INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Empirische Identifikation von Parametern mit Einfluss auf die Effizienz von Virtualisierung am Beispiel von Openvz-Virtuozzo

Evangelia Tsiouprou

Draft vom 4. März 2010

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Empirische Identifikation von Parametern mit Einfluss auf die Effizienz von Virtualisierung am Beispiel von Openvz-Virtuozzo

Evangelia Tsiouprou

Aufgabensteller:	Prof. Dr. D. Kranzlmüller
Betreuer:	Dr. Vitalian Danciu Dr. Nils gentschen Felde Tobias Lindinger
Abgabetermin:	23. Dezember 2009

Hiermit versichere ich, dass ich die vorliegende Projektarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 13. Januar 2010

. . . . . .

(Unterschrift des Kandidaten)

Zusammenfassung

In den letzten Jahren ist die Leistungsfähigkeit von Servern sehr stark gestiegen. Gleichzeitig ist die Auslastung von den meisten Maschinen meistens gering. Die Technologie der Virtualisierung ermöglicht es die nicht genutzten Ressourcen zu nutzen und gleichzeitig die Auslastung deutlich zu erhöhen. Die Vorteile der Nutzung der Virtualisierung sind eine geringere Anzahl an physischen Maschinen, Reduzierung der Stromkosten für die Betreiber von Rechenzentren und nicht zu vernachlässigen der ökologische Faktor, der eine immer größere Rolle erhält. Mit der Virtualisierung wird es ermöglicht eine einheitliche Konfiguration der Systeme zu erstellen, die das Management dieser erheblich vereinfacht. Ziel dieser Arbeit ist es, auf die Virtualisierungstools OpenVZ und Virtuozzo einzugehen und mit Hilfe von ausgesuchten Benchmarks das Verhalten von virtuellen Maschinen im Vergleich zu physischen Servern zu untersuchen. Es wird auf drei Benchmark-Tools eingegangen: Linpack, Iometer und Ramspeed. Diese drei Tools untersuchen die Leistung von CPU, Disk, Netz und RAM und können für die Anforderungen nach Bedarf angepasst werden. Laut Hersteller [PH09] beträgt der Virtualisierungsaufwand für Virtuozzo nur 1 bis 3 Prozent der gesamten Systemleistung und kann eine sehr effektive Systemauslastung erzeugt werden. In dieser Arbeit wird der Aufwand der Virtualisierung von OpenVZ und Virtuozzo im Vergleich zum nativen Server untersucht. Mit Hilfe von durchgeführten Messungen (CPU, Disk, Netz-Schnittstellen und Arbeitsspeicher) werden Aussagen gemacht über die gesamte Systemleistung, Performance und Effzienz von OpenVZ und Virtuozzo. Zuerst wird auf die Begriffe der Virtualisierung und Betriebssystem-Virtualisierung eingegangen. Danach wird das Konzept von OpenVZ-Virtuozzo erklärt und anschließend die Messungen präsentiert.

Inhaltsverzeichnis

1 Einleitung

Die in den letzen Jahren gestiegene Rechenleistung von Servern hat eine nicht dagewesene Erhöhung der Leistungsfähigkeit von Rechenzentren ermöglicht. Trotzdem werden bei Serverfarmen zu heutigen Zeitpunkt nur kleine Anteile der verfügbaren Ressourcen genutzt. Diese Diskrepanz führt zur intensiven Suche nach möglichen Auswegen um die Auslastung der Ressourcen deutlich zu erhöhen. Der Weg, der heute durch eine Mehrheit der Rechenzentren gegangen wird, ist die Erhöhung der Auslastung durch die Nutzung der neuen Technologien der Virtualisierung. Durch die Virtualisierung können Server konsolidiert und die Auslastung erhöht werden. Virtualisierung bedeutet, dass ein vorhandener physischer Server, in mehrere kleine virtuelle Maschinen aufgeteilt werden kann. Nach Popek und Goldberg [PG74] wird die Virtualisierung mit drei Kriterien definiert. Das erste Kriterium ist die Äquivalenz. Äquivalenz besagt, dass ein Programm, das in einer virtuellen Maschine läuft, ein identisches Verhalten aufweist, wie wenn das Programm direkt auf der Maschine läuft. Das zweite Kriterium ist die Ressourcenkontrolle. Dies bedeutet, dass die virtuelle Maschine die vollständige Kontrolle über die virtualisierten Ressourcen haben muss. Das dritte Kriterium ist die Effizienz. Eine virtuelle Maschine soll genau so effizient arbeiten können wie ein physischer Server. Ein gewünschter positiver Nebeneffekt der Nutzung der Virtualisierung ist, dass aufgrund der Virtualisierung weniger physische Maschinen benötigt werden, da die notwendigen Rechenaufgaben auf besser ausgelasteten Maschinen durchgeführt werden können. Die Migration der physischenn maschinen auf virtuelle Maschinen bietet viele Vorteile, wie der gesunkene Platzbedarf, reduzierte Strom- und Verwaltungsaufwände. Dadurch wird die Effizienz der Rechenzentren deutlich erhöht. Durch die Virtualisierung wird die direkte Verknüpfung mit dem Hostsystem beseitigt und Installationsänderungen vereinfacht. Die Virtualisierung eines gesamten Computersystems kapselt die Konfiguration in eine Datenstruktur, die einfacher zu verwalten ist und mehr Fähigkeiten für die Sicherung und die Widerherstellung anzubieten hat [Tho08]. Ist es aber wirklich so, dass eine virtuelle Maschine eine Rechenaufgabe genau so schnell und effizient bearbeitet wie ein physischer Server? Um diese Frage zu beantworten, muss die Performance der virtuellen Maschine im Vergleich zu einem physischen Server mit Verwendung der gleichen Hardware und Software Vorrichtungen getestet werden.

1.1 Ansatz

Um die Performance von virtuelle Maschinen zu testen, gibt es unterschiedliche Möglichkeiten. Diese Arbeit beschäftigt sich mit der theoretischen Benchmark-Performance auf der Basis von zwei bestimmten Virtualisierungs-Tools. Ein Virtualisierungs-Tool ist ein Software-Programm mit Hilfe dessen eine virtuelle Maschine gestartet und betrieben werden kann. Es gibt am Markt unterschiedlichste Virtualisierungs-Tools, mit unterschiedlichen Spezialisierungen. Die bekanntesten sind VMware, Xen, Hyper-V, ESX, Virtuozzo und OpenVZ. Benchmarks sind genormte Testverfahren, mit deren Hilfe die Leistung von Rechnern ermittelt und nach bestimmten Kriterien miteinander verglichen werden können. Bekannte Benchmark-Tests existieren für die Berechnung der CPU, Disk, Netz und RAM-Leistung.

