

Der Logical Volume Manager (LVM) - Teil 1

[Michael Hasenstein](#)

Dieser Text beschreibt den LVM in SuSE Linux. Er kann in unveränderter Form unbegrenzt vervielfältigt werden. Das Original liegt als PDF-Datei unter <http://www.suse.com/us/support/oracle/>.

Seit SuSE Linux 6.3 enthält SuSE Linux einen Logical Volume Manager. Der LVM in SuSE Linux ist Heinz Mauelshagens Implementierung, die Homepage ist <http://www.sistina.com/lvm/>.

SuSE Inc., 2001

Michael Hasenstein <mha@suse.com>

Inhalt

- [Logical Volume Management - Einführung](#)
- [Wie es funktioniert](#)
- [Was LVM kann](#)
- [Wie es wirklich funktioniert - Physical Extents \(PE\), Logical Extents \(LE\)](#)
- [Unterschiede zwischen LVM und RAID Systemen](#)
- [Durch LVM verursachter Aufwand](#)

Logical Volume Management - Einführung

Volume Management generiert über dem Speicher eine abstrakte Ebene. Applikationen benutzen virtuellen Speicher, der durch eine Volume Management Software, einen *Logical Volume Manager (LVM)*, verwaltet wird. Dieser LVM versteckt die Einzelheiten vor dem System, z.B. wo Daten gespeichert werden, auf welcher Hardware und wo auf dieser Hardware. Mit Hilfe von Volume Management können Sie die Speicherkonfiguration editieren, ohne irgend etwas auf der Hardwareseite zu verändern und umgekehrt. Durch das Verbergen von Hardwaredetails ist das Hardware-Speichermanagement vollständig vom Software Speichermanagement getrennt. Dadurch ist es möglich, die Hardwareseite zu verändern, ohne daß dies die Softwareseite je bemerkt..

Mit LVM kann die Verfügbarkeit eines Systems deutlich verbessert werden, denn keine Applikation braucht vor Änderungen in der Speicherkonfiguration deaktiviert zu werden.

Wie es funktioniert

Kein LVM:

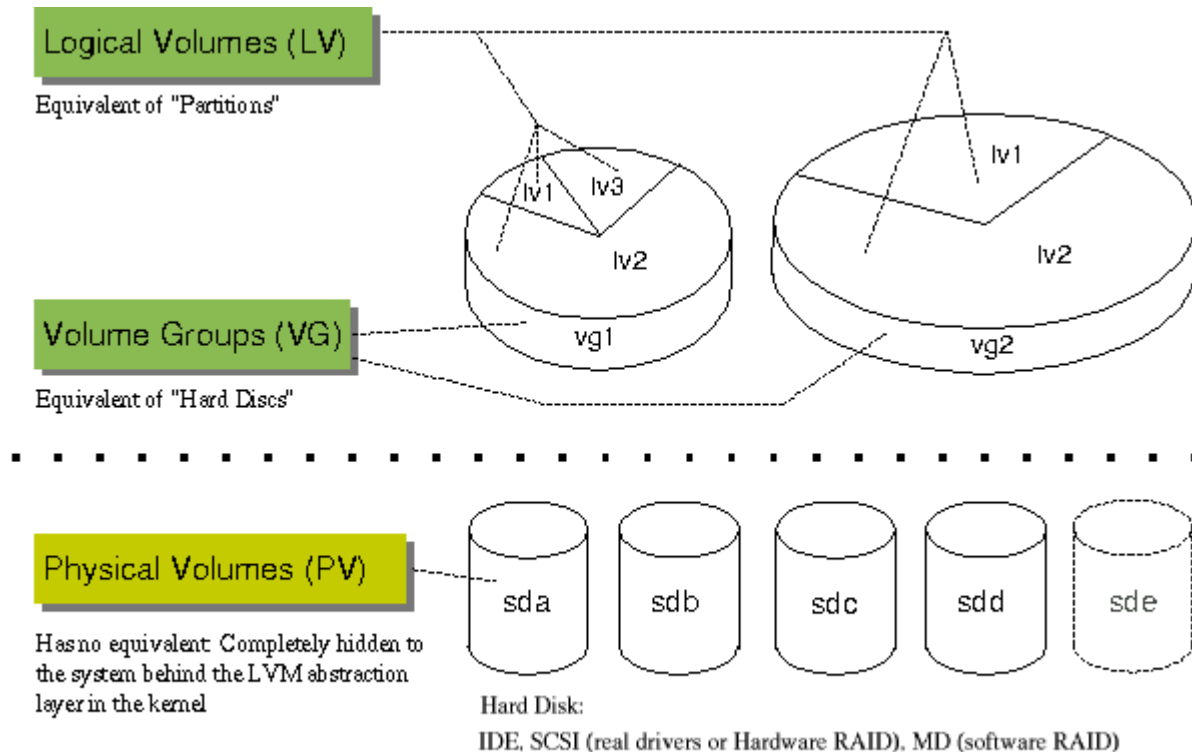
Ohne einen LVM gibt es physikalische Laufwerke mit Partitionen. Auf diesen Partitionen gibt es entweder Dateisysteme, oder aber sie werden *raw* benutzt, (d.h. nicht durch den Kernel verwaltet, sondern durch direkten low-level Zugriff von der Applikation), so z.B. von Datenbanken wie Oracle.

