Informatik Projektpraktikum

XEN Benchmarks

Arne Klein

1. März 2008

Inhaltsverzeichnis

1 Allgemeines

1.1 Einleitung

Das Ziel des Praktikums war die Installation und Inbetriebnahme der Virtualisierungssoftware XEN auf zwei Servern des Clusters der Abteilung Parallele und Verteilte Systeme (PVS) am Institut für Informatik der Universität Heidelberg. Darauf aufbauend sollten sowohl auf den normalen als auch auf virtualisierten Systemen Performancemessungen durchgeführt werden, um die Leistungseinbußen durch die Benutzung von XEN zu quantifizieren.

1.2 Virtualisierung

Das Ziel der Virtualisierung von Hardware ist hauptsächlich die Loslösung von Betriebssystem, Hardware und Software. Dies sorgt dafür, dass sowohl ein paralleler Betrieb von mehreren Betriebssystemen auf einem Rechner als auch eine problemlose Migration auf andere Hardware möglich ist.

Momentan wird Virtualisierung hauptsächlich genutzt um auf einem Server jeweils das passende Betriebssystem für verschiedene Applikationen bereit zu stellen und zur Erhöhung der Sicherheit verschiedene Anwendungen in einzelnen virtuellen Maschinen voneinander abzuschotten. Des weiteren ist es gerade im Bereich des Webhostings mittlerweile üblich virtuelle Server statt ganzer Rechner zu vermieten, was dies deutlich günstiger macht. Da auch Live-Migrationen von einem Server auf einen anderen ohne Ausfälle möglich sind, wird auch die Wartung in diesem Bereich deutlich vereinfacht.

Grundsätzlich unterscheidet man heute zwischen drei grundlegend verschiedenen Virtualisierungsarten:

- Vollständige Virtualisierung (z.B. VMware, VirtualBox)
  - komplette Hardware wird simuliert
  - Virtualisierung ist vor Gastsystem verborgen, also sind keine Anpassungen nötig
  - aber: langsam
- Single Kernel Image (z.B. Linux Vserver)
  - mehrere Instanzen des selben Betriebssystems
  - alle greifen auf selben Kernel zurück, also keine Hardwaresimulation nötig
  - aber: auf einen Kernel und somit sehr ähnliche Betriebssysteme beschränkt

- Paravirtualisierung (z.B. XEN, VMware ESX Server)
  - Hypervisor als Ebene zwischen Betriebssystem und Hardware
  - Hypervisor verteilt Ressourcen an Betriebssysteme
  - Anpassung am Kernel nötig, falls keine spezielle Hardwareunterstützung existiert
  - ähnlich performant wie ohne Virtualisierung

1.3 XEN

XEN ist eine seit Oktober 2003 verfügbare Virtualisierungssoftware, die als Open Source unter der GPL verfügbar ist. Wie bereits oben erwähnt, verfolgt XEN hauptsächlich den Ansatz der Paravirtualisierung - dabei ist das Hostsystem die privilegierte Domain (auch dom0 genannt) und steuert den Hypervisor, welcher die Ressourcen an die verschiedenen virtuellen Maschinen verteilt (siehe Abbildung 1.1). Momentan werden die meisten Unix-artigen Betriebssysteme sowohl als Host- als auch Gastsystem unterstützt. Mit der seit einigen Monaten vorhandenen Virtualisierungs-Hardwareunterstützung in allen neuen Prozessoren von Intel und AMD wird auch der Betrieb beliebiger x86-Betriebssysteme, u.a. Windows, als Gast möglich.

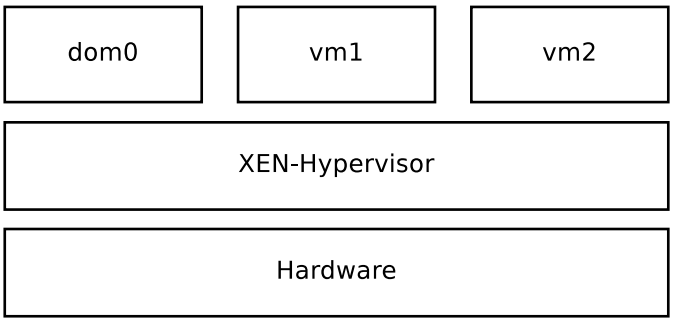


Abbildung 1.1: Schematischer Aufbau von XEN

2 Installation

2.1 Einleitung

Bei der folgenden Anleitung zur Installation wird von einem Debian Etch System ausgegangen.

2.2 Vorbereitung

Es ist empfehlenswert zunächst ausstehende Patches einzuspielen um das System auf die aktuelle Version zu bringen.

```
aptitude update && aptitude upgrade
```

2.3 XEN-Kernel

Unter Debian Etch gibt es bereits vorkompilierte Kernelversionen mit XEN Patches, welche den normalen Kernel ersetzen.

```
aptitude install xen-linux-system-2.6.18-5-xen-686 xen-tools
```

Außerdem sollte noch die Zahl der erlaubten loop-devices erhöhen, was mit dem Befehl

```
echo "options␣loop_max=50" > /etc/modprobe.d/loop
```

möglich ist.

Danach ist ein Neustart mittels `reboot` sinnvoll. Mit `uname -r` kann man nun testen, ob alles funktioniert hat. testen ob es funktioniert hat, die Ausgabe sollte `2.6.18-5-xen-686` sein.

2.4 Grundkonfiguration von XEN

Da aus Performancegründen ein Bridge-basiertes Netzwerksetup empfohlen wird, muss dies zunächst in `/etc/xen/xend-config.sxp` aktiviert werden.

Listing 2.1: /etc/xen/xend-config.sxp

```
(network-script network-bridge)     # kommentar entfernen
# (network-script network-dummy)    # auskommentieren

(vif-script vif-bridge)
```

Um die Einstellung zu übernehmen ist ein Neustart des XEN-Daemons via `invoke-rc.d xend restart` nötig.

Die Bridge dient zur Verbindung mehrerer virtueller Netzwerkkarten der einzelnen Virtuellen Maschinen sowie der externen Netzwerkkarte. Sie funktioniert ähnlich wie ein interner Switch, das heißt die Pakete werden anhand ihrer Ethernet MAC-Adresse nur an den Zielhost weiter geleitet. Sie setzt also auf der OSI Sicherungsschicht auf und funktioniert damit unabhängig von dem darauf aufsetzenden Protokoll, wie z.B. IP.

