Volume Manager Engine

LVM

Semesterarbeit

im Fachgebiet Betriebsysteme

HOCHSCHULE
FÜR TECHNIK
ZÜRICH

vorgelegt von:	Lucien Stucker
Studienbereich:	Informatik
Matrikelnummer:	06-557-540
Erstgutachter:	Beat Seeliger
Datum:	21. März 2011

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

1.1. Motivation

Die IT-Infrastruktur wird stetig komplexer. Gleichzeitig haben viele Unternehmungen vermehrt das Bedürfnis, wiederkehrende Arbeiten an den IT-Betrieb zu übergeben. Damit technisch komplexe Aufgaben nicht von Fachspezialisten ausgeführt werden müssen, benötigt es geeignete Werkzeuge, die dem Betriebspersonal helfen, solche Aufgaben schnell und korrekt auszuführen.

Im Storage Umfeld fehlen meist geeignete Werkzeuge, die es dem Betrieb ermöglichen, zentral für den einzelnen Server oder Cluster, das Volume zu verwalten. Ein solches Tool wird Gegenstand dieser Arbeit sein. Für einfachere IT-Strukturen wird das Volume individuell serverseitig verwaltet, hingegen bei komplexen Strukturen wäre ein solches Werkzeug wie hier beschrieben von grossem Vorteil. Zu berücksichtigen ist auch, dass die Komplexität beim Verwalten von Volume mit der Anzahl zugeordneten Disks und Funktionen wie Mirroring, Snapshot und Striping ansteigt. Es ist heute nicht selten, dass einem Server 30 und mehr Disk zugeordnet werden. Steht dem Unternehmen und somit dem IT-Betrieb kein geeignetes Werkzeug zur Verfügung, müssen weiterhin die wiederkehrende Pflege am Volume von Fachspezialisten “manuell“ ausgeführt werden. IT-Projekte, neue Technologien und Architekturen können nur durch qualifizierte Fachspezialisten nach vorne gebracht werden. Zusätzliche betriebliche Aufgaben verringen die meist ohnehin knappen Kapazitäten von Fachspezialisten, wodurch die effiziente Umsetzung von Projekten leicht verzögert werden und damit die Business-Verantwortlichen hart treffen könnte.

Es soll deshalb ein Konzept und Prototyp für ein Werkzeug erarbeitet werden, welches den Anwendern hilft, aus den verschiedenen Disks ein Volume zu erstellen, Volume zu spiegeln, zu vergrössern oder einen Snapshot zu generieren, ohne über ein fundiertes Fachwissen verfügen zu müssen.

Das Werkzeug soll primär für die Verwaltung von Volumes auf Linux Systemen entwickelt werden. Auf Linux wird meist der Linux Logical Volume Manager (LVM) eingesetzt.

Eine mögliche Erweiterung auf das von Oracle (Sun) entwickelte Dateisystem SZFSünd somit die Unterstützung von Solaris Systemen soll später möglich sein. ZFS fasst die Technologien aus Volume Manager, RAID Systemen und Dateisystem zusammen und macht die Verwaltung mit wenigen Befehlen einfacher.

Die Zuordnung von Speicher in der Storagebox und das Verknüpfung des Servers mit dem Storagebox Controller über ein SAN, soll nicht vom geplanten Werkzeug abgedeckt werden. Das Werkzeug soll keine automatische Optimierung am Volume vornehmen, z.B. Zuordnung von mehr Speicher während des Betriebs. Die Umsetzung eines verkaufsfertigen Produktes ist nicht Teil dieser Arbeit. Dafür wären zusätzliche Funktionen wie Mehrsprachigkeit, Anpassungsfähigkeit an die betrieblichen- und technischen Bedürfnisse des Unternehmens und die Umsetzung von erweiterten Sicherheitsstandards notwendig. Mit Hilfe eines Prototyps soll die Machbarkeit des Werkzeugs untersucht werden.

1.2. Ziel der Arbeit

Es soll ein Konzept für eine Server Client Engine für den Volume Manager LVM erstellt werden. Für das Konzept sollen die Eigenschaften von LVM untersucht werden. Anhand der gewonnenen Erkenntnissen soll ein Lösungsentwurf ausgearbeitet und mit der Entwicklung eines Prototyps soll die Machbarkeit des Konzept festgestellt werden.

Zusätzlich soll der Student in diesem Projekt Erfahrung und Wissen mit der Grundlagentechnik erlangen.

1.3. Aufgabenstellung

- Einarbeiten in LVM
- LVM Eigenschaften aufzeigen
- Anforderungsanalyse die auch zwischen zwingenden und nicht zwingenden Anforderungen unterscheidet
- Konzept für die Client basierten Volume Manager Engine erarbeiten
- Prototyp der VM Engine für LVM entwickeln
- Die Ergebnisse dokumentieren

1.4. Erwartete Resultate

- Dokumentation LVM
- Anforderungsanalyse
- Konzept LVM Engine
- Prototyp LVM Engine

1.5. Der Aufbau der Arbeit

Die Arbeit ist in mehrere Kapitel gegliedert, die im Folgenden kurz vorgestellt werden. Das Kapitel *2 LVM* behandelt das Thema LVM und bietet einen Einstieg in die Eigenschaften des Volume Managers. Zudem wird gezeigt, welche Architekturen mit dem Volume Manager realisiert werden können. In diesem Kapitel werden ebenfalls die wichtigsten Befehle zum Anlegen und Manipulieren von LVM Objekten dokumentiert. Im Kapitel *3 Analyse* werden die Anforderungen seitens der Anwender für eine zentrale Volume Manager Anwendung behandelt. Zusätzlich wird das Marktumfeld einer solchen Lösung analysiert und besprochen. Das Konzept der Applikation wird im Kapitel *4 Konzept Client basierte Volume Manager Engine* erläutert. Das Kapitel *5 Prototyp LVM Engine* zeigt die Ergebnisse und Umsetzung des Prototyps, welche den praktischen Teil der Arbeit umfasst.

2. LVM

2.1. Entstehung von LVM

Unter der Bezeichnung Logical Volume Manager (LVM) existieren unter Unix Betriebsystemen verschiedene Volume Manager. Geschichtlich betrachtet wurde der erste Volume Manager unter dem Namen LVM im Jahr 1989 von IBM für deren Unix Betriebsystem AIX entwickelt. Im Jahr 1995 hat Hewlett Packard auf Basis der IBM LVM Lösung ebenfalls einen Volume Manager entwickelt.

Der Linux Logical Volume Manager wurde von Heinz Maulshagen bei der Firma Sistina entwickelt. Dieser wurde anschliessend im Februar 2000 in den Linux Kernel Release 2.3.47 aufgenommen. Die Linux Version ist von der Bedienung her an die HP UX Version angeglichens. Die Firma Sistina hat einiges zur Storage und Clustering Fähigkeit von Linux beigetragen. Unter anderem wurde der Device-Mapper und das Global File System (GFS) von Matthew O'Keefe, welcher der Gründer von Sistina ist, entwickelt. Es ist daher nicht weiter verwunderlich, dass die grösseren Linux Distributoren auf Sistina aufmerksam wurden. RedHat sicherte sich mit der Übernahme von Sistina im Dezember 2003 wertvolles Knowhow im Bereich Linux Storage.

Noch unter dem Namen Sistina hat Heinz Maulshagen die zweite Version von LVM entwickelt. Dieser wurde unter dem Namen "LVM2" auf den Markt gebracht und war schnell von der Unix-Gemeinde wohlwollend aufgenommen worden. Im Vergleich zur Vorgängerversion ist LVM2 nicht mehr direkt in den Linux-Kernel integriert. Für eine nahtlose Integration wurde der generische Framework Device-Mapper entwickelt, welcher als Schnittstelle zum Volume Manager dient und die Block Devices verwaltet. Der Device-Mapper unterstützt den Linux Kernel ab Version 2.6 mit dem Local Volume Management. LVM2 blieb trotz dieser Architekturänderung abwärts kompatibel zu LVM, was sehr geschätzt wird. Bestehende mit dem LVM angelegten Volumes können somit mit LVM2 direkt verwaltet werden, oder können in LVM2 Volumen konveniert werden. Heute wird LVM2 mit den meisten Distributionen mitgeliefert, ist also zu einem quasi-Standard geworden. Die Enterprice Distributionen

von RedHat Enterprise und Suse Enterprise Linux verwenden in der Standardkonfiguration den LVM.

Für den Einsatz von LVM in der Cluster Umgebung mit Shared Storage Anbindung hat RedHat die Clustering Erweiterung CLVM für LVM2 entwickelt, welche es erlaubt, den LVM Command sicher und einfach in einer Shared Storage Umgebung zu verwenden.

2.2. LVM Eigenschaften

2.2.1. Architektur von LVM2

LVM besteht aus drei Komponenten[1]. Die Abbildung *2.1 LVM: Mind Map* zeigt die Eigenschaften von LVM, bzw. dessen Komponenten, Befehle und Funktionen in einem Mind-Map dargestellt.

[1] http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Logical_Volume_Manager/

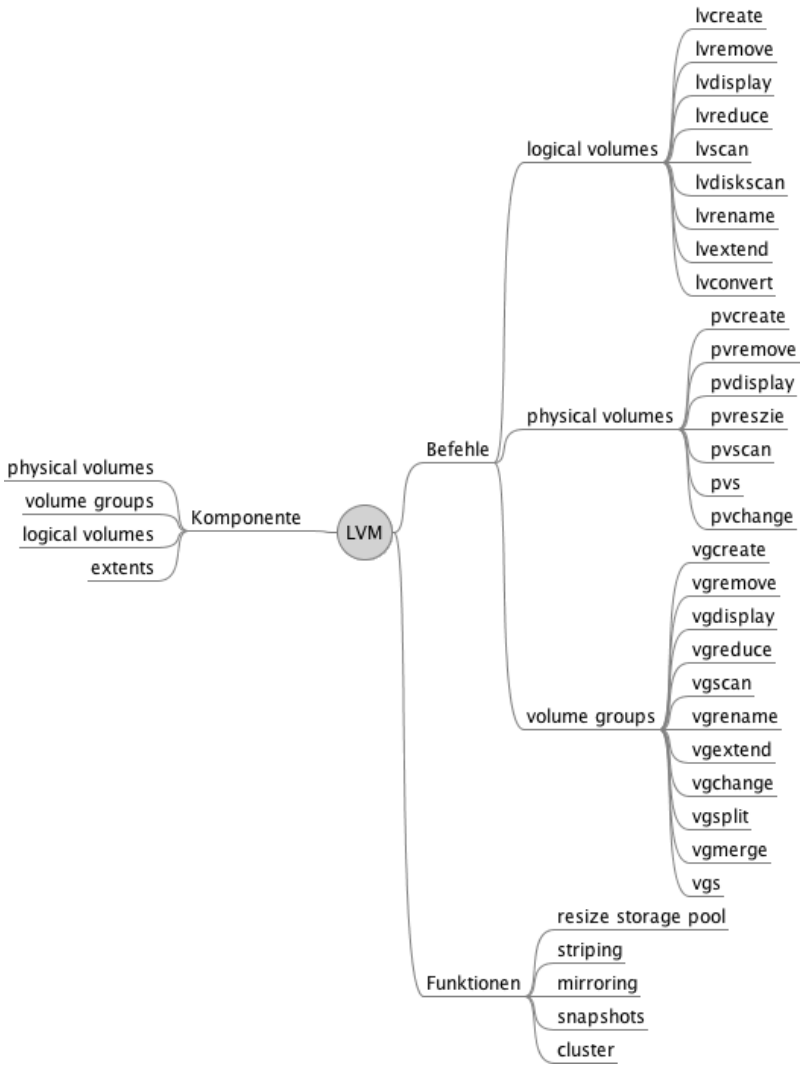
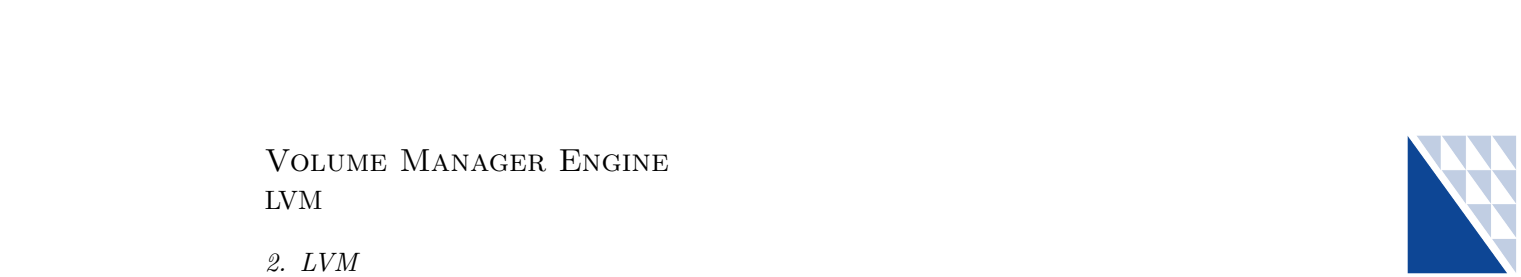


Abbildung 2.1.: LVM: Mind Map

2.2.1.1. Physical Volumes (PV)

Die unterste Schicht eines LVM Logical Volume ist eine physische Disk. Meist ist diese eine Festplatte, eine LUN, welches über Storage Network SAN zugeteilt wurde. Es kann aber auch nur eine Partition oder eine Datei sein, die als Disk verwendet wird. Eine physische Disk muss für die Verwendung im LVM als Physical Volume (PV) initialisiert werden. Dazu wird am Anfang der Disk ein entsprechender Label gesetzt, im Fachchargon auch “labeling“ genannt. Wie in Abbildung *2.2 Physical Volume: Layout* dargestellt, wird diese Kennzeichnung auch “Label“ genannt, welches in den zweiten 512-Byte Sektor der Disk geschrieben wird. Dieses Kennzeichnung



kann jedoch übersteuert werden, um den Label in einen der ersten Sektoren zu schreiben.

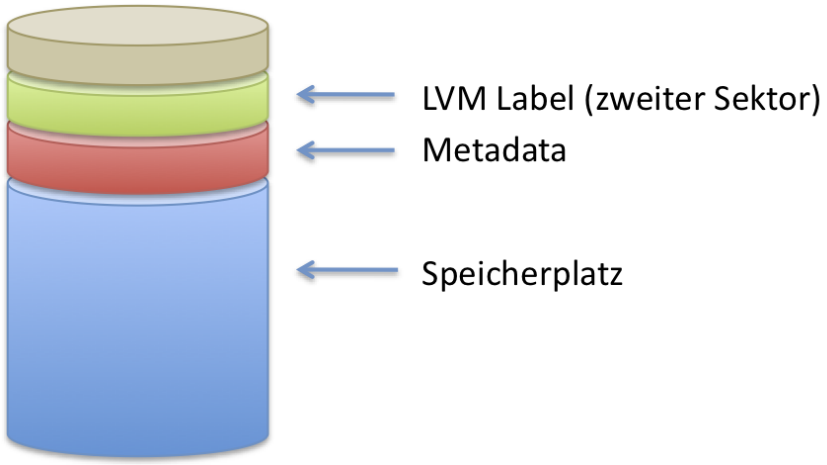


Abbildung 2.2.: Physical Volume: Layout

Der LVM Label dient dazu, die Disk korrekt zu identifizieren. Die Reihenfolge der Disk ist bei einen Neustart nicht gewährleistet, weshalb im Label ein eindeutiger zufälliger Identifikator bzw. eine Kennung, genannt “UUID“, gespeichert wird. Anhand dieser UUID kann die Disk auch über einen Cluster wieder erkannt werden. Zusätzlich zur UUID sind im Label die Grösse der Disk in Bytes und die Record Adressen, in welche die LVM Metadata gespeichert sind, festgehalten.

Die LVM Metadaten enthalten die Konfigurationsdatei von LVM Volume Groups des Systems.

2.2.1.2. Volume Group (VG)

Die Physical Volumes werden bei LVM zu Volume Groups kombiniert. Die erstellten Volume Groups dienen als Pool von Diskdpeicherplatz, welche zu Logical Volumes alloziert werden können. Beim Erstellen eines Volume Group aus ein oder mehreren Physical Volumes wird der Diskspeicherplatz des Physical Volumes in Einheiten bestehend aus einer fixen Grösse unterteilt, die sogenannten Physical Extents (PE). Extents sind die kleinste physikalische Speichereinheit in einem LVM System. Die Extentsgrösse kann beim Erstellen des Volume Group gestgelegt werden, oder den Standardwert von 4 Megabyte übernommen werden. Die maximale Grösse eines Extent ist 1 Gigabyte. Die Anzahl an Extents in einer Volume Group ergibt sich aus der Summe aller Physical Volumes in einer Volume Group. Extents dienen zur “n to

m“ (Physical-to-Logical) Volumenzuordnung. Mehr dazu wird im Abschnitt *2.2.1.3 Logical Volumes (LV)* erläutert.

