ELEKTRONIK TIDNINGEN

Konsten att partitionera SSD-flash

Claus Giebert
Kontron

Flashminnesbaserade SSD-enheter kan bara partitioneras logiskt, inte fysiskt. Här får du veta vilka problem detta medför, och hur de kan lösas.

Redaktör
Jan Tångring
jan@etn.se
0734-17 13 09

EMBEDDED EXPERT

Konsten att partitionera SSD-flash

Eftersom filsystemet hos flashbaserade SSD-enheter inte ger någon information om tilldelningen från logiska sektorer till fysiska sektorer kan operativsystemet inte tillhandahålla mekanismer som garanterar säker användning av en ensam SSD-enhet. Men det finns sätt att komma förbi problemet.

Claus Giebert, Kontron

Claus Giebert är produktchef för inbyggnadsdatorkort på Kontron i Eching, Tyskland. Efter examen vid Erlangen/Nürnbergs tekniska universitet, där hans specialitet var medicinsk teknik, kom han 1996 till Kontron Group (dåvarande Boards AG), där han först arbetade med utveckling av CPU-moduler. Efter att Kontron Elektronik GmbH (nu Kontron AG) tagit över ledde han i över tre år ett utvecklingsteam inriktat på kundanpassade CPU-moduler. År 2002 blev han produktchef för ”slot CPU”-produkter. Idag är han också produktchef i EMEA för M2M-produktfamiljen.

Flashminnesbaserade SSD-enheter (Solid State Drive) blir allt vanligare i inbyggnadstillämpningar som kräver kompakt utförande. De är platsbesparande och robusta, och de utlovar en livstid så lång som en till fem miljoner möjliga skrivcykler för industrikompatibla flashceller.

Detta låter ju bra, kan man tycka. Men när det gäller att uppnå hög tillgänglighet, säkerhet och lång livslängd måste man komma ihåg – åtminstone idag – att SSD:er bara kan partitioneras logiskt, och inte fysiskt. Och detta har sina konsekvenser.

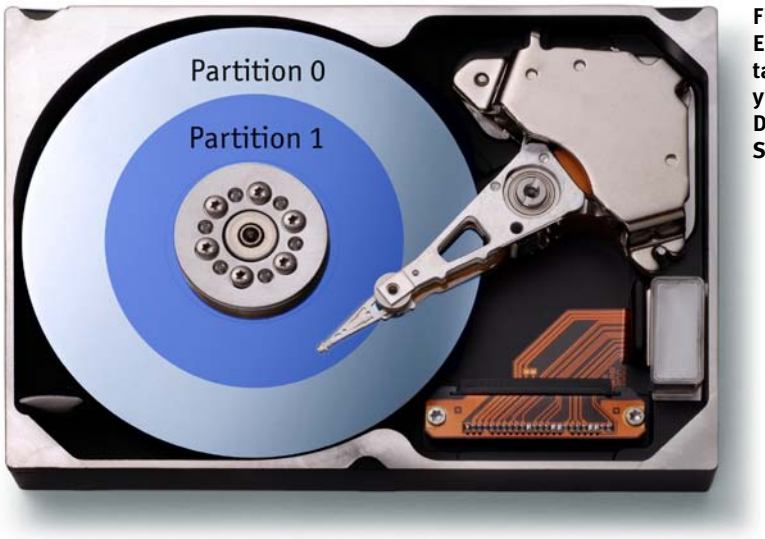
Under ett antal diskussioner med kunder på sistone, särskilt när det handlat om kompakta system som kräver hög tillgänglighet – som M2M-tillämpningar – har vi i allt högre grad mött uppfattningen att tillförlitlighetsproblem med flashminnen är relativa sällsynta. Artiklar om SSD-enheter och deras användning [1] föreslår att partitionering är den rätta metoden för att undvika ”förslitning” orsakad av ett stort antal skrivoperationer.

Vid en första anblick tycks detta också vara en rätt logisk och väl fungerande lösning, eftersom man kan lagra viktiga operativsystemsdata och tillämpningar på en partition och olika swap- och tillämpningsdata på en annan. Om sektorer i den senare partitionen – som ofta får hantera skrivoperationer – skulle råka sluta fungera är systemet ändå fortfarande tillgängligt, om än med vissa restriktioner.