Die richtige Auswahl eines Tools spielt eine wichtige Rolle, da jedes davon unterschiedliche Eigenschaften aufweist. Es muss also ein Tool ausgesucht werden, das bestimmte Eigenschaften aufweist, die für diese Arbeit wichtig sind. Multiplattformfähigkeit ist eine wichtige Eigenschaft unserer Arbeit. Multiplattform heißt unter anderem, dass das Virtualisierungs-Tool kompatibel mit allen gängigen Betriebssystemen ist. Für die Auswahl von großer Bedeutung ist, dass das Tool sowohl auf Windows als auch auf einer Unix-Plattform läuft, um eine Testumgebung zu erstellen mit der die Messungen, durchgeführt werden können. Aus den Messungen können Aussagen über die Ergebnisse in gleichen Umgebungen getroffen werden. Es werden einzelne Messungen durchgeführt, um die virtuelle Maschine mit dem nativen Server zu vergleichen und eine Reihe von parallelen Messungen durchgeführt, um die Skalierbarkeit von mehreren virtuellen Maschinen zu testen. Unter Skalierbarkeit von mehreren virtuellen Maschinen ist die Leistungsfähigkeit beim parallelen Betreiben von mehr als einer virtuellen Maschine auf einem physischen Server zu verstehen. Praktisch ausgedrückt heißt das, dass die Ressourcen konstant beibehalten werden, während die Parallelität der virtuellen Maschinen weiter anwächst. Nachdem alle Messungen durchgeführt sind, werden Schlüsse über die Performance von virtuelle Maschinen gezogen. Dies geschieht sowohl im Vergleich zu physischen Servern bezüglich der CPU Leistung, der Disk-, Netz- und Speicher-Performance als auch mit Bezug auf die Skalierbarkeit von mehreren virtuellen Maschinen auf einem nativen Server.

1.2 Realisierung

Das Thema dieser Arbeit ist das Virtualisierungs-Tool OpenVZ für eine Linux-Plattform und Virtuozzo für eine Windows-Plattform. Im Kapitel 2 wird das Konzept der Virtualisierung und speziell das Modell der Betriebssystemvirtualisierung erklärt. Im Kapitel 3 wird die Vorgehensweise und die Messmethodik der Arbeit geschildert, die verwendeten Benchmarks und die Messungen erklärt. Danach, im Kapitel 4 geht es um die Hardware und Software Vorrichtung, auf der die Virtualisierungs-Tools installiert werden. Die zwei Tools werden auf einen physischen Server installiert und bilden das Hostsystem, das die Basis für die Erzeugung von virtuellen Maschinen ist. Im Kapiteln 5 und 6 werden die einzelnen und parallele Messungen präsentiert und im Kapitel 7 die Interpretation der Ergebnisse dargestellt.

1.3 Ergebnisse

Nach der Durchführung aller einzelnen Messungen konnte die Performance der virtuellen Maschinen im Gegensatz zu dem nativen Server beobachtet und interessante Ergebnisse, in Bezug auf OpenVZ und Virtuozzo aufgezählt werden. Bei fast allen vier Benchmark Messungen(CPU, Disk, Netz und RAM) war die Leistung der virtuellen Maschinen im Vergleich zu den physischen Servern vergleichbar, was die Ausführungszeiten und die Transferraten betrifft. Die CPU, Disk, Netz und RAM Performance einer virtuellen Maschine ist im Fall von Virtuozzo und Openvz vergleichbar mit der Performance eines nativen Servers. Zu beachten ist, dass Virtuozzo ein geeigneter Virtualisierer ist, wenn keine heterogene Gastsystemlösungen realisiert werden sollen. Heterogenes Gastsystem bedeutet, dass der Mischbetrieb von unterschiedlichen Betriebssystemen wie Windows und Linux ermöglicht wird.

Neben den einzelnen Messungen wurde die Skalierbarkeit auf mehr als eine virtuelle Maschine mit den parallelen Messungen getestet. Die parallelen Messungen haben als Ziel zu

ermitteln, wie gut das System auf mehr als eine virtuelle Maschine skaliert werden kann. Alle parallele Messungen wurden unter Windows Server 2003(R2) durchgeführt. Die Performance von Virtuozzo in diesem Fall entspricht der Performance des nativen Servers beim Betrieb bis zu zwei virtuellen Maschinen. Sobald eine dritte Maschine zum Einsatz kommt, ist die Kernzuteilung beim Virtuozzo etwas problematisch. Virtuozzo skaliert am besten da die virtuelle Maschinen nur die Ressourcen beanspruchen, die sie tatsächlich benötigen. Sowohl der Overhead bei der Speichernutzung als auch bei der Prozessorbelastung ist hier minimiert, da die virtuellen Maschinen deutlich weniger Ressourcen des Hostbetriebssystems belegen. Der offensichtliche Nachteil ist, dass alle Gäste auf das Hostbetriebssystem festgelegt sind. Für den Einsatz bei Webhostern, die möglicherweise unterschiedliche Betriebsysteme zur Verfügung stellen wollen kann dies ein Hindernis für den Einsatz bedeuten. Die detaillierte Interpretation der Messungen wird im Kapitel 7 präsentiert.

2 Grundlagen

Es gibt verschiedene Arten der Virtualisierung. Die Voll- und die Betriebssystemvirtualisierung sind zwei Arten von vielen.