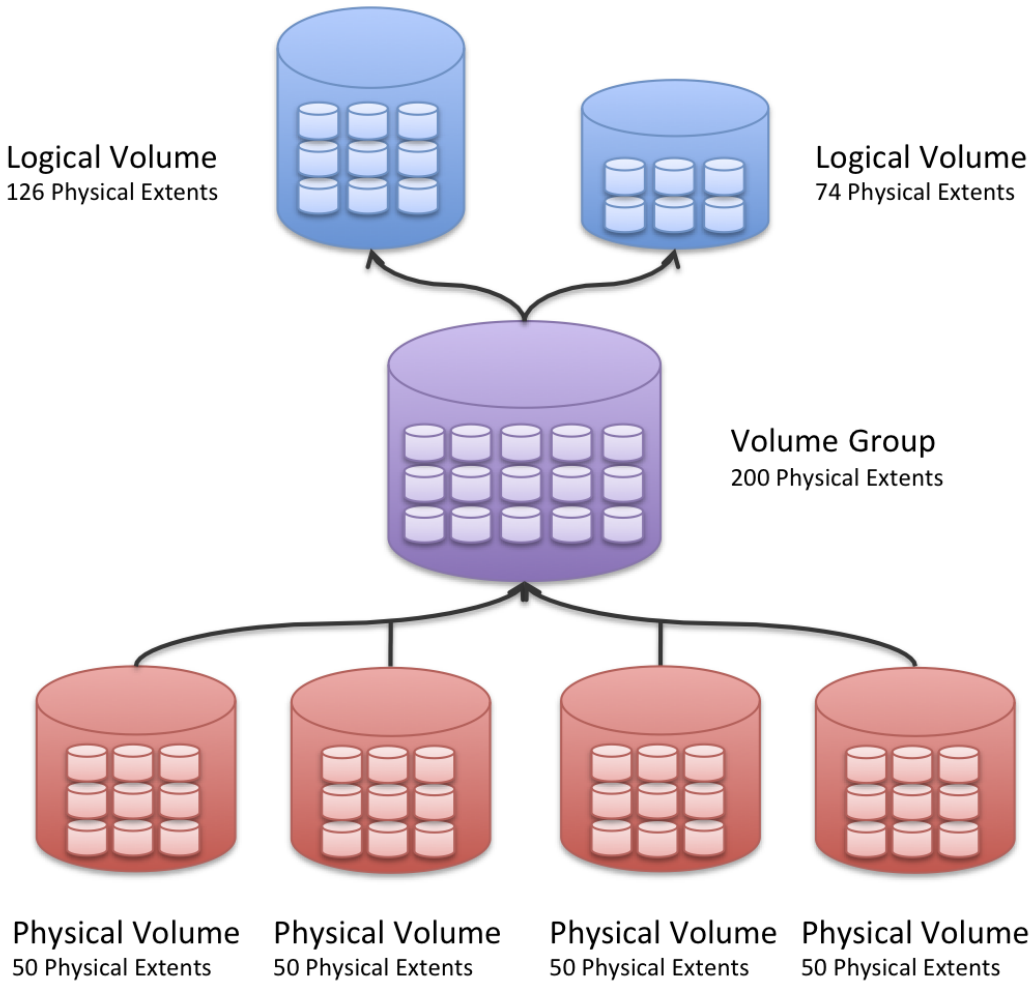


Abbildung 2.3.: Extents Mapping: Layout

2.2.1.3. Logical Volumes (LV)

Wie bereits beschrieben werden Logical Volumes aus einer Volume Group erstellt. Eine Volume Group kann in mehrere Logical Volumes unterteilt werden, jedoch kann sich ein Logical Volume nicht über mehrere Volume Groups erstrecken. Wie im Abschnitt Volume Groups erwähnt, dienen Extents zum Zuordnen von Physical-to-Logical Volumes. Ein solches Mapping wird in Abbildung *2.3 Extents Mapping: Layout* veranschaulicht. Eine Logical Volume wird analog zum Physical Volume in Extents unterteilt, sogenannte Logical Extents (LE). Die Logical Extents besitzen die gleiche Grösse wie die Physical Extents der Volume Groups, in welcher sich

das Logical Volume befindet. Die Grösse des Logical Volume setzt sich aus den Anzahl Physical Extents zusammen, welche aus der Volume Groups hinzugefügt wurde. Wird beim Erstellen der Volume Groups eine Extentgrösse von 32 Megabyte gewählt, kann die Logical Volume Grösse nur eine Vielfaches von 32 Megabyte sein. Ein Volume kann auf einen 64bit maximal 8 Exabytes gross sein.

Die Abbildung *2.3 Extents Mapping: Layout* zeigt eine LVM Beispielkonfiguration. In diesem Beispiel wurden 4 Physical Volume mit je 50 Extents einer Volume Group zugeordnet. Somit verfügt die Volume Group über 200 Extents (4x50), welche sie ihren Logical Volumes zur Verfügung stellen kann. Aus diesen 200 Extents wurden in diesen Beispiel zwei Logical Volumes mit 126 Extents und 74 Extents erstellt.

2.2.1.3.1. Flexible Kapazität

Bei traditionellen Volumes kann ein Volume maximal die Grösse eines physischen Disks annehmen. Ein Logical Volume hingegen kann durch die Abstraktion fast beliebig vergrössert werden. Durch die Zuordnung von weiteren Extents, welche in der Volume Group als “frei“ makiert sind, kann das Logical Volume vergrössert werden. Wächst der Speicherplatz eines der beiden Logical Volumes gemäss Abbildung *2.3 Extents Mapping: Layout*, müsste die Volume Group mit einer oder mehreren zusätzlichen Physical Volume ergänzt werden, um der Logical Volumes mehr Extents zuordnen zu können.

Umgekehrt kann das Logical Volume durch das Entfernen von Extents verkleinert werden. Sind in der Volume Group keine freien Extents mehr verfügbar, kann die Volume Group durch Zuordnung von weiteren Physical Volume vergrössert werden, welche wiederum einer Logical Volume Group zugeordnet werden kann. Die kleinstmöglichste Wachstumgrösse der Logical Volume ist eine Extentsgrösse. Mehr zur Extentsgrösse wird im Abschnitt *2.2.1.2 Volume Group (VG)* erklärt.

2.2.1.3.2. Unterbruchsfreie Umplatzierung der Daten

Durch LVM können die Logical Volumes inkl. Dateisystem und deren Daten zur Laufzeit auf neuere, performantere Speichersysteme umplatziert werden, ohne deswegen den Betrieb unterbrechen zu müssen. Trotz Verwendung der Disks ist es möglich, die Daten im Volume neu zu organisieren. Somit ist es kein Problem, eine verwendete physische Disk von Daten zu befreien, bevor diese ersetzt bzw. entfernt wird (z. B. bei der Renovation der physischen Umgebung).

2.2.1.3.3. Striping Volume

Bei hoher Anzahl von sequenziellen schreib und lese zugriffe auf eine Volume ist es empfehlenswert eine Strips Volume zu verwenden. Striped Volumes

2.2.1.3.4. Spiegelung von Volumes (Mirroring)

Mirrored Logical Volumes sind exakte Kopien eines Volumes, welche durch Spiegelung, auch Mirroring genannt, dauernd mit dem Original in Synchronisation gehalten werden. Wenn bei einem LVM für ein Volume eine Mirrored Logical Volume erstellt wird, stellt LVM sicher, dass wenn Daten auf ein darunter liegendes Physical Volume geschrieben werden soll, die Daten automatisch auf ein weiteres Physical Volume gespiegelt werden. LVM unterstützt zudem das Spiegeln auf mehrere Mirrored Volume. Durch die redundante Speicherung der Daten auf verschiedene Physical Volumes, werden die Daten und das System von Diskfehler besser geschützt. Tritt der Fall eines physischen Diskdefekt ein, steht zwar dieses Laufwerk dem System nicht mehr zur Verfügung, erlaubt aber dem System ohne Unterbruch mit den zusätzlichen Mirrored Volumes weiterzuarbeiten. Für das Mirroring wird das Physical Volume in Regionen unterteilt, der Standardwert für die Region-Grösse ist 512 KB, zusätzlich wird ein kleines Log geführt, um nachzuvollziehen, welche Regionen deckungsgleich mit dem Original sind. Das Log wird zumeist aus Sicherheitsgründen auf eine weitere Disk bzw. Volume abgelegt. Manchmal wird es im Memory geführt, was aber bei einem Systemabsturz gefährliche Nachteile in sich birgt.

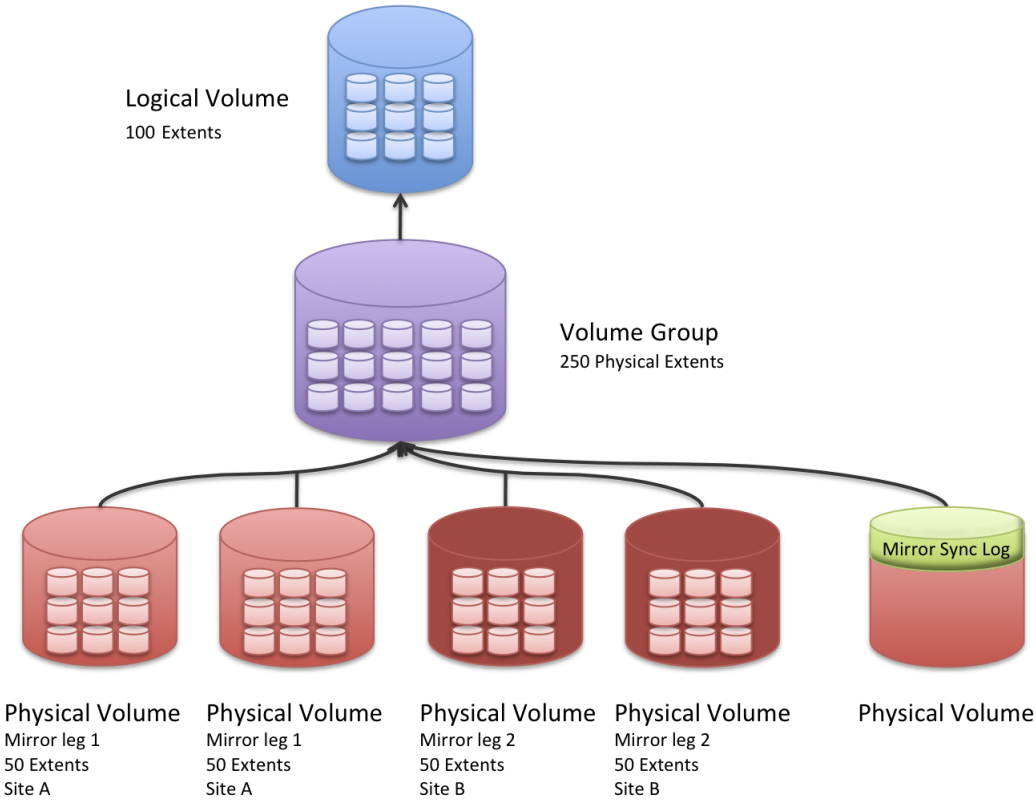


Abbildung 2.4.: LVM: Mirror Architektur

Mirroring wird in den meisten Fällen nicht als Ersatz eines RAID-1 Systeme betrachtet. In vielen Modernen IT-Landschaften werden die Disk bzw. LUN über eine Storage Area Network (SAN) einem System zur Verfügung gestellt. Diese LUN stammen meist aus einer RAID-1 oder RAID-5 Diskgruppe und sind somit im Storage System redundand. Das Mirroring wird in solchen Fällen meist zur Spiegelung der Daten in ein weiteres Rechenzentrum angewendet. Indem das LVM mehrere mirrored Disks unterstützt, können die Daten in mehrere Rechenzentren gespiegelt werden.

Die Abbildung *2.4 LVM: Mirror Architektur* zeigt eine mögliche Architektur eines LVM Mirrors, welche über zwei Rechenzentren Site A und Site B gespiegelt wird. Dazu wurden je zwei Physical Volume über jeweils eine Network Storage System pro Rechenzentrum zugeordnet. Das Mirror Log wird auf eine separaten Physical Volume geschrieben. Mit dieser Architektur lassen sich in Cluster Umgebungen hoch redundante Systeme bauen.

2.2.1.3.5. Snapshot

LVM unterstützt das Erstellen von Snapshots eines Logical Volume. Wenn ein Snapshot erstellt wurde, werden alle Änderungen im Volume im Snapshot gespeichert und das Original Volume bleibt unverändert im Originalzustand. Beim Erstellen eines Snapshot ist darauf zu achten, dass die Snapshot-Grösse etwas mehr Speicherplatz zur Verfügung steht, als die geplante Änderung benötigt. Sobald der reservierte Speicherplatz eines Snapshot voll ausgenutzt wurde, können neue Änderungen nicht mehr nachvollzogen bzw. gespeichert werden, wodurch das Snapshot ungültig wird und alle Änderungen verloren gehen. Um ein "Überlaufen" zu verhindern, sollten Snapshot regelmässig und zeitnah überwacht werden und bei Bedarf genügend vergrössert werden.

Snapshots bieten beim Betrieb eines Systems einige Vorteile:

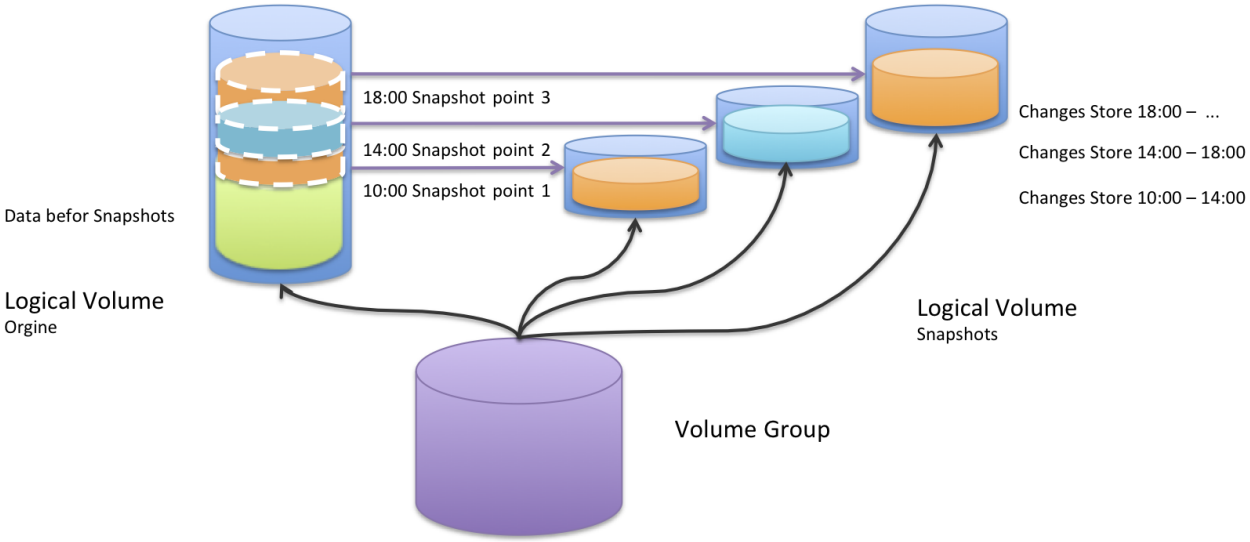


Abbildung 2.5.: LVM: Snapshot

- Es können Tests gegen produktive Daten durchgeführt werden
- Sollen z.B. Datenbanken während des laufenden Betriebes gesichert werden, können mit Hilfe der LVM Snapshot Technologie die Daten parallel gesichert werden, auch wenn zur gleichen Zeit die Originaldaten (zu sichernde Daten) sich verändern sollten (erhalten der Datenkonsistenz).
- Ein File-Systemcheck kann mit Hilfe der LVM Technologie auf dem Filesystem getestet werden.

2.2.1.3.6. Clustered Logical Volume Manager

Clustered Logical Volume Manager (CLVM) ermöglicht es, wie im Abschnitt "Entstehung LVM" erwähnt, die Verwendung von LVM im Cluster Umfeld. CLVM besteht aus dem Deamon clvmd welche auf allen Cluster Nodes installiert und während des Bootvorgangs gestartet sein müssen. Bei einer Änderung an der Volume Konfiguration verteilt der Deamon clvmd die aktuellen LVM Metadaten an alle Nodes und ermöglicht damit, dass alle Nodes dieselbe Sicht auf die Logical Volumes haben. In der Konfigurations Datei lvm.conf von LVM muss für CLVM der Locking Typ der Volumes auf Typ 3 "Built-in clustered locking" angepasst werden.

2.3. Einarbeiten in LVM (LVM2 Commands)

2.3.1. Physical Volume anlegen

Mit folgendem Beispielbefehl werden aus einem Block-Laufwerk */dev/sdb*, */dev/sdc*, */dev/sdd* und */dev/sde* vier Physical Volumes erstellt und gelabelt.

```
pvcreate /dev/sdb /dev/sdc /dev/sdd /dev/sde
```

Mehr Informationen sind in den MAN Pages[2] zu finden.

2.3.2. Physical Volume entfernen

Mit dem Befehl **pvremove** kann ein Physical Volume Label auf aus einem Block-Laufwerk entfernt werden.

```
pvremove /dev/sdb
```

Mehr Informationen sind in den MAN Pages[3] zu finden.

2.3.2.1. Volume Group anlegen

Mit dem Befehl vgcreate kann man aus einem oder mehreren Physical Volumes eine Volume Group erstellen. Die Extentsgrösse kann mit der Option **-s** bestimmt werden. Wird diese Option nicht genutzt, so verwendet das LVM die Standardgrösse, welche in der lvm.conf festgelegt ist. Die Extentgrösse von 32 MB hat sich in der Praxis bewährt. Grössere Extents haben keinen Einfluss auf eine verbesserte I/O

[2] http://linux.die.net/man/8/pvcreate
[3] http://linux.die.net/man/8/pvcreate

Performance. Die Zuordnung der Physical Extents, kann mit –**alloc** definiert werden. Die Standardkonfiguration **normanl** muss nur in speziellen Fällen angepasst werden, wenn aus speziellen Performancegründen evtl. beim Backup parallele Strips gewünscht wird.

```
vgcreate VolGroupData
```

Volume Groups in einer Cluster-Umgebung, welche mit CLVM auf einem geteilten Storage erstellt wurden, sind von allen Cluster Nodes sichtbar. Mit der Option **-c** wird die Sichtbarkeit der Volume Group auf den lokalen Node beschränkt.

```
vgcreate -c VolGroupCluData1 /dev/sdb /dev/sdc /dev/sdd /dev/sde
```

Mehr Informationen sind in den MAN Pages[4] zu finden.

2.3.2.2. Konfiguration von Volume Group auslesen

Mit dem Befehl **vgdisplay vgs** kann die Konfiguration der Volume Group ausgelesen werden. Der Befehl **vgdisplay** eignet sich um eine schnelle Übersicht zu gewinnen. Der Befehl **vgs** eignet zum Auslesen der Konfiguration per Script, da mit Optionen die Ausgabe individuell angepasst werden kann.

```
vgdisplay VolGroupCluData1
```

```
vgs
```

Mehr Informationen sind in den MAN Pages[5] [6] zu finden.