Als Alternativen zu dem Bridge-Setup gibt es noch ein vorkonfiguriertes Routing- und ein NAT-Setup, welche auch intern eine Bridge nutzen. Die Unterschiede bestehen darin, dass bei dem Bridge-Setup direkt auf der Netzwerkkarte vom XEN Hypervisor eine Bridge aufgesetzt wird, und sowohl das Serversystem als auch alle Virtuelle Maschinen nur über diese Bridge auf die Netzwerkkarte zugreifen. Beim Routing- und NAT-Setup ändert sich hingegen der Zugriff des Serversystems auf die Netzwerkkarte nicht. Dafür wird intern ein zweites virtuelles Netzwerkinterface gestartet und dieses mit einer Bridge mit den Virtuellen Maschinen verbunden. Jeglicher Traffic muss also vom Serversystem weitergeleitet werden. Beim Routing-Setup hat jede VM ihre eigentliche öffentliche IP (und MAC), welche per Routing weiter geleitet wird. Das NAT-Setup vergibt hingegen nur interne IPs, welche dann per NAT vom Serversystem angesprochen werden können. Dies wird vor allem bei Root-Servern von Hostinganbietern genutzt, da dort oft ein MAC-Spoofing Schutz aktiviert ist der den Server blockt, sobald er sich mit mehreren MAC-Adressen nach außen meldet.

2.5 LVM für VMs anlegen

Um die automatische Erstellung der XEN VMs in LVM Partitionen zu ermöglichen, muss zunächst eine passende LVM Volume Group angelegt werden. Diese besteht im Normalfall aus einer Partition (hier `/dev/hdaX`).

```
pvcreate /dev/hdaX
vgcreate mainvg /dev/hdaX
```

Möchte man eine schon bestehende Volume Group um eine weitere Partition erweitern, so ist dies mittels `vgextend mainvg /dev/hdaX` möglich.

2.6 Debian Gastsystem

Für Debian Gastsysteme gibt es ein sehr komfortables Skript, welches automatisch passende LVs im LVM anlegt und das System mittels debootstrap installiert. Dafür muss zunächst eine passende Konfiguration angelegt werden.

Listing 2.2: /etc/xen-tools/xen-tools.conf

```
# dir = /home/xen
lvm = mainvg
```

```
# copy = /pfad/zur/bestehenden/installation
debootstrap = 1
# rpmstrap = 1
# tar = /pfad/zur/img.tar


size    = 4Gb       # Größe des Datenträgers
memory = 256Mb     # Zugewiesene Arbeitsspeicher
swap    = 256Mb     # Größe der Swap
# noswap = 1       # Wenn wir keine Swap möchten
fs      = ext3      # Das verwendete Dateisystem
dist    = etch     # Die Default Distribution
image   = sparse    # Sparse legt die Datei nur so groß an, wie
    nötig, "full" alloziert die volle Größe der Datei.
# LVM ignoriert diesen Wert. Die Option '''sparse''' muss vom
    Dateisystem unterstützt werden.


# Möchten wir eine Statische IP vergeben, so können wir sie hier
    vorgeben
gateway    = 10.0.0.1
netmask    = 255.255.255.0
# Andernfalls aktivieren wir Netzwerk per DHCP
#
# dhcp = 1

# Hier können wir interaktiv nach einem Root  Kennwort gefragt
    werden, für den neuen Gast
passwd = 1

# Hier wird der zu verwendende Kernel und dessen Ramdisk angegeben
kernel = /boot/vmlinuz-2.6.18-5-xen-686
initrd = /boot/initrd.img-2.6.18-5-xen-686

mirror = http://ftp.de.debian.org/debian/
```

Ein neues Gastsystem lässt sich nun mit folgendem Befehl erzeugen.

```
xen-create-image --hostname=vm1 --ip 10.0.0.2x
```

Es kann dann mit

```
xm create /etc/xen/vm1.cfg
```

gestartet werden.

3 Performancemessungen

3.1 Messungen

Im folgenden werden die Ergebnisse verschiedener Performancemessungen auf wie oben beschrieben konfigurierten XEN Systemen vorgestellt. Dabei wurde die IO Geschwindigkeit mittels Iozone, die Netzwerkleistung mit netperf und die CPU Belastung mit partdiff getestet. Des weiteren wurde ein abschließender Test mit dem parallelen Dateisystem PVFS2 durchgeführt, welches alle drei Komponenten, insbesondere aber Netzwerk und IO, gleichzeitig belastet.

Bei den IO Messungen war das Ziel ein Geschwindigkeitsvergleich zwischen virtualisierten Systemen und dem Gastsystem auf dem selben Rechner. Auch bei den CPU Tests fanden die Messungen auf nur einem Rechner statt, allerdings ging es hier um die Geschwindigkeit bei paralleler Belastung in verschiedenen VMs und dem Hostsystem. Das Ziel der Netzwerkuntersuchungen lag sowohl in der Geschwindigkeit zwischen zwei Virtual Machines auf dem selben Rechner als auch zwischen verschiedenen Systemen.

3.2 Server

Folgende Server und Virtuelle Maschinen werden für die Performancemessungen benutzt

```
node6 (XEN dom0) - Debian auf Server 6
node7 (XEN dom0) - Debian auf Server 7
vm1 - Debian auf node 6
vm2 - Debian auf node 6
vm3 - Debian auf node 7
vm4 - Debian auf node 7
vm5 - CentOS auf node 6
```

Beide Server sind identisch mit folgender Hardware ausgestattet:

- zwei Intel Xeon 2GHz CPUs
- Intel Server Board SE7500CW2
- 1 GB DDR-RAM
- 80GB IDE HDD
- CD-ROM Drive
- Floppy Disk Drive

- zwei 100-MBit/s-Ethernet-Ports (nicht in Benutzung)
- zwei 1-GBit/s-Ethernet-Port (einer in Benutzung)
- 450 Watt Single Power Supply
- Debian Etch 4.0 (Linux 2.6.18-5-xen)