2.1 Betriebssystemvirtualisierung

Das Prinzip der Betriebssystemvirtualisierung unterscheidet sich vor allem in einem Punkt von anderen Virtualisierungslösungen wie zum Beispiel der Vollvirtualisierung. Die Schicht der Abstraktion setzt auf dem Hostbetriebssystem auf und nicht auf der physischen Hardware. Als Hostsystem werden hier die Betriebssysteme verstanden (Windows Server 2003, Ubuntu Server 9.04), die auf dem Rechner installiert sind. Die Schicht der Abstraktion liegt in diesem Fall über dem Hostsystem (Windows oder Linux-Server) und das führt meistens zur Verbesserung der Performance. Nachteilig ist aber, dass die virtuelle Maschine die Fähigkeit verliert, unwissend gegenüber anderen Maschinen zu sein. Bei der Betriebssystemvirtualisierung funktioniert somit, dass das Hostsystem zum Basisbetriebssystem wird und alle gehosteten Gastsysteme werden darauf gesetzt. Jeder gehostete Container ist wie ein Schnappschuss des Hostsystems und das bedeutet, dass die Nutzung von unterschiedlichen Betriebssystemen nebeneinander nicht möglich ist. Container heißt ein virtueller Server, der sih wie ein normaler Computer verhält mit IP-Adresse, Files, Konfiguration Dateien und Bibliotheken. Container teilen sich einen physischen Server und einen Betriebssystem Kernel, aber sie sind unabhängig von einander. Jeder Container einer virtuellen Maschine kann geändert werden, um ihre Konfiguration individuell anzupassen. Die Dateien jeder virtuellen Maschine, die sich auf dem Hostsystem befinden, in vielen Fällen dieselben Dateien, sind wie diejenigen, die den Hostsystem konstituieren. Wenn man eine Datei auf dem Hostsystem ändert, kann man damit zugleich die verknüpfte Datei innerhalb jeder virtuellen Maschine ändern. Das gemeinsame Betriebssystem wird in diesem Fall zu einem Punkt zentraler Kontrolle für alle gehosteten virtuellen Maschinen. Wenn ein System-Patch oder Service Pack auf das Betriebssystem nötig ist, aktualisiert die Installation dieses Updates in das Basisbetriebssystem automatisch jede virtuelle Maschine. Wenn eine der virtuellen Maschinen einen solchen Patch nicht braucht, kann sie davon ausgenommen werden. Die physische Hardware wird im Gegensatz zu Vollvirtualisierung nicht virtualisiert. Deswegen steigt auch die allgemeine Performance, weil es nicht notwendig ist, komplette Hardwareressourcen für jede virtuelle Maschine bereitzustellen. Der zusätzliche Overhead für die Bereitstellung der Ressourcen im Vollvirtualisierungsmodell verbraucht weitere Ressourcen. Bei der Betriebssystemvirtualisierung ist es nicht der Fall, weil hier ohne virtualisierte Hardware gearbeitet wird, die den Overhead der Anfragen zwischen virtuellen Maschinen und dem Hostsystem verursachen. Dieses Verfahren bedient sich der Partitionierung des vorhandenen Betriebssystems und nicht der vorhandenen Hardware. Somit werden bei der Erstellung der virtuellen Maschinen nur die Individualdaten für die VM angelegt. Alle Betriebssystemdaten (etwa Betriebssystem-Bibliotheken) des Hostsystems, die gleich sind, werden demnach von den

Gästen mitgenutzt. Abweichende Daten werden im Verzeichnis der virtuellen Maschine abgelegt. Dank dieser Technik besteht eine virtuelle Maschine in der Grundausstattung nur aus sehr wenigen Ressourcen und verbraucht deutlich weniger Festplatten- und Hauptspeicher als ein normal installierter Server. Der Benutzer hat wie bei der Vollvirtualisierung den Eindruck, dass er seinen eigenen Rechner besitzt. Es gibt aber ein Unterschied zu der Vollvirtualisierung. Der Benutzer sucht, beim Windows speziel, vergeblich nach einem Pagefile oder nach einer Registry. Der Diskmanager zeigt auch keine physischen Platten an. Die Zuteilung der physischen Ressourcen (CPU, Disk, RAM und Netz) erfolgt bei der Betriebssystemvirtualisierung dynamisch. Disk, RAM und CPU-Leistung können im laufenden Betrieb einer virtuellen Maschine verändert werden ohne, dass diese neu gestartet werden. Bei der Vollvirtualisierung ist es nicht möglich. Da keine Hardware Ressourcen emuliert werden müssen, entfällt der Performanceverlust, der mit der Emulation verbunden ist. Grundsätzlich teilen sich alle virtuelle Maschinen bei der Betriebssystemvirtualisierung alle physischen Ressourcen. Wenn zum Beispiel ein Hostbetriebssystem 4 GB RAM, 1 TB Festplattenspeicher und 4 CPUs, so stehen diese physischen Ressourcen alle virtuellen Maschinen zu Verfügung. Diese Ressourcen werden von dem Virtualisierungsschicht quotiert, so dass eine virtuelle Maschine nicht beispielsweise 95 Prozent des freien Diskplatzes verwenden kann.

> Zitat: In der Regel ist die Betriebssystemvirtualisierung eine bessere Alternative, da sie sparsamer mit den physischen Ressourcen umgeht und eine effektivere Rechen- und I/O-Leistung für die Aufgabe zur Verfügung hat. Betriebssystemvirtualisierung steht für sehr hohe Ausnutzung der Systemressourcen beziehungsweise geringen Virtualisierungsschwund.

Ferner ist aufgrund der geringen Leistungsansprüche der Gastsysteme eine sehr hohe Anzahl virtuelle Maschinen (im Vergleich zur Vollvirtualisierung) bei verhältnismäßig bescheidender Hardwareausstattung möglich. Eine bessere Performance und optimierte Ressourcenauslastung haben aber ihren Preis. Wie schon erwähnt muss jede virtuelle Maschine das gleiche Betriebssystem (oft auch das gleiche Service Pack) wie das Hostsystem haben. Parallels Virtuozzo Containers sowohl für Windows als auch für Linux und OpenVZ für Linux sind Beispiele für eine Lösung, die eine Architektur der Betriebssystemvirtualisierung beinhaltet [Sol]. Die nächste Abbildung zeigt die Virtualisierungstechnik von Betriebssystemvirtualisierung (vgl. Abbildung 2.1);

> Zitat: Laut Hersteller Parallels Virtuozzo Containers besitzt die Betriebsystemvirtualisierung eine bessere Performance, Verwaltung und Effizienz im Vergleich zu anderen Techniken, die nicht mit dem Betriebssystemvirtualisierung arbeiten.

