SSD-Performance optimieren

Schnelligkeit ist keine Hexerei

Nachdem ein Artikel im letzten ADMIN-Magazin erläutert hat, wie SSDs funktionieren und wann sich deren Einsatz lohnt, geht es diesmal um die Optimierung der SSD-Performance. Werner Fischer

SSDs werden so wie viele Festplatten per SATA-Schnittstelle mit dem Rechner verbunden. Sofern der Rechner halbwegs aktuell ist, erkennt er die SSD beim Hochfahren und kann sie sofort verwenden. Wer die optimale Performance erreichen will, für den lohnt es sich allerdings, vor dem Einsatz ein paar Einstellungen zu überprüfen.

AHCI aktivieren

Der erste Schritt auf dem Weg zu einer optimalen I/O-Performance der SSD besteht darin, den Advanced Host Controller Interface-Modus (AHCI) im BIOS zu aktivieren. Er ermöglicht im Gegensatz zum IDE-Modus Native Command Queuing (NCQ). Damit bekommt die SSD immer gleich mehrere I/O-Anfragen parallel vom Betriebssystem und muss nicht nach jeder einzelnen Abfrage auf die nächste warten. Die Pipeline der SSD bleibt damit voll, der SSD-Controller kann durchgängig I/O-Anfragen abarbeiten, die Performance steigt. Neben NCQ bietet der AHCI-Modus darüber hinaus auch noch das Device Initiated Power Management (DIPM) der SATA-Schnittstelle, das den Stromverbrauch von SSDs im Idle-Betrieb deutlich minimiert. Das erhöht zwar nicht die Performance, steigert aber etwa die Akkulaufzeit bei einem Laptop. Sowohl der Microsoft MSAHCI-Treiber als auch Linux ab Kernel 2.6.24 unterstützen DIPM. Der MSAHCI-Treiber nutzt DIPM standardmäßig nur im Power Saver Modus. Mit dem Windows Tool »powercfg« lässt sich die Nutzung von DIPM auch in anderen Modi aktivieren. Unter Linux kann DIPM über das Sysfs aktiviert werden (Listing 1). Als dritten Vorteil ermöglicht AHCI, Laufwerke im Betrieb per Hot-Plug zu tauschen.

Spare Area vergrößern

Wie bereits im ersten Artikel dieser Serie erwähnt, kann ein SSD Controller die Daten von einer einmalig beschriebenen Page (die aus mehreren Speicherzellen besteht) nicht verändern. Vor einem erneuten Beschreiben dieser Page müsste der Controller den gesamten Block löschen, in dem sich die Page befindet. Um dies zu vermeiden, schreibt der Controller in einem solchen Fall die geänderten Daten einfach in eine andere, bisher unbenutzte Page und aktualisiert seine interne Zuordnungstabelle entsprechend (Abbildung 1).

Sind auf der SSD zu Beginn nur wenige Daten gespeichert, gibt es noch ausreichend freie Pages. Darüber hinaus hat jede SSD eine Spare Area, deren Pages der SSD Controller ebenfalls für diese Zwecke nutzt. Bevor die freien Pages ganz zur Neige gehen, räumt der SSD Controller mit seinem Garbage Collector auf. Er kopiert dabei verstreute belegte Pages aus verschiedenen Blöcken in noch freie Pages. Die dadurch freigeschaufelten Blöcke kann er nun löschen. Damit stehen ihm in der Summe wieder mehr unbenutzte Pages zur Verfügung. Diese Garbage Collection kostet jedoch Zeit und damit Performance. Außerdem erhöht sie durch das interne Kopieren die Anzahl der Schreibvorgänge auf die einzelnen Speicherzellen (die Write Am-

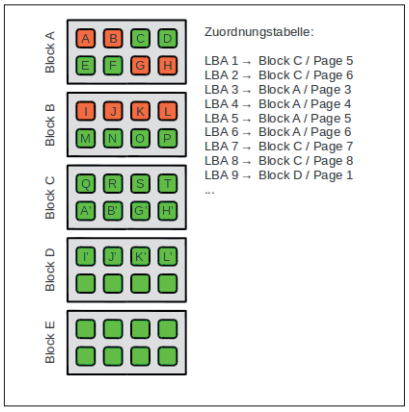

Abbildung 1: Viele Pages (rot markiert) sind ungültig, weil deren Daten verändert und woanders abgespeichert wurden. Der SSD Controller aktualisiert seine interne Zuordnungstabelle entsprechend.