2.3.2.3. Konfiguration von Volume Group anpassen

Mit dem Befehl **vgchange** kann die Konfiguration einer bestehende Volume Group angepasst werden.

Mehr Informationen sind in den MAN Pages[7] zu finden.

[4] http://linux.die.net/man/8/vgcreate
[5] http://linux.die.net/man/8/vgdisplay
[6] http://linux.die.net/man/8/vgs
[7] http://linux.die.net/man/8/vgchange

2.3.2.4. Volume Group vergrössern

Durch Hinzufügen von weiteren Physical Volumes kann eine Volume Group vergrössert werden und somit Speicherplatz für weitere Logical Volumes oder vergrösserte Logical Volumes zur Verfügung gestellt werden.

```
vgextent VolGroupCluData1 /dev/sdf /dev/sdg
```

Mehr Informationen sind in den MAN Pages[8] zu finden.

2.3.2.5. Volume Group entfernen

Sind alle Logical Volumes einer Volume Group entfernt worden, kann mit dem Befehl **vgremove** die Volume Group vom System entfernt werden. Die Eigenschaft "Cur LV" der Ausgabe von **vgdisplay** zeigt die Anzahl Logical Volumes der Volume Groupe und sollte vor dem Entfernen der Volume Group 0 stehen.

```
vgremove VolGroupCluData1
```

Mehr Informationen sind in den MAN Pages[9] zu finden.

2.3.2.6. Logical Volume anlegen

Mit dem Befehl **lvcreate** können lineare, striped oder mirrored Volumes angelegt werden. Grundvoraussetzung für das Anlegen eines Logical Volumes ist, dass auf dem System bereits mindestens eine Volume Group angelegt wurde.

In diesen Beispiel wird eine Logical Volume mit der Grösse von 2 GB auf der Volume Groupe VolClusterData1 angelegt.

```
lvcreate -L 2G VolClusterData1
```

Wird kein Namen für das Logical Volume mit der Option **-n lvname** festgelegt, vergibt LVM automatisch einen generierten Namen "lvol0" (dieser wird durchnummeriert).

```
lvcreate -L 2G VolClusterData1 -n clvdata1
```

Die Grösse eines LVM kann auch prozentual zur Volume Group Grösse mit dem Zusatz **VG** definiert werden oder prozentual zum restlichen freien Speicherplatz der Volume Group mit dem Zusatz **FREE**.

[8] http://linux.die.net/man/8/vgextent
[9] http://linux.die.net/man/8/vgremove

```
lvcreate -L 50%VG VolClusterData1 -n clvdata1
```

```
lvcreate -L 50%FREE VolClusterData1 -n clvdata1
```

Mehr Informationen sind in den MAN Pages[10] zu finden.

2.3.2.7. Striped Locical Volume anlegen

Eine Striped Logical Volume wird mit **lvcreate** und der Option **-i** und der Anzahl an Physical Volumes aus der Volume Group angelegt, welche für LVM verwendet werden soll. Die Grösse eines einzelnen Strips kann mit der Option **-I** festgelegt werden. In der Praxis haben sich Stripgrössen von 64kB oder 128kB bewährt.

```
lvcreate -L 2G -i2 -I64 VolClusterData1 -n clvdata1
```

2.3.2.8. Mirrored Locical Volume anlegen

Eine Mirrored Logical Volume wird mit der Option **-m** plus der Anzahl der Mirrored Volumes angelegt. Das unten aufgeführte Beispiel legt zwei Mirrored Volumes an und legt somit fest, dass die Daten dreifach gespeichert werden sollen.

```
lvcreate -L 2G -m2 VolClusterData1 -n clvdata1
```

2.3.2.9. Logical Volume grösse anpassen

Wird mehr Speicherplatz im Logical Volume benötigt, kann der Speicherplatz mit dem Befehl **lvresize** angepasst werden. Bevor man das Logical Volume vergrössert, sollte man prüfen, ob die darunter liegende Volume Groupe noch genügen freie Extents hat.

```
lvextend -L +2G /dev/VolGroupCluData1/VolClusterData1
```

Eine Logical Volume kann auf dieselbe Weise auch wieder verkleinert werden. Hier ist darauf zu achten, dass man das darüber liegende Filessystem vorher verkleinert hat.

```
lvextend -L -1G /dev/VolGroupCluData1/VolClusterData1
```

Mehr Informationen sind in den MAN Pages[11] zu finden.

[10] http://linux.die.net/man/8/lvcreate
[11] http://linux.die.net/man/8/lvextend

2.3.2.10. Logical Volume entfernen

Wird das Logical Volume nicht mehr benötigt, kann das Logical Volume mit dem Befehl **lvremove** entfernt werden. Die Verwendeten Extents des Logical Volume werden in der Volume Group als frei markiert. Achtung alle Daten auf dem Volume gehen verloren.

```
lvremove /dev/VolGroupCluData1/VolClusterData1
```

Mehr Informationen sind in den MAN Pages[12] zu finden.

[12] http://linux.die.net/man/8/lvremove

3. Analyse

3.1. Anforderungsanalyse

Für die Anforderungsanalyse wurde aufgrund der Projekt- und Applikationsgrösse das User-Stories Verfahren gewählt. Für die Ausarbeitung der Anforderungen wurde ein kleines Gremium zusammengestellt, welche die einzelnen Interessengruppen vertreten. Zu den Intressengruppen gehören der Teamleiter, die Storage Admins, die Capacity Planer und der System Admin und System Engineer. Die Use Cases wurden anschliessend in zwingende Anforderungen und nicht-zwingende Anforderungen unterteilt.

Lukas (System Engineer):

Mit dem Tool vorgenommene Änderungen müssen aufgezeichnet (geloggt) werden.
Bevor die Änderungen vorgenommen werden, soll eine Zusammenfassung angezeigt werden (idealierweise mit graphischer Darstellung)

Thorsten (Storage Admin):

Als Storage Admin kann ich den Storage End-to-End bis zum Filesystem bereitstellen
Als Capacity Planer kann ich mir schnell und einfach einen Überblick über den benutzten und verfügbaren Diskspace verschaffen
Als User kann ich einfach neuen Diskspace auf meinem Server bestellen und kann den Workflow verfolgen.

Daniel (System Admin):

Als System Admin kann ich das Volumes des Servers vergrössern, ohne LVM Befehle kennen zu müssen

Als System Admin kann ich beim Vergrössern der Volumes automatisch das Filesystem vergrössern
Als System Admin kann ich ein Site-übergreifendes Mirroring einrichten
Als System Admin kann ich den Diskspace auf ein anderes Storage System migrieren
Als System Admin werde ich beim sicheren Entfernen eines Disk eines gewählten Servers von der Anwendung unterstützt
Als System Admin erhalte ich einen raschen Überblick über die Diskkonfiguration des zu wartenden Servers

David (Teamleiter):

Als Teamleiter kann ich nachvollziehen wer, wann welche Änderung vorgenommen hat
Als Applikation Owner kann ich den zugewiesenen Diskplatz prüfen

Philipp (Capacity Admin):

Als Capacity Planer kann ich eine Storagereservation vornehmen
Als Capacity Planer erhalte ich einen raschen Überblick auf die noch unbenutzten Storages

3.1.1. Zwingend

Die folgenden Anforderungen müssen zwingend umgesetzt werden:

- Als System Admin kann ich das Filesystem des Server vergrössern, ohne LVM Befehle zu kennen
- Als System Admin kann ich Storage am Server wieder reduzieren
- Als System Admin erhalte ich einen raschen Überblick über die Diskkonfiguration des Servers
- Als User kann ich einfach neuen Diskspace auf meinem Server bestellen und kann den Workflow verfolgen.
- Die dem vom Tool vorgenommenen Änderungen müssen aufgezeichnet (geloggt) werden.

3.1.2. Nicht zwingend

Die folgenden Anforderungen, werden in diesem Projekt nicht umgesetzt, auch wenn einige davon für eine Produktiveversion interessante und wichtige Anforderungen wären:

- Als Capacity Planer kann ich Storage reservieren
- Als Applikation Owner kann ich den zugewiesenen Diskplatz prüfen
- Als Capacity Planer kann ich mir schnell und einfach einen Überblick über den benutzten und verfügbaren Diskspace verschaffen
- Bevor die Änderungen vorgenommen werden, soll eine Zusammenfassung (idealerweise graphisch dargestellt) angezeigt werden

3.2. Marktanalyse

3.2.1. Potenzielle Kunden

Unternehmen konkurrenzieren sich nicht nur im Produkteverkauf und bei der Aquisition von neuen Kunden, sonder auch bei der effizienten Gestaltung der internen Abläufe und optimalen Nutzung der eigenen Ressourcen. Seit Beginn des Computerzeitalters hat die Bedeutung der IT in ausnahmslos allen Unternehmen stark zugenommen. Unternehmen mit einer schlecht funktionierenden IT-Infrastruktur verlieren am Markt an Konkurrenzstärke und manövrieren sich über kurz oder lang auf das Abstellgeleise mit einer unsicheren Zukunft. Eine moderne Geschäftsleitung hat dies längst erkannt und legt Wert auf die Erhaltung und den Ausbau der inneren Stärke in Bezug auf die Abstimmung der Business-Strategie mit der Leistung der IT. Was nützt der beste Verkäufer, wenn es nach dem Verkaufsabschluss mit der Umsetzung des Auftrages happert.

Die Autoren Li und Tan haben in einem Artikel über das Thema IEEE, die Frage der CIO-Charakteristik zur Unternehmensstrategie untersucht:

> "'In a diverse, ever-changing marketplace, organizations are constantly seeking to harness technology to improve their core competency and gain competitive advantage. **Explicitly, knowing how to apply Information Systems (IS) in an appropriate and timely way and in harmony with business strategies, goals, and needs could bring the organization to steps closer to business success.** In other

> words, aligning IS strategy to business strategy becomes a critical issue in most organizations. Indeed, there is an increasing amount of research and understanding on the linkages between business and IS strategies, the role of partnerships between IS and business management, and the need to understand the transformation of business strategies resulting from the competitive use of IS.“’ [13]

Trotz der zunehmenden Bedeutung der IT-Leistung für die innerbetrieblichen Abläufe aber auch vermehrt direkt an der Verkaufsfront, steigt der Druck an stetiger Optimierung und effizienteren Verarbeitung der IT-Prozesse in den IT-Abteilungen. Zusätzlicher Druck von Aussen bezüglich politischen und marktwirtschaftlichen Veränderungen, ich denke da zB. an neue Gesetze und Regulatorien, erhöht die Erwartungen an die IT Dienstleistung. Unternehmen vergleichen sich heute immer mehr an der Leistungsfähigkeit ihrer IT-Leistung wie zB. bezüglich der Kostenstruktur oder Funktionalität mit der Konkurrenz. Schneidet die Effizienz der IT-Abteilung (Leistung versus Kosten) schlecht ab, sind die Gegenmassnahmen oft drastisch und erfordert höchste Konzentration des IT-Managements. Zu nennen sind da unter anderem massive Kosteneinsparungen, Streichung oder Aufschiebung von Projekten, Entlassung von internen und externen Mitarbeiter, Outsourcing bzw. Insourcing, um nur einige der möglichen Massnahmen aufzuführen. Dass solche Eingriffe den IT-Betrieb zusätzlich lähmen kann versteht sich von selbst. Eine laufende Überprüfung und Verbesserung der eigenen Organisation und Kostenstruktur gehört zum Tagesgeschäft des erfolgreichen IT-Managers. Es verwundert nicht, dass daraus auch neue Modell entstehen, wie zB. das anbieten der IT-Dienstleistung als Profit-Center innerhalb des Gesamtbetriebes, die Auslagerung der Dienste ins Ausland nach dem Outsourcing Modell der Inder oder die Aufsplittung der Services in Production und Engineering. In diesem Sinn ist dieses Projekt zu verstehen, indem eben die Storage-Verwaltung vereinfacht und effizienter gestaltet werden soll, um damit einen vielleicht kleinen, aber wichtigen Beitrag zur Verringerung der IT-Kosten zu erfüllen.

3.2.2. Server Markt

3.2.2.1. Betriebsystem

Waren vor ein paar Jahren Linux-Installationen im Server-Bereich, meist als ideologisch und experimentell angesehen, hat sich dies in der Zwischenzeit gewandelt.

[13] [Li und Tan 2009, S. 1]

Vermehrt wird Linux für Hosting- und Web-Installationen in Klein- bis Grossbetrieben professionell eingesetzt und hat Boden gegenüber von dominierenden Windows-Server-Installationen gutgemacht. Dank zunehmender Unterstützung seitens Software Herstellern für die Linux-Plattform steigt das Interesse an Linux im professionellen Geschäftsumfeld. Vergleicht man das Angebot der Betriebsystem Hersteller Oracle, IBM und Microsoft mit dem Angebot der beiden Linux Distributoren Red Hat und Novell Suse, ist ersichtlich, dass sich die Angebote bezüglich Wartung, Support und Dienstleistungen nicht wesentlich voneinander unterscheiden. Trotz diesem heute breiten Angebot an Funktionalität im Linux Umfeld, konnte sich dieses Betriebsystem bis anhin im Finanzdienstleistungs-Branche bei den Unternehmen nicht durchsetzen, ausser für dedizierte Web-Installationen. Traditionell dominieren in diesem Marktsegment immer noch die Betriebsysteme wie Windows, Oracle Solaris (SUN) bzw. IBM AIX als die strategisch geführten Plattformen. Es gibt jedoch Anzeichen, dass sich dieses Verhältnis künftig ändern könnte, denn einige Unternehmen haben für neue Projekte ein Auge auf Linux neben Windows geworfen und liebäugeln Linux nun auch als strategische Betriebsystem Plattform einzusetzen.

3.2.2.2. Hardware Platform

Ein möglicher Faktor für die zunehmende Ablösung der Betriebsysteme Solaris und AIX durch Linux könnte deren Hardware Plattform, welche zumeist auf RISC bzw. Intanium Prozessoren basieren, sein. Im Low- und Midrange Bereich sind solche Hardware Systeme nicht gleich Leistungsfähig wie Systeme welche auf den x86 bzw. die x64 Prozessor-Architektur aufgebaut sind. Zudem lassen die höheren Kosten für den Betrieb und Unterhalt von zwei unterschiedlichen Hardware-Plattformen innerhalb deselben Unternehmens solche Projekte verständlicherweise in den Hintergrund drängen. Diese Einschätzung decken sich auch mit der Server-Hardware Marktanteil Analyse vom zweiten Quartal 2010 von Gartner. Die x86 basierenden Systeme sind um 28.9 Prozent in der verkauften Stückzahl und in 37 Prozent im Umsatz gewachsen. RISC/Itanium basierenden Systeme haben im gleichen Zeitraum im Vergleich zum selben Vorjahresquartal 16.5 Prozent bezüglich der verkauften Stückzahlen sowie 8.8 Prozent beim Umsatz verloren.

Das Linien-Diagramm der Abbildung 3.1 zeigt die Anzahl verkauften Servern der einzelnen Herstellern vom ersten Quartal 2007 bis zum zweiten Quartal 2010. Aus dem Diagramm kann man nicht direkt erkennen, dass die x86/x64 Prozessor-Architektur bei den Servern im Vergleich zu RISC bzw. Itanium Prozessoren überwiegt. Neben HP und Dell, welche hauptsächlich x86/X64 Server verkaufen, ist Oracle einer der

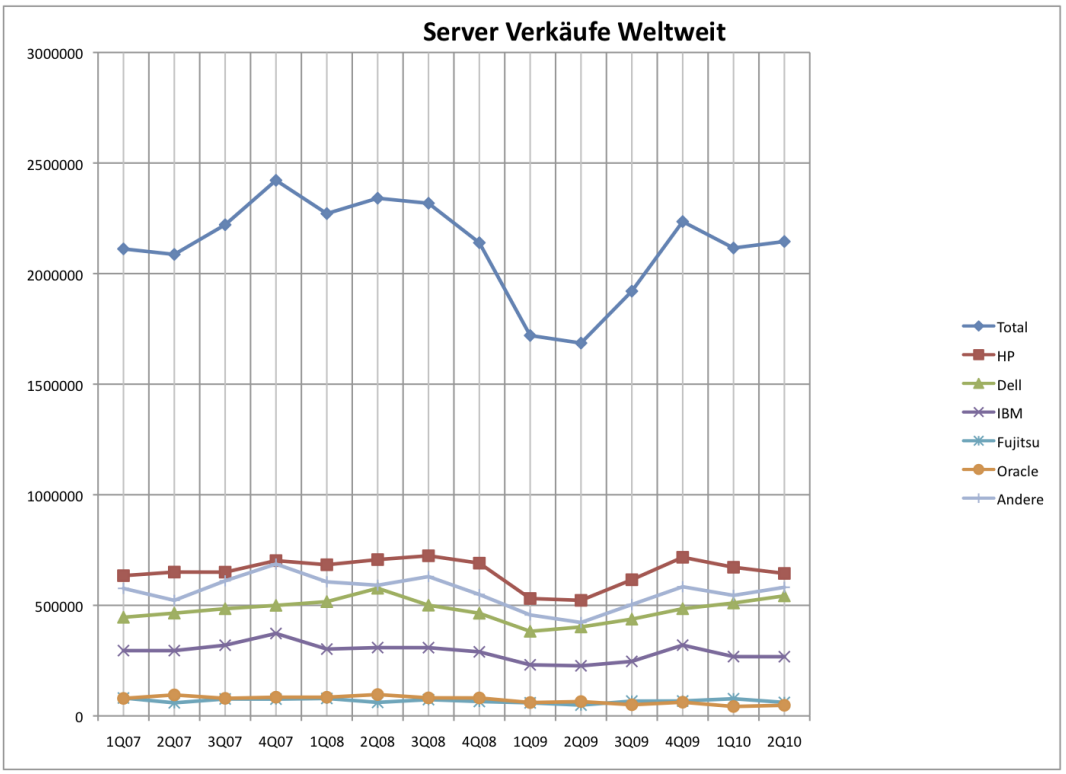


Abbildung 3.1.: Diagramm - Server Verkauf: Weltweit pro Quartal *Quelle Gartner*

führenden RISC Server Hersteller bezüglich der Anzahl verkaufter Server weit abgeschlagen. Zudem kann man aus den Tabelle 3.1 und Tabelle 3.3 erkennen, dass Oracle im zweiten Quartal 2010 gegenüber der Konkurrenz den stärksten Abastzrückgang erleiden musste.