Mit LVM:

Ein *Volume Manager* verlegt die physikalischen Objekte (*Physical Volumes*) in Speicherpools, die als *Volume Groups* bezeichnet werden. Der LVM in SuSE Linux kann ganze SCSI- oder IDE-Laufwerke und -Partitionen innerhalb einer Volume-Group, wie auch Hardware- und sogar Software-RAID-Geräte verwalten.

Die Volume Group ist aus der Sicht des Systems die Entsprechung eines physikalischen Laufwerks. Die Entsprechung der Partitionen, in die dieser Speicherplatz eingeteilt ist, um die Schaffung verschiedener Dateisysteme und rohen Partitionen zu ermöglichen, wird als *Logical Volume* bezeichnet.

Was ein Logical Volume Manager tut:



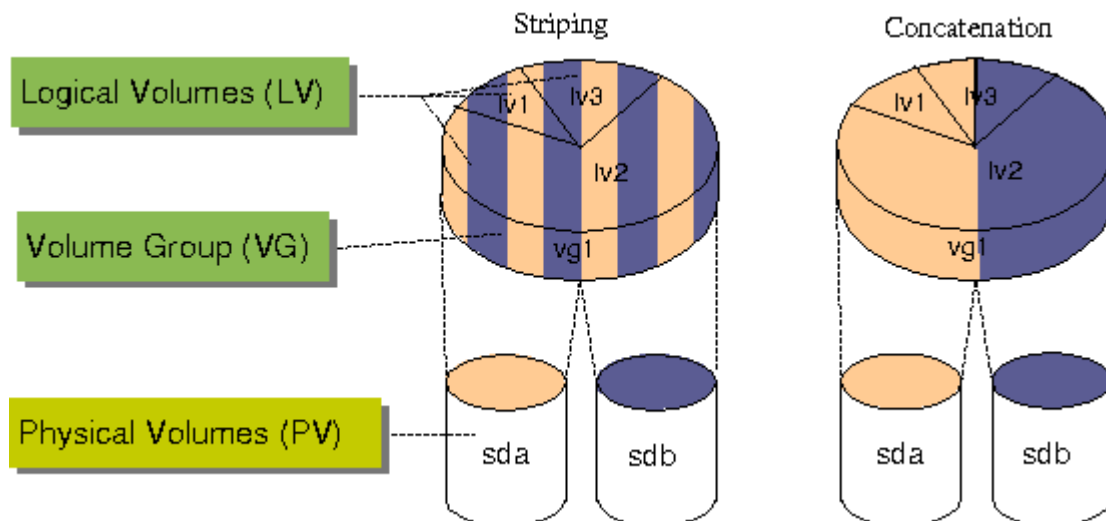
Was LVM kann

- **HA-Systeme:** Die Verwaltung einer Volume Group und mehrerer Logical Volumes kann geschehen, während diese vom System benutzt werden. So ist es zum Beispiel möglich, in Home-Verzeichnissen auf Logical Volumes zu lesen und zu schreiben, während das gesamte Laufwerk-Subsystem, auf dem die Bits und Bytes gespeichert sind, ausgetauscht wird. Wir nennen dieses extreme Beispiel für Flexibilität, da es auch mit "einfachem" Hardware- oder Software-RAID möglich ist, einzelne defekte Festplatten auszutauschen. Außerdem betont es den Vorteil der völligen Trennung des tatsächlichen physikalischen Speichers von dem durch die Applikationen genutzten logischen Speicher.
- **Logische Speicherung verschieben:** Ein Beispiel wurde bereits oben genannt. Es ist möglich, den logischen Speicher an einen anderen physikalischen Ort zu verschieben. Dies kann auch ohne LVM getan werden; was jedoch LVM so besonders macht, ist daß dies im laufenden Betrieb geschehen kann, während die Speicherung benutzt wird.
- **Snapshots:** Es ist möglich (seit SuSE Linux 7.0 mit dem Kernel- und LVM-Update), ein *Snapshot Device* zu erstellen, welches für ein existierendes Logical Volume ein Alias ist, jedoch nur Lesezugriff ermöglicht und ein "eingefrorenes" Abbild des LV ist. Das heißt, während die

