
## Dateisystemwahl für SSDs

# Solides System

Bei einer Festplatte, die mit über 100 MByte/s schreibt und doppelt so schnell liest, denkt man eigentlich kaum ans Optimieren. Doch mit dem richtigen Dateisystem und etwas Tuning liegt noch etwas mehr drin. Marcel Hilzinger

**Linus Torvalds** hat eine klare Meinung zu SSDs: „Wenn der Flash-Hersteller anfängt, über Grenzen im Wear Levelling zu erzählen und wie man die Platte beschreiben soll, dann lauf einfach davon. Geh nicht – renne so schnell du kannst." So brachte der Linux-Übervater 2008 im Real-World-Tech-Forum [1] seine Meinung zu (schlechten) Solid-State-Disks zum Ausdruck.

Die Platten der ersten oder zweiten Generation verkrafteten aufgrund eines relativ schlechten Designs und unzureichender Kapazität nur rund 100 000 Schreibzyklen, was bei einer 8-GByte-Platte und permanenter Belastung eine theoretische kürzeste Lebensdauer von 115 Tagen ergibt. So machte in Linux-Kreisen schnell der Tipp die Runde, auf einer Flash-Festplatte bloß kein Journaling-Filesystem zu benutzen.

## Nicht totzukriegen

Spätestens mit der Markteinführung von Intels X25-Serie (Abbildung 1), von der auch Linus ein Testexemplar erhielt, gehören diese Ratschläge jedoch definitiv der Vergangenheit an: Wer heute eine Solid-State-Disk kauft, muss keinesfalls auf ein aktuelles Dateisystem verzichten – im Gegenteil.

Aktuelle SSDs halten im Normalbetrieb praktisch ein Leben lang. Selbst bei intensivsten Schreibarbeiten beträgt die rechnerische Lebensdauer einer 64 GByte großen SSD rund 51 Jahre, geht man von den aktuell durchschnittlichen 2 Millionen Zyklen und einer Schreibgeschwindigkeit von 80 MByte/s aus [2]. Die Regel, bei SSDs kein Journaling Dateisystem einzusetzen, kann man demnach getrost als alten Hut betrachten: Sie gilt höchstens noch für Eee-PCs der ersten Generation oder sehr günstige Solid-State-Disks.

Seit 2008 hat sich die Schreibtechnik stark verbessert, sodass sich aktuelle Platten selbst darum kümmern, wann sie Da-


**Abbildung 1: Aktuelle SSDs, wie Intels in 34nm-Technik gefertigtes SATA-Model Intel X25-M, halten dank ausgefeilter Wear-Levelling-Algorithmen im Normalbetrieb jahrzehntelang.**

ten wohin schreiben. Optimierungen am Dateisystem laufen deshalb immer Gefahr, dem SSD-eigenen Schreibverfahren („Wear Levelling“) entgegenzuarbeiten oder kommen – wie beim ATA-Trim-Support von Ext4 (siehe unten) – mangels Support durch die Festplattenhersteller gar nicht oder nur schleppend zum Einsatz.

## Gar nicht so schlimm

Der Ext4-Dateisystementwickler Theodore Ts’o untersuchte die Schreibzugriffe bei den Dateisystemen Ext2/3/4 und kam zu dem Schluss, dass das Journaling im Durchschnitt lediglich rund zehn Prozent mehr Schreibzugriffe verursacht [3]. Hinzu kommen die neuen Features von Ext4 und anderen aktuellen Filesystemen, die Dateien erst dann auf die Platte schreiben, wenn es unbedingt nötig ist („Delayed Allocation“). Daher darf man getrost behaupten, dass Ext4 auch für Solid-State-Disks das zurzeit beste und ausgereifteste Dateisystem darstellt (siehe Tabelle „Benchmark-Resultate“).

Übrigens lohnt es sich aus Performance-Gründen bei sämtlichen Partitionen, diese mit den Mount-Optionen »noatime« und »nodiratime« einzuhängen. Das vermeidet unnötige Schreibzugriffe beim Durchstöbern des Dateisystembaums. Einige Distributionen setzen hier generell auf »norelatime«: Damit aktualisiert der Kernel die Access Time nur bei Dateien, die über eine aktuellere Mtime oder Ctime verfügen – also solchen, die sich tatsächlich verändert haben. Der damit verbundene Performance-Gewinn lässt sich auf jeder Hardware messen.

## Nummer Sicher

Wer Angst um die Lebensdauer einer alten SSD hat, kann Ext4 zudem ohne Journaling benutzen. Dazu muss man das Dateisystem lediglich über folgenden Befehl neu anlegen:

```
mke2fs -t ext4 -O ^has_journal /dev/sdXX
```

Dabei ist »*/dev/sdXX*« durch den Namen der passenden Gerätedatei zu ersetzen. Ext4 ohne Journaling kombiniert die Geschwindigkeit von Ext2 mit den erweiterten Fähigkeiten aktueller Dateisysteme. Ohne Journal steht zwar nach einem Absturz ein Dateisystemcheck an, der aber in der Regel nicht sehr lange dauert: Die SSDs der ersten Generation sind maximal 64 GByte groß und verfügen zudem über eine sehr hohe Lesegeschwindigkeit.