Det är åtminstone så som teorin säger att det skall fungera.

Tyvärr förbiser man nästan alltid samma effekter i verkligheten. Ett system utan en fungerande swap-partition kommer till exempel sällan att starta utan olika aktiviteter från användarens sida. Så det är inte speciellt meningsfullt att skriva swap-data till en egen partition. Detta är dock det minst viktiga problemet. Viktigare är översättningen av logiska adresser till fysiska adresser, och detta är inte någon enkel uppgift när det handlar om flashbaserade SSD-enheter.

Varje sektor på en konventionell hårddisk motsvarar ett exakt definierat område på ytan av det magnetiska



Figur 1
En partition på en hårddisk upptar en exakt definierad area på ytan av det magnetiska mediet. Detta gäller tyvärr ännu inte SSD-enheter.

minnet. Därför finns det en säker rumslig separation, vilket man omedvetet förutsätter gälla även för flashbaserade SSD-enheter. Men i verkligheten ser det annorlunda ut.

Åtgärder som Wear-Leveling (förslitningsutjämning) som används för att öka SSD-enhetens livslängd, liksom att fördela skrivcykler över flera flashkanaler för att förbättra skrivegenskaperna, undergräver sådana antaganden.

Det är fortfarande viktigt att inse det praktiska värdet av att sprida riskerna för att möjliggöra dataåtervinning [3]. Speciellt kommer vi att se närmare på vilken effekt Wear-Leveling har.

Wear-leveling har beskrivits på många ställen [2]. För att förstå metoden bättre måste vi se på vilka åtgärder som används för att förbättra livslängden. Här följer en mycket enkel algoritm för Wear-Leveling, baserad på antagandet att radering medför åldrande, men läsning inte gör det:

- En cell raderas bara om ingen annan cell redan raderats
- De sektorer som har raderats minst ofta får raderas
- Skrivoperationer fördelas jämnt mellan sektorerna

En sådan algoritm måste också ta hänsyn till geometrin och storleken hos filen, men detta kommer vi inte att se närmare på här, så vi förutsätter i fortsättningen att filerna har en hanterbar storlek.

Wear-Level	3	3	3	3	3	3	3	2
Sektor/Page	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	xx	
3								

Figur 2
Uppdelningen av en SSD i sektorer och sidor med angivande av Wear-Level.

Utgångsläge

Wear-Level	10	9	10	10	10	10	10	10
Sektor/Page	0	1	2	3	4	5	6	7
0	0	xx	2	3	4	5	6	7
1	8	9	xx	11	12	13	14	15
2	16	17	18	19	20	21	xx	10
3	22	23	24	26	1	xx	xx	xx

Planerat resultat

Wear-Level	10	10	10	10	10	10	10	10
Sektor/Page	0	1	2	3	4	5	6	7
0	0	26	2	3	4	5	6	7
1	8	9	xx	11	12	13	14	15
2	16	17	18	19	20	21	xx	10
3	22	23	24	xx	1	xx	xx	xx

Figur 5
Om spänningen bortfaller och Wear-Leveling-algoritmen säger att hela sidan 1 måste raderas och data återskrivas, då kan under ett visst tidsfönster allt innehåll på sidan gå förlorat trots alla säkerhetsmekanismer.

Sektor 10

Wear-Level	3	3	4	3	3	3	3	4
Sektor/Page	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	xx	11	12	13	14	15
2	16	17	18	19	20	21	22	10
3								

Sektor 22

Wear-Level	4	3	4	3	3	3	4	4
Sektor/Page	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	xx	11	12	13	14	15
2	16	17	18	19	20	21	xx	10
3	22							

Neue Sektoren

Wear-Level	4	4	5	3	3	3	4	4
Sektor/Page	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	xx	11	12	13	14	15
2	16	17	18	19	20	21	xx	10
3	22	23	24					

Figur 3
Vid ändring av befintliga data och insamling av nya data flyttas de fysiska positionerna för de logiska sektorerna vad gäller såväl de fysiska sektorerna som de fysiska sidorna.

