Fazladan Bağımsız Disk Dizisi (RAIDs)

Bir diski kullandığımızda, bazen daha hızlı olmasını isteriz; Giriş/Çıkış işlemleri yavaştır ve tüm sistemin darboğaz olmasına sebep olur. Bir diski kullandığımız zaman, daha fazla yere sahip olmasını yani daha büyük olmasını isteriz; zamanla daha fazla veri kullanıyor oluyoruz ve böylece disklerimiz doldukça doluyor. Bir diski kullandığımız zaman daha güvenilir olmasını isteriz; ne zaman bir disk hata verse ve verimizi kurtaramadığımız zaman, tüm değerli verimiz gitmiş olur.

SORUNUN MERKEZİ: NASIL DAHA BÜYÜK, HIZLI VE GÜVENİLİR BİR DİSK YAPARIZ

Nasıl daha büyük, hızlı ve güvenilir bir depolama sistemi yapabiliriz? Anahtar teknikler nelerdir? Farklı yaklaşımlar arasındaki değiş tokuşlar nelerdir?

Bu bölümde, size fazladan bağımsız disk dizisi "RAID [P+88]" adıyla daha iyi bilinen bir taktikle tanıştıracağım, bu teknik birden fazla diskin anlaşarak daha hızlı, büyük ve güvenilir bir disk sistemi olmasını anlatacağım. Bu terim, 1980'lerin sonlarına doğru U.C Berkeley'deki bir grup araştırmacı (Profesörler David Patterson ve Randy Katz ve Garth Gibson adındaki bir öğrenci liderliğinde) tarafından tanıtıldı; bu süre zarfında, birçok farklı araştırmacı aynı anda daha iyi bir depolama sistemini oluşturmak içi birden fazla disk kullanma fikrine ulaştı [BG88, K86, K88, PB86, SG86].

Dışarıdan, RAID bir diske benziyordu: birinin okuyabileceği veya yazabileceği bir grup blok. İçten, RAID birkaç diskten oluşan, hafıza (hem geçici hem kalıcı), ve bir veya daha fazla işlemcinin sistemi yönettiği kompleks bir canavar. Bir RAID bir grup diski yönetmek için geliştirilmiş, bir bilgisayar sistemine benzer.

RAID'ler bize tek bir diskten sayılarca fazla avantaj sağlar. Bunlardan biri ise **performanstır**. Birden fazla diski paralel olarak kullanmak Giriş/Çıkış işlemlerinde hız kazandırır. Diğer bir iyi yanı ise **kapasite**. Büyük veri setleri büyük disklerde yer kaplıyor. Son olarak, RAID'ler güvenilirliği arttırabilirler; veriyi birden fazla diske dağıtmak (RAID teknikleri olmadan) verileri tek bir diskin kaybına savunmasız hale getirir; bir çeşit fazlalık ile, RAID'ler bir diskin kaybını tolere eder ve bir hata yokmuş gibi devam eder.

İPUCU: ŞEFFAFLIK DAĞITIMI SAĞLAR

Bir sisteme nasıl yeni bir işlevsellik ekleneceği düşünülürken, her zaman böyle bir işlevselliğin **şeffaf** bir şekilde eklenip eklenemeyeceği düşünülmelidir, sistemin geri kalanında bir değişiklik gerektirmeyecek bir şekilde. Mevcut yazılımın tamamen yeniden yazılmasını (veya radikal donanımı değişiklikleri) gerektirmek, bir fikrin etki şansını azaltır. RAID mükemmel bir örnek ve kesinlikle şeffaflığı başarısında katkıda bulundu. Yöneticiler SCSI diski yerine SCSI-tabanlı RAID depolama dizisini kurmalı ve sistemin kalanı (ana bilgisayar, işletim sistem, vb.) kullanmaya başlamak için bir bit bile değiştirmek zorunda değildi. Bu dağıtım problemi çözerek, RAID ilk gününden daha başarılıydı.

İnanılmaz, RAID'ler bu avantajları **şeffaf** bir şekilde sistemlerde kullanır, yani RAID ana sistem için büyük bir disk gibi görünür. Şeffaflığın güzelliği ,tabi ki, birinin bir diski RAID ile değiştirmesine ve tek bir yazılım satırını değiştirmemesine olanak sağlamasıdır; işletim sistemi ve kullanıcı uygulamaları modifikasyon gerektirmeden işlemlerine devam eder. Böylelikle, şeffaflık RAID'in **konuşlandırılabilirliğini** büyük ölçüde arttırır, kullanıcıların ve yöneticilerin yazılım uyumluluğu endişeleri duymadan kullanmalarına olanak tanır.

Şimdi RAID'lerin bazı önemli yönlerini tartışalım. Arayüz, hata modeli ile başlıyoruz ve ardından bir RAID tasarımın üç önemli eksen boyunca nasıl değerlendirileceğini tartışacağız: kapasite, güvenilirlik ve performans. Daha sonra, RAID tasarımı ve uygulanması için önemli olan bir dizi başka konuyu ele alacağız.

Arayüz ve RAID Dahili Parçaları

Yukarıdaki bir dosya sistemine, bir RAID büyük gibi görünüyor, (umarım ki) hızlı ve (umarım ki) güvenilirdir. Tek bir diskmiş gibi, kendisini, her biri dosya sistemi (veya başka bir istemci) tarafından okunabilen veya yazılabilen doğrusal bir blok dizisi olarak sunar.

Ne zaman bir dosya sistemi RAID'e mantıksal bir G/Ç isteği gönderdiğinde, RAID, talebi tamamlamak için hangi diske (veya disklere) erişebileceğini dahili olarak hesaplamalı ve ardından bunu yapmak için bir veya daha fazla fiziksel G/Ç vermelidir. Bu fiziksel G/Ç'lerin tam doğası, RAID düzeyine bağlıdır, aşağıda ayrıntılı olarak tartışacağımız gibi. Ancak, basit bir örnek olarak, her bloğun iki kopyasını tutan bir RAID düşünün (her biri ayrı bir disk olacak şekilde); Böyle bir ikizlenmiş RAID sistemine yazarken, RAID'in verdiği her bir mantıksal G/Ç için iki fiziksel G/Ç gerçekleştirmesi gerekecektir. RAID sistemi genellikle bir ana

bilgisayara standart bir bağlantıyla (örn. SCSI veya SATA) ayrı bir donanım kutusu olarak oluşturulur. Bununla birlikte, dahili olarak, RAID'ler, RAID'in çalışmasını yönlendirmek için bellenimi çalıştıran bir mikrodenetleyiciden oluşan oldukça karmaşıktır, veri bloklarını okunurken ve yazılırken arabelleğe almak için DRAM gibi geçici bellek ve bazı durumlarda, güvenli bir şekilde ara belleğe yazmak için geçici olmayan bellek ve hatta eşlik hesaplamaları yapmak için özel mantık (bazı RAID seviyelerinde yararlıdır, örneğin: aşağıda da göreceğiz). Yüksek düzeyde, RAID daha çok uzmanlaşmış bir bilgisayar sistemidir: bir işlemcisi, belleği ve diskleri vardır; ancak uygulamaları çalıştırmak yerine RAID'i çalıştırmak için tasarlanmış özel yazılımları çalıştırır.