Die Abbildung 2.1 zeigt ganz unten die Hostsystemschicht, die entweder ein Windows Betriebssystem oder Linux sein kann. Danach folgt die Virtualisierungsschicht mit einem proprietären Dateisystem und einem Kernel, der für die Isolierung und die Sicherheit von Ressourcen unterschiedlicher virtuellen Maschinen zuständig ist. Diese Schicht lässt die virtuelle Maschinen als eigenständigen Server erscheinen. Schließlich kommen die virtuellen Maschinen mit ihren eigenen Applikationen. Die Virtualisierungsschicht erzeugt auf dem Server und der Betriebssysteminstanz zahlreiche isolierte Umgebungen. Die Umgebungen werden auch Container genannt.

> Zitat: Die Vorteile dieser Technologie ist, dass die Hardware nicht virtualisiert und das Betriebssystem nur einmal auf dem Server installiert werden muss und

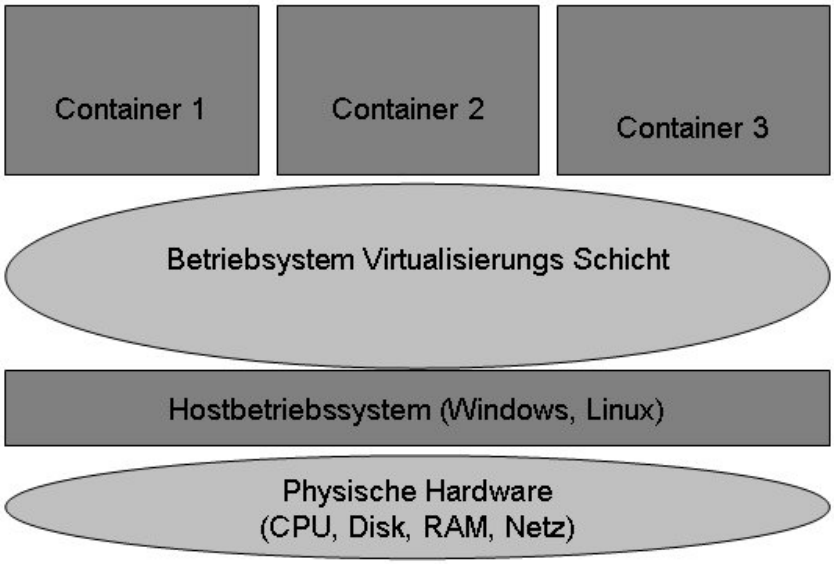

Abbildung 2.1: Betriebssystemvirtualisierungstechnik

nicht zusätzlich in jedem einzelnen Container. Bei der Betriebssystemvirtualisierung werden weniger Ressourcen pro Container benötigt als bei der Vollvirtualisierung der Fall ist.

2.2 Vollvirtualisierung

Vollvirtualisierung fußt auf der Idee, Virtualisierung durch eine Schicht unterhalb des Betriebssystems zu implementieren. Bei der Vollvirtualisierung agiert diese Virtualisierungsschicht - oft als Hypervisor bezeichnet - als eine Art Vertreter zwischen einzelnen virtuellen Systemen, die sich darüber befinden, und physischen Ressourcen, die sich darunter befinden. Im Falle der virtuellen Maschinen ist jede einzelne virtuelle Maschine komplett und völlig getrennt von den anderen. Die virtuelle Maschine besitzt kein Wissen darüber, dass sie mit anderen virtuellen Maschinen auf dem Hostsystem angesiedelt ist. Jede Maschine weiß lediglich, dass wenn sie eine Anfrage über physischen Ressourcen (CPU, Festplatte, Speicher oder Netz) abschickt, diese Anfrage von einer von ihr als solche eingeschätzten physischen Ressource stammt. Die virtuelle Maschine wird auf einer Schicht direkt oberhalb der physischen Hardware abstrahiert, daher besteht einer der Vorteile von virtuellen Maschinen darin, dass sie agnostisch gegenüber Betriebssystemen sind. Der Hypervisor sorgt dafür, als Vertretung Anfragen zwischen Code in der virtuellen Umgebung und ihren physischen Ressourcen zu behandeln. Bei der Vollvirtualisierung beinhaltet jede einzelne virtuelle Maschine alle Ressourcen, die sie benötigt, um selbst innerhalb ihrer Virtualisierungsumgebung laufen zu können. Für fünf Instanzen eines Betriebssystems werden also fünf Kopien der Dateien des Betriebssystems und anderer Konfigurationen benötigt. Da jedoch jede virtuelle Maschine isoliert ist und die Aufrufe an physikalische Ressourcen über einen Hypervisor vorgenommen werden, ist es möglich, Maschinen von verschiedenen Betriebssystemen auf demselben

Host auszuführen. Es gibt viele Marktlösungen, die auf dieses Prinzip der Vollvirtualisierung basiert sind. Die berühmtesten davon sind der Microsoft Virtual Server und VMware ESX Server. [Fuj09] Die Abbildung 2.2 zeigt die Technik der Vollvirtualisierung.

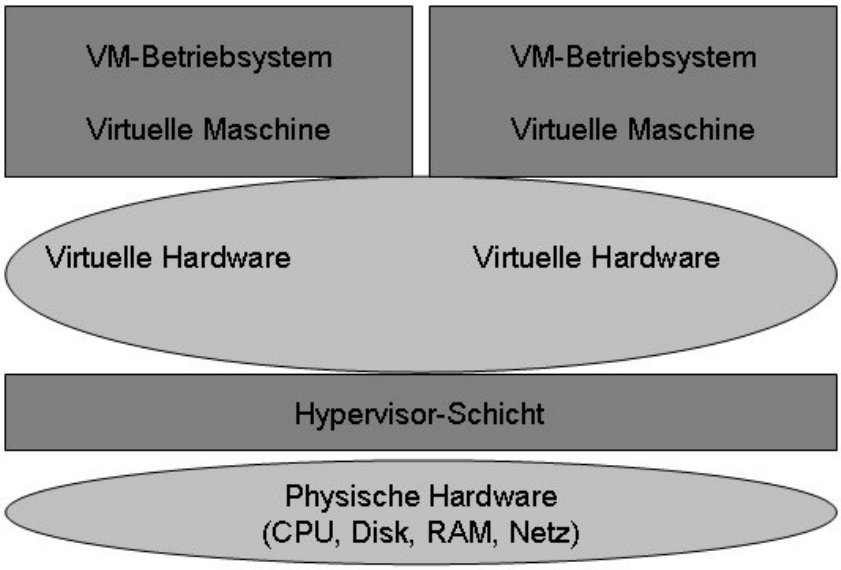

Abbildung 2.2: Vollvirtualisierungstechnik

Um die Performance von virtuelle Maschinen zu messen und die nötigen Beweise zu liefern, wird in dieser Arbeit der OpenVZ für eine Linux Plattform und Virtuozzo 4.5 für eine Windows Plattform als Virtualisierungslösung gewählt. Zunächst folgt eine Erklärung wie OpenVZ und Virtuozzo funktioniert und auf das System installiert ist. Danach kommen die Messungen, die im Rahmen dieser Arbeit durchgeführt wurden. Zum Schluss werden die Tests von den virtuellen Maschinen mit den Messungen von nativen Servern verglichen. Nur dann ist es möglich, die Aussagen von den Hersteller, was die Performance, Skalierbarkeit und Effizienz von OpenVZ und Virtuozzo als Virtualisierungslösung angeht, zu bestätigen oder zu widerlegen.