plification steigt). Somit sinkt auch die Lebensdauer dieser Speicherzellen. Ein einfacher Trick verhindert häufige Aufrufe der Garbage Collection: Werden bei der erstmaligen Nutzung der SSD etwa nur 90 Prozent des verfügbaren Datenbereiches partitioniert (Over-Provisioning), erfolgen auf die restlichen zehn Prozent niemals Schreibzugriffe. Die entsprechenden Pages bleiben ungenutzt, der SSD Controller kann diese Pages somit wie Pages aus der Spare Area nutzen. Untersuchungen von Intel zeigen, dass bei einer solchen Einsparung von zehn Prozent der Datenmenge die Random-I/O-Performance auf das Zweieinhalbfache steigt und sich die Lebensdauer der SSD mehr als verdoppelt (Abbildung 2).

Secure Erase

Wie bereits erläutert kann ein SSD Controller die Daten von einer einmalig beschriebenen Page nicht verändern, sondern muss zuvor den gesamten Block mit mehreren Pages löschen. Für die Wiederverwendung einer zuvor bereits benutzten SSD ist es daher sinnvoll, vor dem Einsatz alle Blöcke der SSD zu löschen. Bei den meisten SSDs klappt dies mit einem einfachen Secure Erase [2]. Ein Secure Erase soll laut ATA-Spezifikation das sichere Löschen aller gespeicherten Daten eines Datenträgers garantieren. Bei den meisten SSDs, die Secure Erase unterstützen, führt dies zum physischen Löschen aller Blöcke der SSD. Die SSD ist dann wieder mit der ursprünglichen optimalen Performance nutzbar, da alle Pages direkt beschrieben werden können. Bei einigen neueren SSDs ist das Secure Erase allerdings anders implementiert.

Listing 1: DIPM-Konfiguration unter Linux

```
01 root@ubuntu-10-10:~# hdparm -I /dev/sda | grep
   Device-initiated
02                 Device-initiated interface power
   management
03 root@ubuntu-10-10:~# echo min_power > /sys/class/
   scsi_host/host0/link_power_management_policy
04 root@ubuntu-10-10:~# hdparm -I /dev/sda | grep
   Device-initiated
05            *    Device-initiated interface power
   management
```

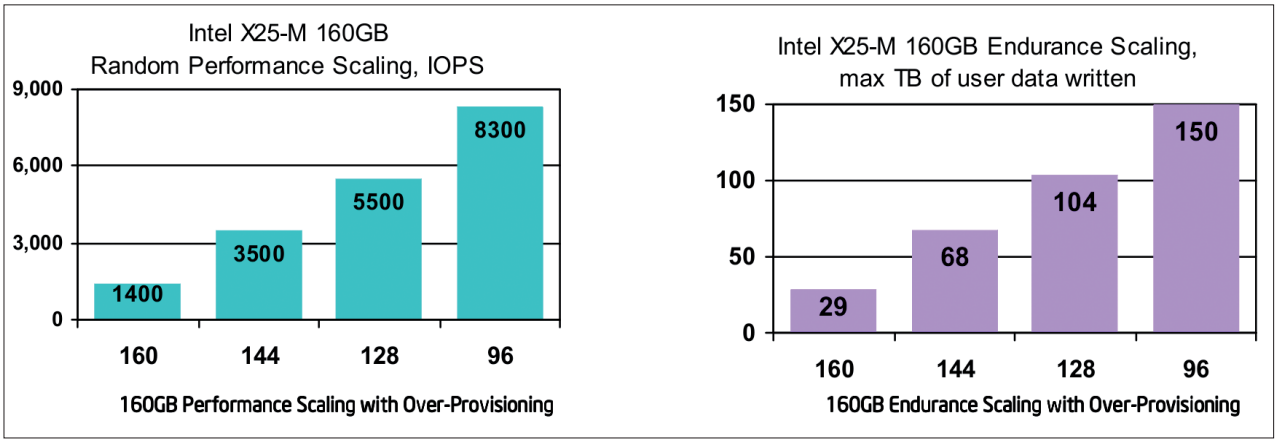

Abbildung 2: Eine etwas vergrößerte Spare Area hat positive Auswirkungen. Nutzt man nur 144GB einer 160GB X25-M SSD, steigen I/O Performance und Lebensdauer.