Gemäss Herrn O'Connell von Garnter sind die Rückgänge vom RISC/Itanium Segment auf die folgenden Ursachen rückzuführen:

> "'The first half of 2010 has seen continued challenges for vendors in the server segment of RISC and Itanium Unix. The combination of longer sales cycles for these systems, product refreshes and ongoing economic uncertainty has resulted in RISC/Itanium Unix revenue in the second quarter of 2010 being less than half that of the second quarter of 2008, we expect this market to improve in the second half of 2010, but vendors in this segment will face an increasingly difficult challenge in minimizing migrations to Windows and Linux platforms."' [14]

[14] O'Connell, http://www.gartner.com/it/page.jsp?id=1426834

Wie man in der Tabelle 3.1 von Gartner über die weltweiten Server Warenlieferungen im zweiten Quartal 2010 erkennen kann, haben die beiden grössten Server Hersteller mit RISC Architektur eine Wachstumsverlangsamung gegenüber dem gleichen Quartal 2009 in Kauf nehmen müssen. Die restlichen Hersteller hatten ein der selben Periode ein positives Wachstum verzeichnet. Bei IBM ist diese Zahl wenig aussagefähig, da nicht hervorgeht wie der Rückgang der Zahlen auf die Produkte bezüglich Prozessor Architektur aufzuteilen sind.

Tabelle 3.1.: Weltweite: Hersteller Server Warenlieferung Q2 2010 (Stück)

Hersteller	Q2 10 **Stückzahl**	Q2 10 **Markt-Anteil**	Q2 09 **Stückzahl**	Q2 09 **Markt-Anteil**	Q2 09 - Q2 10 **Wachstum**
HP	644’172	30,0 %	522’447	31,0 %	23,3 %
Dell	542’799	25,3 %	402’187	23,8 %	35,0 %
IBM	267’614	12,5 %	226’570	13,4 %	18,1 %
Fujitsu	60’974	2,8 %	48’819	2,9 %	24,9 %
Sun	47’968	2,2 %	64’810	3,8 %	-26,0 %
Andere Hersteller	581’512	27,1 %	422’371	37,7 %	27,1 %
Total	2’145’039	100,0 %	1’687’204	100,0 %	27,1 %

Tabelle 3.2.: Weltweite: Hersteller Server Warenlieferung Q2 2009 (Stück)

Hersteller	Q2 09 **Stückzahl**	Q2 09 **Markt-Anteil**	Q2 08 **Stückzahl**	Q2 08 **Markt-Anteil**	Q2 08 - Q2 09 **Wachstum**
HP	522’447	31,0 %	706’724	30,2 %	-26,1 %
Dell	402’187	23,9 %	577’163	24,7 %	-30,3 %
IBM	226’570	13,4 %	308’835	13,2 %	-26,6 %
Sun	63’412	3,8 %	96’510	4,1 %	-26,6 %
Fujitsu	48’819	2,9 %	61’077	2,6 %	-20,1 %
Andere Hersteller	422’371	25,1 %	590’468	25,2 %	-28,5 %
Total	1’685’806	100,0 %	1’687’204	100,0 %	-28,0 %

Tabelle 3.3.: EMEA: Server Hersteller Warenlieferung Q2 2010 (Stück)

Hersteller	Q2 10 **Stückzahl**	Q2 10 **Markt-Anteil**	Q2 09 **Stückzahl**	Q2 09 **Markt-Anteil**	Q2 09 - Q2 10 **Wachstum**
HP	241'898	41,5 %	200'421	40,6 %	20,7 %
Dell	111'328	19,1 %	90'273	18,3 %	23,3 %
IBM	63'701	10,9 %	62'359	12,6 %	2,2 %
Fujitsu	36'575	6,3 %	31'207	6,3 %	17,2 %
Sun	15'473	2,7 %	24'053	4,9 %	-35,7 %
Andere Hersteller	114'558	19,6 %	84'743	17,2 %	35,2 %
Total	583'533	100,0 %	492'056	100,0 %	18,4 %

Tabelle 3.4.: EMEA: Hersteller Server Warenlieferung Q2 2009 (Stück)

Hersteller	Q2 09 **Stückzahl**	Q2 09 **Markt-Anteil**	Q2 08 **Stückzahl**	Q2 08 **Markt-Anteil**	Q2 08 - Q2 09 **Wachstum**
HP	200'421	40,8 %	293'621	40,6 %	-31,7 %
Dell	90'273	18,4 %	138'000	19,1 %	-34,6 %
IBM	62'359	12,7 %	95'861	13,3 %	-34,9 %
Fujitsu	31'207	6,3 %	40'775	5,6 %	-23,5 %
Sun	22'655	4,6 %	30'584	4,2 %	-25,9 %
Andere Hersteller	84'743	17,2 %	123'561	17,1 %	-31,4 %
Total	491'658	100,0 %	722'402	100,0 %	-31.9 %

Der Betriebsystemhersteller Microsoft hat angekündigt, nach Windows 2008 R2 keine Software mehr für Itanium Prozessoren zu entwickeln. Ferner verzichten auf den Support von Itanium in ihren zukünftigen Softwareversionen die beiden Linux Distrbutoren RedHat (RedHat Enterprise Linux) und Canonical (Ubuntu Linux). Canonical wird zusätzlich den Sparc Support einstellen. Diese Hersteller sind zwar bei den Installationen auf den genannten Prozessoren eher Nischenanbieter, jedoch un-

terstützt dies die These, dass die Hersteller künftig ihre Produktstrategie nicht mehr auf diese Prozessoren richten.

3.2.2.3. Konkurrenzprodukt

Der internationale Softwarekonzern Symantec Corporation mit Hauptsitz in Cupertino Kalifornien, hat mit der Veritas Storage Foundation Suite ein umfangreiches Portfolio für Volume Manager, File System und Storage Manager im Angebot. Mit dem Produkt Veritas Storage Foundation Manager hat Symantec eine vergleichbare Lösung wie die geplante Volume Managment Engine. Mit dem Veritas Storage Foundation Manager lässt sich zentral über ein Web-GUI die Server Volumes verwalten. Der Veritas Storage Foundation Manager unterstützt jedoch nur ihren eigenen Volume Manager. Deshalb kann der Betriebsystem eigene Volume Manager nicht verwendet werden. Folgende Betriebsysteme werden unterstützt: Oracle Solaris, HP HP-UX, IBM AIX, Linux und Microsoft Windows.

RedHat bietet mit der Opensource Applikation Conga, eine webbasierende Linux Cluster und LVM Verwaltungslösung an. Die Applikation ist jedoch auf das Verwalten der Konfiguration beschränkt.

4. Konzept Client basierte Volume Manager Engine

4.1. Spezifikation

4.1.1. Objekte

4.1.1.1. Server

Im folgenden wird der Volume Manager Verwaltungsserver als “Server“ bezeichnet, welche den Client und deren Volume Manager verwaltet. Auf dem Server wird die eigentliche Logik und die Daten verwaltet. Die Daten der Disk und Volumes der einzelnen Clients werden auf dem Server in einer Datenbank gespeichert. Beim Anlegen eines neuen Volumes, werden die Informationen aus der Datenbank verwendet.

4.1.1.2. Client

Als Client wird der zu verwaltende Server bezeichnet, auf welchem die Disks, die Volumes und das Filessystem verwaltet und konfiguriert werden sollen. Auf dem Client ist das Betriebsystem Linux mit dem Volume Manager LVM installiert. Für die Verwaltung des Client mittels Volume Manager Engine wird vorausgesetzt, dass der Client mit dem Server über TCP/IP kommunizieren kann. Mehrere Server zusammen können einen Cluster mit gemeinsamen Disk und Volumes bilden. Die einzelnen Servern können in verschiedenen Rechenzentren stehen.

4.1.1.3. Speicher/Strorage

Der Speicher kann eine im Server eingebaute Festplatte sein, oder der Speicher eines Storage-System, welches über eine Storage Area Network (SAN) als LUN zugeordnet ist. Für eine Mirrored Volume ist es wichtig zu wissen, aus welchem Rechenzentrum das LUN stammt. Aus diesen Grund wird dem Speicher ein Ort bzw. ein Rechenzentrum zugeordnet

4.1.1.4. Speichernetzt/Storage Area Network

Über das Storage Area Network (SAN) wir der Server mit dem Speichersystem verbunden.

4.1.1.5. Ethernet Netzwerk

Über das Ethernet Netzwerk kommuniziert der Server und die Clients miteinander.

4.1.1.6. Standort/Site

Als “Site“ bezeichnen wir das Rechenzentrum.

4.1.1.7. Dateisystem/Filesystem

Auf dem Logischen Volume wird ein Dateisystem erstellt, in welches die Daten gespeichert werden.

4.1.1.8. Mountpoint

Der Mountpoint ist der Pfad in welchem das Dateisystem eingehängt wird.

4.1.1.9. Browser

Der Webbrowser wird für den Dialog mit dem Anwender und für die Interaktion mit dem Volume Manager System verwendet.

4.1.2. Annahmen

Grundsätzlich ist es technisch möglich, Befehle für das Erstellen, Ändern und Löschen eines Clusters, einer Volume Group, eines Disk etc. direkt über das LVM oder via des neuen Tools vom Benutzer auszuführen. Für diese Arbeit wird angenommen, dass jedoch sämtliche Befehle durch das neue Tool erfasst werden, damit das Log-File lückenlos alle Befehle speichert und ausschliesslich mit den Daten aus der zentralen Datenbank für die anstehenden Mutationen gearbeitet wird. Würde diese Anforderung nicht erfüllt, so müsste angenommen werden, dass die Informationen zum System in der Datenbank nicht aktuell sind. Dann müsste vor jeder Mutationen

das System, auf welchen die Mutation ausgeführt werden soll, neu eingelesen werden. Der Einlesevorgang kann bei Servern mit hoher Diskauslastung lange dauern und sich für den Benutzer als inakzeptabel auswirkt.

4.1.3. Funktionen

4.1.3.1. Rollen und Benutzer

Die Web-Applikation soll eine integrierte Benutzer- und Rollen-Verwaltung enthalten. Um die Web-Applikation benützen zu können, muss man sich mit seinem persönlichen Benutzername und Passwort anmelden.

Beim Erfassen eines Benutzer muss dem Benutzer eine Rolle zugewiesen werden. Damit wird die Funktionsberechtigungen auf einfache Weise gesteuert. Folgende Rollen werden von der Anwendung vorgeschlagen:

Admin

- Kann Benutzer erfassen und verwalten
- Kann neue Server hinzufügen und abbauen
- Kann neue Cluster hinzufügen und abbauen
- Kann neue Volume Objekte auf einem Server/Cluster erstellen
- Kann bestehende Volume Objekte auf einem Server/Cluster bearbeiten

System-Admin-Teamleader

- Kann neue Server hinzufügen und abbauen
- Kann neue Cluster hinzufügen und abbauen
- Kann neue Volume Objekte auf einem Server/Cluster erstellen
- Kann bestehende Volume Objekte auf einem Server/Cluster bearbeiten
- Kann nachvollziehen wer welche Mutationen durchgeführt hat

System-Admin

- Kann neue Server hinzufügen und abbauen
- Kann neue Cluster hinzufügen und abbauen
- Kann neue Volume Objekte auf einem Server/Cluster erstellen
- Kann bestehende Volume Objekte auf einem Server/Cluster bearbeiten

System-Operator

- Kann neue Volume Objekte auf einem Server/Cluster erstellen
- Kann bestehende Volume Objekte auf einen Server/Cluster bearbeiten

System-Operator-Teamleader

- Kann neue Volume Objekte auf einen Server/Cluster erstellen
- Kann bestehende Volume Objekte auf einem Server/Cluster bearbeiten
- Kann nachvollziehen wer welche Mutationen durchgeführt hat

Weitere Rollen aus der Anforderungsanalyse sollen zu einen späteren Zeitpunkt umgesetzt werden.

4.1.3.2. Erfassen eines Clusters

Neue Clusters sollen über ein Formular auf dem Webinterface erfasst werden können. Beim Erfassen eines neuen Clusters, werden folgende Angaben benötigt.

- Clustername

4.1.3.3. Erfassen eines Servers

Neue Server sollen über eine Formular auf dem Webinterface erfasst werden können. Beim Erfassen eines neuen Servers, benötigt es zwingend folgende Angaben:

- Servername
- IP-Adresse

Optional kann der Server einem Cluster als Mitglied zugeordnet werden. Sollte der Cluster noch nicht erfasst sein, soll es möglich sein an dieser Stelle diesen zu erfassen:

Nach dem Abspeichern der Serverinformationen, soll sich die Applikation mit dem Server verbinden, um das Betriebsystem, den installierten Volume Manager und alle vorhandenen LVM Objekte auszulesen.

4.1.3.4. Einlesen der LVM Objekte

Bei den Disks werden die folgenden Attribute eingelesen:

- Name
- Grösse

Bei den LUN werden folgende Attribute eingelesen:

- Name
- Grösse
- WWN

Bei Physical Volume werden die folgenden Attribute eingelesen:

- Name
- UUID
- Grösse
- Freier Speicher
- Verwendeter Speicher
- Anzahl Physical Volume Extents
- Anzahl verwendeter Physical Volume Extents
- Physical Volume Status
- Volume Group UUID

Bei Volume Group werden die folgenden Attribute eingelesen:

- Name
- UUID
- Grösse
- Freier Speicher
- Verwendeter Speicher
- Anzahl Volume Group Extents

- Grösse der Volume Group Extents
- Maximale Anzahl erlaubte Physical Volume
- Maximale Anzahl erlaubte Logical Volume
- Stauts der Volume Group

Bei Logical Volume werden die folgenden Attribute eingelesen:

- Name
- UUID
- Grösse
- Anzahl Extens
- Berechtigung
- Mirrorstatus
- Volume Group UUID

4.1.3.5. Ausführen von LVM Mutationen

Bei Mutationen an einem Clustermietglied soll die Konfiguration, wenn gewünscht, an allen Clustermitglieder vorgenommen werden

4.1.3.6. Physical Volume erstellen

Aus den zugeordneten leeren Volumes eines Servers bzw. Clusters sollen neue Physical Volumes über das Webinterface erstellt werden können.

4.1.3.7. Volume Groups erstellen

Aus einem oder mehreren freien Physical Volumes eines Servers bzw. Clusters soll über das Webinterface eine neue Volume Groupe erstellt werden können. Beim erstellen von Volume Groups soll zwischen Server Volume Group, einfachen Volume Group und hochverfügbaren Volume Group unterschieden werden können.

Bei Server Volume Groups können nur lokale Physical Volumes verwendet werden.

Bei einfachen Volume Groups können jeweils einzelne Physical Volumes unabhängig vom Standort verwendet werden. Der Benutzer soll beim Erstellen der Volume Groups erkennen können, aus welchem Standort die Physical Volumes stammen.

Bei hochverfügbaren Volume Groups können jeweils nur Physical Volumes in Paaren zugeordnet werden, wobei jeweils innerhalb eines Paares die Physical Volumes aus unterschiedlichen Standorten stammen müssen.

Einfache und hochverfügbare Volume Groups können clustered oder nicht clustered Volume Groups sein. Server Volume Groups können nur “nicht Clustered Volume Groups“ sein.

4.1.3.8. Volume Group erweitern

Bestehenden Volume Groups sollen weitere Physical Volumes nach dem Schema zugeordnet werden können.

4.1.3.9. Volume Group reduzieren

Nicht mehr benötigter Speicherplatz soll bei Volume Groups mit dem Abbau von Physical Volumes wieder freigegeben werden können.

4.1.3.10. Logical Volume erstellen

Aus freien Extents einer Volume Group sollen über das Webinterface Logical Volumes erstellt werden können.

4.1.3.11. Logical Volume Mirrored erstellen

Aus freien Extents einer hochverfügbaren Volume Group sollen über das Webinterface Logical Volumes, welche gespiegelt sind, erstellt werden können.

4.1.3.12. Logical Volume erweitern

Bestehende Logical Volumes sollen durch Zuordnen von weiteren freien Extents aus der Volume Group erweitert werden können.

4.1.3.13. Logical Volume reduzieren

Bestehende Logical Volumes sollen durch entfernen von Extents verkleinert werden können.

4.1.3.14. Report über alle freien Disks

Mit einem Report über alle Servers bzw. Clusters sollen ungenutzte freie Disks angezeigt werden

4.1.3.15. Report über alle freien Physical Volume

Mit einem Report über alle Servers bzw. Clusters sollen ungenutzte freie Physical Volumes angezeigt werden

4.1.3.16. Report durchschnittliche freie Extents pro Volume Group

Mit einem Report über alle Servers bzw. Clusters soll der Durchschnitt aller freien Extents pro Volume Group angezeigt werden.

4.1.3.17. Report über Mirrored Volumes

Mit einem Report sollen alle ungespiegelte Logical Volumes ausgegeben werden.

4.1.3.18. Report über alle nicht Multipath Luns

Mit einem Report sollen alle Luns ausgegeben werden, welche nicht über zwei SAN-Verbindungen am Server angehängt sind.