Wear-Level	99	98	99	99	99	99	99	99
Sektor/Page	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	xx	11	12	xx	14	15
2	xx	17	18	19	xx	21	xx	10
3	22	23	24	xx	20	13	16	xx

Figur 4
Om en sida bortfaller kan detta medföra förlust av data även på den skrivskyddade partitionen för operativsystemet. Skrivskyddet är i detta fall verkningslöst.

Det är redan uppenbart att om man skriver med Wear-Leveling, men utan fysisk partitionering, är det stor risk att det uppstår problem. Låt oss se närmare på de olika effekter som uppstår.

För det första måste vi ta hänsyn till strukturen hos en SSD (figur 2). Låt oss anta att den är uppdelad i åtta sidor (0–7), var och en med lagringsutrymme för fyra sektorer (0–3). För varje sida finns en separat räknare som registrerar hur ofta den redan raderats (till exempel betyder ”Wear-Level 3” att sidan skrivits tre gånger).

De sektorer som markerats med blått innehåller data som skrivits av användaren, medan de grönmarkerade inte tycks ha skrivits alls. Talen i sektorerna representerar de logiska sektorerna. Om data har skrivits, med sektorn inte innehåller några färska data, kommer den att identifieras med ”xx”. I detta exempel ska vi se närmare på de sidor som markerats med grått.

Låt oss anta att partition 1 motsvarar de logiska sektorerna 1–8, och att swap- och tillämpningsdata lagras i de logiska sektorerna 9 till slutet. I praktiken skulle en sådan konfiguration kunna produceras genom att man exempelvis registrerar en avbildning av en ”ny” enhet.

Mycket förenklat liknar detta proceduren vid produktion av serieenheter. Låt oss se på hur en serieenhet arbetar på fältet.

På fältet samlas processdata in och lagras, och sektorer i filsystemet modifieras i enlighet med detta. Den inverkan som filsystemet ger kan negligeras, för den gör bara situationen värre. Anta att data modifieras, först i sektor 10 och sedan i sektor 22 (figur 3). För detta behövs två nya sektorer. Datavolymen växer konstant.

Redan här, utan optimal Wear-Leveling (se Wear-Level 5 mot Wear-Level 3), är tilldelningen från logiska till fysiska sektorer inte på något vis en enkel uppgift. Efter bara fyra skrivoperationer har positionen i relation till såväl sidorna som sektorerna ändrats. Så vad händer om det antingen blir fel på en sida, eller – mindre dramatiskt mer troligt – man får ett spänningsbortfall?

Scenario 1:
Fel på en sida
Här förutsätter vi att det blir fel på en hel flashsida. Detta skulle naturligtvis hända vid en mycket hög Wear-Level. Men sektorernas Wear-Level ökar snabbt vid konstant skrivning, och sannolikheten för fel ökar därför också. Figur 4 visar att detta påverkar sektorerna 1, 9, 17 och 23.

Det framgår att sektorerna såväl i den skrivskyddade partitionen för operativsystemet som i partitionen för swap- och tillämpningsdata påverkas. Så skrivskyddet av partitionen visar sig vara ineffektivt. Det medför att filsystemet inte vet vilka sektorer som finns i en viss sida, så det går inte att vidta några säkerhetsåtgärder.

Scenario 2:
Spänningsbortfall under skrivning
Efter ett spänningsbortfall kan en hel del räddas tack vare den interna algoritm och det filsystem som används. Men det finns alltid ett specifikt tidsfönster under skrivprocessen då den ”nya” statusen inte kan registreras.

Figur 5 visar detta: Data i sektor 26 skall modifieras, och sida 1 kommer att användas eftersom den har lägst Wear-Level. Vi antar att SSD-enheten är full – det vill säga att inga logiska sektorer har kvar sitt ursprungliga tillstånd ”FF” (vilket dock inte betyder att minnet är fullt). Enligt den ovan beskrivna Wear-Level-algoritmen måste därför hela sida 1 raderas och därefter data återskrivas.

Om spänningen skulle falla bort under detta steg kommer alla data på denna sida att hamna i ett odefinierat tillstånd. Med hjälp av interna skyddsmekanismer och bättre implementering än i detta exempel kan detta tidsfönster krympas avsevärt. Men det finns fortfarande kvar i alla hittills kända implementeringar.