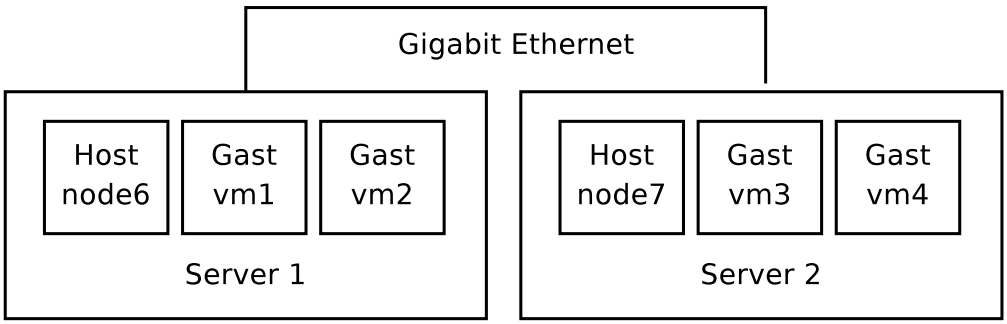


Abbildung 3.1: Schematischer Aufbau der Testsysteme

3.3 Iozone

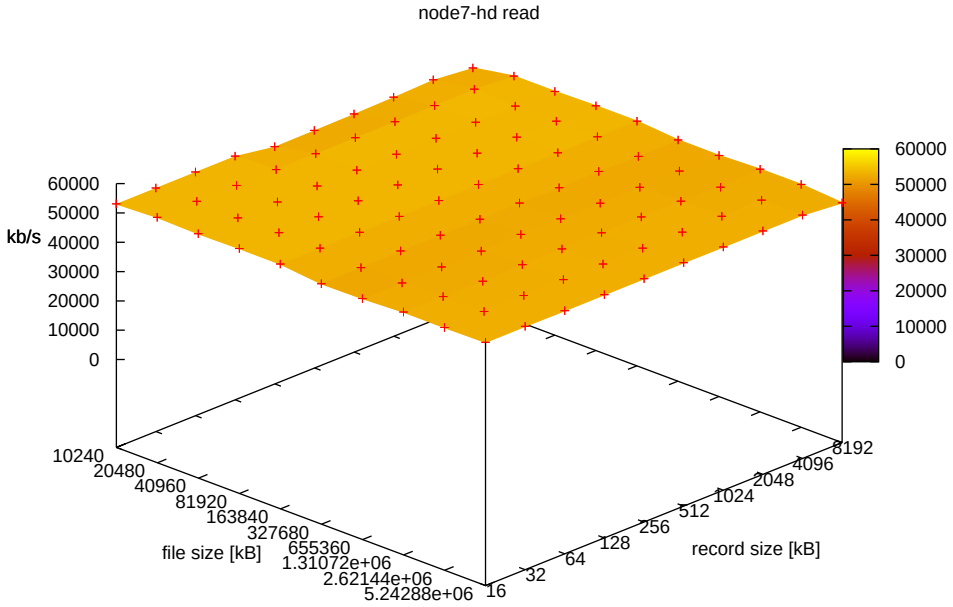
3.3.1 Allgemeines

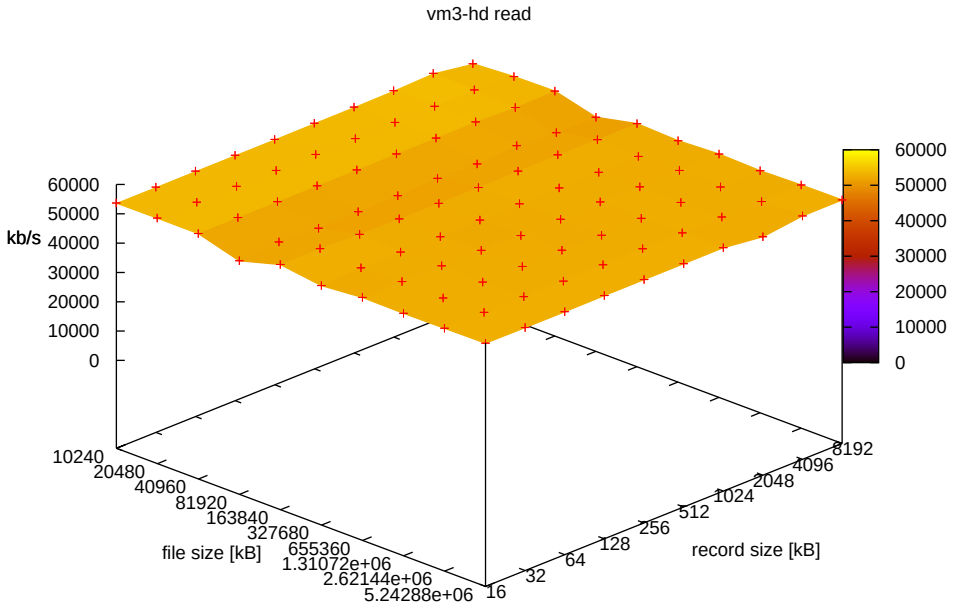
Die Iozone Dateisystem Benchmarks wurden mit folgendem Kommando durchgeführt:

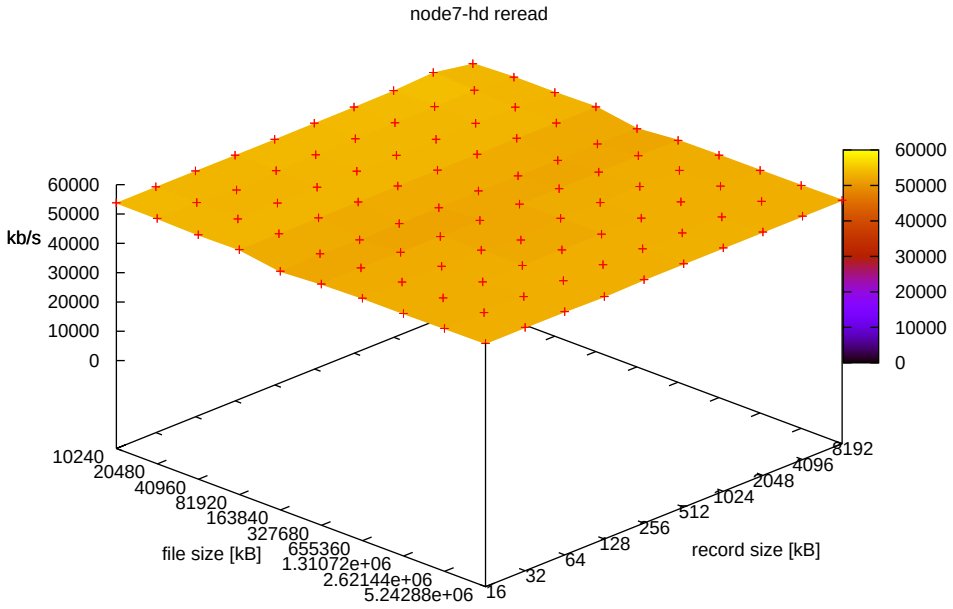
```
iozone -a -U /partition -n 10m -g 5g -q 10m -y 16k -f /partition/
   testfile -i 0 -i 1 -i 2 -i 9 -i 10 > filename.csv
```