Hata Modeli

RAID'i anlamak ve farklı yaklaşımları karşılaştırmak, aklımızda bir hata modeli olmalı. RAID'ler, belirli türdeki disk hatalarını algılamak ve bunlardan kurtulmak için tasarlanmıştır; bu nedenle, tam olarak hangi hataların bekleneceğini bilmek, çalışan bir tasarıma ulaşmada kritik öneme sahiptir. Varsayacağımız ilk hata modeli oldukça basit ve **başarısız-durdurma** arıza modeli [S84] olarak adlandırıldı. Bu modelde, bir disk tam olarak iki durumdan birinde olabilir: çalışır ya da başarısız. Çalışan bir disk ile tüm bloklar okunabilir veya yazılabilir. Buna karşılık, bir disk arızalandığında kalıcı olarak kaybolduğunu varsayarız. Başarısız-durdurma modelinin kritik bir yönü, arıza tespiti hakkında ne varsaydığıdır. Spesifik olarak, bir disk arızalandığında bunun kolayca tespit edildiğini varsayarız. Örneğin, bir RAID dizisinde, RAID denetleyici donanımının (veya yazılımının) bir disk arızalandığında hemen gözlemleyebileceğini varsayarız. Bu nedenle, şimdilik disk bozulması gibi daha karmaşık "sessiz" arızalar hakkında endişelenmemize gerek yok. Ayrıca, başka türlü çalışan bir diskte (bazen gizli sektör hatası olarak adlandırılır) tek bir bloğun erişilemez hale gelmesi konusunda endişelenmemize gerek yoktur. Bu daha karmaşık (ve maalesef daha gerçekçi) disk hatalarını daha sonra ele alacağız.

RAID Nasıl Değerlendirilir?

Yakında göreceğimiz gibi, RAID oluşturmak için birkaç farklı yaklaşım vardır. Bu yaklaşımların her biri, güçlü ve zayıf yönlerini anlamak için değerlendirilmeye değer farklı özelliklere sahiptir. Özellikle, her bir RAID tasarımını üç eksende değerlendireceğiz. İlk eksen **kapasitedir**; her biri B bloklu bir dizi N disk verildiğinde, RAID istemcileri için ne kadar yararlı kapasite kullanılabilir? Fazlalık olmadan cevap N ·B'dir; tersine, her bloğun iki kopyasını tutan bir sistemimiz varsa (yansıtma denir), (N · B) /2'lik bir yararlı kapasite elde ederiz. Farklı şemalar

(Örneğin, parite tabanlı olanlar) ikisinin arasına girme eğilimindedir. Değerlendirmenin ikinci ekseni **güvenilirliktir**. Verilen tasarım kaç disk hatasını tolere edebilir? Hata modelimize uygun olarak, yalnızca tüm bir diskin arızalanabileceğini varsayıyoruz; sonraki bölümlerde (yani veri bütünlüğü üzerine), daha karmaşık hata modlarının nasıl ele alınacağını düşüneceğiz. Son olarak, üçüncü eksen performanstır. Performans, büyük ölçüde disk dizisine sunulan iş yüküne bağlı olduğundan, değerlendirilmesi biraz zordur. Bu nedenle, performansı değerlendirmeden önce, dikkate alınması gereken bir dizi tipik iş yükü sunacağız. Şimdi üç önemli RAID tasarımını ele alıyoruz: RAID Düzey 0 (şeritleme), RAID Düzey 1 (yansıtma) ve RAID Düzeyleri 4/5 (eşlik tabanlı artıklık). Bu tasarımların her birinin bir "seviye" olarak adlandırılması, Berkeley'de Patterson, Gibson ve Katz'ın öncü çalışmalarından kaynaklanmaktadır [P+88].

RAID Seviye 0: Şeritleme

İlk RAID seviyemiz aslında tam olarak bir RAID seviyesi değil çünkü fazlalık değildir. Ancak, RAID seviye 0 ya da daha iyi bilinen adıyla şeritleme, perforamns ve kapasite üzerinde mükemmel bir üst sınır görevi görür ve bu nedenle anlaşılmaya değerdir.

En basit formuyla şeritleme, blokları sistemin diskleri boyunca aşağıdaki gibi şeritler. (Burada 4-disk dizisi olduğunu varsayalım):

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Yukarıdaki diziden, basit fikri anlayabilirsiniz: dizinin bloklarını disklere dairesel bir şekilde yayar. Bu yaklaşım, dizinin bitişik parçaları için istek yapıldığında diziden en fazla paralelliği çıkarmak için tasarlanmıştır (büyük, sıralı bir okumada olduğu gibi, örneğin). Aynı sıradaki bloklara **şerit** diyoruz; bu nedenle, 0, 1 ,2 ve 3 blokları yukarıdaki aynı şerittedir.

Bu örnekte, bir sonrakine geçmeden önce her diske yalnızca 1 bloğun (her biri 4KB boyutunda) yerleştirildiği basitleştirici varsayımını yaptık. Ancak, bu düzenlemenin bu şekilde olması gerekmiyor. Örnek vermek gerekirse, blokları

aşağıdaki gibi diskler arasında düzenleyebiliriz:

Disk 0	Disk 1	Disk 2	Disk 3	
0	2	4	6	chunk size:
1	3	5	7	2 blocks
8	10	12	14	
9	11	13	15	

Bu örnekte, bir sonraki diske geçmeden önce her diske iki adet 4KB blok yerleştiriyoruz. Böylece, bu RAID dizisinin öbek boyutu 8 KB'dir ve bu nedenle bir şerit 4 parçadan veya 32 KB veriden oluşur.

Ayrı olarak: RAID Haritalama Sorunu

RAID'in kapasitesini, güvenilirliğini ve performans özelliklerini incelemeden önce, önce haritalama sorunu dediğimiz şeyi bir kenara bırakıyoruz. Bu sorun tüm RAID dizilerinde ortaya çıkar; Basitçe söylemek gerekirse, okuma veya yazma için mantıksal bir blok verildiğinde, RAID tam olarak hangi fiziksel diske ve ofsete erişileceğini nasıl bilir? Bu basit RAID seviyeleri için, mantıksal blokları fiziksel konumlarına doğru bir şekilde eşlemek için fazla karmaşıklığa ihtiyacımız yok. Yukarıdaki ilk şeritleme örneğini alın (yığın boyutu = 1 blok = 4 KB). Bu durumda, mantıksal blok adresi A verildiğinde, RAID istenen diski kolayca hesaplayabilir ve iki basit denklemle dengeleyebilir:

Disk = A % number_of_disks

Offset = A / number_of_disks

Bunların hepsinin tamsayı işlemleri olduğuna dikkat edin (ör. 4 / 3 = 1, 1.33333 değil...). Basit bir örnek için bu denklemlerin nasıl çalıştığını görelim. Yukarıdaki ilk RAID'de 14. blok için bir istek geldiğini hayal edin. 4 disk olduğuna göre, bu demektir ki bizim ilgilendiğimiz disk (14 % 4 = 2): disk 2. Kesin blok şu şekilde hesaplanır (14 / 4 = 3): blok 3. Bu nedenle, blok 14, üçüncü diskin (disk 2, 0'dan başlayarak) dördüncü bloğunda (blok 3, 0'dan başlayarak) bulunmalıdır, ki bu tam olarak bulunduğu yerdir. Farklı yığın boyutlarını desteklemek için bu denklemlerin nasıl değiştirileceğini düşünebilirsiniz. Dene! Çok zor değil.