Aktuelle Optimierungsversuche von Dateisystementwicklern haben nicht in erster Linie das Ziel, das Leben einer SSD zu verlängern, sondern diese schneller zu machen beziehungsweise Ermüdungserscheinungen vorzubeugen. Letztere treten bei den meisten Solid-State-Platten spätestens dann auf, wenn sämtliche Blöcke einmal belegt waren.

## Trimm-Dich

Die Spezifikation von SSDs sieht als Erfrischungskur den ATA-Trim-Befehl vor, der dem Laufwerk mitteilt, welche unbenutzten Blöcke es zurücksetzen und anschließend komplett neu beschreiben kann. Ted Ts’o hat bereits seit einiger Zeit eine entsprechende Trim-Funktion in das Dateisystem Ext4 eingebaut [4], seit Version 2.6.33 ist der entsprechende Support auch Teil des Linux-Kernels. Ältere Linux-Versionen verfügen – im Gegensatz zu Windows 7 – über keinen Trim-Support. Das Trimmen einer SSD klappt generell nur, wenn diese über eine passende Firmware-Version verfügt. Die bieten zurzeit nur wenige Modelle von Intel und OCZ.

## Alles in Butter

Neben Ext4-Maintainer Ts’o arbeiten auch die Entwickler des Dateisystems Btrfs an einem speziellen SSD-Modus [5]. Die Mount-Option »ssd« existiert bereits seit Längerem. Sie bewirkt, dass das Dateisystem nach Möglichkeit auf unbenutzte Bereiche schreibt. Seit Kernel 2.6.31 benutzt »mount« automatisch die passende Option, sobald das Btrfs eine SSD erkennt.

Neben dieser Quasi-Standard-Option gibt es noch »-o ssd_spread« und »-o discard«. Die Option »ssd_spread« arbeitet laut Dokumentation [6] auf günstigerer SSD-Hardware schneller, da sie unbedingt versucht, einen freien Bereich zu finden. Per »discard« lässt sich Btrfs anweisen, nicht mehr benutzte Blöcke zum Trimmen freizugeben. Da sich dieser Vorgang

**Tabelle 1: Benchmark-Resultate**

| | Ext2 | Ext4 | Ext4 ohne Journal | Btrfs | Btrfs »-o ssd_spread« |
|---|---|---|---|---|---|
| »dbench -D /test 10« | 520 MByte/s | 407 MByte/s | 428 MByte/s | 347 MByte/s | 347 MByte/s |
| »bonnie++ -d /test -s 2048« | 38 MByte/s | 58 MByte/s | 72 MByte/s | 64 MByte/s | 67 MByte/s |

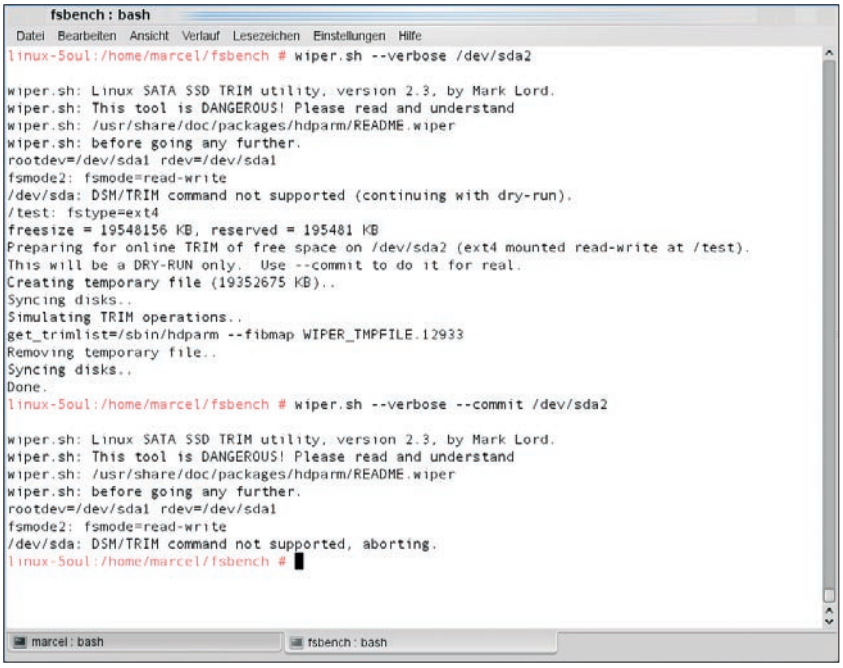


**Abbildung 2: Bei SSDs, die über keinen passenden Trim-Support verfügen, verweigert das Wiper-Skript seine Mitarbeit.**

aber bei vielen SSD-Platten unter Umständen auch negativ auswirken kann, ist die Option in der Grundeinstellung ausgeschaltet.