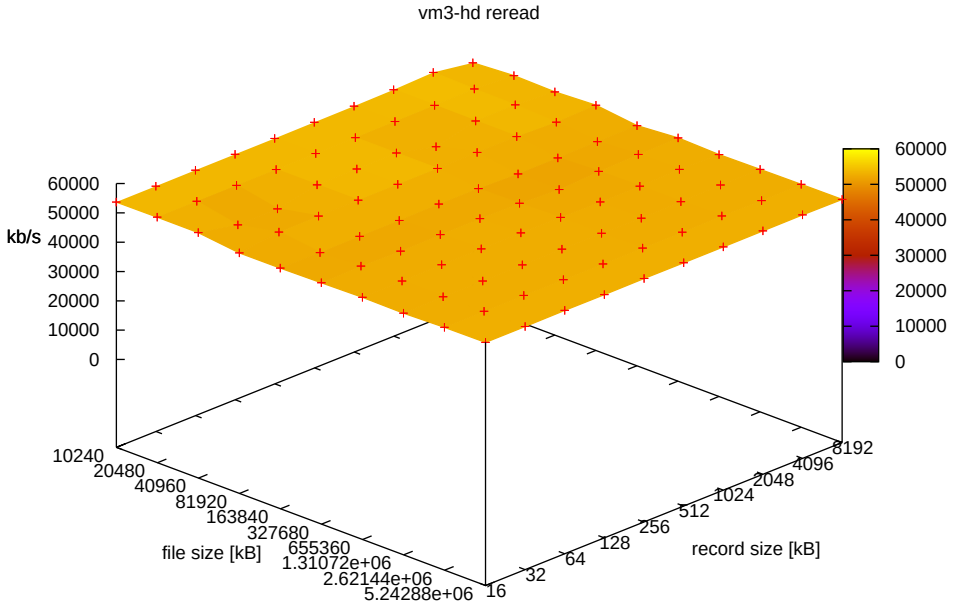
Sie fanden auf node 7 bzw vm3 (Virtual machine auf node7) statt. Die Testpartition lag nicht im LVM, sondern war beide male die selbe normale Partition mit 10 GB.