Yığın Boyutları

Yığın boyutu çoğunlukla dizinin performansını etkiler. Örnek vermek gerekirse, küçük yığın boyutu, birçok dosyanın bir çok diskte şeritleneceği anlamına gelir, böylece tek bir dosyaya okuma ve yazma işlemlerinin paralelliğini artırır; ancak, Birden çok diskteki bloklara erişmek için konumlandırma süresi artar, çünkü tüm

istek için konumlandırma süresi, tüm sürücülerdeki isteklerin maksimum konumlandırma süreleri tarafından belirlenir.

Büyük yığın boyutu, diğer yandan, bu tür dosya içi paralelliği geliştirir ve bu nedenle yüksek verim elde etmek için birden çok eşzamanlı isteğe dayanır. Ancak, büyük parça boyutları konumlandırma süresini azaltır; örneğin, tek bir dosya bir yığının içine sığarsa ve bu nedenle tek bir diske yerleştirilirse, ona erişim sırasında gerçekleşen konumlandırma süresi yalnızca tek bir diskin konumlandırma süresi olacaktır. Bu nedenle, "en iyi" yığın boyutunun belirlenmesi zordur, çünkü bir disk sistemine [CL95] sunulan iş yükü hakkında çok fazla bilgi. Bu tartışmanın geri kalanında, dizinin yığın boyutunu tek bir bloğun (4KB) kullandığını varsayacağız. Çoğu dizi daha büyük yığın boyutları kullanır (ör. 64 KB), ancak aşağıda tartışacağımız sorunlar için tam yığın boyutu önemli değildir; bu nedenle basitlik adına tek bir blok kullanıyoruz.

RAID-0 Analizine Geri Dön

Şimdi şeritlemenin kapasitesini, güvenilirliğini ve performansını değerlendirelim. Kapasite açısından mükemmel: her biri B boyutlu bloklardan N disk verildiğinde, şeritleme N ·B faydalı kapasite bloğu sağlar. Güvenilirlik açısından, şeritleme de mükemmeldir, ancak kötü bir şekilde: herhangi bir disk arızası veri kaybına yol açacaktır. Son olarak, performans mükemmeldir: kullanıcı G/Ç isteklerine hizmet vermek için tüm diskler genellikle paralel olarak kullanılır.

RAID Performans Değerlendirme

RAID performansını analiz ederken, iki farklı performans ölçümü göz önünde bulundurulabilir. İlki, tek istek gecikmesidir.

Bir RAID'e yapılan tek bir G/Ç talebinin gecikmesini anlamak, tek bir mantıksal G/Ç işlemi sırasında ne kadar paralellik olabileceğini ortaya koyduğu için yararlıdır. İkincisi, RAID'in sabit durum verimidir, yani birçok eşzamanlı isteğin toplam bant genişliğidir. Çünkü RAID'ler genellikle yüksek performansta kullanılır.

Ortamlarda, kararlı durum bant genişliği kritiktir ve bu nedenle analizlerimizin ana odak noktası olacaktır. Verimi daha ayrıntılı olarak anlamak için, ilgilenilen bazı iş yüklerini ortaya koymamız gerekiyor. Bu tartışma için iki tür iş yükü olduğunu varsayacağız: sıralı ve rastgele. Sıralı bir iş yüküyle, diziye yapılan isteklerin büyük bitişik parçalar halinde geldiğini varsayarız; örneğin, x bloğunda başlayan ve blokta (x+1 MB) biten 1 MB veriye erişen bir istek (veya istek dizisi), sıralı kabul edilir. Sıralı iş yükleri birçok ortamda yaygındır (bir anahtar sözcük için büyük bir dosyada arama yapmayı düşünün) ve bu nedenle önemli kabul edilirler. Rastgele iş yükleri için her isteğin oldukça küçük olduğunu varsayarız.

Ve her isteğin diskte farklı bir rasgele konuma yapıldığını. Örneğin, rastgele bir istek akışı önce 10 mantıksal adresinden, ardından 550.000 mantıksal adresinden ve ardından 20.100'den vb. 4KB'ye erişebilir. Bir veritabanı yönetim sistemindeki (DBMS) işlemsel iş yükleri gibi bazı önemli iş yükleri, bu tür bir erişim modeli sergiler ve bu nedenle önemli bir iş yükü olarak kabul edilir. Elbette, gerçek iş yükleri o kadar basit değildir ve genellikle sıralı ve rastgele görünen bileşenlerin yanı sıra ikisi arasındaki davranışların bir karışımına sahiptir. Basitlik için, sadece bu iki olasılığı göz önünde bulunduruyoruz. Sizin de görebileceğiniz gibi sıralı ve rastgele iş yükleri, bir diskten çok farklı performans özellikleriyle sonuçlanacaktır. Sıralı erişimle, bir disk en verimli modunda çalışır, dönüşü aramak ve beklemek için çok az zaman harcar ve zamanının çoğunu veri aktarmak için harcar. Rastgele erişimde bunun tam tersi doğrudur: çoğu zaman rotasyonu aramak ve beklemek için harcanır ve veri aktarımı için nispeten daha az zaman harcanır. Analizimizde bu farkı yakalamak için, bir diskin sıralı bir iş yükü altında S MB/s hızında ve rastgele bir iş yükü altında R MB/s hızında veri aktarabileceğini varsayacağız. Genel olarak, S, R'den çok daha büyüktür (yani, $S \gg R$).

Bu farkı anladığımızdan emin olmak için basit bir alıştırma yapalım. Spesifik olarak, aşağıdaki disk özellikleri göz önüne alındığında S ve R'yi hesaplayalım. Ortalama olarak 10 MB boyutunda sıralı bir aktarım ve ortalama 10 KB boyutunda rastgele bir aktarım varsayalım. Ayrıca, aşağıdaki disk özelliklerini varsayın:

Ortalama arama süresi 7 ms

Ortalama dönüş gecikmesi 3 ms

Disk aktarım hızı 50 MB/sn

S'yi hesaplamak için, önce tipik bir 10 MB aktarımda zamanın nasıl harcandığını bulmamız gerekir. Önce 7 ms aramaya, ardından 3 ms dönmeye harcıyoruz. Sonunda transfer başlar; 50 MB/s'de 10 MB, aktarımda harcanan saniyenin 1/5'ine veya 200 ms'ye yol açar. Böylece, her 10 MB istek için, isteği tamamlamak için 210 ms harcıyoruz. S'yi hesaplamak için, sadece bölmemiz gerekiyor:

$$S = \frac{Amount\ of\ Data}{Time\ to\ access} = \frac{10\ MB}{210\ ms} = 47.62\ MB/s$$

Gördüğümüz gibi, veri aktarımı için harcanan çok zaman nedeniyle S, diskin tepe bant genişliğine çok yakındır (arama ve döndürme maliyetleri amortize edilmiştir).

