temel yaklaşım, her bir disk sektörü (veya bloğu) ile bir sağlama toplamı depolar. Verilen bir veri bloğuD,bu veri üzerinden sağlama toplamını çağıralım CD).Böylece, sağlama toplamları olmadan disk düzeni şöyle görünür:

D0	D1	D2	D3	D4	D5	D6

Sağlama toplamları ile düzen, her blok için tek bir sağlama toplamı ekler:

Sağlama toplamları genellikle küçük olduğundan (örneğin, 8 bayt) ve diskler yalnızca sektör boyutunda parçalar (512 bayt) veya bunların katları halinde yazabildiğinden, ortaya çıkan sorunlardan biri yukarıdaki mizanpaja nasıl ulaşılacağıdır. Sürücü üreticileri tarafından kullanılan bir çözüm, sürücüyü 520 baytlık sektörlerle biçimlendirmektir; sağlama toplamını depolamak için sektör başına fazladan 8 bayt kullanılabilir.

Böyle bir işlevselliğe sahip olmayan disklerde, dosya sistemi sağlama toplamlarını 512 baytlık bloklar halinde depolamanın bir yolunu bulmalıdır. Böyle bir olasılık aşağıdaki gibidir:

Bu şemada,nsağlama toplamları bir sektörde birlikte depolanır ve ardındann veri blokları, ardından bir sonraki için başka bir sağlama toplamı sektörünbloklar vb. Bu yaklaşım, tüm disklerde çalışma avantajına sahiptir, ancak daha az verimli olabilir; örneğin dosya sistemi bloğun üzerine yazmak istiyorsaD1,içeren sağlama toplamı sektöründe okumak zorundadır. CD1),GüncellemeCD1)içine girin ve ardından sağlama toplamı sektörünü ve yeni veri bloğunu yazınD1 (böylece bir okuma ve iki yazma). Daha önceki yaklaşım (sektör başına bir sağlama toplamı) yalnızca tek bir yazma gerçekleştirir.

45.4 Sağlama toplamlarını kullanma

Bir sağlama toplamı düzenine karar verildiğinde, artık nasıl yapılacağını gerçekten anlamaya devam edebiliriz.kullanmaksağlama toplamları. Blok okurken D,istemci (ör. dosya sistemi veya depolama denetleyicisi) sağlama toplamını da diskten okur$C_s(D)$,dediğimiz**saklanan sağlama toplamı**(dolayısıyla alt simgeC_s). müşteri o zamanhesaplaralınan blok üzerindeki sağlama toplamıD,dediğimiz **hesaplanan sağlama toplamı**$C_c(D)$.Bu noktada müşteri, saklanan ve hesaplanan sağlama toplamlarını karşılaştırır; eğer eşitlerse (yani,$C_s(D) == C_c(D)$,veriler muhtemelen bozulmamıştır ve bu nedenle kullanıcıya güvenli bir şekilde iade edilebilir. Yaparlarsaolumsuzlukmaç (yani,$C_s(D) != C_c(D)$),bu, verilerin depolandığı andan itibaren değiştiği anlamına gelir (çünkü depolanan sağlama toplamı, verilerin o andaki değerini yansıtır). Bu durumda, sağlama toplamımızın tespit etmemize yardımcı olduğu bir yolsuzluğumuz var.

Bir yolsuzluk göz önüne alındığında, doğal soru, bu konuda ne yapmalıyız? Depolama sisteminde fazladan bir kopya varsa, cevap kolaydır: onun yerine onu kullanmayı deneyin. Depolama sisteminde böyle bir kopya yoksa, olası yanıt şudur:

bir hata döndürmek için. Her iki durumda da, yolsuzluk tespitinin sihirli bir değnek olmadığının farkına varın; bozulmamış verileri almanın başka bir yolu yoksa, şansınız kalmaz.

45.5 Yeni Bir Sorun: Yanlış Yönlendirilmiş Yazmalar

Yukarıda açıklanan temel şema, bozuk blokların genel durumunda iyi çalışır. Bununla birlikte, modern disklerde, farklı çözümler gerektiren birkaç sıra dışı arıza modu bulunur.