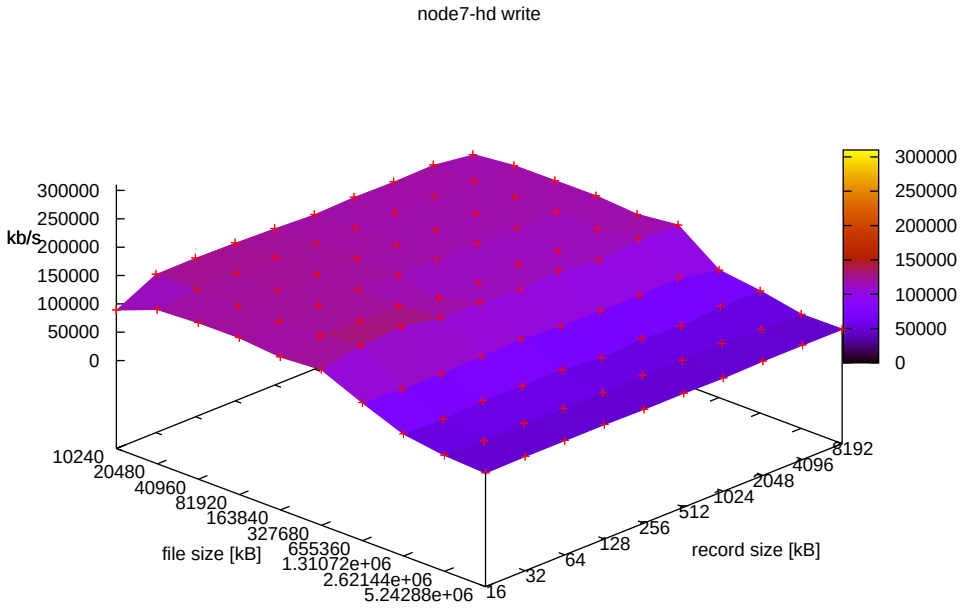
3.3.2 Messdaten im Vergleich

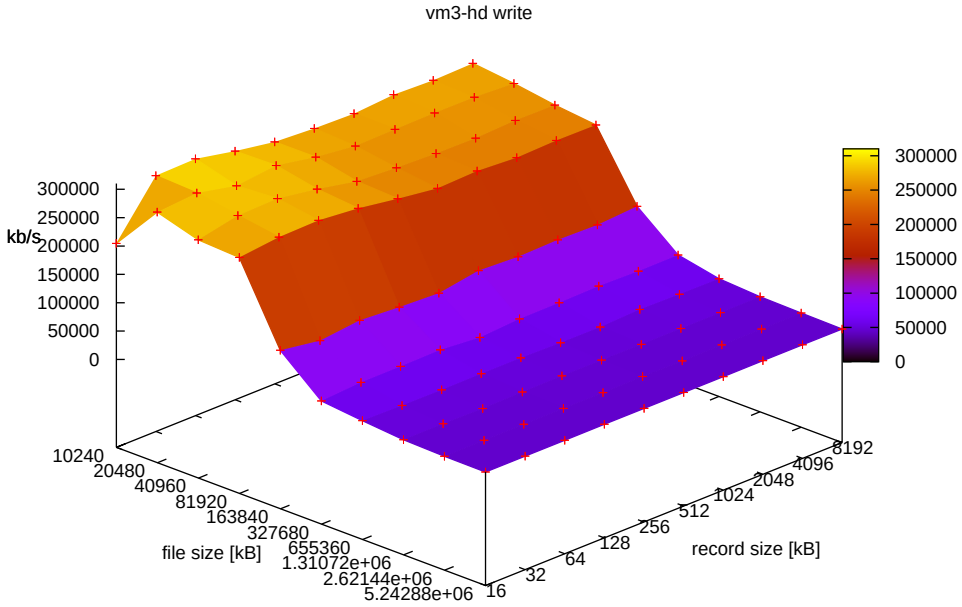




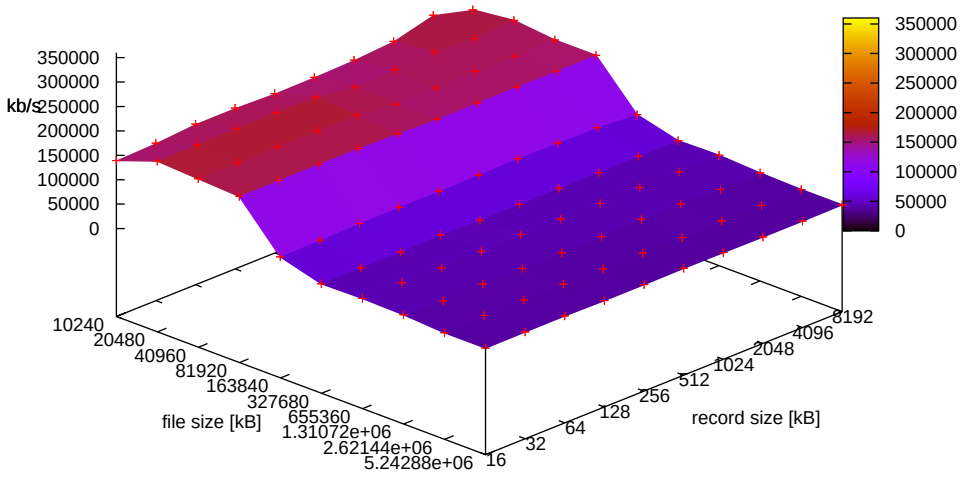

node7-hd reread
kb/s
60000
50000
40000
30000
20000
10000
0
10240
20480
40960
81920
163840
327680
655360
1.31072e+06
2.62144e+06
5.24288e+06
file size [kB]
16
32
64
128
256
512
1024
2048
4096
8192
record size [kB]


vm3-hd reread
kb/s
60000
50000
40000
30000
20000
10000
0
10240
20480
40960
81920
163840
327680
655360
1.31072e+06
2.62144e+06
5.24288e+06
file size [kB]
16
32
64
128
256
512
1024
2048
4096
8192
record size [kB]

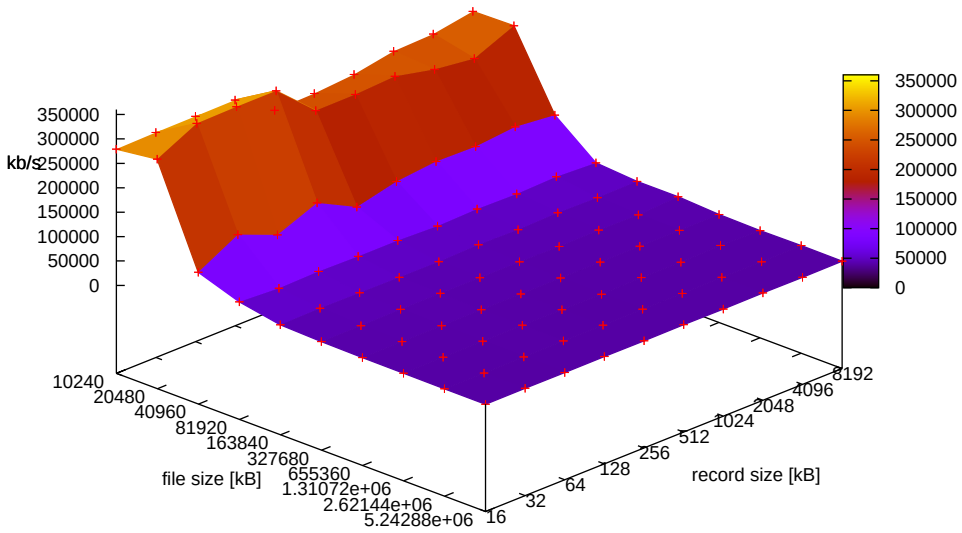

node7-hd write
kb/s
300000
250000
200000
150000
100000
50000
0
10240
20480
40960
81920
163840
327680
655360
1.31072e+06
2.62144e+06
5.24288e+06
file size [kB]
16
32
64
128
256
512
1024
2048
4096
8192
record size [kB]


vm3-hd write
kb/s
300000
250000
200000
150000
100000
50000
0
10240
20480
40960
81920
163840
327680
655360
1.31072e+06
2.62144e+06
5.24288e+06
file size [kB]
16
32
64
128
256
512
1024
2048
4096
8192
record size [kB]