I vårt exempel är det sektorerna 1, 26, 9, 17 och 23 som påverkas. Om vi jämför detta med den ursprungliga partitio-

Figur 6
Om det finns tillräckligt med utrymme kan användaren välja vanliga hårddiskar, som här i en Kontron KISS-server med fordonsspecificerade diskar i en RAID-konfiguration för digital passagerarinformation på tåg. Jämfört med idag tillgängliga, flashbaserade SSD-enheter ger detta fördelarna av tydlig fysisk partitionering, plus större lagringskapacitet till lägre kostnad.

Figur 7
Helt enligt SFF-trenden: Denna box-PC ur Kontrons CB-serie (i detta fall CB 511) erbjuder både en SSD och interna eller externa kortplatser för CompactFlash-kort (här externa). Så de kan lagra operativsystemdata mycket säkert och under lång tid, eftersom dessa data endast skrivs om vid förändring av konfigurationen. Om SSD-enheten för swap- och tillämpningsdata skulle fallera p.g.a. sidfel eller spänningsbortfall kan systemen som regel boota om utan någon aktion från användaren.

neringen med operativsystemsdata i sektorerna 1-8, ser vi att såväl sektorer i den skrivskyddade operativsystemspartitionen som swap- och tillämpningsdata påverkats. Så skrivskyddet har varit verkningslöst också här. Filsystemet vet alltså inte vilka sektorer som finns på sidan, och därför går det inte att utforma någon fungerande säkerhetsfunktion.

Så länge som filsystemet hos flashbaserade SSD-enheter inte ger någon information om tilldelningen från logiska sektorer till fysiska sektorer, har operativsystemet inga möjligheter att tillhandahålla några mekanismer som garanterar säker användning av en ensam SSD-enhet.

De många fördelar som ett icke-roterande minne med mycket korta accesstider ger kan naturligtvis inte förnekas. Och det finns sätt att komma förbi problemen, även om man måste räkna med ökade arbetsinsatser och kostnader.

Lösning 1:
Använd två separata enheter
Detta är naturligtvis den enkla vägen, och man slipper ytterligare problem. Men den ökar kostnaderna och utrymmesbehoven för den totala lösningen.

Figur 8
Kontron M2M Smart Services Development Kit är en kompakt plattform med hög tillgänglighet avsedd för utveckling av M2M-tillämpningar. Den integrerar ett 4 Gigabyte MicroSD-kort för M2M smart-servicetillämpningar, middleware samt ett operativsystem. Integration av ytterligare en minnelösning rekommenderas för tillämpningar som regelbundet behöver lagra stora mängder information.

Källor:
[1]: http://en.wikipedia.org/wiki/Solid-state_drive
[2]: http://en.wikipedia.org/wiki/Wear_leveling
[3]: c't: Heise, nr 22/2011, sid. 150–151

Dessutom kan en andra enhet inte alltid läggas till i efterhand.

Lösning 2:
Använd speciella enheter som kan ge en fixerad uppdelning av sidorna
Nya typer av flashbaserade SSD-enheter är på gång: dels sådana som kan betraktas som två separata enheter, och dels sådana där man kan "frysa" tilldelningen av sektorer och sidor.

Innan dessa nya enheter finns tillgängliga och har testats tillräckligt är det inte tillrådligt att använda lösningar baserade på en enda SSD-enhet. Istället är det lämpligt att antingen använda vanliga hårddiskar eller att fördela sina uppgifter över två separata lagringsmedia.

Därför erbjuder Kontron möjligheten att använda två självständiga SSD-minnen i sina panel-PC och box-PC. Förutom en flash-SSD för operativsystemet kan man antingen använda ett ickeflyktigt RAM-minne eller en andra flashdisk.

Detta enkla exempel visar att svårigheterna vid utveckling av inbyggda system ofta uppstår på detaljnivå, och att sådant som tycks vara innovativa teknologier inte alltid leder till bättre totallösningar. Så nya komponenter skall väljas med omsorg. Om man är tveksam skall man anlita konstruktörer med stor erfarenhet av inbyggnadshårdvara för planering och utvärdering av sina plattformar.

Man bör tänka på problemet redan i början av ett projekt för att undvika otrevliga överraskningar senare på fältet. Att behöva sätta in åtgärder senare skapar nästan alltid stora problem, eftersom både hårdvara och mjukvara kan påverkas.