R'yi benzer şekilde hesaplayabiliriz. Arama ve döndürme aynıdır; daha sonra transferde harcanan süreyi hesaplıyoruz, bu 10 KB @ 50 MB/s veya 0,195ms'dir.

$$R = \frac{Amount\ of\ Data}{Time\ to\ access} = \frac{10\ KB}{10.195\ ms} = 0.981\ MB/s$$

Gördüğümüz gibi, R 1 MB/sn'den az ve S/R neredeyse 50.

Yeniden RAID-0 Analizine Dön

Şimdi şeritleme performansını değerlendirelim. Yukarıda söylediğimiz gibi, genellikle iyidir. Gecikme açısından, örneğin, tek bloklu bir isteğin gecikmesi, tek bir diskinkiyle hemen hemen aynı olmalıdır; ne de olsa RAID-0, bu isteği disklerinden birine yönlendirecektir.

Kararlı durum sıralı iş hacmi açısından, sistemin tam bant genişliğini elde etmeyi bekleriz. Böylece verim, N (disk sayısı) ile S (tek bir diskin sıralı bant genişliği) çarpımına eşittir. Çok sayıda rasgele G/Ç için yine tüm diskleri kullanabilir ve böylece N · R MB/s elde edebiliriz. Aşağıda göreceğimiz gibi, bu değerler hem hesaplanması en basit olanlardır hem de diğer RAID seviyeleri ile karşılaştırıldığında bir üst sınır görevi görecektir.

RAID Seviye 1: Yansıtma

Şeritlemenin ötesindeki ilk RAID seviyemiz, RAID seviye 1 veya ikizleme olarak bilinir. Yansıtılmış bir sistemle, sistemdeki her bloğun birden fazla kopyasını yaparız; her kopya elbette ayrı bir diske yerleştirilmelidir. Bunu yaparak, disk arızalarını tolere edebiliriz.

Tipik bir ikizlenmiş sistemde, RAID'in her mantıksal blok için bunun iki fiziksel kopyasını tuttuğunu varsayacağız. İşte bir örnek:

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

Örnekte, disk 0 ve disk 1 aynı içeriğe sahiptir ve disk 2 ve disk 3 de aynıdır; veriler bu ayna çiftleri boyunca şeritlenir. Aslında, blok kopyaları diskler arasında yerleştirmenin birkaç farklı yolu olduğunu fark etmişsinizdir. Yukarıdaki düzenleme yaygın bir düzenlemedir ve bazen RAID-10 (veya RAID 1+0, şerit aynalar) olarak adlandırılır çünkü yansıtılmış çiftler (RAID-1) ve ardından bunların

üzerinde şeritler (RAID-0) kullanır; diğer bir yaygın düzenleme, iki büyük şeritleme (RAID-0) dizisi ve ardından bunların üzerinde ikizlemeler (RAID-1) içeren RAID-01'dir (veya RAID 0+1, aynalı şeritler). Şimdilik, yukarıdaki düzeni varsayarak yansıtma hakkında konuşacağız.

Yansıtılmış bir diziden bir bloğu okurken, RAID'in bir seçeneği vardır: her iki kopyayı da okuyabilir. Örneğin, RAID'e mantıksal blok 5'e bir okuma verilirse, onu disk 2'den veya disk 3'ten okumak serbesttir. Bununla birlikte, bir blok yazarken böyle bir seçenek yoktur: RAID, mantıksal bloğun her iki kopyasını da güncellemelidir. Verilerin güvenilirliğini korumak için. Ancak, bu yazma işlemlerinin paralel olarak gerçekleşebileceğini unutmayın; örneğin, 5. mantıksal bloğa bir yazma aynı anda 2. ve 3. disklere ilerleyebilir.

RAID-1 Analiz

Hadi RAID-1'i değerlendirelim. Kapasite açısından, RAID-1 pahalıdır; yansıtma seviyesini 2 aldığımıza, en yüksek faydalı kapasitemizin yalnızca yarısını elde ederiz. n kadar diskin b blokları ile birlikte, RAID-1 kullanışlı kapasite (n.b) 2 'dir.

Güvenirlik açısından, RAID-1 çok iyi iş çıkartır. Herhangi bir diskteki hatayı tolere edebilir. RAID-1'in biraz şansla bundan daha iyisini yapabileceğini de fark edebilirsiniz. Hayal edin, yukarıdaki şekilde, disk 0 ve disk 2'in aynı anda hata verdiğini. Böyle bir durumda, hiçbir veri kaybı olmaz! Genellikle, yansıtmalı sistem (yansıtma seviye 2 de) 1 diskin hatasını kesinlikle ve hangi diskin hata verdiğine göre N/2 sine kadar tolere edebilir. Pratikte, biz genellikle böyle durumları şansa bırakmak istemeyiz; ancak çoğu kişi, yansıtmanın tek bir hata için iyi olduğunu düşünür. Son olarak, performansı analiz edeceğiz. Tek bir okuma isteğinin gecikmesi açısından, tek bir diskteki gecikmeyle aynı olduğunu görebiliriz; tüm RAID-1 ler okuma işlevini kopyalarından birine yönlendirir. Yazmak ise biraz farklı: işlem bitmeden 2 fiziksel yazım uygular. Bu iki yazım paralel ve neredeyse tek yazımla aynı sürede; ancak, mantıksal yazım iki fiziksel yazımında tamamlanmasını bekler. İki istek arasında en kötü arama ve dönüş gecikmesinden muzdariptir ve bu nedenle (ortalama olarak) tek bir diske yazmaya göre biraz daha yüksek olacaktır.

BİR YANDAN: RAID TUTARLI GÜNCELLEME PROBLEMİ

RAID-1 analiz etmeden önce, ilk olarak birden fazla diskli RAID sistemlerinde yükselen bir problemi tartışalım, **tutarlı güncelleme problemi [DAA05]** olarak bilinen. Sorun, tek bir mantıksal işlem sırasında birden çok diski güncellemesi gereken herhangi bir RAID'e yazma sırasında ortaya çıkar. Bu durumda, ikizlenmiş bir disk dizisini düşündüğümüzü varsayalım. Yazmanın RAID'e verildiğini ve ardından RAID'in bunun iki diske, disk 0 ve disk 1'e yazılması