İlgilenilen ilk başarısızlık modu,**yanlış yönlendirilmiş yazma**. Bu, verileri diske doğru şekilde yazan disk ve RAID denetleyicilerinde ortaya çıkar.yanlışyer. Tek diskli bir sistemde bu, diskin blok yazdığı anlamına gelir.$D_x$hitap etmemekx (istendiği gibi) değil, ele almak yerine sen (böylece "bozucu"D_y); ek olarak, bir çoklu disk sisteminde, denetleyici ayrıca yazabilir$D_{ben,\ x}$hitap etmemekxdiskinibunun yerine başka bir diske j.Böylece sorumuz:

CRUX: HÇOKTÖHANDLEMYÖNLENDİRİLMİŞWAYİNLER

Bir depolama sistemi veya disk denetleyicisi yanlış yönlendirilmiş yazmaları nasıl algılamalıdır? Sağlama toplamından hangi ek özellikler gerekir?

Cevap, şaşırtıcı olmayan bir şekilde basit: her bir sağlama toplamına biraz daha fazla bilgi ekleyin. Bu durumda, bir ekleme**fiziksel tanımlayıcı**(**fiziksel kimlik**) oldukça faydalıdır. Örneğin, saklanan bilgiler artık sağlama toplamını içeriyorsaCD)ve bloğun hem disk hem de sektör numaraları, istemcinin belirli bir yerel ayarda doğru bilginin bulunup bulunmadığını belirlemesini kolaylaştırır. Spesifik olarak, müşteri disk 10'daki blok 4'ü okuyorsa ($D_{10,4}$), saklanan bilgiler, aşağıda gösterildiği gibi bu disk numarasını ve sektör ofsetini içermelidir. Bilgiler eşleşmezse, yanlış yönlendirilmiş bir yazma gerçekleşmiş ve artık bir bozulma algılanmıştır. İşte bu eklenen bilgilerin iki diskli bir sistemde nasıl görüneceğine dair bir örnek. Sağlama toplamları genellikle küçük (örn. 8 bayt) ve bloklar çok daha büyük (örn. 4 KB veya daha büyük) olduğundan, bu rakamın, kendinden önceki diğerleri gibi ölçekli olmadığına dikkat edin:

Disk 1	C[D0]	disk=1	blok=0	D0	C[D1]	disk=1	blok=1	D1	C[D2]	disk=1	blok=2	D2
Disk 0	C[D0]	disk=0	blok=0	D0	C[D1]	disk=0	blok=1	D1	C[D2]	disk=0	blok=2	D2

Disk üstü biçiminden, diskte artık makul miktarda fazlalık olduğunu görebilirsiniz: her blok için, disk numarası her blokta tekrarlanır ve söz konusu bloğun ofseti de bloğun yanında tutulur. . Gereksiz bilgilerin varlığı yine de şaşırtıcı olmamalıdır; fazlalık, hata saptamanın (bu durumda) ve düzeltmenin (diğerlerinde) anahtarıdır. Kusursuz disklerde kesinlikle gerekli olmasa da, biraz ekstra bilgi, ortaya çıkmaları durumunda sorunlu durumların tespit edilmesine yardımcı olmak için uzun bir yol kat edebilir.

45.6 Son Bir Sorun: Kayıp Yazmalar

Ne yazık ki, yanlış yönlendirilmiş yazmalar ele alacağımız son sorun değil. Spesifik olarak, bazı modern depolama cihazlarının da şu bilinen bir sorunu vardır:**kayıp yazma**, cihaz üst katmana bir yazmanın tamamlandığını bildirdiğinde meydana gelir, ancak aslında hiçbir zaman devam etmez; bu nedenle geriye kalan, güncellenen yeni içerik yerine bloğun eski içeriğidir.

Buradaki bariz soru şudur: Yukarıdan sağlama toplama stratejilerimizden herhangi biri (örneğin, temel sağlama toplamları veya fiziksel kimlik) kayıp yazma işlemlerini tespit etmeye yardımcı olur mu? Ne yazık ki cevap hayır: eski bloğun muhtemelen eşleşen bir sağlama toplamı vardır ve yukarıda kullanılan fiziksel kimlik (disk numarası ve blok ofseti) de doğru olacaktır. Böylece son sorunumuz:

CRUX: HÇOKTÖHANDLELOSTWAYİNLER
Bir depolama sistemi veya disk denetleyicisi kayıp yazmaları nasıl algılamalıdır? Sağlama toplamından hangi ek özellikler gerekir?