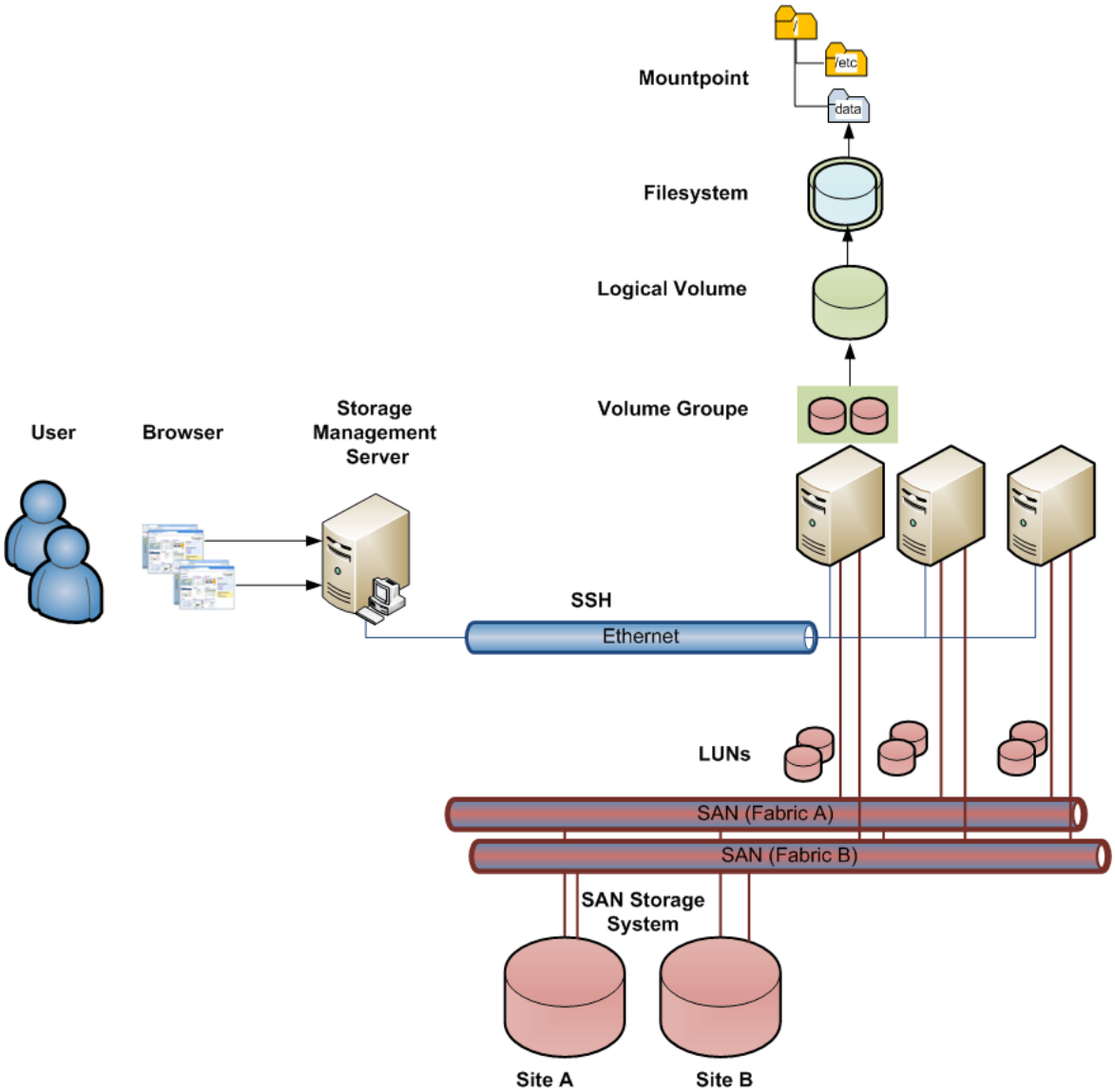


Abbildung 4.1.: Konzept: LVM Top-Level Architektur

4.2. Architektur

4.2.1. Programmiersprache

Ruby ist eine interpretierte und objektorientierte Programmiersprache und beinhaltet einige bewährte Prinzipien wie z.B. “DuckTyping“ und “Principle of Least Suprice“. Die Entwickler von Ruby stellen sich selber den Anspruch eine Programmiersprache zu schaffen, die durch Ihre Natürlichkeit einfach erlernbar ist und es den Programmierern ermöglicht, einfachen und übersichtlichen Code zu schreiben, welcher aber nicht seine Mächtigkeit und innere Komplexität verliert. Ruby hat sich in den letzten Jahren von einer kaum beachteten Programmiersprache zu einem Publikums-Magneten entwickelt. Es gibt eine stetig wachsende offene Community “Gemeinschaft“, welche sich und die Sprache durch Austausch von Erfahrungen und Ideen weiterbringen möchte. Ein Grund für die hohe Bereitschaft der Community die Sprache Ruby weiter zu bringen ist der Umstand, dass die Programmiersprache vollständig OpenSource ist und unter der Lizenz der Ruby-License und GPL steht. Zudem ist die Sprache fast beliebig erweiterbar und bestehende Funktionen können einfach durch eigene Funktionen ausgetauscht werden. Neben den oben genannten Vorteilen, gibt es sicherlich auch diverse Eigenschaften, welche im Vergleich zu anderen Sprachen gegen Ruby sprechen würden. Der hauptsächliche Grund die Sprache Ruby in diesem Projekt einzusetzen, ist der Entscheid des aktuellen Arbeitgebers, bei möglichst allen neuen Projekten die Sprache Ruby einzusetzen.

4.2.2. Web-Framework

Das auf Ruby basierende Web-Framework Ruby on Rails auch “Rails“ genannt, ist eines der beliebtesten Web-Framework für klein- bis mittelgrosse Anwendungen. Ruby on Rails wurde vom Dänen David Heinemeier Hannson geschrieben und gilt als Vorbild für weitere Web Applikations-Framework wie zB. Grails (Java), CakePHP (PHP), Django (Python) und “softies on rails“ (.Net), welche die Phylosophie von Rails übernommen haben. Wie die Programmiersprache Ruby ist auch das Framework OpenSource, steht jedoch unter der MIT License.

Alle Rails Applikationen unterliegen der gleichen Basisarchitektur. Die sich daraus ergebenden Vorteile liegen auf der Hand. Neue erfahrene Entwickler können sich rasch in ein neues Projekt einarbeiten, haben keine Mühe den bestehenden Code schnell zu erfassen und im Sinne der anderen Programmierer den Code weiterzuentwickeln.

Rails Applikationen sind nach dem Design Pattern Model-View-Control (MVC) in drei Schichten unterteilt, die nachfolgend kurz besprochen werden sollen.

Model-Schicht Die Model-Schicht entkoppelt die Daten von der Businesslogik zum Manipulieren von Daten.

View-Schicht Die View-Schicht, auch Präsentationsschicht, stellt das User-Interface der Applikation dar.

Controler-Schicht Die Controller-Schicht ist verantwortlich, die Daten von der Benutzereingabe und externer Input zu interpretieren, indem sie mit dem beiden Schichten View und Model kommunizieren.

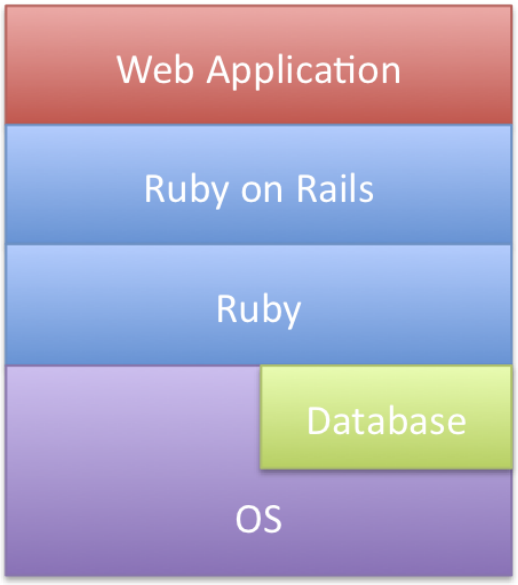


Abbildung 4.2.: Rails Architektur: Applikation Stack

Ruby On Rails ist nach den Architektur Stilen als Independent Components und Call-and-Return aufgebaut.

Als Independent Components besteht die Architektur aus unabhängigen Komponenten, welche wiederum aus unabhängig ablaufenden Elementen bestehen, die über Nachrichten miteinander interagieren.

Bei Call-and-Return bzw. dessen Implementierung “Layered“ werden sämtliche Elemente einzelnen Schichten zugeordnet, die hierarchisch geordnet sind. Jeder dieser Schichten erfüllt eine klar definierte Aufgabe.

Die Basis-Architektur wiederspiegelt sich auch in der Verzeichnisstruktur, welche bei allen Rails Applikationen gleich aufgebaut ist.

- **app:** Hier befindet sich die eigentliche Rails Applikation in der MVC Struktur
- **conf:** Applikation Konfigurationsdateien
- **db:** Datenbank Schema und Migrations Dateien
- **doc:** Dokumentation der Applikation
- **lib:** Applicationsspezifischer Code, welcher nicht Teil des MVC Code ist
- **log:** Applications Log
- **public:** JavaScript, CSS, Bilder und andere statische Inhalte
- **script:** Rails Scripts
- **test:** Unit-test Code
- **tmp:** Cache und Session Informationen
- **vender:** Verwendete Fremd-Plugins

Neben der einheitlichen Dateistruktur und dem MVC Pattern, haben die Entwickler von Rails das Framework nach dem Design Pattern "Don't Repeat Yourself" (DRY) und "Convention over configuration" entwickelt.

Die "Don't Repeat Yourself" Phylosophie geht davon aus, dass man sich bei der Entwicklung möglich nicht wiederholen soll und stattdessen einen generischen wiederverwendbaren Code entwickeln soll (projektoreintierte Entwicklung). Bei Rails wird die Phylosophie auf das ganze Framework angewendet, der Entwickler soll eine Definition nur an einem Ort vornehmen, ohne sich im Quelltext, in der Konfiguration oder in der Dokumentation wiederholen zu müssen. Durch diese vollständige Umsetzung von DRY erspart sich der Entwickler haufenweise Arbeit und verringert gleichzeitig die Fehleranfälligkeit seiner Applikation. Bei einer nachträglichen Anpassung z.B. eines Objekts, Funktion oder Konfiguration muss der Entwickler diese nur an einer Stelle anpassen. Ist der angepasste Quellcode genügend mit Unit-Test abgedeckt, muss der Entwickler bei dieser Anpassung nicht befürchten, dass er neue Fehler in die Applikation eingebaut haben könnte. Natürlich muss man bei diesem stark objektierten Konzeptansatz sich im klaren sein, dass in der Praxis in der Instanz viele Objekte entstehen können, die zwar ähnlich zu einem anderen Objekt sind, aber in dieser Ausprägung nicht weiter vererbt werden kann.

Eine weitverbreitete Eigenschaft von einem viel eingesetzten Applikations-Framework ist es, dass für die Verwendung des Frameworks grosse zum Teil komplexe Konfigurationen vorgenommen werden müssen. Mit der Philosophie "Convention over configuration" wollten die Rails Entwickler den Anwendern ihres Frameworks diese

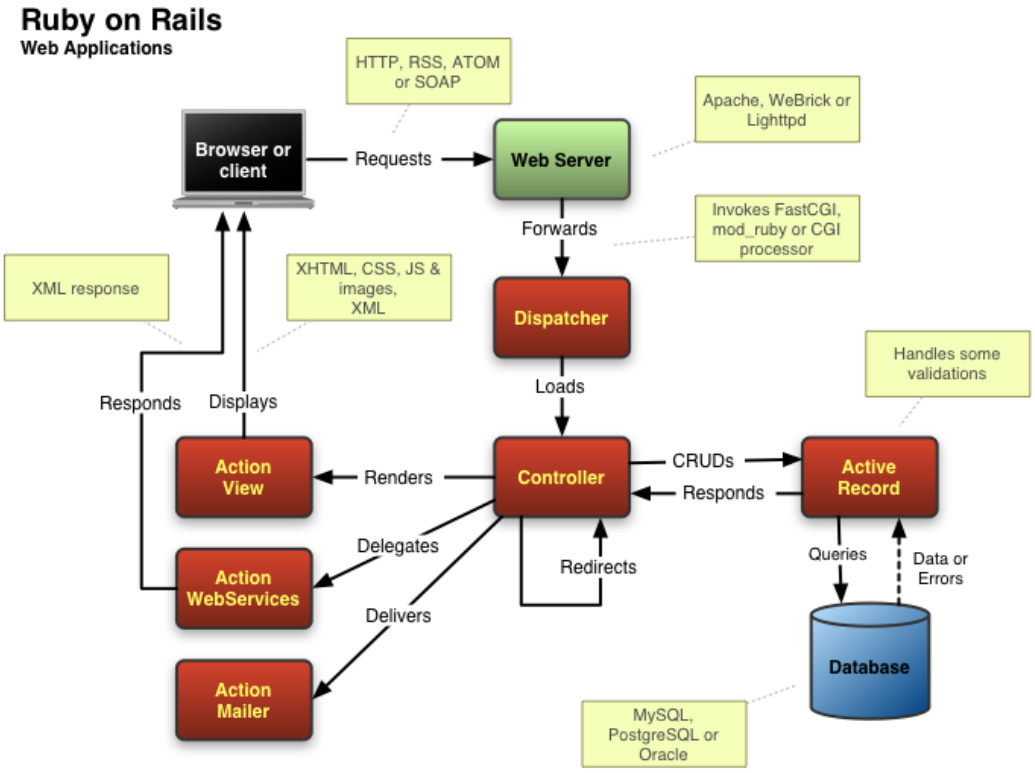


Abbildung 4.3.: Architektur: Ruby On Rails [http://blog.roi-office.com]

Arbeit ersparen. So verhält sich Rails in den meisten Fällen so, wie man es erwarten würde, ohne eine Spezifikation in einer Konfigurationsdatei vornehmen zu müssen. So weiss z.B. der Routing Mechanismus von Rails ohne eine Konfiguration, welche Klasse und Methode den Seitenaufruf abhandeln muss. Wenn notwendig und gewünscht können diese Standardverhalten einfach überschrieben werden und geben dem Entwickler somit genügend Flexibilität.

4.2.3. Datenmodell

Das Datenmodell soll nach Möglichkeit die reale Welt abbilden, um die gegebenen Anforderungen und die Eigenschaften der Objekte im Datenmodell zu erfüllen. Es wurden folgende Überlegungen gemacht:

- Ein Cluster besteht aus mindestens zwei oder mehr Servern.
- Ein Server kann Mitglied eines Clusters sein, oder stand-a-lone betrieben werden.

- Für die Hochverfügbarkeit eines Clusters ist es wichtig zu wissen, in welcher Site ein Server steht. Aus diesen Grund ist ein Server immer einer Site zugeordnet.

- Ein Server kann ein oder mehrere Volumemanger installiert haben.
- Ein Server hat ein oder mehrere Disks, wobei es sich um Serverdisks oder LUN Disks handeln kann.
- Ein Server kann Mitglied eines Clusters sein.
- Ein Server hat ein oder mehrere Volumemanger
- Ein Server befindet sich in einem Rechenzentrum/Site.
- Ein Server hat ein oder mehre Disks.

- Eine Disk kann ein Serverdisk oder LUN-Disk sein.
- Eine Serverdisk ist immer einem Server zugeteilt.
- Eine LUN-Disk kann ein oder mehreren Servern zugeteilt sein.
- Eine Disk wird nicht in Partitionen aufgeteilt.
- Eine Disk kann für einen Volume-Manager verwendet werden.
- Der Volumemanager bestimmt die dazugehörigen Objekte für die Disk.

- Ein Physical Volume gehört immer einer Disk an.
- Ein Physical Volume kann einer Volume Group zugeordnet werden.

- Eine Volume Group besteht immer aus einer oder mehreren Physical Volumes.
- Eine Voume Group kann eine oder mehrere Logical Volume Groups haben.

- Ein Logical Volume gehört immer einem Volume Group an.
- Ein Logical Volume hat eine Filesystem.

- Ein Filesystem gehört immer einem Logical Volume an.

- Ein Filesystem hat ein Mountpoint.

Das im Abbild 4.4 dargestellte Datenmodell, zeigt die Objekte und deren Verbindungen zueinander, zusätzlich enthält es eine mögliche Erweiterung für den Oracle ZFS Volume Manager.

Abbildung 4.4.: Architektur: Datenmodell

4.2.4. LVM-Engine

Die LVM-Engine inklusive Client-Server Kommunikation wird als Libary-Module für das Web-Framework umgesetzt. Somit ist die Logik der LVM-Engine vom Web-Framework gelöst und kann für andere Projekte wieder verwendet werden.

4.2.4.1. ruby-lvm-wrapper

Das von den Entwicklern Matthew Kent und John Hampton auf github veröffentlichte Projekt ruby-lvm-wrapper[15] und dessen dazu gehörende Subprojekt ruby-lvm-attributes footnote[16], bietet eine gute Ausgangslage für die LVM-Engine.

Der LVM-Wrapper des Projekts liest alle Logical Volumes, Logical Volume Segments , Volume Groups, Physical Volumes und Physical Segments aus einem lokalen Server aus und gibt diese als Ruby Objekt zurück. Die Attribute der einzelnen LVM Objekten sind pro Objekt im YAML Format definiert und in das separate Subprojekt ruby-lvm-attributes ausgelagert. Somit ist die Definition der Objekte vom eigentlichen Quellcode entkoppelt. Für jedes Attribut wird in der YAML Datei eine *:methode* mit dem LVM Attribute Namen, *:type_hint* mit dem Datentyp des Atttribues, *:column* Name des Attributes im RubyObjekt und *:description* mit der Beschreibung definiert. Ein Beispiel einer Definition wird in dem Listing *4.1 YAML: Beispiel Code aus `PVS`* gezeigt. Von der LVM Version abhängig wird im ruby-lvm-attributes Projekt ein Gruppe von YAML Dateien für die Objekte erstellt.