Applikationen die Daten auf dem LV ändern, enthält dieses Logical Device das unveränderte Abbild des LV zum Zeitpunkt der Erstellung des Snapshots. Diese ermöglicht einen Sicherungslauf, ohne irgend etwas abschalten zu müssen und ohne besondere Software zu erfordern. Diese Methode ist softwareunabhängig, da sie in der LVM-Abstraktionsschicht geschieht.

- **Größe ändern:** Es ist möglich, schon existierenden Logical Volumes Speicherplatz hinzuzufügen und ihre Größe im laufenden Betrieb zu verändern. Wenn das Dateisystem auf dem Logical Volume die entsprechende Unterstützung bietet (wie zum Beispiel ReiserFS auf SuSE Linux), kann die Größe im laufenden Betrieb angepaßt werden und der zusätzliche Platz sofort genutzt werden, ohne daß ein Unmounten notwendig wäre, so daß Anwendungen und der User den Vorgang nicht einmal bemerken werden.
- **Verkettung (Concatenation) & Striping:** Es ist möglich, Festplatten und/oder Partitionen auf unterschiedlichen Speicherungssystemen zu einer Volume Group (ein logisches Laufwerk) zu gruppieren. Beispiel: Es gibt im System zwei IDE-Laufwerke mit je 30 GB, /dev/hdc und /dev/hdd, aber sie sollen unter einem Mountpoint als ein fortlaufender Speicherblock genutzt werden. Beide können in einer Volume Group zu einem großen logischen Laufwerk zusammengefaßt werden. Physikalischer Speicher kann verkettet werden (Concatenation), d.h. das System schreibt auf das nächste Laufwerk, sobald eines voll ist, oder er kann abschnittsweise beschrieben werden (Striping), indem ein Datenabschnitt auf einem Laufwerk gespeichert wird, der nächste auf einem anderen usw. Dies geschieht abwechselnd, so daß der Datenfluß beschleunigt wird, indem jedes Laufwerk kleinere Aufträge bekommt. Dies kann auch mit Hardware- oder Software-RAID realisiert werden. Nur ist es bei LVM egal, wo die Laufwerke sich befinden. Das gleiche Prinzip läßt sich auch mit zwei großen externen Speichersystemen verwirklichen, nicht nur mit zwei Laufwerken.

Beispiel: Verkettung & Striping von zwei Laufwerken zu einem logischen Laufwerk:



- **Shared Storage Cluster Option:** Der von SuSE benutzte LVM kann auch in gemeinsamen Speicherclustern genutzt werden. Die Konfiguration ist auf dem eigentlichen physikalischen Speicher gespeichert. Beim Booten durchforstet SuSE Linux alle dem System bekannten Physical Volumes (alle Laufwerke und Partitionen auf allen SCSI- und IDE-Kanälen und allen Software-RAID-Laufwerken). Wenn es Platz findet, der durch LVM beansprucht wird, liest es die dort gespeicherte Konfiguration aus und initialisiert den entsprechenden lokalen Knoten einschließlich aller Device-Dateien. Daher braucht LVM auf einem gemeinsamen Speichercluster nur einmal auf einem Knoten aufgesetzt zu werden; alle anderen Knoten, die den Speicherplatz erkennen können, werden beim Booten automatisch konfiguriert. Natürlich kann dies auch bewerkstelligt werden, ohne die anderen Knoten neu zu starten. Allerdings ist diese Version von LVM noch nicht völlig Cluster-bewußt, d.h. es verteilt die Setup-Information nicht automatisch an die Knoten. Aus diesem Grunde erfordern manche Arbeitsschritte die Deaktivierung von LVM auf allen Knoten bis auf einen. Diese Knoten werden nach der

Modifizierung der Konfiguration neu gestartet. Wenn beispielsweise der logische Speicher an einen anderen physikalischen Ort verlegt wird, merkt dies nur der Knoten, der diese Operation durchführt. Alle anderen sind über diesen Arbeitsschritt nicht informiert und benutzen weiterhin die alte Position. Dieses Beispiel verdeutlicht einen Konfigurationsschritt, der nicht durchgeführt werden kann, solange alle Knoten auf den Speicher zugreifen.