[K+08]'e yardımcı olabilecek bir dizi olası çözüm var. Klasik bir yaklaşım [BS04],**yaz doğrulama**veya**yazdıktan sonra oku**; Bir sistem, bir yazma işleminden sonra verileri hemen geri okuyarak, verilerin gerçekten disk yüzeyine ulaşmasını sağlayabilir. Ancak bu yaklaşım oldukça yavaştır ve bir yazmayı tamamlamak için gereken G/Ç sayısını iki katına çıkarır.

Bazı sistemler, kayıp yazmaları algılamak için sistemin başka bir yerine bir sağlama toplamı ekler. Örneğin, Güneş'in**Zettabyte Dosya Sistemi**(**ZFS**) her dosya sistemi inode'unda bir sağlama toplamı ve bir dosyaya dahil edilen her blok için dolaylı blok içerir. Bu nedenle, bir veri bloğuna yazma kaybolsa bile, inode içindeki sağlama toplamı eski verilerle eşleşmeyecektir. Ancak hem inode'a hem de verilere yazma işlemleri aynı anda kaybolursa, böyle bir şema başarısız olur, bu pek olası olmayan (ama ne yazık ki mümkün!) bir durumdur.

45.7 Fırçalama

Tüm bu tartışma göz önüne alındığında, merak ediyor olabilirsiniz: Bu sağlama toplamları gerçekte ne zaman kontrol ediliyor? Tabii ki, bir miktar kontrol

Verilere uygulamalar tarafından erişildiğinde oluşur, ancak çoğu veriye nadiren erişilir ve bu nedenle denetlenmeden kalır. Bit çürümesi sonunda belirli bir veri parçasının tüm kopyalarını etkileyebileceğinden, denetlenmeyen veriler güvenilir bir depolama sistemi için sorunludur.

Bu sorunu çözmek için birçok sistem**disk temizleme**çeşitli biçimlerde [K+08]. Periyodik olarak okuyarakhersistemin bloğu ve sağlama toplamlarının hala geçerli olup olmadığını kontrol eden disk sistemi, belirli bir veri öğesinin tüm kopyalarının bozulma olasılığını azaltabilir. Tipik sistemler, taramaları gecelik veya haftalık olarak planlar.

45.8 Sağlama Toplama Genel Giderleri

Bitirmeden önce, veri koruması için sağlama toplamlarını kullanmanın bazı ek yüklerini tartışacağız. Bilgisayar sistemlerinde yaygın olduğu gibi, iki farklı genel gider türü vardır: uzay ve zaman.

Uzay giderleri iki biçimde gelir. İlki, diskin (veya başka bir depolama ortamının) kendisinde; saklanan her sağlama toplamı, diskte artık kullanıcı verileri için kullanılamayan yer kaplar. Tipik bir oran, %0,19'luk bir disk alanı ek yükü için 4 KB veri bloğu başına 8 baytlık bir sağlama toplamı olabilir.

İkinci tip alan yükü, sistemin belleğinde gelir. Verilere erişirken, artık bellekte verilerin kendisi kadar sağlama toplamları için de yer olmalıdır. Ancak, sistem yalnızca sağlama toplamını kontrol eder ve bir kez yaptıktan sonra atarsa, bu ek yük kısa ömürlüdür ve çok da endişe verici değildir. Yalnızca sağlama toplamları bellekte tutulursa (bellek bozulmasına karşı ek bir koruma düzeyi için [Z+13]), bu küçük ek yük gözlemlenebilir olacaktır.

Alan genel giderleri küçük olsa da, sağlama toplamının neden olduğu zaman ek yükleri oldukça belirgin olabilir. En azından CPU, hem veri depolandığında (depolanan sağlama toplamının değerini belirlemek için) hem de veriye erişildiğinde (sağlama toplamını yeniden hesaplamak ve depolanan sağlama toplamıyla karşılaştırmak için) her blok üzerinden sağlama toplamını hesaplamalıdır. Sağlama toplamları (ağ yığınları dahil) kullanan birçok sistem tarafından kullanılan CPU ek yüklerini azaltmaya yönelik bir yaklaşım, veri kopyalama ve sağlama toplamını tek bir kolaylaştırılmış etkinlikte birleştirmektir; kopya her halükarda gerekli olduğu için (örneğin, verileri çekirdek sayfası önbelleğinden bir kullanıcı arabelleğine kopyalamak için), birleştirilmiş kopyalama/sağlama toplamı oldukça etkili olabilir.