Listing 4.1: YAML: Beispiel Code aus `PVS`

```
1 - :method: free
2   :type_hint: Integer
3   :column: pv_free
4   :description: Total amount of unallocated space in current units.
```

Für die Ausführung und das Auslesen der Ausgabe-Kanäle (englisch: streams), verwendet der Wrapper das API POpen4[17] von John-Mason und P. Shackelfords. POpen4 erstellt für die Ausführung der einzelnen Befehlen einen childprocess. POpen4 liest dabei die von POSIX Stream stdin, stdout und stderr aus.

- **stdin** ist der Eingabe-Kanal, meistens mit der Tastatur verbunden.

[15] https://github.com/johnhampton/ruby-lvm
[16] http://rubyforge.org/projects/ruby-lvm-attrib/
[17] https://github.com/pka/popen4

- **stdout** ist der Ausgabe-Kanal einer Shell und wird normalerweise auf dem Terminal ausgegeben.
- **stderr** ist der Fehler-Ausgabe-Kanal. Die Fehler werden standardmässig auf dem gleichen Gerät wie stdout ausgegeben.

4.2.4.2. Anpassungen am ruby-lvm-wrapper

Damit das Projekt ruby-lvm-wrapper die Anforderungen der LVM-Engine erfüllt, muss das Projekt angepasst und erweitert werden. Das Projekt wurde deshalb in einer Forke Version vom ursprünglichen Projekt abgespaltet.

Im ursprünglichen Ruby-lvm-wrapper Projekt, gibt es keinen Wrapper für die LVM Funktionen für das Erstellen und Modifizieren von LVM Objekten. Manipulationen sind zwar indirekt über die RAW Schnittstelle möglich, indem man die LVM Befehle der Schnittstelle übergibt. Dies bedeutet aber, dass die Logik ausserhalb der Ruby-lvm-wrapper zu implementieren ist. Die RAW-Schnittstelle wird deshalb im Forke durch weitere Wrappers für das Erstellen von Physical Volumes, Volume Groups und Logical Volumes ersetzt.

Das Auslesen von LVM Objekten und das Ausführen von Commandline Befehlen ist im ursprünglichen Ruby-lvm-wrapper Projekt auf den lokalen Rechner beschränkt. Die Ausführung im Fork wird mit einer Client-Server Software ersetzt bzw. ergänzt.

Ruby-lvm-wrapper prüft das System bei der Initialisierung auf die LVM Version des Systems und verwendet dazu je Version die YAML Dateien von Projekt ruby-lvm-attrib für das Auslesen der Objekt-Attribute. Weil pro Version die YAML Dateien vorhanden sein müssen, birgt dies die folgenden Nachteile:

- Der Verwaltungsaufwand steigt mit der Anzahl der Versionen.
- Die LVM-Engine könnte nach einer Aktualisierung eines Systems mit Patches, nicht mehr kompatibel (läuffähig) sein, wenn die Version von LVM wechselt.

Aus diesen Grund wird im Fork nur die im Datenmodell verwendeten Objekt-Attribute ausgelesen. Zusätzlich wird das Projekt ruby-lvm-attrib in den Fork integriert, was den Vorteil bringt, nicht mehr zwei Projekte parallel pflegen zu müssen.

Die Abbildung *4.5 Architektur: Klassendiagramm LVM Wrapper* zeigt das Klassendigramm nach der Anpassung.

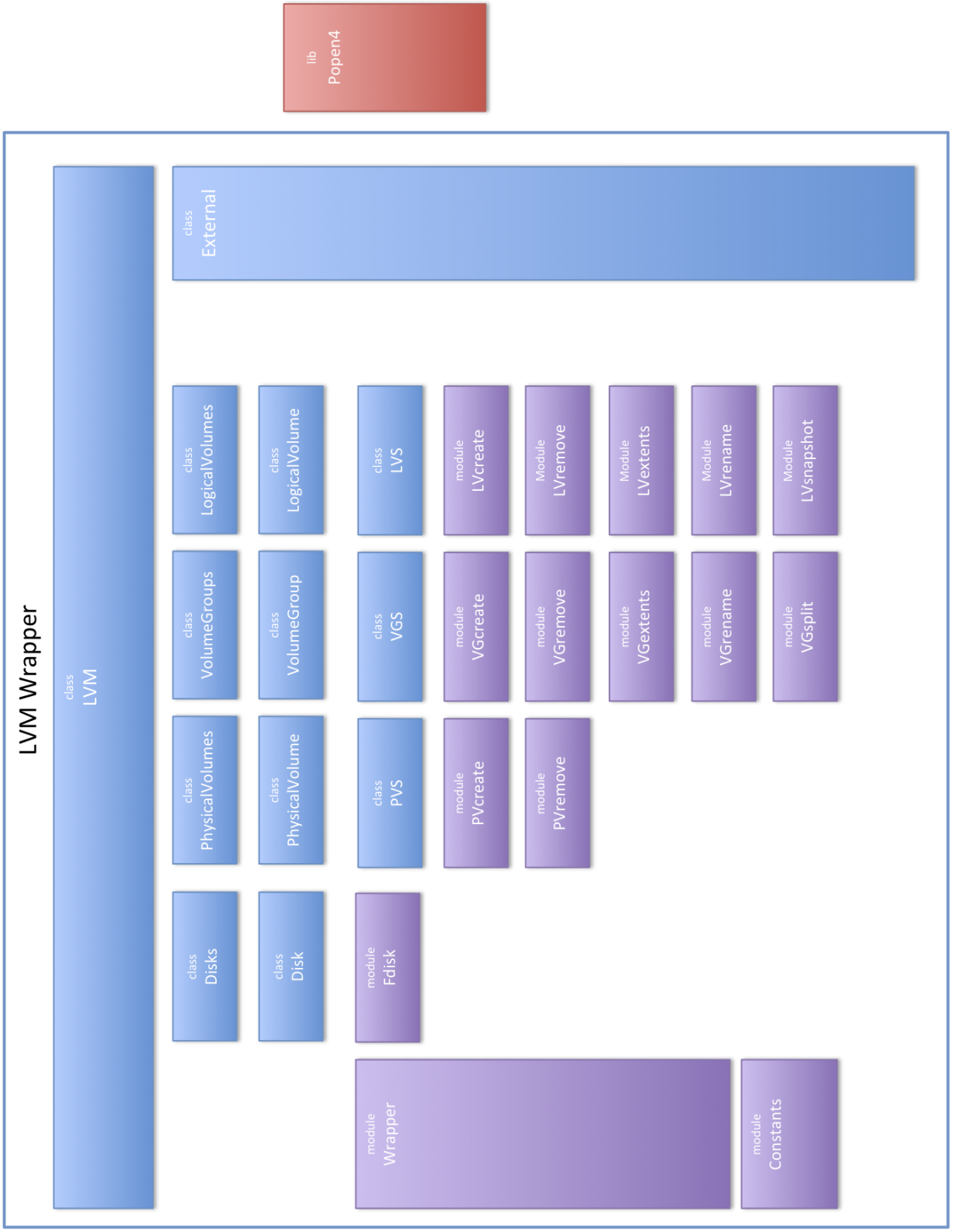


Abbildung 4.5.: Architektur: Klassendiagramm LVM Wrapper

4.2.5. Objekte Auslesen

4.2.5.1. Lokale Disk auslesen

Die SCSI Disks, welche vom Linux-Kernel Ring erkennt werden, können mit *dmesg* ausgelesen werden.

Befehl:

```
dmesg | grep "SCSI device" | uniq
```

Ausgabe Beispiel:

```
SCSI device sda: 16777216 512-byte hdwr sectors (8590 MB)
```

Aus der Ausgabe lässt sich der Gerätename (sda), die Sektorgrösse (512-byte), die Sektoren Anzahl (16777216) auslesen. Mit diesen Angaben lässt sich die Diskgrösse in byte berechnen.

$$GrösseInByte = Sektorgrösse * AnzahlSektoren \tag{4.1}$$

4.2.5.2. Physical Volumes auslesen

LVM bietet die Möglichkeit, angepasste Reports der LVM Objekte auszugeben. Somit kann die Ausgabe optimal für das automatische Auslesen per Parser vorbereitet werden.

Befehl:

```
pvs --separator=";" --noheadings --nosuffix --units=b --unbuffered
--options pv_name,pv_uuid,pv_size,pv_used,pv_pe_count,
pv_pe_alloc_count,pv_attr,vg_uuid
```

Ausgabe Beispiel:

```
/dev/sdb;hAL4Ok-vah7-V682-eQqh-n8Wj-geMA-JJKyqV;
213909504;0;51;0;a-;MTzZmV-yQua-cycm-msiv-fUM4-5718-wbUp1u
```

Das Zeichen zwischen den einzelnen Attributen kann mit der Option –separator bestimmt werden. Bei der Ausgabe wird standardmässig eine Beschriftungzeile ausgegeben. Diese kann mit der Option –noheadings übersteuert werden. Die Einheit der Ausgabe kann mit der Option –units bestimmt werden. Für die Engine wird die Einheitsgrösse Bytes gewählt, somit werden keine kommaseparierten Zahlen ausgegeben.

4.2.5.3. Volume Groups auslesen

Befehl:

```
vgs --separator=; --noheadings --nosuffix --units=b --unbuffered
--options vg_name,vg_uuid,vg_size,vg_free,vg_extent_count,
vg_extent_size,pv_max,lv_max,lv_attr,vg_uuid
```

Ausgabe Beispiel:

```
test;MTzZmV-yQua-cycm-msiv-fUM4-5718-wbUp1u;
213909504;213909504;51;4194304;0;0;wz--n-
```

4.2.5.4. Logical Volumes auslesen

```
lvs --separator=";" --noheadings --nosuffix --units=b --unbuffered
--options lv_name,lv_uuid,lv_size, pv_used,pv_pe_count,
pv_pe_alloc_count,pv_attr, vg_uuid
```

4.2.6. Client-Server

Die Client Server Komponente wird mittels Secure Shell (SSH) realisiert. Diese hat gegenüber einem proprietären Socket - oder einer XMLRPC Lösung den Vorteil, dass auf dem Client keine zusätzliche Software-Installation notwendig ist, die verteilt und später gewartet werden muss. Neue Anpassungen an der Software müssen nur auf einem zentralen Server installiert werden und vereinfacht somit den Versionenwechsel.

SSH gehört zu den Internet Standard Protokoll und ist unter den Request for Comments (RFC) 4250[18], 4251[19], 4252[20], 4253[21] und 4254[22] standardisiert. Der SSH Server ist bei den wichtigen Linux Distributionen im Enterprice Umfeld standardmässig installiert und aktiviert. Mit SSH ist es möglich, auf einem entfernten Rechner bzw. Server, CLI befehle auszuführen, als würde man sich direkt am Terminal des Systems befinden. Durch die relative einfach und sichere Nutzung, wird SSH von vielen Konfigurations-Applikationen verwendet, um Server zu verwalten.

[18] http://www.ietf.org/rfc/rfc4250.txt
[19] http://www.ietf.org/rfc/rfc4251.txt
[20] http://www.ietf.org/rfc/rfc4252.txt
[21] http://www.ietf.org/rfc/rfc4253.txt
[22] http://www.ietf.org/rfc/rfc4254.txt

4.2.7. Kommunikation

Das Anwendungs-Protokoll SSH kommuniziert über TCP auf dem Standard Port 22. Für die Kommunikation zwischen Server und Client sind keine weiteren Ports mehr nötig. Da SSH oft für die Server-Administration verwendet wird, ist dieser Port auf vielen Firewall geöffnet. Somit sind meist keine langwierigen Firewall Change-Requests notwendig.

4.2.8. Sicherheit

Mit dem Einsatz von SSH als Client-Server Komponente, ist durch die symmetrische Verschlüsselung mittels TrippleDES, BlowFish oder AES die Kommunikation sichergestellt. Damit sind keine Manipulationen beim Übertragen der Befehle auf den Client möglich.

Für die Authentifizierung und den Schlüsselaustausch kann bzw. wird bei SSH das Public-Key Verfahren mittels RSA eingesetzt. Mit diesem Verfahren ist sichergestellt, dass sich gleichzeitig nur ein Benutzer bzw. eine Applikation am System anmelden kann, wenn dieser im Besitz des privaten Schlüssels ist.

Für die Ausführung von LVM Befehlen auf dem System sind hohe Berechtigungsrechte auf dem System notwendig. Würde ein Angreifer Zugriff auf den Server erhalten, könnte er auf allen Clients Befehle mit hohen Rechten ausführen. Durch den Einsatz von SUDO kann auf dem Client ein Benutzer mit niedrigen Rechten angelegt werden, welcher mit SUDO nur die LVM-Befehle mit hohen Rechten ausführen kann.

4.3. Testing

Ein ausgereiftes white-box Testing hat einen spürbaren Einfluss auf die Code-Qualität. Ferner bietet es dem Entwickler die Möglichkeit, schneller auf Anforderungsänderungen einzugehen. Der Entwickler hat bei einem Projekt, bei welchem der Code gut mit Units-Test abgedeckt ist, bei Anpassungen am bestehenden Code eine gute Leitlinie, welche ihm hilft, weitgehend fehlerfrei zu entwickeln. Für die Entwicklung des Projektes wird das in Ruby implementierte UnitTest Framework verwendet.

5. Prototyp LVM Engine

5.1. Ausgangslage

Mit dem Prototyp soll die Machbarkeit der Entwicklung einer LVM Engine aufgezeigt werden. Die LVM Engine ist der Teil der Anwendung, welche auf dem Clients die LVM-Befehle ausführt, um LVM Objekte einzulesen und anzulegen. Die LVM Engine baut auf dem Projekt ruby-lvm-wrapper auf. Dadurch sind die Funktionen für das Einlesen der LVM Objekte bereits umgesetzt.

5.2. SSH Client

Im ruby-lvm-wrapper wurden die LVM an die Funktion cmd des Module External übergeben. Für das Ausführen eines Befehles auf einem entfernten Rechner habe ich die Funktion cmd mit der Funktion Parameter Server ergänzt. Ursprünglich wollte ich für SSH die Ruby Libary NET::SSH verwenden. Für die Sicherheit auf dem Client sollten die LVM Befehle mittels SUDO auf dem System ausgeführt werden. Für das Ausführen von SUDO ist eine Shell notwendig. Die Libary hat diese jedoch nicht unterstützt, weshalb ich die LVM Befehle an den SSH Client des Systemes als Parameter übergeben habe.

Mit der Option **-q** (Quite Mode) wird verhindert, dass Login Banner Informationen des Clients nicht ausgegeben werden. Die Unterdrückung der Ausgabe hat den Vorteil, dass keine störende Zeichen oder Zeilen für das Parsen ausgegeben werden. Für SUDO wird mit der Option **-t** in doppelter Ausführung eine TTY alloziert. Der Loginname wird mit der Option **-l** (Login) und dem Benutzernamen definiert. Mit der Option **-i** und dem Pfad zum SSH Private-Key des Benutzers kann die PublicPrivate Key Authentifizierung von SSH verwendet werden.

Der Auszug der Klasse External, welche im Listing *5.1 LVM-Engine: Klass* `External` abgebildet ist, zeigt den ganzen SSH Befehl.

Für alle zu verwendeten Clients wird der gleiche SSH-Key und der gleiche SSH-Benutzer verwendet. Somit kann dieser vom System klar als Application-Benutzer

identifiziert werden und allenfalls automatisch über ein Konfiguration Management Tool angelegt werden. Einfachheitshalber kann man den SSH Benutzer und den Pfad zu dessen Private Key über eine AppConfig File im YAML Format für die Engine konfigurieren bzw. bereitstellen.

Listing 5.1: LVM-Engine: Klass `External`

```
require 'rubygems'
require 'open4'
require 'net_client'

module LVM
  module External
    class ExternalFailure < RuntimeError;
    end

    def cmd(server, cmd)

      if server != 'localhost'
        ssh_user = AppConfig.ssh_user
        ssh_key = AppConfig.ssh_key
        cmd = "ssh -q -t -t -l #{ssh_user} -i #{ssh_key} #{server} '#{cmd}'"
      end

    ...

  end # module External
end # module LVM
```

Zusätzlich zu den SSH Anpassungen wurden in der Klasse der Code für die pOpen4 Stream-Verarbeitung vereinfacht.

5.3. Physical Volume erstellen

Für das Erstellen von Physical Volumes habe ich einen Wrapper für den LVM-Befehl pvcreate als Module PVcreate gebaut. Das Module PVcreate stellt die Funktion pv_create zur Verfügung. Die Funktion pv_create verlangt eine Device Objekt als Parameter. Für das Device-Objekt wird geprüft, ob dessen Attribute "Partitionünd "Physical Volume Label"gesetzt sind. Ist dies nicht der Fall, wird anhand des Device Namen ein Physical Volume angelegt.

Listing 5.2: LVM-Engine: Module `PVCreate`

```
require 'lvm/wrapper'
```

```
2  require 'lvm/wrapper/pvs'
3
4  module LVM
5    module Wrapper
6      module PVCreate
7
8        # Create a LVM Physical Volume out of a device
9        # if disk has now partition and now physical volume label
10       def pv_create(device)
11         External.cmd(@server, "#{@command} pvcreate #{device.name}") if not device.
               partition || device.pv_label
12       end
13
14     end # module PVCreate
15   end # module Wrapper
16 end # module LVM
```

5.4. VolumeGroup erstellen

Das Module VGCreate bietet die beiden Funktionen vg_create und vg_create_cluster an. Beide Funktionen erstellen anhand eines oder mehrerer Physical Volume Objekte eine Volume Group mit dem Unterschied, dass bei der Funktion vg_create_cluster ein Clustered Volume Group erstellt wird. Das Physical Volume Object wird in einem Array zusammen mit dem gewünschten Volume Group Namen an die Funktionen übergeben. Die Funktionen prüfen alle übergebenen Physical Volume Objekte, ob diese bereits einer Volume Group zu geordnet sind. Ist diese nicht der Fall, wird die Volume Group mit dem gewünschten Namen erstellt.