- **Unabhängigkeit vom Standort einer Festplatte:** Da die LVM-Konfiguration auf den vom LVM verwalteten Physical Volumes gespeichert wird, ist das System unabhängig von Devicenamen. Beispiel: Die Laufwerksbezeichnungen in Linux für SCSI-Geräte heißen `/dev/sda` für die erste Festplatte, `/dev/sdb` für die zweite usw. Die Reihenfolge wird durch die Reihenfolge der Festplattenerkennung beim Booten bestimmt. Wenn eine andere Festplatte mit SCSI-ID mitten in die Kette der bereits existierenden Laufwerke eingefügt wird, wird sie beim Booten vor den anderen Laufwerken erkannt und es wird ihr ein Devicename zugewiesen, der bisher von einem anderen Laufwerk benutzt wurde. Da die LVM-Konfiguration auf den Laufwerken gespeichert wird und die Konfiguration des laufenden LVM während dem Booten neu aufgebaut wird (bzw. wenn im laufenden Betrieb bestimmte LVM-Befehle benutzt werden), wird LVM in einer solchen Situation problemlos funktionieren. Es verwendet zur Identifizierung der vom LVM verwalteten Laufwerke nämlich nicht diese variablen Devicenamen, sondern die Information, die auf den Laufwerken selbst gespeichert ist.

Wie es wirklich funktioniert - Physical Extents (PE), Logical Extents (LE)

Jetzt, da wir wissen, was LVM kann, können wir die interne Funktionsweise von LVM betrachten. Angenommen, die Konfiguration ist bereits geschehen. Wir betrachten eine feste LVM-Konfiguration, da wir nur daran interessiert sind, wie es jetzt funktioniert und nicht an irgendwelchen administrativen Themen.

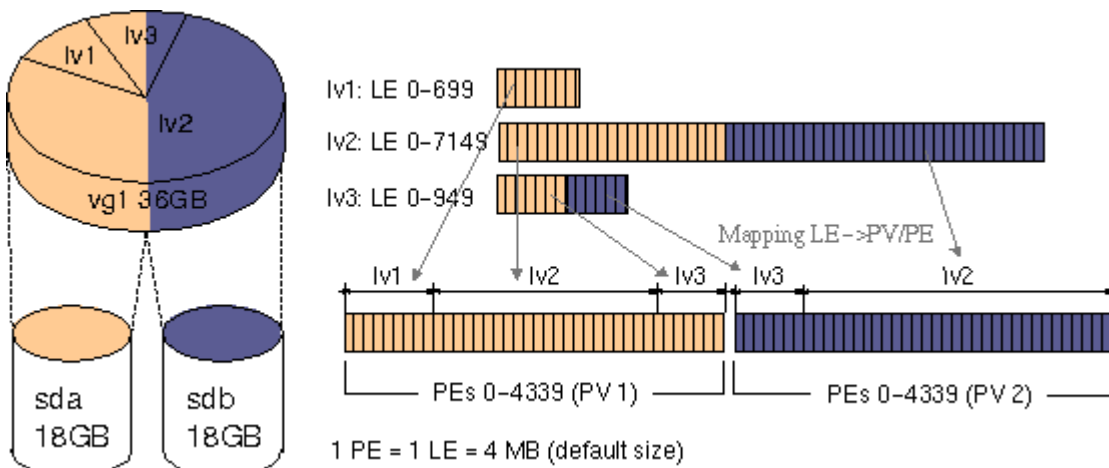
Jedes Physical Volume (PV) ist in PEs von gleicher Größe unterteilt. Die Größe eines PE ist variabel, jedoch für alle PEs in einer Volume Group (VG) gleich. Innerhalb eines PV ist die ID-Nummer jedes PE eindeutig. Jedes PV verfügt über PEs, die von 0 bis $(\text{Größe-des-PV} / \text{Größe-eines-PE} = \text{Gesamtanzahl-der-PEs})$ durchnummeriert sind. Ein PE ist die kleinste Einheit, die vom LVM auf der physikalischen Speicherung adressiert werden kann.