## Mit der Brechstange

Bis der Kernel und die Dateisysteme über einen umfassenden Support für SSDs verfügen, werden somit noch ein paar Monate verstreichen. Der Hdparm-Entwickler Mark Lord hat deshalb die aktuellen Versionen seines Festplattentools um das Skript »wiper.sh« ergänzt [7]. Es sucht im Dateisystem nach freien (unbenutzten) Blöcken und meldet diese der Firmware der SSD. So weiß die Platte, dass sie diese Blöcke für das Wear Levelling oder eine allgemeine Speicherbereinigung nutzen kann. Dieses Feature gehört seit Version 9.27 zu Hdparm (veröffentlicht im Oktober 2009), die meisten Distributionen bringen dafür bereits entsprechende Pakete mit. Allerdings funktioniert »wiper.sh« nur bei den teureren SSD-Modellen wie der Vortex-Serie von OCZ und der X25 von Intel, unser Testgerät verweigerte die Zusammenarbeit (Abbildung 2).

Das Wiper-Skript lässt sich bei Ext4 und XFS im normalen Betrieb nutzen, für Ext2/3 und ReiserFS muss die entsprechende Partition read-only eingehängt sein. Die Readme-Datei weist darauf hin, dass man »wiper.sh« am besten bei einem nicht eingehängten Datenträger benutzt. Zudem gilt das Feature noch als experimentell – es empfiehlt sich also ein Backup sämtlicher Daten auf eine zweite Festplatte. Mit Disktrim (Abbildung 3) steht Ubuntu-Nutzern zudem ein fertiges Debian-Paket für eine grafische Oberfläche zu »wiper.sh« zur Verfügung [8].

Auf Grund einiger Besonderheiten von Btrfs eignet sich das Wiper-Skript nicht für dieses Dateisystem. Die Btrfs-Entwickler setzen hier auf die oben geschilderten Mount-Optionen zur Optimierung von Schreibzugriffen auf SSDs. Neben Optimierungen an bestehenden Dateisystemen gibt es auch Ansätze für komplett

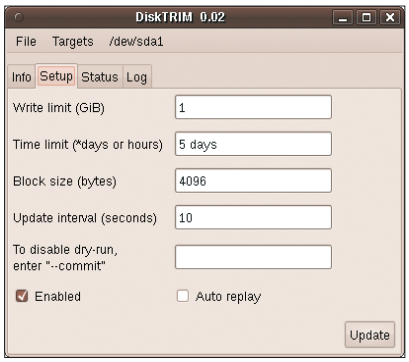


**Abbildung 3: Trotz grafischer Oberfläche handelt es sich bei Disktrim nicht um ein für Anfänger geeignetes Tool.**

neue, auf Flash-Speicher hin optimierte Dateisysteme, zum Beispiel Nilfs2 [9] oder LogFS [10]. Diese reichen aber in Sachen Performance nicht an Ext4 oder Btrfs heran.

## Fazit

Wer eine aktuelle SSD von einem namhaften Hersteller wie Kingston/Intel, OCZ oder Samsung kauft, der braucht sich um die Lebensdauer der Platte nicht groß zu kümmern. Der Straßenpreis für ein 64-GByte-Laufwerk liegt inzwischen bei den meisten Herstellern unter 160 Euro. Auch die in den Tests über mehrere Tage intensiv benutzte 32 GByte große 2,5-Zoll-Warp-Platte von Patriot Memory (Straßenpreis 100 Euro) zeigte keinerlei Ermüdungserscheinungen. Allerdings treten solche bei sehr I/O-intensiven Anwendungen früher oder später auf. Hier schafft unter Linux zurzeit einzig das zu Hdparm gehörende Skript »wiper.sh« Abhilfe. Die Kernel- und Dateisystementwickler arbeiten aber mit Hochdruck an der Integration entsprechender Tools, sodass in naher Zukunft sämtliche Probleme beim SSD-Support von Linux gelöst sein dürften. *(mhi/ofr)* ■

### Infos


[1] Linus über SSDs: [http://www.realworldtech.com/forums/index.cfm?action=detail&id=93409&threadid=92678&roomid=2]

[2] Mythen über SSDs: [http://www.storagesearch.com/ssdmyths-endurance.html]

[3] Schreibzugriffe durch Journaling: [http://thunk.org/tytso/blog/?p=328]

[4] Trim-Support in Ext4: [http://www.linux-mag.com/id/7272]

[5] Discard-Funktion von Btrfs: [http://btrfs.wiki.kernel.org/index.php/Changelog#v2.6.32_.28December_2009.29]

[6] SSD-Optimierungen von Btrfs: [http://btrfs.wiki.kernel.org/index.php/FAQ#Is_Btrfs_optimized_for_SSD.3F]

[7] Hdparm: [http://sourceforge.net/projects/hdparm/]

[8] Disktrim: [https://sourceforge.net/projects/disktrim/]

[9] Nilfs2: [http://www.nilfs.org/en/]

[10] LogFS: [http://logfs.org/logfs/]