2.3 OpenVZ

OpenVZ ist ein freies Virtualisierungstool für Linux-Systeme, das auf den Kernkomponenten der Virtuozzo-Version für Linux basiert. Es handelt sich um eine kostenlose Open-Source Community Lösung. OpenVZ erzeugt eigenständige und isolierte virtuelle Umgebungen, (die auch Virtual Private Server (VPS) heißen) auf einem einzelnen physischen Server. Jede Maschine verhält sich genau so wie ein physischer Server, weil jeder von sich unabhängig voneinander mit eigene IP-Adresse, Speicher und Root-Zugriffe gestartet werden kann. OpenVZ benutzt das Ein-Kernel-Modell und ist deswegen wie der Linux-Kernel 2.6 skalierbar. Es gelten die gleichen Beschränkungen wie beim 2.6er Kernel (max 64 CPUs und 64 GB RAM). Eine einzelne virtuelle Maschine kann auf das komplete Hostsystem skaliert werden, d.h alle CPUs und das gesamte RAM des Hostsystems nutzen. Diese Vorgehensweise virtualisiert die Hardware der virtuelle Maschine. Das Gastsystem der virtuelle Maschine greift nicht mehr direkt auf die physische Hardware des Hostsystems zu, sondern nutzt die Schnittstellen von OpenVZ. Auf diese Weise ist es möglich, einen Server zur Laufzeit zu migrieren, um gestiegene Ressourcen zu nutzen oder um Hardware Ausfälle des Hostsystems zu kompensieren [Bau07].

2.3.1 Installation und Einrichtung

Generell kann OpenVZ auf jedem Rechner betrieben werden, auf dem ein Linux mit Kernel 2.6 läuft. Pro Kernelversion stehen mehrere Versionen für verschiedene Rechnerarchitekturen bereit. Von der openVZ-Seite [ope] können verschiedene Versionen herundergeladen werden. Für die Testmessungen wurde der Linux-Kernel der Version 2.6.24-22-openvz ausgesucht und installiert, weil er bei dem Zeitpunkt der Installation die neuste Version war. Mit dem unten angegebenen Befehl kann man den openVZ-Kernel installieren [SWs].

```
apt-get install 2.6.24-22-openvz
```

Vor der Linux-Kernel Installation wurde das Hostsystem Ubuntu 9.04 Server Edition installiert. Ausgesucht wurde das Hostsystem Ubuntu 9.04 Server Edition aufgrund dessen, weil Ubuntu ein etabliertes und stabiles System ist. Danach die Programme zur Administration des OpenVZ-Systems. Die folgenden OpenVZ Tools sind wichtig für die Verwaltung der virtuellen Maschinen und müssen installiert werden.

```
apt-get install vzctl
apt-get install vzqouta
```

vzctl: mit diesem Paket können Arbeiten durchgeführt werden wie Erzeugen, lLöschen, Starten oder Stoppen der VMs.

vzqouta: das Tool für die Speicherplatzverwaltung.

Bevor die ersten VPS eingerichtet werden können, muss von dem installierten Betriebssystem ein Template erstellt werden. Templates sind die Vorlagen, von denen später die VPS Instanzen abgeleitet werden. Ein Template stellt die Beschreibung dar, wie das Grundsystem einer Distribution aussehen muss, um in einem OpenVZ als VPS verwendet zu werden. Templates können auch OpenVZ eigene Pakete enthalten, die in den VPS installiert werden müssen, um Funktionen von OpenVZ zu nutzen. Um ein Template verwenden zu können,

muss die Datei im Verzeichnis /vz/template/cache abgelegt sein. Die Datei kann von der OpenVZ Webseite (http://openvz.org/download/template/cache/) heruntergeladen werden. Installiert wurde für diese Arbeit das Template:

```
ubuntu-9.04-i386-minimal
```

Nun ist das OpenVZ System installiert und das zugehörige Template eingespielt. Es ist jetzt an der Zeit eine virtuelle Maschine zu erzeugen. Bevor eine virtuelle Maschine erzeugt wird ist es sehr wichtig die Netzkonnektivität des Servers zu überprüfen. Jede virtuelle Maschine braucht auch eine eigene IP-Adresse. Wenn die virtuelle Maschine nicht nur vom Host-Server erreichbar sein soll, muss es sich um eine öffentliche IP-Adresse handeln oder eine entsprechende Weiterleitung eingerichtet worden sein. Grundsätzlich gliedert sich die Erzeugung einer virtuelle Maschine in drei Schritte.

1. Auswahl einer ID für die VM (über welche Nummer die VM angesprochen wird) Bei der ID handelt es sich um eine 32-Bit Integer Zahl. Für die richtige Wahl einer ID muss man aber folgendes beachten: Erstens ist die ID mit der Nummer 0 ausgeschlossen, weil sie für den Host-Server reserviert ist. Zweitens die IDs von 1 bis 100 sind für Openvz interne Zwecke reserviert. Um Updates zu erleichtern sollte man also keine ID unter 101 vergeben. Drittens ist die ID pro physischen Server eindeutig. Es kann zum Beispiel nur eine VM pro Host-Server mit der ID 101 geben. Auf einem anderen Server kann aber ohne Probleme eine VM mit der ID 101 geben, aber pro Maschine ist die Nummer eindeutig. Man kann aber mit dem unten angegebenen Befehl schon überprüfen, ob die ID schon vergeben ist.

   ```
   vzlist -a 101
   VPS not found
   ```

   Die Antwort ist, dass es keine virtuelle Maschine mit der ID 101 gibt.