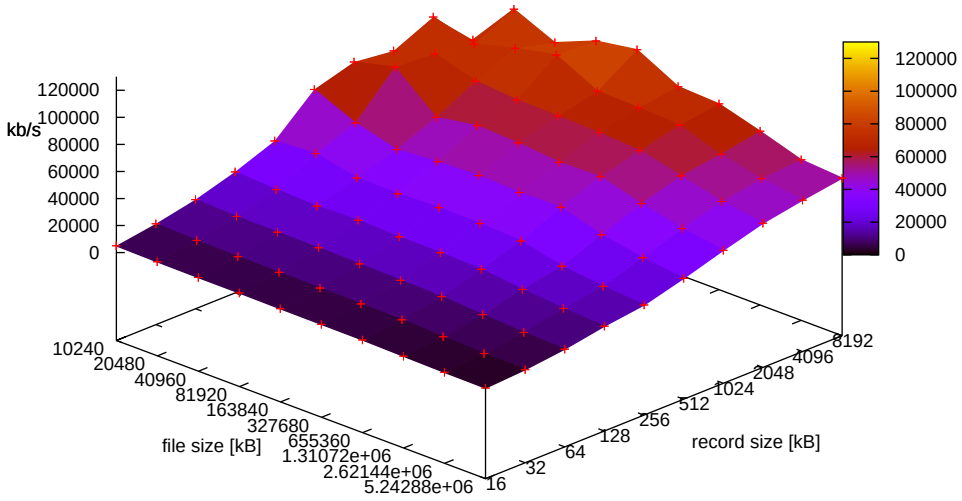
node7-hd rewrite



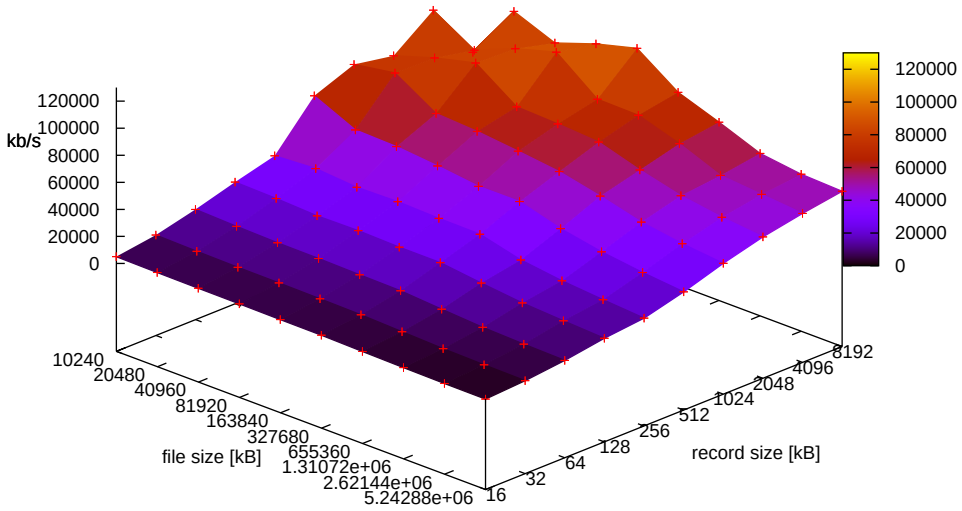
vm3-hd rewrite

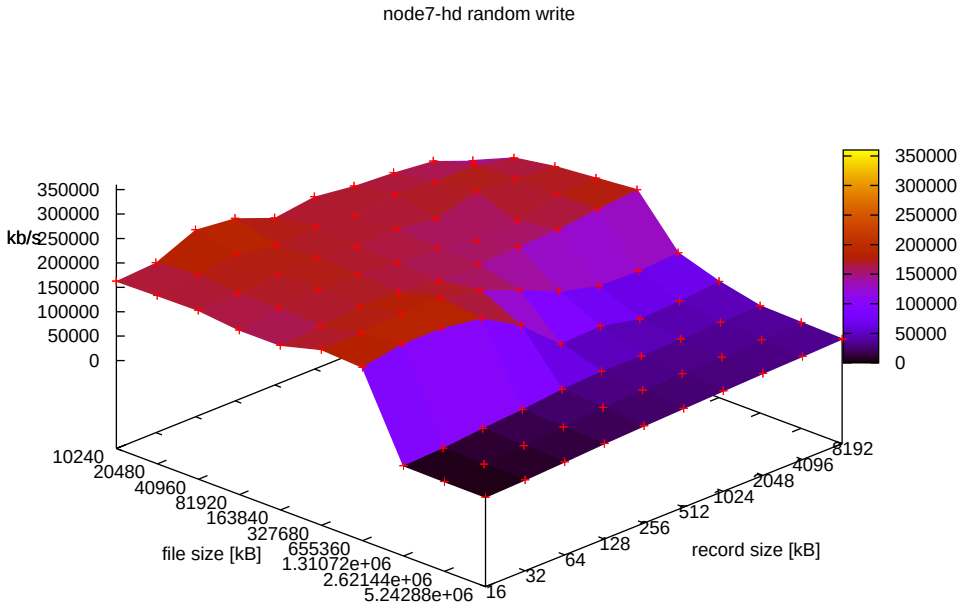


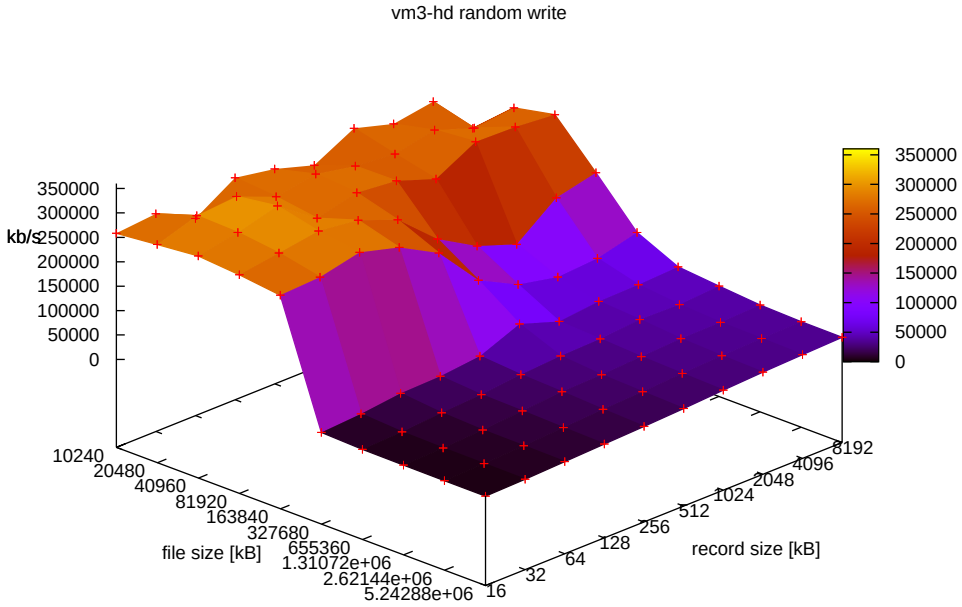
node7-hd random read



vm3-hd random read







3.3.3 Diskussion der Ergebnisse

Man sieht, dass es bei der Benutzung der XEN Virtual Machine keine deutlichen Performance-Einbußen gibt. Allerdings ist auffällig, dass bei der Benutzung von XEN der write, rewrite und random write Datendurchsatz bei kleinen Dateigrößen deutlich höher ist. Es scheint,

als ob hier durch XEN die Daten zunächst gecachet werden. Sowohl bei vm3 als auch bei node7 fällt der write Durchsatz deutlich ab, sobald die Blockgröße über der Größe des Arbeitsspeichers liegt. Der Grund für die höhere Geschwindigkeit der XEN Virtutal Machine im Cache-Bereich wurde nicht klar.

3.4 netperf

3.4.1 Allgemeines

Die netperf Benchmarks wurden mit folgenden Befehlen durchgeführt:

```
netperf -c -C -l 100 -n 2 -f M -t tcp_stream -H targethostname
```

3.4.2 Messdaten

auf/von	node6	vm2	node7	vm3	vm4
node6	232	112	106	30	
vm2	8			28	
node7					
vm3			3		
vm4			2	1	

Tabelle 3.1: Messergebnisse 1, alle Angaben in MB/s

Die zunächst durchgeführten Messungen, dargestellt in Tabelle 3.1 zeigen eine extrem schlechte Performance zwischen zwei Systemen auf dem selben Server auf. Dies lässt die Vermutung aufkommen, dass an dieser Stelle noch Optimierungen bei den Netzwerkeinstellungen durchgeführt werden können, die letzendlich für einen deutlich besseren Datendurchsatz sorgen.

Als erste Verbesserung wurde in XEN Foren vorgeschlagen, die TX Überprüfung auf eth0 auf den nodes auszuschalten, da dies nur virtuelle Netzwerkkarten sind.

```
ethtool -K eth0 tx off
```

Diese Einstellung brachte deutliche Verbesserungen - allerdings komischerweise nur bei Benutzung auf den nodes. Aktiviert man diese Option auf einer der VMs, so senkt sich der Datendurchsatz deutlich.