gerektiğine karar verdiğini hayal edin. RAID daha sonra disk 0'a yazma işlemini gerçekleştirir, ancak RAID disk 1'e istekte bulunmadan hemen önce bir güç kaybı (veya sistem çökmesi) meydana gelir. Bu talihsiz durumda, 0 diskine yönelik talebin tamamlandığını varsayalım (ancak disk 1'e yapılan talebin, hiçbir zaman yapılmadığı için açıkça yapılmadı). Bu zamansız güç kaybının sonucu, bloğun iki kopyasının **tutarsız** olmamasıdır; disk 0'daki kopya yeni sürümdür ve disk 1'deki kopya eskidir. Olmasını istediğimiz şey, her iki diskin durumunun atomik olarak değişmesi, yani ya her ikisi de yeni sürüm olarak bitmeli ya da hiçbiri. Bu sorunu çözmenin genel yolu, bunu yapmadan önce RAID'in ne yapmak üzere olduğunu (yani, iki diski belirli bir veri parçasıyla yükseltmek) kaydetmek için bir tür önceden yazma günlüğü kullanmaktır. Bu yaklaşımı benimseyerek, bir çarpışma durumunda doğru şeyin olmasını sağlayabiliriz; bekleyen tüm işlemleri RAID'de yeniden yürüten bir kurtarma prosedürü çalıştırarak, iki ikizlenmiş kopyanın (RAID-1 durumunda) senkronize olmadığından emin olabiliriz. Son bir not: Her yazma işleminde diskte oturum açmak çok pahalı olduğundan, çoğu RAID donanımı, bu tür bir günlük kaydı gerçekleştirdiği yerde az miktarda geçici olmayan RAM (örn. pil destekli) içerir. Böylece diske giriş yapmanın yüksek maliyeti olmadan tutarlı güncelleme sağlanır. Kararlı durum verimini analiz etmek için sıralı iş yüküyle başlayalım. Sırayla diske yazarken, her mantıksal yazma iki fiziksel yazmayla sonuçlanmalıdır; örneğin, mantıksal blok 0'ı (yukarıdaki şekilde) yazdığımızda, RAID dahili olarak bunu hem disk 0'a hem de disk 1'e yazar. Böylece, yansıtılmış bir diziye sıralı yazma sırasında elde edilen maksimum bant genişliğinin (N/2 * S) veya tepe bant genişliğinin yarısı olduğu sonucuna varabiliriz. Ne yazık ki sıralı bir okumada aynı performansı elde ediyoruz. Verilerin yalnızca bir kopyasını okuması gerektiğinden, her ikisini birden değil, sıralı okumanın daha iyi yapabileceği düşünülebilir. Ancak, bunun neden pek yardımcı olmadığını göstermek için bir örnek kullanalım. 0,1,2,3,4,5,6 ve 7 bloklarını okumamız gerektiğini düşünün. Diyelim ki 0'ın okumasını disk 0'a, 1'in okumasını disk 2'ye, 2'nin okumasını disk 1'e veriyoruz., ve 3'ün okuması disk 3'e yapılır. Sırasıyla 4,5,6'ya ve 7'nin okunması 0,2,1 ve 3 disklerine verilerek devam edilir. Tüm diskleri kullandığımız için dizinin tam bant genişliğine ulaştığımız safça düşünülebilir. Bununla birlikte, durumun (zorunlu olarak) böyle olmadığını görmek için, tek bir diskin (disk 0 diyelim) aldığı istekleri göz önünde bulundurun. İlk olarak, blok 0 için bir istek alır; ardından 4. blok için bir talep alır (2. blok atlanır). Aslında, her disk diğer her blok için bir istek alır. Atlanan blok üzerinde dönerken istemciye faydalı bant genişliği sağlamıyor. Bu nedenle, her disk en yüksek bant genişliğinin yalnızca yarısını sunacaktır. Ve bu nedenle sıralı okuma, yalnızca (N/2 .S) MB/s'lik bir bant genişliği elde edecektir. Rastgele okumalar, ikizlenmiş bir RAID için en iyi durumdur. Bu durumda, okumaları tüm disklere

dağıtabilir ve böylece tam olası bant genişliğini elde edebiliriz. Bu nedenle, rastgele okumalar için RAID-1, N*R MB/s sunar. Son olarak, rastgele yazma işlemleri beklediğiniz gibi çalışır: N/2*R MB/sn. Her mantıksal yazma, iki fiziksel yazmaya dönüşmelidir ve bu nedenle, tüm diskler kullanımdayken, istemci bunu yalnızca mevcut bant genişliğinin yarısı olarak algılayacaktır. Mantıksal blok X'e yazma, iki farklı fiziksel diske iki paralel kabloya dönüşse de birçok küçük isteğin bant genişliği şeritlemede gördüğümüzün yalnızca yarısına ulaşır. Yakında göreceğimiz gibi, mevcut bant genişliğinin yarısını elde etmek gerçekten oldukça iyi!

RAID SEVİYE 4: EŞLİK İLE YER KAZANMA

Şimdi, **eşlik** olarak bilinen bir disk dizisine artıklık eklemenin farklı bir yöntemini sunuyoruz. Parite tabanlı yaklaşımlar, daha az kapasite kullanmaya çalışır ve böylece ikizlenmiş sistemlerin ödediği devasa alan cezasının üstesinden gelir. Ancak bunu bir maliyet karşılığında yaparlar: performans.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

İşte örnek bir beş diskli RAID-4 sistemi. Her veri şeridi için, o blok şeridi için fazlalık bilgileri depolayan tek bir eşlik bloğu ekledik. Eşlik bloğu için P1, 4,5,6 ve 7 bloklarından hesapladığı fazladan bilgiye sahiptir. Eşliği hesaplamak için, şeridimizden herhangi bir bloğun kaybına dayanmamızı sağlayan matematiksel bir fonksiyon kullanmamız gerekiyor. Görünüşe göre XOR basit işlevi oldukça iyi yapıyor. Belirli bir bit kümesi için, tüm bu bitlerin XOR'u, bitlerde çift sayıda 1 varsa 0 ve tek sayıda 1 varsa 1 döndürür. Örneğin: İlk satırda (0,0,1,1), iki 1 (C2, C3) vardır ve dolayısıyla tüm bu değerlerin XOR'u 0 (P) olacaktır; benzer şekilde, ikinci satırda yalnızca bir 1 (C1) vardır ve bu nedenle XOR, 1 (P) olmalıdır.

C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0
0	1	0	0	XOR(0,1,0,0) = 1

Bunu basit bir şekilde hatırlayabilirsiniz: herhangi bir sıradaki 1'lerin sayısı, eşlik biti dahil, çift (tek değil) bir sayı olmalıdır; bu, paritenin doğru olması için RAID'in sürdürmesi gereken değişmezdir. Yukarıdaki örnekten, eşlik bilgilerinin bir

arızadan kurtulmak için nasıl kullanılabileceğini de tahmin edebilirsiniz. C2 etiketli sütunun kaybolduğunu hayal edin. Sütunda hangi değerlerin olması gerektiğini bulmak için, o satırdaki diğer tüm değerleri okumamız (XOR eşlik biti dahil) ve doğru cevabı yeniden yapılandırmamız gerekir. Spesifik olarak, C2 sütunundaki ilk satırın değerinin kaybolduğunu varsayalım (bu bir 1'dir); o satırdaki diğer değerleri okuyarak (C0'dan 0, C1'den 0, C3'ten 1 ve P eşlik sütunundan 0) 0, 0, 1 ve 0 değerlerini alıyoruz. Her satırdaki çift 1 sayısı, eksik verinin ne olması gerektiğini biliyoruz: a 1 . Ve XOR tabanlı bir parite şemasında yeniden yapılanma bu şekilde çalışır! Yeniden yapılandırılmış değeri nasıl hesapladığımıza da dikkat edin: ilk etapta pariteyi hesapladığımız şekilde, veri bitlerini ve eşlik bitlerini birlikte XOR'larız. Şimdi merak ediyor olabilirsiniz: tüm bu bitleri XOR'lamaktan bahsediyoruz ve yine de yukarıdan biliyoruz ki RAID her diske 4 KB (veya daha büyük) bloklar yerleştiriyor; pariteyi hesaplamak için bir grup bloğa XOR'u nasıl uygularız? Bu da kolay çıkıyor. Basitçe, veri bloklarının her bir biti boyunca bit düzeyinde bir XOR gerçekleştirin; her bitsel XOR'un sonucunu parite bloğundaki karşılık gelen bit yuvasına yerleştirin. Örneğin, 4 bit boyutunda bloklarımız olsaydı (evet, bu hala 4 KB'lik bir bloktan biraz daha küçüktür, ancak resmi anladınız), şöyle görünebilirler:

Block0	Block1	Block2	Block3	Parity
00	10	11	10	11
10	01	00	01	10

Şekilden de görebileceğiniz gibi, her bloğun her biti için parite hesaplanır ve sonuç parite bloğuna yerleştirilir.