2. Auswahl eines Templates für die VM (welche Distribution verwendet wird) Das ubuntu-9.04-i386-minimal ist die Template-Version die für die Installation genutzt wird. Durch die Eingabe des folgenden Befehls wird eine neue virtuelle Maschine erzeugt. vzctl create 101 – ubuntu-9.04-i386-minimal -config Creating VPS private area VPS private area was created

   In diesem Fall erstellt OpenVZ eine VM mit der ID 101 und das Template ubuntu 9.04-i386-minimal. Im Prinzip kann man jetzt mit dem Befehl:

   ```
   vzctl create 101
   ```

   die Maschine erzeugen, jedoch ist im Normalfall eine IP-Adresse und ein Hostname nötig. Außerdem sollte man mit dem root Passwort setzen, bevor die virtuelle Maschine das erste Mal gestartet wird.

3. Erzeugung der VM. Nun werden die folgenden Einstellungen noch gebraucht bis die virtuelle Maschine vollkommen konfiguriert ist. Diese werden mit dem Befehl:

   ```
   vzctl set
   ```

durchgeführt. Um über ein Netz auf die virtuelle Maschine zugreifen zu können, muss man eine IP-Adresse und einen Hostnamen konfigurieren. Mit den folgenden Befehlen:

```
vzctl set 101 --hostname test101 --save
vzctl set 101 --ipadd 10.0.174.4 --save
```

ist der Hostname test101 und die IP-Adresse 10.0.174.4 konfiguriert. Die bereits erwähnten Befehlen zum starten und stoppen der virtuellen Maschine lauten:

```
vzctl start 101
vzctl Stopp 101
```

Außerdem kann man mit dem unten angegebenen Befehlen ein Passwort setzen und die Liste der bereits erzeugten Maschinen anschauen.

```
vzctl set 101 -userpasswd root:test
```

In diesem Fall lautet das Passwort test.

```
vzlist -a
```

Der Status aller gestarteten oder gestoppten Maschinen wird angezeigt.

Auf diese Weise sind die virtuelle Maschinen unter Verwendung von OpenVZ für die Messungen im Kapitel 6 und 5 konfiguriert.

2.4 Virtuozzo

Virtuozzo ist ein Virtualisierungstool, welches ermöglicht sowohl in Linux als auch unter Windows-Systemen virtuelle Server einzurichten. Es bietet ein einfaches und effektives Verfahren, um System- und Softwareinstallationen zu verwalten und stellt eine Web-Oberfläche bereit, über die der Administrator die virtuellen Maschinen verwalten kann.

> Zitat: Laut Hersteller umfasst Virtuozzo Fähigkeiten und Werkzeuge, die für den Virtualisierungsbetrieb benötigt werden. Diese beinhalten Ressourcen-Limits, damit man eine bessere Leistung ermöglicht wird. Außerdem die Möglichkeit zur Live-Migration von eine bis hin zu mehreren Maschinen zwischen verschieden Hostsysteme. Die Realisierung von integrierten Backup-Funktionen und eine einheitliche Administration zur Verwaltung einzelner VPS bis zur Kompletten Administration eines Clusters aus zahlreiche Hostsystemen.

Für diese Arbeit wird Virtuozzo verwendet, um die Tests unter Windows laufen zu lassen. Das benutzte Hostsystem unter Virtuozzo 4.5 ist der Windows Server 2003 jeweils für x64 und x86 Bit. Nach der Hostsysteminstallation, eine Windows Server 2003(R2) wurde die entsprechende Installationsdatei(32 Bit,64 Bit) von der Seite (http://www.parallels.com/de/download/virtuozzo4/) runtergeladen und installiert[PH09].

2.4.1 Installation und Einrichtung

Die Installation von Virtuozzo 4.5 ist sehr einfach und nicht so mühsam wie es beim OpenVZ der Fall war. Mit den Programmen:

```
virtuozzo4.5_x64.exe
virtuozzo4.5_x86.exe
```

die auf der Virtuozzo Seite (http://www.parallels.com/de/download/) verfügbar sind, können die beide Versionen (32,-64 Bit) von Virtuozzo installiert werden. Die nächste Abbildung 2.3; zeigt ein Installations Fenster. Während der Installation von Virtuozzo müssen zwei

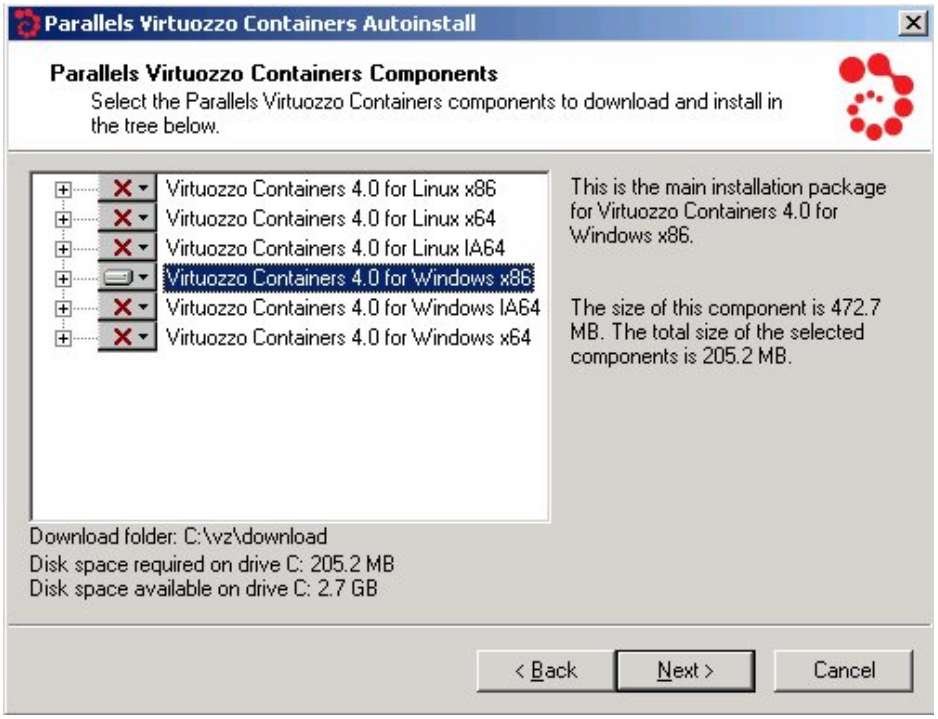

Abbildung 2.3: Installations Beispiel für Windows

Verzeichnisse spezifiziert werden:

```
C:\Program Files\Parallels\Containers
C:\vz
```