Der VGDA, ein Bereich in jedem PV, der für die Speicherung der Verwaltungsinformationen oder Metadaten (vgl. Anhang A) benutzt wird, enthält die Informationen darüber, welcher Volume Group die PV angehört, wie groß ein PE ist, und - was für diesen Abschnitt am wichtigsten ist - welche PEs dieses PVs welchem LV angehören und was die sequentielle Nummer dieses PV in der VG ist. Wichtig: In einer VG, die aus bis zu 255 PVs besteht, kann es ein bis 255 LVs geben, die auf einem beliebigen PV der VG physikalisch gespeichert sind.

Beim Systemstart und wenn die VGs und LVs aktiviert werden, wird diese VGDA-Information in das System eingelesen. Hier wird sie benutzt, um die Mapping-Information zu erstellen. Diese wird gebraucht, um festzulegen, wo die Teile der LVs physikalisch gespeichert werden.

Jedes LV ist in LEs aufgeteilt. LEs haben die gleiche Größe wie die PEs des VG, in dem sich das LV befindet. Jedes LV hat LEs, die von 0 bis $(\text{Größe-des-LV} / \text{Größe-eines-LE} = \text{Gesamtzahl-der-LEs})$ durchnummeriert sind, was der obengenannten Formel für die PEs entspricht. Jedes LE wird genau einem PE auf einem PV zugewiesen, d.h. es findet ein 1:1 Mapping von LE:(PV:PE) statt. Ein PE-Index kann nicht global eindeutig sein, er gilt nur jeweils pro PV. Aus diesem Grund mußte man eine logische Entsprechung zur Kombination PV/PE schaffen.

Beispiel: Festplatte sda wird als PV 1 konfiguriert, sdb als PV 2.



Wenn eine Anwendung nun den Speicher auf einem Logical Volume ansprechen möchte, wird das LE, in dem sich dieser Speicherplatz befindet, identifiziert und sowohl das PV als auch das PE werden anhand der eindeutigen ID-Nummer des LE in der Mapping-Tabelle lokalisiert. Jetzt kann die Ein-/Ausgabe beginnen, da die physikalische Position des Speicherplatzes auf dem LV, auf den zugegriffen wird, identifiziert wurde.

Anhand der obigen Grafiken soll nun ein Beispiel erläutert werden: Eine Anwendung greift auf das LV lv3, Byte 254.123 zu; dementsprechend ist das LE #62 ($62 \cdot 4\text{MB} = 253.952$, #63 wäre das nächste LE). Laut Mapping-Tabelle sind die LEs 0 ~500 des LV lv3 auf dem PV1 gespeichert. Dabei setzt sich die Nummer des PEs aus der Anzahl der PEs von lv1 auf PV1 und der Anzahl der PEs von lv2 auf PV1 plus 62 zusammen.

Der Unterschied zwischen LVM und RAID-Systemen

Logical Volume Management ist kein Ersatz für RAID. Genausowenig wie RAID ein Ersatz für eine Backup-Strategie ist!

Hardware-RAID arbeitet auf einer sehr niedrigen Device-Ebene und ist von einem bestimmten Controller und demzufolge von einem bestimmten Speichersubsystem abhängig. Es soll gegen den Ausfall einer oder mehrerer (abhängig von der konkreten RAID-Konfiguration) Festplatten innerhalb eines Speicherungssystems schützen und den Datendurchsatz erhöhen, was wiederum von der RAID-Konfiguration abhängig ist.

Software-RAID wird eingesetzt, wenn kein Hardware-RAID-Controller verfügbar ist und stellt eine Emulation auf Softwareebene dar.

Das Logical Volume Management arbeitet unabhängig von irgendeinem bestimmten Speicherungssystem. Es kann auch gewisse RAID-Funktionen wie Striping bereitstellen (das LVM in SuSE Linux implementiert bislang nur Striping, aber Sie können MD benutzen, um mit LVM Software-Mirroring zu bewerkstelligen), wie es bei Software-RAID der Fall ist. Aber das ist nur ein (angenehmer) Nebeneffekt.