RAID-4 ANALİZİ

Hadi şimdi RAID-4'ü analiz edelim. Kapasite açısından RAID-4, koruduğu her disk grubu için eşlik bilgisi olarak 1 disk kullanır. Böylece, bir RAID grubu için faydalı kapasitemiz (N-1) * B'dir.

Güvenilirliğin anlaşılması da oldukça kolaydır: RAID-4, 1 disk arızasını tolere eder ve daha fazlasını değil. Birden fazla disk kaybolursa, kaybolan verileri yeniden oluşturmanın hiçbir yolu yoktur.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

Son olarak, performans var. Bu kez, sabit durum verimini analiz ederek başlayalım. Ardışık okuma performansı, eşlik diski dışındaki tüm diskleri kullanabilir ve bu nedenle (N-1) * S MB/s (kolay bir durum) değerinde bir tepe etkin bant genişliği sağlar. Sıralı yazmaların performansını anlamak için önce nasıl yapıldığını anlamalıyız. Diske büyük miktarda veri yazarken RAID-4, **tam şerit yazma** olarak bilinen basit bir optimizasyon gerçekleştirebilir. Örneğin, 0,1,2 ve 3 bloklarının bir yazma talebinin parçası olarak RAID'e gönderildiği durumu hayal edin. Bu durumda, RAID basitçe P0'ın yeni değerini hesaplayabilir (0,1,2 ve 3 blokları arasında bir XOR gerçekleştirerek) ve ardından tüm blokları (parite bloğu dahil) yukarıdaki beş diske paralel olarak yazabilir. (Şekilde gri renkle vurgulanmıştır). Bu nedenle, tam şeritli yazma işlemleri, RAID-4'ün diske yazmasının en verimli yoludur. Tam şerit yazmayı anladıktan sonra, RAID-4'te sıralı yazmaların performansını hesaplamak kolaydır; etkin bant genişliği de (N-1) * S MB/sn'dir. İşlem sırasında parite diski sürekli kullanımda olsa da istemci bundan performans avantajı elde etmez.

Şimdi rastgele okumaların performansını analiz edelim. Yukarıdaki şekilden de görebileceğiniz gibi, 1 blokluk rasgele okumalar sistemin veri disklerine dağıtılacak, ancak parite diskine dağıtılmayacaktır. Böylece etkin performans: (N-1) *R MB/s.

En sona sakladığımız rastgele yazmalar, RAID-4 için en ilginç durumu sunuyor. Yukarıdaki örnekte blok 1'in üzerine yazmak istediğimizi düşünün. Devam edip üzerine yazabilirdik ama bu bizi bir sorunla baş başa bırakırdı: parite bloğu P0 artık şeridin doğru parite değerini doğru bir şekilde yansıtmayacaktı; bu örnekte, P0 da yükseltilmelidir. Hem doğru hem de verimli bir şekilde nasıl güncelleyebiliriz?

Burada iki yöntem olduğu ortaya çıktı. **Ek parite** olarak bilinen ilki, aşağıdakileri yapmamızı gerektirir. Yeni eşlik bloğunun değerini hesaplamak için, şeritteki diğer tüm veri bloklarını paralel olarak okuyun (örnekte, blok 0, 2 ve 3) ve yeni blokla (1) XOR. Sonuç, yeni eşlik bloğunuzdur. Yazmayı tamamlamak için, yeni verileri ve yeni pariteyi yine paralel olarak ilgili disklere yazabilirsiniz.

Bu teknikle ilgili sorun, disk sayısıyla ölçeklenmesi ve bu nedenle daha büyük RAID'lerde hesaplama eşliği için yüksek sayıda okuma gerektirmesidir. Böylece, **çıkarıcı parite** yöntemi.

Örneğin, bu bit dizisini hayal edin (4 veri biti, bir eşlik):

C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0

C2 bitinin üzerine $C2_{new}$ diyeceğimiz yeni bir değer yazmak istediğimizi düşünelim. Çıkarma yöntemi, oradaki adımlarda çalışır. Önce C2'deki eski verileri ($C2_{old}$ = 1) ve eski pariteyi (P_{old} = 0) okuruz. Ardından eski verilerle yeni verileri karşılaştırıyoruz; eğer aynı iseler (örneğin, $C2_{new}$ = $C2_{old}$), o zaman parite bitinin de aynı kalacağını biliyoruz (yani, P_{new} = P_{old}). Ancak farklılarsa, o zaman eski parite bitini mevcut durumunun tersine çevirmeliyiz, yani eğer (P_{old} ==1), P_{new} = olarak ayarlanacak; (P_{old} ==0) ise, P_{new} 1'e ayarlanır. Tüm bu karışıklığı XOR ile düzgün bir şekilde ifade edebiliriz (burada $\oplus$, XOR operatörüdür):

$$P_{new} = (C_{old} \oplus C_{new}) \oplus P_{old}$$

Bitlerle değil bloklarla uğraştığımız için, bu hesaplamayı bloktaki tüm bitler üzerinden yaparız (örneğin, her bloktaki 4096 bayt çarpı bayt başına 8 bit). Bu nedenle çoğu durumda yeni blok eski bloktan farklı olacaktır ve dolayısıyla yeni parite bloğu da farklı olacaktır. Artık ne zaman toplamsal parite hesaplamasını ve ne zaman çıkarma yöntemini kullanacağımızı anlayabilmelisiniz; kesişme noktası nedir? Bu performans analizi için çıkarma yöntemini kullandığımızı varsayalım. Bu nedenle, her yazma işlemi için RAID'in 4 fiziksel G/Ç gerçekleştirmesi gerekir (iki okuma ve iki yazma). Şimdi RAID'e gönderilen çok sayıda yazma olduğunu hayal edin; RAID-4 paralel olarak kaç tanesini gerçekleştirebilir? anlamak için RAID-4 düzenine tekrar bakalım.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

Şimdi RAID-4'e yaklaşık aynı anda 4 ve 13 numaralı bloklara (şemada * ile işaretlenmiş) gönderilen 2 küçük yazma olduğunu hayal edin. Bu disklerin verileri 0 ve 1 disklerindedir ve bu nedenle verilere okuma ve yazma paralelde olabilir ki bu iyidir. Ortaya çıkan sorun, eşlik diskindedir; her iki istek de 4 ve 13 için ilgili eşlik bloklarını, eşlik blokları 1 ve 3'ü (+ ile işaretlenmiş) okumak zorundadır. Umarız sorun artık açıktır: eşlik diski bu tür iş yükü altında bir darboğazdır; bu nedenle bazen buna parite tabanlı RAID'ler için **küçük yazma sorunu** diyoruz. Umarız sorun artık açıktır: eşlik diski bu tür iş yükü altında bir darboğazdır; bu nedenle bazen buna parite tabanlı RAID'ler için küçük yazma sorunu diyoruz.