Listing 5.3: LVM-Engine: Module `VGCreate`

```
1  require 'lvm/wrapper'
2  require 'lvm/wrapper/vgs'
3  require 'lvm/physical_volume_helper'
4
5  module LVM
6    module Wrapper
7      module VGCreate
8
9        include PhysicalVolumeHelper
10
11       # Create a Volume Group with Extents Size of 32 MB.
12       # if physical volumes are note used.
13       # See vor vgcreate command http://linux.die.net/man/8/vgcreate
14       # -c eq. --clustered
```

```
15        # -s eq. --physicalextentsize
16        def vg_create(volume_group_name, physical_volumes_array)
17          External.cmd(@server, "#{@command} vgcreate -s 32M #{volume_group_name} #{
                physical_volumes_to_s(physical_volumes_array)}") if physical_volumes_unused(
                physical_volumes_array)
18        end
19
20        # Create a Clustered Volume Group with Extents Size of 32 MB.
21        # if physical volumes are note used.
22        # See vor vgcreate command http://linux.die.net/man/8/vgcreate
23        # -c eq. --clustered
24        # -s eq. --physicalextentsize
25        def vg_create_cluster(volume_group_name, physical_volumes)
26          External.cmd(@server, "#{@command} vgcreate -s 32M -s #{volume_group_name}
                #{physical_volumes_to_s(physical_volumes_array)}") if
                physical_volumes_unused(physical_volumes_array)
27          #if physical_volumes_unused(physical_volumes)
28        end
29
30      end # module VGCreate
31    end # module Wrapper
32  end # module LVM
```

5.5. Logical Volume erstellen

Für das Erstellen eines Logical Volumes wurde ein Wrapper-Modul für den LVM-Befehl LVcreate erstellt. Das Module LVcreate bietet die Möglichkeit, gespiegelte und nicht-gespiegelte Logical Volumes auf einer Volume Group anzulegen. Dabei kann die Grösse jeweils in Extents oder in Bytes definiert werden. Den dafür benötigten vier Funktionen werden der Logical Volume Name, das Volume Group Objekt und die Grösse übergeben. Die Funktion prüft, ob für das Volume Objekt genügend Speicherplatz vorhanden ist, ein Logical Volume mit der gewünschten Grösse anzulegen.

Listing 5.4: LVM-Engine: Module `LVCreate`

```
1  require 'lvm/wrapper'
2  require 'lvm/wrapper/lvs'
3  require 'lvm/volume_group_helper'
4
5  module LVM
6    module Wrapper
7      module LVCreate
8
```

```
9       include VolumeGroupHelper
10
11      # Create a LVM Logical Volume.
12      # Set the size with extents.
13      # See vor lvcreate command http://linux.die.net/man/8/lvcreate
14      # -l eq. --extents
15      # -n eq. --name
16      def lv_create_extents(logical_volume_name, volume_group, extents)
17          External.cmd(@server, "#{@command} lvcreate -l #{extents} -n #{
                logical_volume_name} #{volume_group.name}") if
                volume_group_check_space_in_extents(volume_group,extents)
18      end
19
20      # Create a LVM Logical Volume.
21      # Set the size in kilobytes.
22      # See vor lvcreate command http://linux.die.net/man/8/lvcreate
23      # -l eq. --extents
24      # -n eq. --name
25      def lv_create_size_in_kb(logical_volume_name, volume_group, size_in_kb)
26          External.cmd(@server, "#{@command} lvcreate -l #{size_in_kb} -n #{
                logical_volume_name} #{volume_group.name}") if
                volume_group_check_space_in_kb(volume_group,size_in_kb)
27      end
28
29      # Create a LVM Logical Volume mirrored with regions size of 8 MB.
30      # Set the size with extents.
31      # See vor lvcreate command http://linux.die.net/man/8/lvcreate
32      # -l eq. --extents
33      # -n eq. --name
34      # -m1 eq. --mirrors
35      # -R eq. --regionsize
36      def lv_create_mirrored_extents(logical_volume_name, volume_group, extents)
37          External.cmd(@server, "#{@command} lvcreate -m1 -R 8 -s #{extents} -n #{
                logical_volume_name} #{volume_group.name}") if
                volume_group_check_space_in_extents(volume_group,extents)
38      end
39
40      # Create a LVM Logical Volume mirrored with regions size of 8 MB.
41      # Set the size in kilobytes.
42      # See vor lvcreate command http://linux.die.net/man/8/lvcreate
43      # -l eq. --extents
44      # -n eq. --name
45      # -m1 eq. --mirrors
46      # -R eq. --regionsize
47      def lv_create_mirrored_size_in_kb(logical_volume_name, volume_group, size_in_kb)
```

```
48            External.cmd(@server, "#{@command} lvcreate -m1 -R 8 -l #{size_in_kb} -n #{
                  logical_volume_name} #{volume_group.name}") if
                  volume_group_check_space_in_kb(volume_group,size_in_kb)
49        end
50
51      end # module LVCreate
52    end   # module Wrapper
53  end # module LVM
```

5.6. Logical Volume entfernen

Für das Entfernen eines Logical Volume wurde das Module LVRemove entwickelt. Das Module bietet die Funktion lv_remove an.

Listing 5.5: LVM-Engine: Module `LVRemove`

```
1  require 'lvm/wrapper'
2  require 'lvm/wrapper/pvs'
3
4  module LVM
5    module Wrapper
6      module LVRemove
7
8        # Remove a LVM Logical Volume.
9        # See vor lvremove command http://linux.die.net/man/8/lvremove
10       # -f eq. --force
11       def lv_remove(logical_volume)
12         External.cmd(@server, "#{@command} lvremove -f #{logical_volume.name}")
13       end
14
15     end # module LVRemove
16   end # module Wrapper
17 end # module LVM
```

5.7. Volume Group entfernen

Das Wrapper Module VGRemove entfernt eine Volume Group, wenn diese keine Logical Volume Objekte enthält.

Listing 5.6: LVM-Engine: Module `VGRemove`

```
1  require 'lvm/wrapper'
2  require 'lvm/wrapper/pvs'
```

```
3
4  module LVM
5    module Wrapper
6      module VGRemove
7
8        # Remove a empty LVM Volume Group.
9        # See vor vgremove command http://linux.die.net/man/8/vgremove
10       def vg_remove(volume_group)
11         #If there is no logical volume than the volumegroup can be removed
12         External.cmd(@server, "#{@command} vgremove #{volume_group.name}") if
               volume_group.logical_volumes.empty?
13       end
14
15     end # module VGRemove
16   end # module Wrapper
17 end # module LVM
```

5.8. Physical Volume entfernen

Das Wrapper Module PVRemove entfernt den Phyiscal Volume Label eines Physical Volume Objektes, wenn dieses keinem Volume Group zugeordnet ist. Die Zuordnung wird anhand des Attributes vg_uuid geprüft.

Listing 5.7: LVM-Engine: Module `PVRemove`

```
1  require 'lvm/wrapper'
2  require 'lvm/wrapper/pvs'
3
4  module LVM
5    module Wrapper
6      module PVRemove
7
8        # Remove a LVM PhyscialVolume Label from a disk
9        # if physical volume does not belong to a volume group
10       def pv_remove(physical_volume)
11         External.cmd(@server, "#{@command} pvremove #{physical_volume.name}") if
               physical_volume.vg_uuid.nil?
12       end
13
14     end # module PVRemove
15   end # module Wrapper
16 end # module LVM
```

5.9. Volume Group erweitern

Das Wrapper Module VGExtents erweitert ein Volume Group Object, nach denselben Kriterien wie beim Erstellen einer Volume Group.

Listing 5.8: LVM-Engine: Module `VGExtend`

```
require 'lvm/wrapper'
require 'lvm/wrapper/pvs'
require 'lvm/physical_volume_helper'

module LVM
  module Wrapper
    module VGExtend

      # Extend a LVM Volume Group.
      # See vor vgextend command http://linux.die.net/man/8/vgextend
      def vg_extend(volume_group, physical_volumes)
        External.cmd(@server, "#{@command} vgextend #{volume_group.name} #{
            physical_volumes_to_s(physical_volumes_array)}") if physical_volumes_unused(
            physical_volumes_array)
      end

    end # module VGExtend
  end # module Wrapper
end # module LVM
```

5.10. Logical Volume erweitern

Das Wrapper Module LVExtents erweitert ein Logical Volume Object, nach denselben Kriterien wie beim Erstellen eines Logical Volumes.

Listing 5.9: LVM-Engine: Module `LVExtend`

```
require 'lvm/wrapper'
require 'lvm/wrapper/lvs'
require 'lvm/volume_group_helper'

module LVM
  module Wrapper
    module LVExtend

      include VolumeGroupHelper

      # Create a LVM Logical Volume.
```

```
12        # Set the size with extents.
13        # See vor lvcreate command http://linux.die.net/man/8/lvcreate
14        # -l eq. --extents
15        # -n eq. --name
16        def lv_extend_extents(logical_volume_name, volume_group, extents)
17            External.cmd(@server, "#{@command} lvextend -l +#{extents} -n #{
                  logical_volume_name}")if volume_group_check_space_in_extents(
                  volume_group,extents)
18        end
19
20        # Create a LVM Logical Volume.
21        # Set the size in kilobytes.
22        # See vor lvcreate command http://linux.die.net/man/8/lvcreate
23        # -l eq. --extents
24        # -n eq. --name
25        def lv_extend_size_in_kb(logical_volume_name, volume_group, size_in_kb)
26            External.cmd(@server, "#{@command} lvextent -L #{size_in_kb} -n #{
                  logical_volume_name}") if volume_group_check_space_in_kb(volume_group,
                  size_in_kb)
27        end
28
29      end # module LVExtend
30    end # module Wrapper
31  end # module LVM
```

5.11. Quellcode

Das Projekt befindet sich auf der Social Coding Plattform "Github"https://github.com/lutsho/ruby-lvm-ssh

6. Fazit / Zusammenfassung

Mit dem Thema Volume Manager Engine habe ich bewusst ein Thema gewählt, welches mir erlaubt, viel neues Wissen anzueignen und mich in neue Technologien einzuarbeiten. Neben den Volume Managern LVM und ZFS war mir die Programmiersprache Ruby zu Beginn der Arbeit nicht bekannt.

Für die Einarbeitung in LVM habe ich mich zuerst mittels Literatur in das Thema eingelesen. Dazu habe ich das Handbuch von RedHat verwendet. Für eine wissenschaftliche Arbeit möchte man sicher noch tiefer in eine neue Technologie hineinblicken, als dies das Studium eines Benutzerhandbuches für Anwender zulässt. Leider war es mir trotz intensiven Recherchen nicht möglich gewesen, an bessere und zuverlässigere Quellen heranzukommen. Beispielsweise ging aus dem Handbuch nicht klar hervor, wie die Segmentierung der Volumes und das Striping der Volumens zusammenhängen. Leider blieb eine anonyme Anfrage in der offiziellen LVM-Mailingliste unbeantwortet und erfolglos. Andere Fragen versuchte ich anhand von virtuellen Linux Servern, welche ich als Cluster und nicht Cluster-Betrieb ausführte, mir zu erarbeiten. Obwohl diese Vorgehensweise eher aufwendig ist, sammelte ich damit viel Erfahrung und es half mir sehr gut, mich in LVM einzuarbeiten.

Anhand von Grafiken und den dazugehörenden Beschreibungen, versuchte ich die Eigenschaften von LVM dem Leser zu erklären. Die Grafiken sollten es dem Leser vereinfachen, die verschiedenen Volume Architekturen, welche LVM unterstützt und wie die einzelnen LVM-Objekte zu einander in Verbindung stehen, zu verstehen. Die Zusammenfassung der wichtigsten Befehle beschreibt, wie sich die aufgeführten Architekturen konfigurieren lassen.

Die Anforderungsanalyse für die Applikation habe ich um eine Marktanalyse ergänzt, obwohl diese nicht Ziel der Arbeit war, aber weitere sinnvolle Informationen liefert. Eine Marktanalyse sollte nach meinem Verständnis in keiner Arbeit fehlen, in der es um die Implementation eines neuen Produktes geht. Ohne zu erforschen, welche Produkte und Optionen der Markt für die bestehenden und zukünftigen Technologien bietet und welche davon sich wahrscheinlich durchsetzen werden, sollte man kein Softwareprojekt beginnen. Gerne hätte ich anhand von Gartner-Analysen dargestellt, wie sich der Markt von x86 zu RISC Architekturen voraussichtlich künftig

entwickeln wird, jedoch waren diese Analysen nur käuflich zu erwerben und nicht öffentlich zugänglich. Eine Firma hätte sicherlich sich die notwendigen Informationen aus dem Projektbudget geleistet.

Mit dem Prototyp konnte ich aufzeigen, dass eine Umsetzung des von mir erstellten Konzept möglich ist. Für den Prototyp habe ich ein bestehendes Projekt verwendet und mit meinen Anforderungen erweitert. Als Novize in einer neuen Programmiersprache hat es mir geholfen, dass ich mich an bestehenden Codebeispielen orientiert habe und meiner Einschätzung nach eine akurate Architektur erstellt habe.

Für mich persönlich fällt die Zwischenbilanz für diese Arbeit positiv aus. Die Arbeit half mir neues und für meinen Beruf notwendiges Wissen anzueignen, welches ich künftig in meinen Arbeiten einsetzen werde. Auf Wunsch meines Arbeitgebers werde ich mich nach Beendung der Semesterarbeit bei der Entwicklung neuer Storage Managment Tools in der Firma einbringen. Dieser neue Aufgabenbereich bietet mir neben der Weiterbildung im System-Engineering, einen nahtlosen und idealen Einstieg in die Softwareentwicklung.

Glossar

AES

Advanced Encryption Standard (AES) wurde von US National Institute of Standards and Technology für den Schutz von elektronischen Daten spezifiziert. AES Algorithmus ist eine symetrische Block Chiffrierung, welche mit 128, 192 und 256 bit Schlüssel Daten in Blocks von 128 bit verschlüsseln und entschlüsseln kann. http://www.nist.gov/manuscript-publication-search.cfm?pub_id=901427. 48

API

Application Programming Interface (API) auch Anwendungsprogrammierschnittstelle genannt. "Ein Satz an Routinen, die vom Betriebsystem des Computers für die Verwendung aus Anwendungsprogrammen heraus angeboten werden und diverse Dienste zur Verfügung stellen." MicrosoftPress [2003]. 43

BlowFish

BlowFish wurde von Brice Schneider 1993 entwickelt. Der Blowfish Algorithmus ist eine symetrische Block Chiffrierung, welche eine variable Schlüssellänge von 32 bits bis 448 bits Daten verschlüsseln und entschlüsseln kann. http://www.schneier.com/blowfish.html. 48

Cluster

Ein Cluster ist ein Verbund von vernetzten Servern, auch Nodes genannt, welche dem Client in Form eines einzelnen Server gesehen wird. Cluster werden meist dazu verwendet, um einen Server-Service hoch verfügbar zu halten, indem ein Cluster Node die Aufgaben eines anderen Cluster Node übernimmt. Zudem werden Cluster für die Erhöhung von Rechenkapazität eingesetzt, wie sie z.B. in der Forschung, Industrie bzw. Metrologie zur Berechnung rechenintensiver Aufgaben verwendet werden. http://de.wikipedia.org/wiki/Computercluster. 5, 13

Cluster Node

Ein Cluster Node, oder oft auch nur als Node (zu deutsch Knoten) bezeichnet, ist ein einzelner Server in einem Cluster-Verbund. 13

LUN

Logical Unit Number kurz LUN, ist eine Nummer um ein Locical Unit bzw. eine SCSI Gerät zu identifizieren. Oft wird bei der Verwendung des Begriffs das Gerät selber gemeint, was technisch nicht ganz korrekt ist. Im Storage Umfeld wird LUN mit Disk gleichgesetzt. 6, 11

POpen4

POpen4 stellt den Ruby Entwicklern eine plattformübergreifende API zur Ausführung von Befehlen in einen Kindprozess, welche die stdout, stderr, stdin stream als auch die Prozess-ID und Exit Status behandelt, zur Verfügung stellt. http://popen4.rubyforge.org/. 43

RFC

Request for Comments (RFC) sind Dokumente, über Internet, inklusive der technischen Spezifikation und Richtlinien, welche von der Organisation Internet Engineering Task Force entwickelt wurde. "Das RFC wird erst nach erfolgter Diskussion unter der Aussicht des Internet Architecture Board (IAB) herausgegeben und fungiert als Quasistandard. Jedes RFC enthält eine eindeutige, vorlaufende Nummer, die kein zweites Mal zu gewiesen wird." MicrosoftPress [2003] http://www.rfc-editor.org/. 47

RSA

RSA ist eine Chiffrierungs Algorithmus für asymetrische Verschlüsselung und wurde von den Mathematikern Ronald L. Rivest, Adi Shamir und Leonard Adleman entwickelt. RSA wird in vielen Verfahren verwendet, in welchen eine sichere Authentifizierung sichergestellt werden muss, unter anderem in SSL, PKI usw. http://www.rsa.com/rsalabs/node.asp?id=2125. 48

SAN

Storage Area Netzwork, ist ein Netzwerk zur Anbindung von Speicher und Tape, welches von einem Storage-System bzw. Tape-Library eines Servers stammt. Die Server und Storage-Systeme kommunizieren im SAN mit Fibre-Channels. 6, 11, 27, 28

Snapshot

Snapshot ist ein besonderer Speicherbereich, der sämtliche Änderungen zu einem älteren festgelegten Datenbestand aufnimmt. http://en.wikipedia.org/wiki/Snapshot_(computer_storage). 12

SUDO

Sudo kommt von dem Befehl su und dem Wort do und erlaubt es System Administratoren-Berechtigung an Benutzer oder Gruppen zu geben und einzelne Befehle als root Benutzer auszuführen. http://www.courtesan.com/sudo/. 48

TrippleDES

TrippleDES ist eine Erweiterung des Data Encryption Standards (DES) für die symmetrische Verschlüsselung und Entschlüsselung. TrippleDES verschlüsselt die Daten drei mal mit DES, dabei werden jeweils drei von einander unabhängige Schlüssel verwendet.. 48

Wrapper

Als Wrapper bezeichnet man in der Informationstechnik ein Stück Software, welches ein anderes Stück Software umgibt (umwickelt). http://de.wikipedia.org/wiki/Wrapper_(Software). 43

YAML

YAML ist eine vereinfachte Auszeichnungssprache zur Datenserialisierung, angelehnt an XML und an die Datenstrukturen in den Sprachen Perl, Python und C. http://de.wikipedia.org/wiki/YAML. 43

Literaturverzeichnis

Fisher 2008

Fisher, Timothy: *Ruby on Rails Bible*. Wiley, 2008. – ISBN 9780470258224

Levine 2009

Levine, Steven: *Red Hat Enterprice Linux 5 Logical Volume manager Administration*. 1. Red Hat Inc., 08 2009

Li und Tan 2009

Li, Yan ; Tan, Chuan H.: Aligning CIO Characteristics to Business Strategy: An Empirical Investigation, 2009. – ISSN 1530–1605, S. 1 –10 3.2.1, 13

MicrosoftPress 2003

MicrosoftPress: *Computer Lexikon Fachwörterbuch*. 7. Microsoft Press, 2003. – ISBN 3860638963 6

Zörner 2009

Zörner, Tom: *E/A - Kanäle*. http://wwwcip.informatik.uni-erlangen.de/old/tree/CIP/Manuals/unix/shell/kanaele.html. Version: 2009, Abruf: 16.03.2010

Eidesstattliche Erklärung

Ich, Lucien Stucker, Matrikel-Nr. 06-557-540, versichere hiermit, dass ich meine Seminararbeit mit dem Thema

Volume Manager Engine LVM

selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Texten entnommen sind, wurden unter Angabe der Quellen (einschliesslich des World Wide Web und anderer elektronischer Text- und Datensammlungen) und nach den üblichen Regeln des wissenschaftlichen Zitierens nachgewiesen. Dies gilt auch für Zeichnungen, bildliche Darstellungen, Skizzen, Tabellen und dergleichen. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschungsversuch behandelt werden und dass bei einem Täuschungsverdacht sämtliche Verfahren der Plagiatserkennung angewandt werden können.