Das erste Verzeichnis beinhaltet Dateien wie Treiber, Skripte und Dienstprogramme für Virtuozzo 4.5. Das zweite Verzeichnis beinhaltet alle Daten, die der Container braucht um virtuelle Maschinen zu erzeugen. Das Verzeichnis kann nicht gemountet werden. Nach der Virtuozzo 4.5 Installation sind zwei Tools für die Verwaltung der Container nötig. Gebraucht wird das Tool Parallels Management Console und Parallels Infrastructure Manager. Das erste Tool ist ein Remote Management Tool mit einer graphischen Oberfläche. Die Konsole verwaltet virtuelle Maschinen auf zahlreichen Servern und Maschinen von einer einzelnen Benutzeroberfläche aus. Der Manager hat die Möglichkeit einen physischen Server und alle gehosteten Container mit einem Web-Browser zu verwalten. Die zwei Tools ermöglichen beim

Virtuozzo eine schnellere und einfache Erzeugung, Löschung oder Änderung einer virtuellen Maschine. Der Vorteil ist natürlich die Übersichtichkeit, weil eine graphische Oberfläche viele Arbeiten übersichtlicher macht im Gegenteil zu OpenVZ, bei dem alle Arbeiten mit speziellen Befehle durchgeführt werden. Die nächste Abbildung 2.4; zeigt wie einfach beim Virtuozzo die Erzeugung eines Containers sein kann.

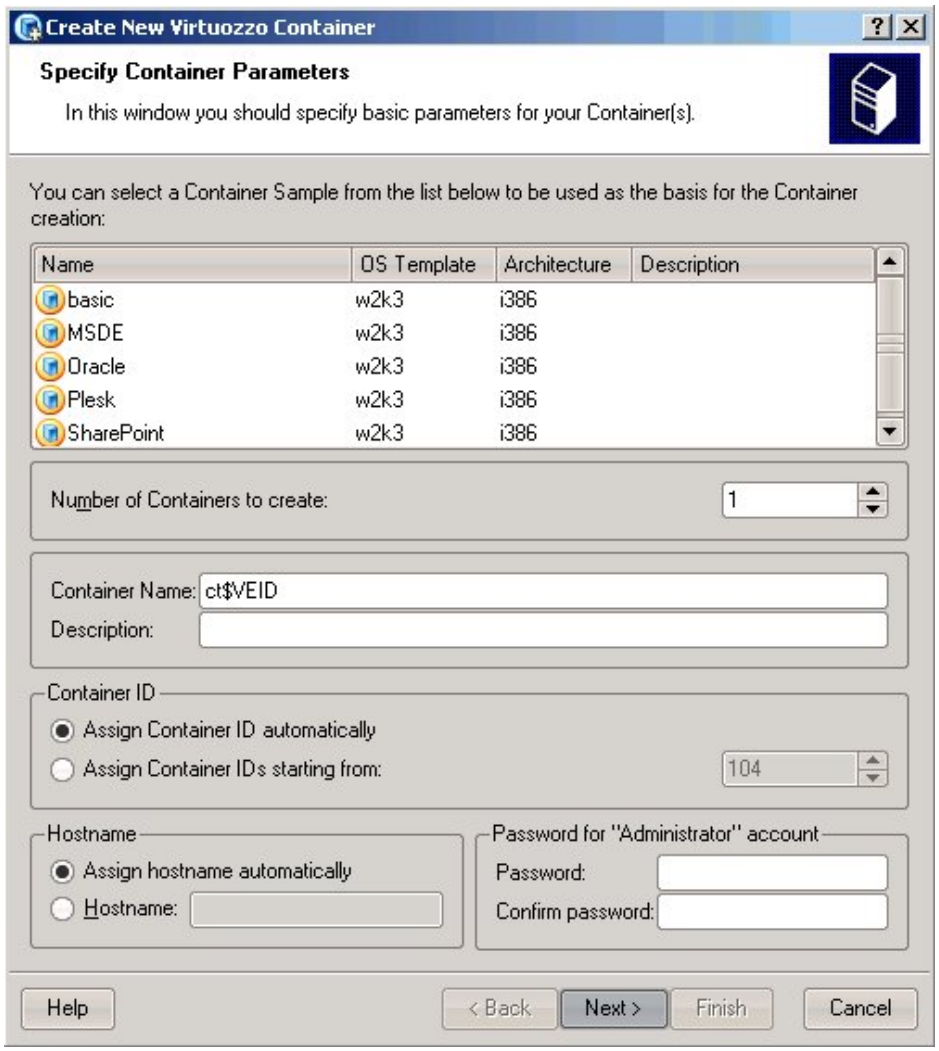

Abbildung 2.4: Erzeugung eines Containers mit dem Management Tool

3 Vorgehensweise

Um die Performance eines physischen Servers im Vergleich zu einer virtuellen Maschine zu messen, ist es notwendig bestimmte Benchmarks zu finden. Die Benchmarks sind die Programme, die bestimmte Eigenschaften testen um diese anschließend vergleichen zu können. Es existieren zwar diverse Tools und Benchmarksuiten zum Beurteilen der Computer-Performance, aber sie sind weder für virtuelle Maschinen angepasst worden noch laufen sie unter allen relevanten Betriebssystemen. Oft sind sie, wie einige Ausgaben von Linpack, auf einen bestimmten Hersteller (im Beispiel von Linpack Intel) zugeschnitten. Gemessen wird die Performance einzelner Komponenten, die für die Gesamtleistung eines Systems relevant sind. Zwar spiegeln solche Messungen nur theoretische Leistung wider, aber die ausgesuchten Tools sollten sowohl unter Linux als auch unter Windows laufen. Für diese Arbeit sind folgende Benchmarks ausgesucht worden: Iometer für Disk- und Netz-Messungen, Linpack für CPU-Messungen und Ramspeed für die Arbeitsspeicher-Messungen. Der Vorteil dieser Benchmarks liegt auf der Hand: ihr Quellcode ist frei erhältlich und, wie vorhin erwähnt, laufen sie sowohl unter Windows- als auch unter Unix-basierten Betriebssystemen. Ein möglicher Nachteil ist, dass die Benchmarks nur die theoretische Leistung testen. Die Leistung entspricht nicht ganz einem echten Anwendungsfall. Dafür sind sie aber einfach zu installieren und auszuführen. Bei vielen großen Benchmarksuiten (zum Beispiel VMmark) müsste man zunächst diverse Anwendungen installieren.