CPU ek yüklerinin ötesinde, bazı sağlama toplamı şemaları, özellikle sağlama toplamları verilerden farklı bir şekilde depolandığında (dolayısıyla bunlara erişmek için fazladan G/Ç'ler gerektiğinde) ve arka plan temizleme için gereken herhangi bir ekstra G/Ç için fazladan G/Ç ek yüklerine neden olabilir. İlki tasarım gereği azaltılabilir; ikincisi ayarlanabilir ve dolayısıyla etkisi, belki de bu tür temizleme faaliyetinin ne zaman gerçekleştiği kontrol edilerek sınırlandırılabilir. Çoğu (tümü değil!) üretken işçinin yattığı gecenin ortası, bu tür temizleme faaliyetini gerçekleştirmek ve depolama sisteminin sağlamlığını artırmak için iyi bir zaman olabilir.

45.9 Özet

Sağlama toplamı uygulamasına ve kullanımına odaklanarak modern depolama sistemlerinde veri korumayı tartıştık. Farklı sağlama toplamları, farklı hata türlerine karşı koruma sağlar; depolama cihazları geliştikçe, şüphesiz yeni arıza modları ortaya çıkacaktır. Belki de böyle bir değişiklik, araştırma topluluğunu ve endüstriyi bu temel yaklaşımlardan bazılarını yeniden gözden geçirmeye veya tamamen yeni yaklaşımlar icat etmeye zorlayacaktır. Zaman gösterecek. Ya da olmaz. Zaman bu şekilde komik.

Referanslar

[B+07] "Disk Sürücülerinde Gizli Sektör Hatalarının Analizi", L. Bairavasundaram, G. Goodson, S. Pasupathy, J. Schindler. SIGMETRICS '07, San Diego, CA.Gizli sektör hatalarını ayrıntılı olarak inceleyen ilk makale. Gazete ayrıca, çok erken vefat eden parlak bir araştırmacı ve harika bir adamın adını taşıyan Kenneth C. Sevcik Üstün Öğrenci Makalesi ödülünü de kazandı. OSTEP yazarlarına ABD'den Kanada'ya taşınmanın mümkün olduğunu göstermek için Ken bir keresinde bize Kanada milli marşını söyledi ve bunu yapmak için bir restoranın ortasında ayağa kalktı. ABD'yi seçtik ama bu hafızayı aldık.

[B+08] "An Analysis of Data Corruption in the Storage Stack", yazan Lakshmi N. Bairavasundaram, Garth R. Goodson, Bianca Schroeder, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. FAST '08, San Jose, CA, Şubat 2008.Bu tür bozulmaların 1,5 milyondan fazla sürücü için üç yıl boyunca ne sıklıkta meydana geldiğine odaklanarak, disk bozulmasını gerçekten ayrıntılı olarak inceleyen ilk makale.

[BS04] "Ticari Hata Toleransı: İki Sistemin Hikayesi", Wendy Bartlett, Lisa Spainhower. Güvenilir ve Güvenli Bilgi İşlem Üzerine IEEE İşlemleri, Cilt. 1:1, Ocak 2004.Hataya dayanıklı sistemler oluşturma konusundaki bu klasik, hem IBM'in hem de Tandem'in en son teknolojiye mükemmel bir genel bakışıdır. Alanla ilgilenenlerin okuması gereken bir başka eser daha.

[C+04] "Row-Diagonal Parity for Double Disk Failure Correction", yazan P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, S. Sankar. FAST '04, San Jose, CA, Şubat 2004.Fazladan fazlalığın birleşik tam disk arızası/kısmi disk arızası sorununu çözmeye nasıl yardımcı olduğuna dair erken bir makale. Ayrıca daha fazla teorik çalışmanın pratikle nasıl karıştırılacağına dair güzel bir örnek.