Wir haben bereits auf viele der Unterschiede hingewiesen. Zum Beispiel ist LVM von den Devicenamen unabhängig, da die LVMS eigene IDs auf jeder Festplatte speichern. Das Snapshot-Device ist ein anderes Beispiel für einen speziellen Dienst, den die Abstraktionsebene des LVM bereitstellen kann. Verkettung und Striping über mehrere Speichersubsysteme hinweg - Hardware-RAID funktioniert nur mit den Laufwerken, die direkt an einem RAID-Controller angeschlossen sind. Oder die HA-Option: **Die meisten Volume-Management-Vorgänge sind im laufenden Betrieb durchführbar, während der logische Speicher durch eine Anwendung in Anspruch genommen wird.** Des Weiteren die Fähigkeit, Dinge zu tun, die über die Möglichkeiten eines bestimmten Hardware-Controllers hinausgehen, selbst wenn ein kompliziertes RAID-System eingesetzt würde: Zum Beispiel könnte ein Oracle Datenbankserver mehr Platz für einen seiner Tablespace benötigen. Normalerweise würde man einfach ein weiteres Laufwerk an das RAID-Array anschließen und zu

diesem Zweck die proprietäre Software des Array-Herstellers einsetzen. Was jedoch, wenn dieses Array bereits voll ist? Mit LVM ist es möglich, irgendwo im System eine Festplatte hinzuzufügen, d.h. sie könnte an den freien Slot eines anderen RAID-Controllers angeschlossen und in die Volume Group, auf der sich der Tablespace befindet, eingefügt werden. Danach könnte das durch den Tablespace benutzte Logical Volume vergrößert werden, um den neuen Speicherplatz in der VG verfügbar zu machen. Daraufhin kann Oracle diesen Tablespace erweitern.

Teure RAID-Arrays werden mit komplizierter Software ausgeliefert, die auf die Speicherhardware, also auf das RAID-Array beschränkt ist und über keine weiteren Fähigkeiten verfügt. Der Logical Volume Manager hingegen, ist von proprietären Speichersystemen unabhängig und behandelt alles als "Speicher".

Beispiel: Sie haben einen Server mit einem externen RAID-Array (z.B. mit Fibre Channel angeschlossen). Sie brauchen mehr Platz, aber alle Festplatten sind voll. Wenn Sie LVM benutzen, können Sie einfach ein weiteres Speichersubsystem an das FC anschließen und LVM **im laufenden Betrieb** umkonfigurieren, um den Platz den Volume Groups hinzuzufügen und die Logical Volumes zu erweitern. Wenn Sie das derzeitige Speichersubsystem durch ein neues, größeres komplett ersetzen möchten, können Sie dies tun und - wiederum im laufenden Betrieb - den gesamten Speicher (alle Logical Volumes und Volume Groups) von einem Array zum nächsten verschieben. Nicht zu vergessen ist die bequeme Verwaltung: Die Speicherverwaltung kann im laufenden Betrieb geschehen. Der LVM und die Verwaltung ist auf allen SuSE Linux Plattformen gleich: i386, IA64, AMD x86-64, S/390, PowerPC, Sparc[64], AXP.