Außerdem sollte die Queuelength der interfaces auf allen vm's und nodes erhöht werden

```
ifconfig eth0 txqueuelen 1000
```

Des weiteren experimentierte ich mit verschiedenen Einstellungen in /etc/sysctl.conf. Da diese jedoch keine merkbaren Veränderungen erbrachten, wurden sie wieder verworfen und auf Debian Standardwerte zurück gestellt.

```
# increase TCP max buffer size setable using setsockopt()
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
# increase Linux autotuning TCP buffer limits
# min, default, and max number of bytes to use
# set max to at least 4MB, or higher if you use very high BDP
    paths
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# don't cache ssthresh from previous connection
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_moderate_rcvbuf = 1
# recommended to increase this for 1000 BT or higher
net.core.netdev_max_backlog = 2500
```

auf/von	node6	vm2	node7	vm3	vm4
node6	323	161	78	112	112
vm2	35	337	35	29	29
node7	112	89	322	28	293
vm3	37	30	32	338	3
vm4	38	30	32	2	294

Tabelle 3.2: Messergebnisse nach Optimierungen, alle Angaben in MB/s

Darauf hin werden die Performancemessungen erneut durchgeführt, die Ergebnisse sind nun allerdings anders als erwartet.

Zu erwarten wäre an dieser Stelle zumindest eine Symmetrie zwischen $vm3$ und $vm4$. Diese ist allerdings bei Messungen, die mit dem Host System $node7$der beiden VMs gemacht werden, in manchen Fällen in keinster Weise gegeben. So ist der Datendurchsatz der Messung $vm4 - no7$ etwa 10 mal größer als der der Messung $vm3 - no7$. Da hierbei zunächst von einem Konfigurationsfehler bzw. Unterschied ausgegangen werden muss, wurden alle gemachten Änderungen nochmals auf Gleichheit auf beiden Virtual Machines überprüft, es haben sich allerdings keine Unterschiede gefunden.

Unter Umständen liegt die Begründung für dieses Verhalten in kleineren Bugs der entsprechenden XEN Version. Im Bugtracker ist ein Fehler im Zusammenhang mit der TX Überprüfung der Virtual Machines zu finden, allerdings gibt es keine genaueren Angaben. Weitere Tests mit neueren XEN Versionen wären an dieser Stelle sinnvoll, falls die Performance gebraucht wird. Des weiteren ist möglicherweise durch ein ausgefeilteres Netzwerktuning noch etwas mehr Leistung zu erreichen. Die hier probierten Einstellungen hatten allerdings - außer dem Ausschalten der TX Überprüfung - keine merkbaren positiven Auswirkungen. Genauere Tests wären auch an dieser Stelle nötig, wenn möglich mit Kenntnissen über den genauen Netzwerkaufbau.

3.5 partdiff

Partdiff ist ein Tool, welches zur CPU Leistungsmessung partielle Differentialgleichungen mittels der Gauß-Seidel oder der Jacobi Methode lösen kann.

Dieses wird mit folgenden Optionen gestartet:

```
./partdiff-seq 0 2 100 1 2 8000
```

Startet man es einmal auf einer node oder einer vm, so braucht es etwa 200 ± 3 Sekunden bis zum Ergebnis. Es besteht kein feststellbarer Unterschied zwischen node und vm.

Falls man es zwei mal parallel auf dem selben Rechner ablaufen lässt (node/vm Kombination ist egal) ändert sich die benötigte Zeit bis zum Ergebnis nicht. Dies liegt am Zweiprozessorsystem, da auf jedem ein Programmablauf stattfinden kann.

Bei vier gleichzeitigen Programmstarts (node/vm Kombination ist wieder egal), benötigt es bis zum Ergebnis etwa 405 ± 5 Sekunden. Diese Zeit liegt leicht über dem erwarteten Wert von 400 Sekunden.

Man sieht hierbei, dass durch zusätzliche XEN VMs bei reiner Prozessorbelastung keine Nachteile auftreten. Das Ergebnis entspricht dem, als ob man alles auf dem Hostsystem laufen lassen würde.

3.6 PVFS2

3.6.1 Einrichten

Zunächst muss PVFS2 kompiliert und installiert werden

```
aptitude install gcc bison libdb4.4-dev make perl-modules
./configure --prefix=/opt/pvfs2
make -j 2
make install
```

Daraufhin kann MPI-IO installiert werden und das später für die Performancemessung genutzte `mpi-io-test` kompiliert werden.

```
./configure -enable-romio --with-file-system=ufs+nfs+pvfs2 --
   prefix=/opt/mpich2 --with-mpe --with-pvfs2=/opt/pvfs2/
make
make install

cd pvfs2/test
./configure --prefix=/opt/pvfs2/ --with-mpi=/opt/mpich2/
make client/mpi-io/mpi-io-test
```

3.6.2 Ergebnisse

Vor jeder Messung wird der PVFS2-Server neu gestartet und der storage-space gelöscht.

```
killall pvfs2-server
rm -r /pvfs2-storage-space/
/opt/pvfs2/sbin/pvfs2-server -f /opt/etc/pvfs2-fs.conf /opt/etc/
   pvfs2-server.conf-node6
/opt/pvfs2/sbin/pvfs2-server /opt/etc/pvfs2-fs.conf /opt/etc/pvfs2
   -server.conf-node6
```

Die Messungen werden dann mit folgenden Optionen durchgeführt:

```
/opt/mpi-io-test -b $((100*1024*1024)) -i 5 -f pvfs2://pvfs2/
   testfile
```

Zunächst wurde Untersucht, ob die angeschaltete Bridge für Performanceeinbrüche sorgt. Dafür wurde der Server auf node6 gestartet und von node7 sowohl mit, als auch ohne Bridge `mpi-io-test` gestartet.

	Ohne Bridge read bandwidth	Ohne Bridge write bandwidth	Mit Bridge read bandwidth	Mit Bridge write bandwidth
Messung 1	96	62	99	70
Messung 2	89	67	98	70
Messung 3	90	67	99	77
Messung 4	97	77	99	69
Messung 5	90	75	97	72

Tabelle 3.3: Vergleich von mit/ohne Bridge, alle Angaben in MB/s

Wie in Tabelle 3.3 zu sehen, treten durch die Bridge keine merkbaren Verlangsamungen bei der Benutzung von PVFS2 auf node7 auf.

Nun wird der selbe Test auf vm3 durchgeführt. Dabei ergibt sich für die Write bandwidth wie bei den Tests zuvor ein Wert von etwa $70MB/s$. Die Read bandwidth liegt allerdings deutlich unter den Ergebnissen von node7, es wird nur ein Durchsatz von etwa $35MB/s$ erreicht.

Dies ist vermutlich darauf zurück zu führen, dass schon bei den Messungen mit netperf nur ein Datendurchsatz von etwa 60% erreicht wurde. Dieser Wert ist nun bei der Benutzung von PVFS2 noch etwas schlechter, der Durchsatz der vm liegt nur noch bei ungefähr 35%.