RAID-4'teki G/Ç gecikmesini analiz ederek sonuca varıyoruz. Artık bildiğiniz gibi, tek bir okuma (hata olmadığı varsayılarak) yalnızca tek bir diske eşlenir ve bu nedenle gecikme süresi, tek disk isteğinin gecikme süresine eşdeğerdir. Tek bir yazmanın gecikmesi, iki okuma ve ardından iki yazma gerektirir; okumalar, yazmalar gibi paralel olarak gerçekleşebilir ve bu nedenle toplam gecikme, tek bir diskin yaklaşık iki katıdır (bazı farklılıklar vardır, çünkü her iki okumanın da tamamlanmasını beklememiz ve böylece en kötü durum konumlandırma süresini elde etmemiz gerekir, ancak sonra yükselmeler arama maliyetine neden olmaz ve bu nedenle ortalamadan daha iyi bir konumlandırma maliyeti olabilir).

RAID SEVİYE 5: DÖNDÜRME PARİTESİ

Küçük yazma sorununu çözmek için (en azından kısmen), Patterson, Gibson ve Katz RAID-5'i tanıttı. RAID-5, eşlik bloğunu sürücüler arasında **döndürmesi** dışında RAID-4 ile hemen hemen aynı şekilde çalışır.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

Gördüğünüz gibi, RAID-4 için eşlik diski darboğazını ortadan kaldırmak için artık her şerit için eşlik bloğu diskler arasında döndürülüyor.

RAID-5 ANALİZİ

RAID-5 için yapılan analizlerin çoğu, RAID-4 ile aynıdır. Örneğin, iki seviyenin etkin kapasitesi ve arıza toleransı aynıdır. Sıralı okuma ve yazma performansı da öyle. Tek bir isteğin (okuma veya yazma) gecikme süresi de RAID-4 ile aynıdır. Rastgele okuma performansı biraz daha iyi çünkü artık tüm diskleri kullanabiliyoruz. Son olarak, rastgele yazma performansı, istekler arasında paralelliğe izin verdiği için RAID-4'e göre belirgin şekilde iyileşir. Blok 1'e bir yazma ve blok 10'a bir telgraf düşünün; bu, disk 1 ve disk 4'e (blok 1 ve paritesi için) ve disk 0 ve disk 2'ye (blok 10 ve paritesi için) yapılan isteklere dönüşecektir.

	RAID-0	RAID-1	RAID-4	RAID-5
Capacity	$N \cdot B$	$(N \cdot B)/2$	$(N-1) \cdot B$	$(N-1) \cdot B$
Reliability	0	1 (for sure) $\frac{N}{2}$ (if lucky)	1	1
Throughput				
Sequential Read	$N \cdot S$	$(N/2) \cdot S^1$	$(N-1) \cdot S$	$(N-1) \cdot S$
Sequential Write	$N \cdot S$	$(N/2) \cdot S^1$	$(N-1) \cdot S$	$(N-1) \cdot S$
Random Read	$N \cdot R$	$N \cdot R$	$(N-1) \cdot R$	$N \cdot R$
Random Write	$N \cdot R$	$(N/2) \cdot R$	$\frac{1}{2} \cdot R$	$\frac{N}{4}R$
Latency				
Read	T	T	T	T
Write	T	T	$2T$	$2T$

Böylece paralel ilerleyebilirler. Aslında, çok sayıda rasgele istek verildiğinde, tüm diskleri eşit şekilde meşgul tutabileceğimizi varsayabiliriz. Durum buysa, küçük yazmalar için toplam bant genişliğimiz N/4 * R MB/s olacaktır. Dört kayıp faktörü, her bir RAID-5 yazmanın hala toplam 4 G/Ç işlemi oluşturmasından kaynaklanır; bu, yalnızca eşlik tabanlı RAID kullanmanın maliyetidir.

RAID-5, daha iyi olduğu birkaç durum dışında temel olarak RAID-4 ile aynı olduğundan, pazarda neredeyse tamamen RAID-4'ün yerini almıştır. Olmadığı tek yer, büyük yazma dışında hiçbir şey yapmayacaklarını bilen, böylece küçük yazma sorununu tamamen ortadan kaldıran sistemlerdir [HLM94]; bu durumlarda, oluşturulması biraz daha basit olduğu için RAID-4 bazen kullanılır.

RAID Karşılaştırması: Özet

Şimdi RAID düzeylerinin basitleştirilmiş karşılaştırmasını özetliyoruz. Analizimizi basitleştirmek için bazı ayrıntıları atladığımızı unutmayın. Örneğin, ikizlenmiş bir sistemde yazarken, ortalama arama süresi maksimum iki aramadır (her diskte bir tane). Bu nedenle, çekme disklerine rasgele yazma performansı genellikle tek bir diskin rasgele yazma performansından biraz daha az olacaktır. Ayrıca, eşlik diskini RAID-4/5'te güncellerken, eski eşliğin ilk okuması büyük olasılıkla tam bir arama ve dönüşe neden olur, ancak eşliğin ikinci yazılması yalnızca dönüşle sonuçlanır. Son olarak, ikizlenmiş RAID'lere yönelik sıralı G/Ç, diğer yaklaşımlara kıyasla 2 kat performans kaybı öder.

Bununla birlikte, karşılaştırma, temel farklılıkları yakalar ve RAID düzeylerindeki ödünleşimlerin anlaşılması için yararlıdır. Gecikme analizi için, tek bir diske yapılan bir isteğin alacağı süreyi temsil etmek için T'yi kullanırız. Sonuç olarak, kesinlikle performans istiyorsanız ve güvenilirliği umursamıyorsanız, sıyırma

kesinlikle en iyisidir. Ancak rastgele G/Ç performansı ve güvenilirliği istiyorsanız, yansıtma en iyisidir; ödediğiniz maliyet kayıp kapasitededir. Kapasite ve güvenilirlik ana hedeflerinizse, RAID-5 kazanır; ödediğiniz maliyet küçük yazma performansındadır. Son olarak, her zaman sıralı G/Ç yapıyorsanız ve kapasiteyi en üst düzeye çıkarmak istiyorsanız, RAID-5 de en mantıklısıdır.

DİĞER İLGİ ÇEKİCİ RAID SORUNLARI

RAID hakkında düşünürken tartışılabilecek (ve belki de tartışılması gereken) bir dizi başka ilginç fikir var. İşte sonunda hakkında yazabileceğimiz bazı şeyler. Örneğin, orijinal taksonomiden Düzey 2 ve 3 ve çoklu disk hatalarını tolere etmek için Düzey 6 dahil olmak üzere birçok başka RAID tasarımı vardır [C+04].