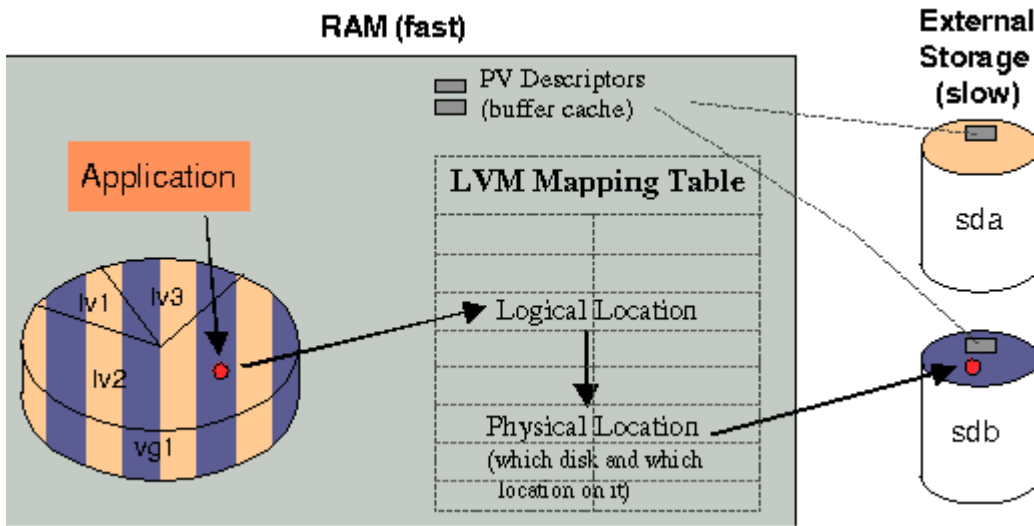
Festplattenausfälle und LVM

LVM bietet keinen Schutz gegen Datenverlust, sollte eine Festplatte ausfallen. In solchen Fällen können Sie entweder (Hard- oder Software-RAID) Level 1 (Mirroring) oder 5 verwenden. Aber weder LVM noch RAID können eine vernünftige Backupstrategie ersetzen. Es gibt RAID-Level, die gegen den Ausfall einer oder mehrerer Festplatten absichern. Von Bedeutung sind hierbei heutzutage RAID 0, 1 und 5, wobei die Kombination von Level 1 und 0 auch als RAID 10 Anwendung findet. LVM ist eine von mehreren Optionen, die HA gewährleisten können. Sie können Speicher im laufenden Betrieb austauschen, ohne daß dies die zugreifende Applikation bemerkt - aber vor Plattenausfall schützt auch das nicht. LVM arbeitet eben auf Systemebene und nicht auf Hardwareebene (siehe den Abschnitt [Wie es funktioniert](#)).

Durch LVM verursachter Aufwand

Die LVM-Ebene im Kernel pflegt ein Mapping der logischen Speicherpositionen zu den realen (physikalischen) Speicherpositionen. Dies bedeutet, daß LVM bei jedem Lese- oder Schreibzugriff zur Speicherung die Position der angeforderten Information auf den Festplatten mittels einer Tabelle übersetzen muß. Obwohl dies im Sinne des Prozessors "Kosten" verursacht, sind die Vorgänge "billig" (d.h. sehr schnell ausgeführt), da das Nachschauen in einer Mapping-Tabelle eine der schnellsten und grundlegendsten Arbeitsschritte ist, die von Computern durchgeführt werden.

Jedes LVM-PV enthält einen kleinen Bereich mit Informationen über das PV (z.B. die ID dieses Volumes, da LVM es mittels dieser ID identifiziert, **nicht** anhand seiner Bezeichnung im System, da die Bezeichnung sich ändern kann, z.B. wenn sich die SCSI-ID ändert!), über die Volume Group, der das PV angehört, und über die LVs, die darauf gespeichert sind. Da dieser Bereich ziemlich klein ist (auf einer VG von 8,5 GB mit 5 PVs werden ungefähr 128k auf jedem PV benutzt) und oft benutzt wird - für jeden einzelnen Lese-/Schreibzugriff - wird er im Buffercache des Linux-Kernels gespeichert, d.h. im RAM. Die Teile davon, die von dem Codeabschnitt benutzt werden, die der Prozessor ausführt, werden wahrscheinlich sogar im 1st-Level Cache, d.h. im Prozessor selbst gespeichert. Daher ist die Zeit, die für das Nachschlagen in der Mapping-Tabelle benötigt wird, auf den heutigen CPUs vernachlässigbar im Vergleich zu der langen Zeit, die für den Zugriff auf den Festplattenspeicher benötigt wird.



Man sollte auch nicht vergessen, daß LVM-Daten nur geschrieben werden, wenn die LVM-Konfiguration geändert wird. Im laufenden Betrieb sind alle Vorgänge im Zusammenhang mit der LVM-Mapping-Tabelle Read-Only. Dies ist für das Caching sehr gut, besonders auf Multi-Prozessor-Systemen, in denen die 1st-Level-Caches auf allen CPUs Teile der Mapping-Tabelle beinhalten.

Ein Benchmark-Test

Hört sich schön an, aber wie sieht's denn nun in Wirklichkeit aus?

Hier sind einige Zahlen von einem (nicht-wissenschaftlichen) Benchmark, der Ein-/Ausgabe auf "rohen" Festplatten mit Ein-/Ausgabe (I/O) über LVM und nur einer Festplatte vergleicht, um den LVM-Aufwand direkt zu messen, und LVM und 4 Festplatten in einer Striping-Konfiguration.

Wir benutzten den *Bonnie*-Benchmark, der unter diesem Namen auch in der SuSE Linux Distribution enthalten ist. Für nähere Einzelheiten über diesen Benchmark ziehen Sie bitte die Dokumentation für dieses Paket heran. Der Rechner, der zum Einsatz kam, war ein Compaq ProLiant 1850 mit 6 Festplatten. Der Smart-Array RAID-Controller in diesem Server wurde so konfiguriert, daß er keinerlei Einfluß hat, d.h. wir haben direkten Zugriff zu den Festplatten. Der Rechner hat 512 MB RAM. Da die Dateigröße in diesem Benchmark 1024MB ist, unterdrücken wir das Caching, um die "echte" I/O-Leistung messen zu können. Das Filesystem ist ext2 mit 4k Blockgröße in Tabelle 1 und reiserfs in Tabelle 2. Dies schafft die zusätzliche Möglichkeit, ext2 und reiserfs zu vergleichen. Aber lassen Sie sich durch diesen Aspekt nicht verwirren: Betrachten Sie jeweils nur eine der Tabellen, wenn Sie die Ergebnisse für LVM überprüfen.