4 Fazit

Die im Rahmen dieses Praktikums gemachten Erfahrungen mit XEN waren insgesamt sehr erfreulich.

Die Installation und Konfiguration des Hostsystems gestaltet sich unter Debian sehr problemlos. Auch das Einrichten von Virtuellen Maschinen mit Debian oder Ubuntu ist mittels weniger Konsolenkommandos möglich.

Auch die Performancemessungen hinterließen einen positiven Eindruck. Bei reiner CPU Belastung mit dem Programm zum lösen von partiellen Differentialgleichungen (partdiff), ist keine messbare Verlangsamung durch das ausführen auf Virtuellen Maschinen feststellbar. Der Vergleich der IO Leistung mittels iozone konnte zeigen, dass auch in diesem Bereich keine deutlichen Performance-Einbußen existieren. Im Gegenteil scheint sogar das Cachen bei Schreibzugriffen in den Virtuellen Maschinen so implementiert, dass diese bei kleinen Dateigrößen schneller sind.

Die Tests der Netzwerkperformance waren hingegen etwas problematischer. Zunächst traten sehr große Leistungseinbußen bei der Kommunikation zwischen verschiedenen Virtuellen Systemen auf dem selben Server auf. Dies konnte durch ein Abschalten der TX Überprüfung deutlich verbessert werden. Nach verschiedenen weiteren Tuning-Tests, die nicht erfolgreich waren, und darauf folgendes Zurückstellen der Einstellungen waren jedoch keine konsistenten Messergebnisse mehr zu erzielen. Es ergaben sich unterschiedliche Ergebnisse für eigentlich gleich konfigurierte virtuelle Maschinen. Dies könnte an einem Konfigurationsfehler oder an einem Bug in XEN liegen. Insgesamt sind im Netzwerkbereich die deutlichsten Leistungseinbrüche festzustellen - sie sind hoch genug, dass ein Betrieb mit PVFS2 nicht sinnvoll scheint. Der Datendurchsatz geht auf etwa 50% von dem eines nicht virualisierten Systems zurück.

A Ausführliche Messdaten

von-auf	Recv Socket Size bytes	Send Socket Size bytes	Send Message Size bytes	Elapsed Time secs.	Through-put MB/s	Utilization Send local % S	Recv remote % S	Service Send local us/KB	Demand Recv remote us/KB
no6-no6	87380	16384	16384	100.02	231.88	50.24	50.24	4.232	4.232
no7-vm3	87380	16384	16384	100.50	3.31	1.90	1.75	11.244	5.152
no7-vm4	87380	16384	16384	100.54	2.02	0.71	0.56	6.890	2.723
vm3-vm4	87380	16384	16384	100.34	1.64	0.03	0.09	0.181	0.534
no7-no6	87380	16384	16384	100.02	106.49	4.02	50.01	0.738	9.171
vm3-no6	87380	16384	16384	137.90	29.67	1.25	13.88	0.411	9.134
vm3-vm2	87380	16384	16384	100.25	28.00	5.13	8.83	1.789	3.080
no6-vm2	87380	16384	16384	100.01	11.63	9.33	9.18	15.663	7.705

Tabelle A.1: Netperf Messergebnisse vor Optimierungen

von-auf	Recv Socket Size bytes	Send Socket Size bytes	Send Message Size bytes	Elapsed Time secs.	Through-put MB/s	Utilization Send local % S	Utilization Recv remote % S	Service Send local us/KB	Demand Recv remote us/KB
vm3-no6	87380	16384	16384	20.03	112.19	16.74	49.68	1.457	8.650
vm3-vm2	87380	16384	16384	20.12	29.41	5.81	10.68	1.931	3.548
vm3-no7	87380	16384	16384	78.82	27.54	3.64	8.61	1.291	6.108
vm3-vm3	87380	16384	16384	20.01	338.34	99.95	99.95	2.885	2.885
vm3-vm4	87380	16384	16384	20.51	1.68	0.09	0.04	0.536	0.246
vm4-no6	87380	16384	16384	20.01	112.17	21.98	48.74	1.914	8.488
vm4-vm2	87380	16384	16384	20.01	28.79	5.10	9.70	1.730	3.290
vm4-no7	87380	16384	16384	20.01	293.56	99.95	99.95	3.325	3.325
vm4-vm3	87380	16384	16384	20.30	2.66	0.08	0.18	0.305	0.658
vm4-vm4	87380	16384	16384	20.01	294.32	99.95	99.95	3.316	3.316
no7-no6	87380	16384	16384	20.01	77.89	51.44	42.50	12.900	10.656
no7-vm2	87380	16384	16384	20.01	35.31	15.62	14.07	8.640	3.891
no7-no7	87380	16384	16384	20.01	322.94	85.68	85.68	5.182	5.182
no7-vm3	87380	16384	16384	20.01	31.76	51.60	49.08	31.733	15.090
no7-vm4	87380	16384	16384	20.01	31.95	51.67	50.62	31.587	15.473
vm2-no6	87380	16384	16384	20.01	161.88	17.85	49.93	1.077	6.024
vm2-vm2	87380	16384	16384	20.01	337.05	99.95	99.95	2.896	2.896
vm2-no7	87380	16384	16384	20.01	88.93	13.99	39.65	1.536	8.708
vm2-vm3	87380	16384	16384	20.22	29.80	5.27	10.27	1.729	3.365
vm2-vm4	87380	16384	16384	20.21	29.48	4.81	10.50	1.594	3.478
no6-no6	87380	16384	16384	20.01	323.53	86.43	86.43	5.218	5.218
no6-vm2	87380	16384	16384	20.00	35.35	34.71	33.91	19.179	9.368
no6-no7	87380	16384	16384	20.03	112.20	14.16	50.05	2.464	8.713
no6-vm3	87380	16384	16384	20.01	36.59	7.51	15.08	4.010	4.025
no6-vm4	87380	16384	16384	20.01	38.04	7.82	15.49	4.018	3.978

Tabelle A.2: Netperf Messergebnisse nach Optimierungen

B Quellen, Literatur

- Andrej Radonic und Frank Meyer, XEN3 (Franzis Verlag GmbH, 2006)
- Xen hypervisor
  `http://www.xen.org`
- Xen Wiki
  `http://wiki.xensource.com`
- Penguin User Group (PUG), XEN-Installation
  `http://www.pug.org/index.php/Xen-Installation`
- Netperf
  `http://www.netperf.org`
- IOzone Filesystem Benchmark
  `http://www.iozone.org`
- SelfLinux, Linux LVM-HOWTO
  `http://www.nextop.de/selflinux/html/lvm01.html`