Diese SSDs verschlüsseln automatisch alle geschriebenen Daten. Bei einem Secure Erase wird dann einfach der Schlüssel sicher gelöscht – die Daten können damit nicht mehr entschlüsselt werden, sind aber noch physisch vorhanden. Bei solchen SSDs werden somit nicht alle Blöcke der SSD gelöscht. In diesem Fall müssen die Blöcke per TRIM gelöscht werden, um für die neue Verwendung der SSD die optimale Performance zu bekommen. Windows 7 führt ein solches TRIM bei der Formatierung automatisch durch. Unter Linux bietet »hdparm« dazu zwar die »--trim-sector-ranges«-Option, die Manpage von hdparm 9.37 rät allerdings nach wie vor von ihrer Verwendung ab.

Partition Alignment

Unter Partition Alignment versteht man das Ausrichten von Partitionen an bestimmten Grenzen eines Datenträgers [3]. Ein korrektes Partition Alignment gewährleistet eine optimale Performance bei Datenzugriffen. Speziell bei SSDs (mit internen Page-Größen von beispielsweise 4.096 oder 8.192 Bytes), Festplatten mit 4 KiB-Sektoren (4.096 Bytes) und RAID-Volumes führt eine fehlerhafte Ausrichtung von Partitionen zu einer verminderten Performance [4].

In der Vergangenheit begann die erste Partition stets auf LBA-Adresse 63 (entspricht dem 64. Sektor), um kompatibel zu DOS und zur alten CHS-Adressierung (Cylinder/Head/Sector) zu bleiben. Die Größe eines solchen (logischen) Sektors beträgt 512 Byte. Bei normalen Festplatten (mit einer physischen Sektorgröße von 512 Byte) bringt das keine Nachteile. Neuere Festplatten mit einer physischen Sektorgröße von 4.096 Byte (4 KiB) emulieren zwar nach außen hin eine Sektorgröße von 512 Byte, arbeiten intern aber mit 4.096 Byte. Und auch SSDs arbeiten mit einer Pagegröße von 4 KiB beziehungsweise 8 KiB. Bei diesen neuen Festplatten und SSDs ist eine solche Partitionierung beginnend bei LBA-Adresse 63 daher sehr problematisch.

Formatiert der Benutzer eine solche Partition mit einem Dateisystem mit einer typischen Blockgröße von 4 KiB, passen die 4-KiB-Dateisystem-Blöcke nicht direkt in die 4 KiB oder 8 KiB großen Pages der SSD (Abbildung 3). Beim Schreiben eines einzelnen 4-KiB-Dateisystem-Blockes müssen dann zwei 4 KiB Pages verändert werden. Erschwerend kommt dabei hinzu, dass die jeweiligen 512-Byte-Sektoren erhalten bleiben müssen – es kommt damit zu einem Read/Modify/Write. Die Folge ist eine bis zu 25fach schlechtere Schreibperformance bei kleinen Dateizugriffen, wie Analysen von IBM zeigen [5].

Um diese Probleme zu vermeiden, empfiehlt sich ein Alignment auf 1 MiB – damit ist man auf lange Sicht auf der sicheren Seite. Mit der aktuellen Adressierung in 512 Byte großen logischen Sektoren

Abbildung 3: DOS-kompatible Partitionen, die bei LBA Adresse 63 beginnen, führen zu erheblichen Performance-Nachteilen bei Datenzugriffen.

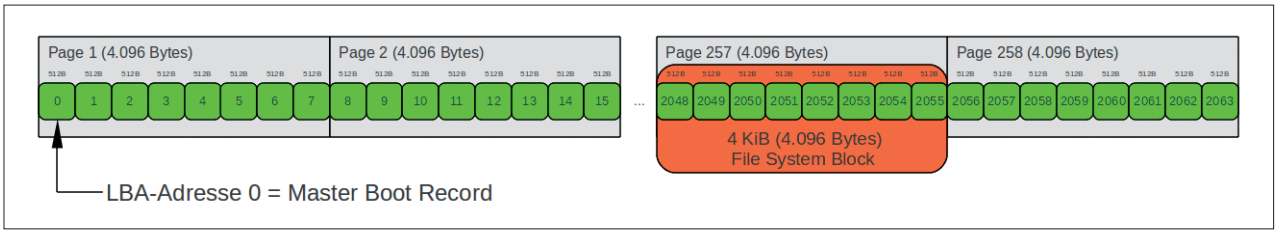

Abbildung 4: Eine korrekt ausgerichtete Partition bringt optimale Performance bei Lese- und Schreiboperationen.