Bir disk arızalandığında RAID'in yaptığı da vardır; bazen arızalı diski doldurmak için bekleyen etkin bir yedeği vardır. Başarısız durumdaki performansa ve arızalı diskin yeniden oluşturulması sırasındaki performansa ne olur? Gizli sektör hatalarını hesaba katmak veya bozulmayı engellemek [B+08] için daha gerçekçi hata modelleri ve bu tür hataları işlemek için birçok teknik vardır (ayrıntılar için veri bütünlüğü bölümüne bakın). Son olarak RAID'i bir yazılım katmanı olarak bile oluşturabilirsiniz: bu tür yazılım RAID sistemleri daha ucuzdur ancak tutarlı güncelleme sorunu [DAA05] dahil olmak üzere başka sorunları vardır.

ÖZET

RAID'i tartıştık. RAID, bir dizi bağımsız diski büyük, daha geniş ve daha güvenilir tek bir varlığa dönüştürür; daha da önemlisi, bunu şeffaf bir şekilde yapar ve bu nedenle yukarıdaki donanım ve yazılım, değişiklikten nispeten habersizdir. Aralarından seçim yapabileceğiniz pek çok olası RAID düzeyi vardır ve kullanılacak tam RAID düzeyi büyük ölçüde son kullanıcı için neyin önemli olduğuna bağlıdır. Örneğin, ikizlenmiş RAID basit, güvenilirdir ve genellikle yüksek bir kapasite maliyeti karşılığında iyi performans sağlar. Buna karşılık RAID-5, güvenilirdir ve kapasite açısından daha iyidir, ancak iş yükünde küçük yazmalar olduğunda oldukça düşük performans gösterir. Belirli bir iş yükü için bir RAID seçmek ve parametrelerini (yığın boyutu, disk sayısı vb.) uygun şekilde ayarlamak zordur ve bilimden çok bir sanat olmaya devam etmektedir.

SORULAR

1. Bazı temel RAID eşleme testleri gerçekleştirmek için simülatörü kullanın. Farklı seviyelerde (0, 1, 4, 5) ÇALIŞTIRIN ve bir dizi isteğin eşlemelerini çözüp çözemeyeceğinize bakın. RAID-5 için sol simetrik ve sol asimetrik düzenler arasındaki farkı anlayıp anlayamadığınıza bakın. Yukarıdakinden farklı problemler oluşturmak için bazı farklı rasgele seedler kullanın.

Cevap: Sol asimetrik düzenin garip bir yol olduğunu düşünüyorum ve nasıl yardım ettiği konusunda fikrim yok.

2. İlk problemin aynısını yapın, ancak bu sefer yığın boyutunu -C ile değiştirin. Yığın boyutu eşlemeleri nasıl değiştirir?
   Cevap: Daha büyük yığın kullanmak, seviye 5 için aynı diske birden fazla eşlik bloğu koyduğunu söyleyebilirim.
3. Yukarıdakinin aynısını yapın, ancak her sorunun doğasını tersine çevirmek için -r bayrağını kullanın.
   Cevap:

```
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse True
ARG timing False

8444 1
LOGICAL READ from addr:8444 size:4096
  read  [disk 0, offset 2111]

4205 1
LOGICAL READ from addr:4205 size:4096
  read  [disk 1, offset 1051]

5112 1
LOGICAL READ from addr:5112 size:4096
  read  [disk 0, offset 1278]

7837 1
LOGICAL READ from addr:7837 size:4096
  read  [disk 1, offset 1959]

4765 1
LOGICAL READ from addr:4765 size:4096
  read  [disk 1, offset 1191]

9081 1
LOGICAL READ from addr:9081 size:4096
  read  [disk 1, offset 2270]

2818 1
LOGICAL READ from addr:2818 size:4096
  read  [disk 2, offset 704]

6183 1
LOGICAL READ from addr:6183 size:4096
  read  [disk 3, offset 1545]

9097 1
LOGICAL READ from addr:9097 size:4096
  read  [disk 1, offset 2274]

8102 1
LOGICAL READ from addr:8102 size:4096
  read  [disk 2, offset 2025]
```

4. Şimdi ters bayrağı kullanın, ancak -s bayrağıyla her isteğin boyutunu artırın. RAID seviyesini değiştirirken 8k, 12k ve 16k boyutlarını belirtmeyi deneyin. İsteğin boyutu arttığında temeldeki G/Ç modeline ne olur? Bunu sıralı iş yükü ile de denediğinizden emin olun (-W sıralı); RAID-4 ve RAID-5 hangi istek boyutları için G/Ç açısından çok daha verimlidir?
   Cevap: RAID 4 ve 5, dört diskin tümü tek bir istekte kullanıldığında çok daha verimli olacaktır. Okumalar o zaman atlanabilir.
5. 4 disk kullanarak RAID seviyelerini değiştirirken RAID'e 100 rastgele okumanın performansını tahmin etmek için simülatörün zamanlama modunu (-5) kullanın.
   Cevap:

   ./raid.py -L 0 -t -n 100 -c
   ```
   STAT totalTime 275.69999999999993
   ```

   ./raid.py -L 1 -t -n 100 -c
   ```
   STAT totalTime 278.7
   ```

   ./raid.py -L 4 -t -n 100 -c
   ```
   STAT totalTime 386.1000000000002
   ```

   ./raid.py -L 5 -t -n 100 -c
   ```
   STAT totalTime 276.7
   ```
6. Yukarıdakilerin aynısını yapın, ancak disk sayısını artırın. Disk sayısı arttıkça her bir RAID düzeyinin performansı nasıl ölçeklenir?
   Cevap:
   ./raid.py -L 0 -n 100 -t -D 4 -c
   ```
   STAT totalTime 275.69999999999993
   ```
   ./raid.py -L 0 -n 100 -t -D 8 -c
   ```
   STAT totalTime 156.49999999999994
   ```
   ./raid.py -L 0 -n 100 -t -D 16 -c
   ```
   STAT totalTime 86.79999999999998
   ```
   ./raid.py -L 0 -n 100 -t -D 32 -c
   ```
   STAT totalTime 58.6
   ```

   Diğer seviyelerde de aynı şekilde ölçekleniyor.

7. Yukarıdakilerin aynısını yapın, ancak okumalar yerine tüm yazmaları (-w 100) kullanın. Okuma RAID düzeyi performansı şimdi nasıl ölçekleniyor? 100 rasgele yazmanın iş yükünü tamamlamanın ne kadar süreceğini kabaca tahmin edebilir misiniz?
   Cevap: 4. Seviye kötü ölçekleniyor ama diğerleri benzer şekilde.
8. Zamanlama modunu son bir kez çalıştırın, ancak bu kez sıralı bir iş yüküyle (-W sıralı). Performans, RAID düzeyine ve okuma ve yazma işlemlerine göre nasıl değişir? Her isteğin boyutunu değiştirirken ne dersiniz? RAID-4 veya RAID-5 kullanıldığında RAID'e hangi boyutta yazmalısınız?
   Cevap: Performans, sıralı iş yükü için okuma ve yazma ile değişmez. Ancak rastgele iş yükü ile, yığın boyundan* disk sayısından daha büyük istek boyutu kullanmak, daha büyük isteklerle çok iyi ölçeklenir