[F04] Peter M. Fenwick tarafından yazılan "Sağlama Toplamları ve Hata Kontrolü". Burada çevrimiçi olarak kopyalanabilir: http://www.ostep.org/Citations/checksums-03.pdf.Muhteşem bir ücretsiz maliyetle kullanımınıza sunulan, sağlama toplamları hakkında harika ve basit bir eğitim.

[F82] John G. Fletcher'ın "Seri İletimler İçin Aritmetik Sağlama Toplamı". İletişimde IEEE İşlemleri, Cilt. 30:1, Ocak 1982.Fletcher'ın kendi adını taşıyan sağlama toplamına ilişkin orijinal çalışması. Buna Fletcher sağlama toplamı demedi, bunun yerine herhangi bir şey demedi; daha sonra diğerleri ona onun adını verdi. O yüzden bu palavra gibi görünen davranış için yaşlı Fletcher'ı suçlama. Bu anekdot size Rubik'i hatırlatabilir; Rubik asla "**Rubik küp**"; bunun yerine, ona sadece "benim küpüm" adını verdi.

[HLM94] Dave Hitz, James Lau, Michael Malcolm tarafından yazılan "NFS Dosya Sunucusu Cihazı için Dosya Sistemi Tasarımı". USENIX İlkbahar '94.NetApp'ın özündeki fikirleri ve ürünü açıklayan öncü makale. Bu sisteme dayalı olarak NetApp, multi-milyar dolarlık bir depolama şirketi haline geldi. NetApp hakkında daha fazla bilgi edinmek için Hitz'in otobiyografisi "How to Castrate a Bull"u okuyun (asıl başlık bu, şaka yok). Ve CS'ye girerek boğa kastrasyonundan kaçınabileceğinizi düşündünüz.

[K+08] "Parity Lost and Parity Regained", yazan Andrew Krioukov, Lakshmi N. Bairavasundaram, Garth R. Goodson, Kiran Srinivasan, Randy Thelen, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. FAST '08, San Jose, CA, Şubat 2008.Bu çalışma, verileri korumada farklı sağlama toplamı şemalarının nasıl çalıştığını (veya çalışmadığını) araştırıyor. Mevcut koruma stratejilerindeki bir dizi ilginç kusuru ortaya koyuyoruz.

[M13] "Döngüsel Artıklık Kontrolleri" bilinmiyor. Mevcut:http://www.mathpages.com/home/kmath458.htm.CRC'lerin süper net ve özlü bir açıklaması. Görünüşe göre internet bilgi dolu.

[P+05] "IRON File Systems", V. Prabhakaran, L. Bairavasundaram, N. Agrawal, H. Gunawi, A. Arpaci-Dusseau, R. Arpaci-Dusseau. SOSP '05, Brighton, İngiltere.Disklerin nasıl kısmi arıza modlarına sahip olduğuna dair makalemiz ve modern dosya sistemlerinin bu tür arızalara nasıl tepki verdiğine dair ayrıntılı bir çalışma. Görünüşe göre, oldukça zayıf! Bu çalışmada çok sayıda hata, tasarım kusuru ve diğer tuhaflıklar bulduk. Bunların bir kısmı Linux topluluğuna geri dönerek dosya sistemi güvenilirliğini artırdı. Rica ederim!

[RO91] Mendel Rosenblum ve John Ousterhout tarafından "Log Yapılı Dosya Sisteminin Tasarımı ve Uygulanması". SOSP '91, Pacific Grove, CA, Ekim 1991.O kadar güzel ki tekrar alıntı yapıyoruz.

[S90] "Durum Makinesi Yaklaşımını Kullanarak Hataya Dayanıklı Hizmetleri Uygulama: Bir Öğretici", Fred B. Schneider. ACM Anketleri, Cilt. 22, No. 4, Aralık 1990.Hataya dayanıklı hizmetler nasıl oluşturulur? Dağıtılmış sistemler inşa edenler için okunması gereken bir kitap.

[Z+13] "Esnek Uçtan Uca Veri Bütünlüğü ile Zettabyte Güvenilirliği", Y. Zhang, D. Myers, A. Arpaci-Dusseau, R. Arpaci-Dusseau. MSST '13, Long Beach, Kaliforniya, Mayıs 2013.Bir sistemin sayfa önbelleğine veri koruması nasıl eklenir. Yerimiz kalmadı, yoksa bir şeyler yazardık...