Listing 2: Partition-Alignment mit »fdisk«

```
01 root@ubuntu-10-10:~# fdisk -l -u /dev/sda
02
03 Disk /dev/sda: 160.0 GB, 160041885696 bytes
04 255 heads, 63 sectors/track, 19457 cylinders, total 312581808 sectors
05 Units = sectors of 1 * 512 = 512 bytes
06 Sector size (logical/physical): 512 bytes / 512 bytes
07 I/O size (minimum/optimal): 512 bytes / 512 bytes
08 Disk identifier: 0x9349dd6c
09
10    Device Boot      Start         End      Blocks   Id  System
11 /dev/sda1   *        2048     2457599     1227776    7  HPFS/NTFS
12 Partition 1 does not end on cylinder boundary.
13 /dev/sda2         2457600    61051349    29296875    7  HPFS/NTFS
14 Partition 2 does not end on cylinder boundary.
15 /dev/sda3        61052928    80582655     9764864   83  Linux
16 Partition 3 does not end on cylinder boundary.
17 /dev/sda4        80582656   312580095   115998720   83  Linux
18 Partition 4 does not end on cylinder boundary.
```

entspricht das 2048 Sektoren (Abbildung 4). Neuere Windows-Versionen (Windows Vista, Windows 7, Windows Server 2008) führen bei Partitionen größer 4 GiB ein solches Alignment auf 1 MiB durch. Kleinere Partitionen richten diese Windows-Versionen auf 64 KiB aus. Ältere Versionen (Windows XP, Windows Server 2003) benötigen ein manuelles Alignment. Aktuelle Linux-Distributionen verwenden ebenfalls ein Alignment von 1 MiB bei der Installation. Beim späteren Einrichten von Partitionen mit »fdisk« sind dessen Optionen »c« (Deaktiveren des DOS-Kompatibilitätsmodus) und »u« (verwendet Sektoren statt Zylinder als Einheiten) nötig, um ein korrektes Alignment zu erhalten.

Zur Kontrolle zeigt »fdisk -l -u« die Startsektoren aller Partitionen. Sind diese Sektoren durch 2048 teilbar, sind alle Partitionen korrekt ausgerichtet (Listing 2). Neue LVM-Versionen (ab Version 2.02.73) verwenden dank eines Patches von Mike Snitzer ebenfalls ein Alignment von 1 MiB [6].

ATA TRIM

Eine weitere Funktion zur Steigerung der Performance sowie der Lebensdauer ist ATA TRIM. Mit dem TRIM-Kommando teilt das Betriebssystem der SSD mit, welche Datenbereiche (etwa die Datenbereiche einer gelöschten Datei) es nicht mehr benötigt. Der SSD-Controller kann damit betroffene Blöcke löschen, was ähnlich wie eine vergrößerte Spare Area die Performance und Haltbarkeit der SSD steigern soll. Damit ATA TRIM funktioniert, muss es von der SSD, vom Betriebssystem und vom Dateisystem unterstützt werden. Windows 7 mit NTFS oder Linux ab Kernel 2.6.33 mit Ext4 mit Discard-Option erfüllen diese Anforderungen.

Ab Kernel 2.6.38 unterstützen Ext4 und XFS zudem das zeitversetzte Batched Discard, das bei SSDs mit langsamer TRIM-Funktion wichtig ist. Über den tatsächlichen Performance-Gewinn aufgrund von ATA TRIM gibt es unterschiedliche Aussagen.

Tatsache ist, dass ATA TRIM nur für einzelne SSDs genutzt werden kann. RAID-Controller unterstützen diese Funktion nicht. Eine etwas vergrößerte Spare-Area durch Nutzung von 90 Prozent der SSD-Kapazität für das RAID-Volume sollte in diesem Fall ein Fehlen der TRIM-Funktion ausgleichen. *(jcb)* ■

Infos:

[1] Over-provisioning an Intel SSD : [http://cache-www.intel.com/cd/00/00/45/95/459555_459555.pdf]

[2] Secure Erase: [http://www.thomas-krenn.com/Secure-Erase]

[3] Partition Alignment: [http://www.thomas-krenn.com/Alignment]

[4] Ben Martin, RAID-Systeme unter Linux optimal konfigurieren, ADMIN 02/2011, S. 80

[5] Linux on 4KB-sector disks: [http://www.ibm.com/developerworks/linux/library/l-4kb-sector-disks]

[6] LVM Alignment Patch: [http://www.redhat.com/archives/lvm-devel/2010-August/msg00035.html]

Der Autor

Werner Fischer ist Technology Specialist bei der Thomas-Krenn AG und Chefredakteur des Thomas Krenn Wiki. Seine Schwerpunkte sind die Bereiche Hardware-Monitoring, Virtualisierung, I/O-Performance und Hochverfügbarkeit.