Zürich, den 21. März 2011

Lucien Stucker

A. Anhang

A.1. Projektplanung

A.1.1. Meilensteine

Milestone-Trend

The following project milestones have been defined:

#	Milestone	Group	Planned	Prio	Expected	Complete
15	Freigabe Arbeit	Volume Manager Engine	02.09.2010 08:00	Normal	02.09.2010 08:00	0 %
22	Kick-Off	Volume Manager Engine	04.11.2010 08:00	Normal	04.11.2010 08:00	0 %
26	Volume Manager	Dokumentphase	05.11.2010 12:00	Normal	05.11.2010 12:00	0 %
38	Konzept	Dokumentphase	24.12.2010 14:00	Normal	24.12.2010 14:00	0 %
64	Ende der Entwicklung des Prototyp	Volume Manager Engine	08.01.2011 12:00	Normal	08.01.2011 12:00	0 %
71	Design-Review	Volume Manager Engine	20.01.2011 08:00	Normal	20.01.2011 08:00	0 %
49	Dokument	Volume Manager Engine	12.02.2011 12:00	Normal	12.02.2011 12:00	0 %
82	Präsentation	Präsentation	30.03.2011 08:00	Normal	30.03.2011 08:00	0 %

Abbildung A.1.: Projektplanung: Meilensteine

A.1.1.1. Trend

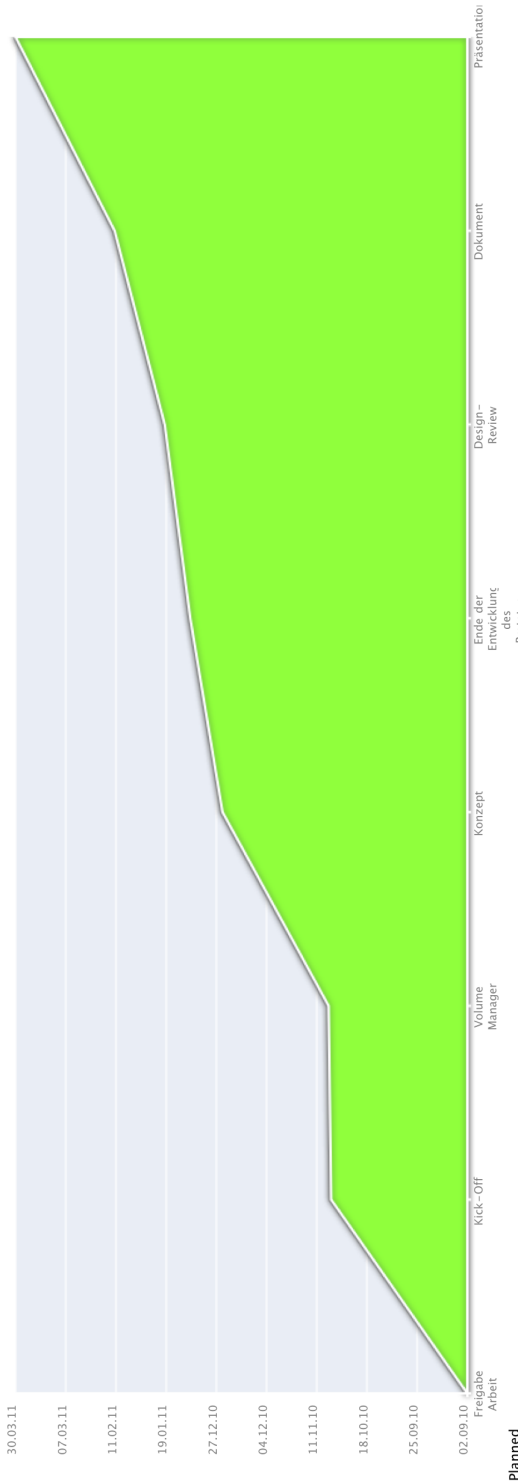


Abbildung A.2.: Projektplanung: Meilensteine Trend

A.1.2. Netzplan

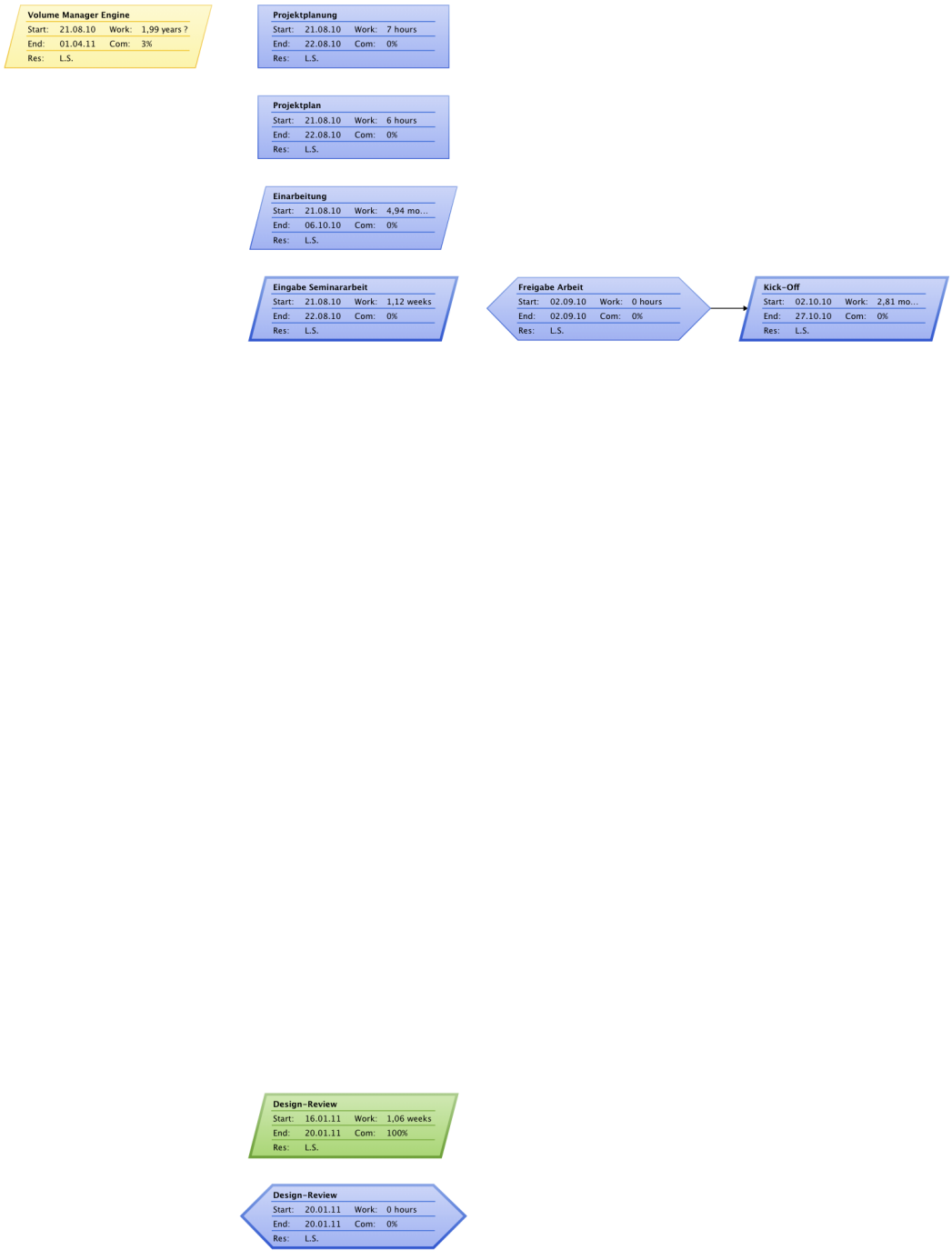


Abbildung A.3.: Projektplanung: Netzplan Teil 1

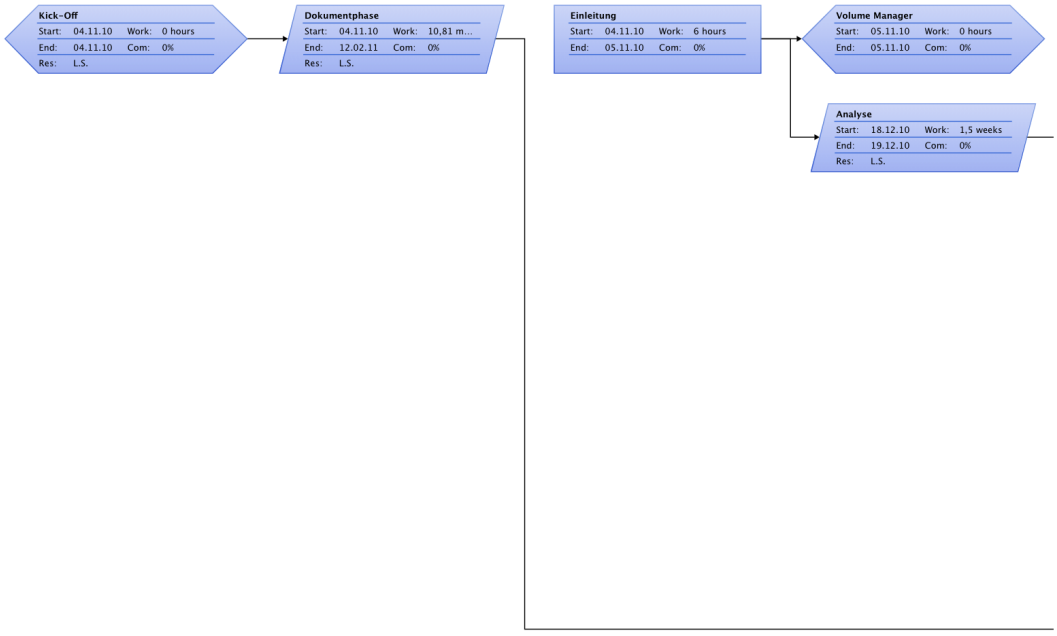
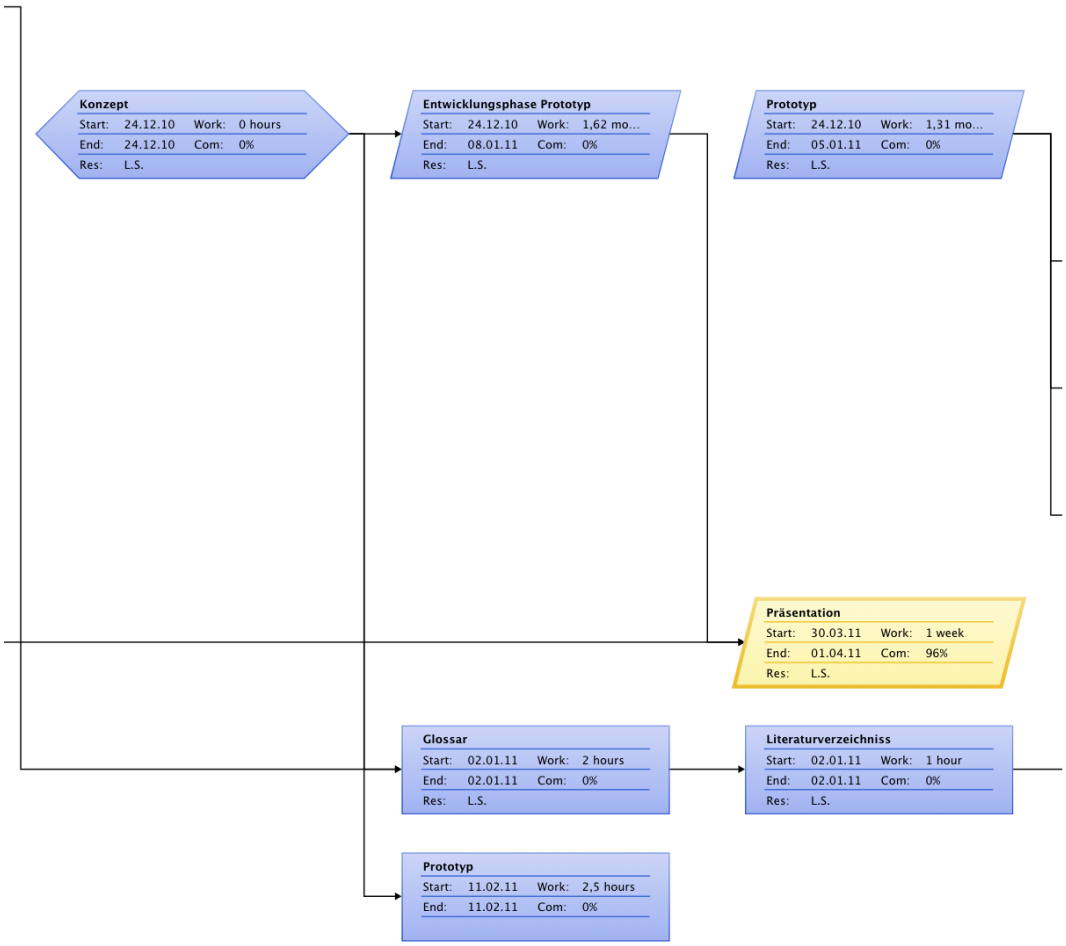


Abbildung A.4.: Projektplanung: Netzplan Teil 2


Konzept
Start: 24.12.10 Work: 0 hours
End: 24.12.10 Com: 0%
Res: L.S.
Entwicklungsphase Prototyp
Start: 24.12.10 Work: 1,62 mo...
End: 08.01.11 Com: 0%
Res: L.S.
Prototyp
Start: 24.12.10 Work: 1,31 mo...
End: 05.01.11 Com: 0%
Res: L.S.
Präsentation
Start: 30.03.11 Work: 1 week
End: 01.04.11 Com: 96%
Res: L.S.
Glossar
Start: 02.01.11 Work: 2 hours
End: 02.01.11 Com: 0%
Res: L.S.
Literaturverzeichniss
Start: 02.01.11 Work: 1 hour
End: 02.01.11 Com: 0%
Res: L.S.
Prototyp
Start: 11.02.11 Work: 2,5 hours
End: 11.02.11 Com: 0%

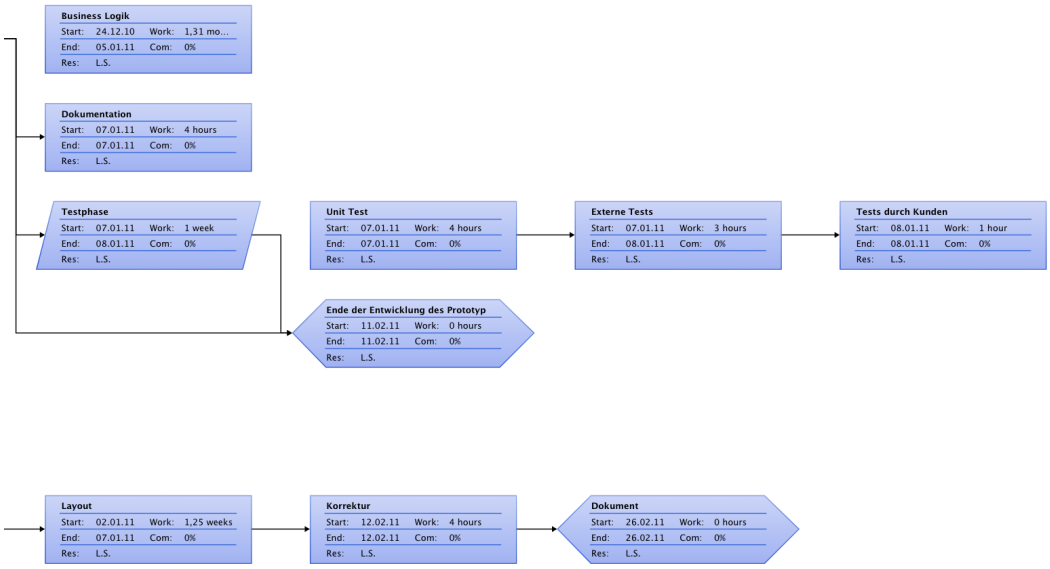


Abbildung A.6.: Projektplanung: Netzplan Teil 4