Zunächst werden die vier Festplatten ohne LVM getestet. Diese Zahlen können später zum Vergleich herangezogen werden.

Der nächste Test schließt LVM mit ein. Eine VG wird erstellt, die nur einen PV beinhaltet, und das ist die Festplatte HD 1. Wir können die Ergebniszahlen direkt mit denen vergleichen, die wir beim Test von HD 1 ohne LVM erhielten, um so den LVM-Aufwand zu erhalten. Wie versprochen, gibt es keinen merklichen Aufwand. Die Daten, die wir erzielen, sehen nicht anders aus als die, die wir beim Testen der Festplatten ohne LVM erhielten.

Der letzte Test erstellt eine VG, die alle vier Festplatten beinhaltet, die in diesem Test enthalten sind. Auf dieser VG wird ein LV erstellt, bei dem über alle vier PVs (Festplatten) hinweg Striping angewandt wird, um uns die Möglichkeit zu geben, mögliche Leistungsgewinne oder -Einbußen durch die Benutzung von LVM-Striping (Software-RAID0) zu messen.

Tabelle 1: Filesystem ist ext2

---Sequential Output (nosync)--- ---Sequential Input--- --Rnd Seek--

-Per Char-		--Block---		-Rewrite--		-Per Char-		--Block---		--04k (03)-	
K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	/sec	%CPU

HD 1 (kein LVM)											
8500	98.9	20073	17.2	9070	15.7	7970	88.4	22082	16.2	251.2	2.6
HD 2 (kein LVM)											
8358	97.1	20927	18.0	9129	16.2	7968	88.6	22096	15.7	258.9	3.6
HD 3 (kein LVM)											
8309	96.5	20525	17.8	9079	15.8	7729	85.9	22091	15.7	249.9	3.4
HD 4 (kein LVM)											
8343	97.0	21295	18.4	9065	16.3	7938	88.2	22017	14.9	258.7	3.2
LVM - nur Festplatte 1, zeigt reine LVM-Zahlen, vergleichen Sie mit dem obigen Ergebnis von Festplatte 1											
8395	97.8	21574	18.8	9106	16.0	7957	88.5	22097	16.6	256.0	2.6
LVM - Festplatten 1, 2, 3, 4 in einer VG, ein LV mit Striping (Striping über alle 4 Festplatten)											
8474	98.9	32048	27.8	6138	11.1	7142	79.8	23983	17.8	385.6	6.4
8428	98.3	31942	28.4	6064	10.7	6875	76.5	24198	17.6	372.1	4.4

Tabelle 2: Filesystem ist reiserfs

---Sequential Output (nosync)---						---Sequential Input--				--Rnd Seek-	
-Per Char-		--Block---		-Rewrite--		-Per Char-		--Block---		--04k (03)-	
K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	/sec	%CPU

HD 1 (kein LVM)											
7775	98.9	22793	76.2	8686	16.9	7265	81.4	21856	17.1	255.9	3.1
HD 2 (kein LVM)											
8399	97.8	22599	19.8	9233	16.2	7958	88.4	22077	16.6	250.4	2.6
HD 3 (kein LVM)											
8481	98.8	21080	18.5	9080	15.9	7969	88.5	22069	16.3	249.5	3.1
HD 4 (kein LVM)											
8487	98.8	21149	18.4	9052	16.0	7956	88.5	22010	15.0	260.9	3.8
LVM - nur Festplatte 1, zeigt reinen LVM-Aufwand, vergleichen Sie mit dem obigen Ergebnis von Festplatte 1											
7827	98.7	22609	81.8	8682	16.9	7223	80.9	21787	18.0	263.7	2.8
LVM - Festplatten 1,2,3,4 in einer VG, ein LV mit Striping (Striping über alle 4 Festplatten)											
7837	99.1	26304	85.9	5718	11.1	6744	75.6	23229	18.3	372.2	5.7
7824	98.9	29375	93.4	5787	11.1	6822	76.6	24296	18.9	392.7	4.7