3.1 Messungen im Detail

In den nächsten Abschnitten werden die Komponenten CPU, Disk, Netz und RAM im Detail präsentiert. Alle diese vier Komponenten sind relevant für die Gesamtleistung eines Systems. Dabei sind, die Performance der CPU, die Transferraten für Disk und Netz und die Speichergeschwindigkeit, die Anhaltspunkte dieser Arbeit.

3.1.1 CPU Performance

Die CPU Leistung für einen Rechner ist ein entscheidender Faktor um die Performance eines gesamten Systems zu beschreiben. In der Regel betrachtet man sowohl die Interger- als auch Fließkommaoperationen. Für diese Arbeit wurde die Interger Leistung außer Acht gelassen und auf die Fließkommaoperationen eingegangen. Die CPU Messungen werden mit dem Linpack Tool ausgeführt. Das Tool entstammt dem Linpack Projekt von 1979, der Autor ist Jack Dongarra. Die ursprüngliche Version ist in Fortan geschrieben. Es gibt allerdings bereits Java und C Portierungen. Ermittelt wird die Leistung der CPU beim Lösen eines linearen Gleichungssystems bei vorgegebener Komplexität. Für die Messungen wurde die aktuelle Version verwendet. Sie wurde dahingehend angepasst, dass die Zeitgebung von einem Zeitserver geholt wird. Der Grund liegt darin, dass durch das komplexe Scheduling in der virtuellen Maschine die Zeitzyklen verfälscht werden können. Ziel der Berechnungen sind vier Matrizendurchläufe mit den im Voraus festgelegten Größen (Matrizen in beiden zweier

Dimensionen verdoppelt) 1000x1000, 2000x2000, 4000x4000 und 8000x8000. Es sind im ideal Fall 10 Durchläufe pro Matrizengröße durchgeführt worden. Die Durchläufe werden sowohl unter Ubuntu 9.04 Server Edition (nativ) und Ubuntu-Openvz für Linux, als auch unter Windows Server 2003 R2 (nativ) und Virtuozzo für Windows jeweils für 32 und 64 Bit realisiert. Wichtig ist es auch zu erwähnen, dass die maximale Zeitdauer pro Matrizenberechnung auf dreißig Minuten beschränkt wurde.

3.1.2 Disk und Netz Messungen

Für die Disk und Netz Messungen wird das Iometer Tool verwendet[iom]. Iometer ist ein komplexes Benchmarkprogramm, mit dem sich durch Erstellen von verschiedenen Zugriffsprofilen fast jedes erdenkliche Szenario testen lässt. Das Tool wurde von Intel entwickelt und später 2001 als Open Source Tool übergegeben. Wir beschränken uns auf vier Tests für die Disk Messungen: zufälliges Lesen und Schreiben und sequenzielles Lesen und Schreiben von Blöcken. Die Blockgröße beträgt 4 KB, 32 KB, 256 KB, 1 MB, 4 MB. Die virtuelle Maschine führt Schreib-/Leseoperationen aus. Die Zeitdauer für eine Messung beträgt 15 Minuten für sequenzielles Lesen, 15 Minuten für sequenzielles Schreiben, 15 Minuten für zufälliges Lesen und 15 Minuten für zufälliges Schreiben und die Gesamtdauer beträgt 1 Stunde. Genau wie beim Linpack kommt hier wegen VSchedulling die Client-/Server-Architektur zum Einsatz: Auf dem Router läuft die Serverkomponente Iometer GUI, in virtuelle Maschine ein Agent namens Dynamo. Die Netz Messungen laufen im Prinzip genau wie die Diskmessungen ab mit einen kleinen Unterschied. Für die Messungen wird eine dedizierte Netzleitung verwendet. Die dafür verwendete Ethernetkarte ist die Intel(R) PRO/1000 MT Desktop Adapter 2.

3.1.3 RAM Messungen

Eine ebenfalls wichtige Rolle für die Gesamt-Performance eines Rechners spielt der Hauptspeicher. Insbesondere für Datenbankserver, aber auch für diverse Anwendungen, die mit großen Datenmengen operieren müssen, ist Speicher-Geschwindigkeit entscheidend. Hauptspeicher Performance ist relativ einfach zu messen, sofern man einige grundlegende Voraussetzung beachtet. Wir ermitteln sie mit dem Open-Source Kommandozeilen-Dienstprogramm RamSpeed. Es unterstützt drei Hardware-Plattformen (i386, AMD64, Alpha) und mehrere populäre Unix-ähnliche Betriebssysteme. Das Programm versucht den Arbeitspeicher voll auszulasten. Die Grundidee ist, dass synthetische Benchmarks mit solchen kombiniert werden, die die tägliche Arbeit am Rechner simulieren. Wir setzen zwei synthetische Tests ein: INTmark und FLOATmark. Je nach Test werden unterschiedliche Bereiche der CPU gefordert. Dabei werden Lese- und Schreibraten ermittelt, indem sequentielle Datenströme durch ALU- beziehungsweise FPU-Einheit geschickt werden. Zunächst wird der erstbeste freier Speicherbereich alloziiert, der groß genug ist. Anschließend werden Speicherblöcke fester Größe gelesen bzw. geschrieben. Bei diesen Tests werden nur Blöcke in zweier Dimensionen verdoppelt (1 KB, 2 KB, 4 KB, ..., 512 KB, 1 MB,..., bis zu 32 MB). Diese Synthetische-Benchmarks sind in der Praxis allerdings nicht von großer Bedeutung - sie zeigen lediglich die theoretischen Speichergeschwindigkeiten. Um die Lese- bzw. Schreib-Leistung bei echten Anwendungen zu testen, werden daher weitere zwei Tests durchgeführt: INTmem sowie FLOATmem. Jeder von ihnen besteht aus vier Teilschritten: Copy, Scale, Add and Triad (Kopieren, Skalieren, Hinzufügen und Triad). Zwar handelt es sich dabei ebenfalls um syn-

thetische Messungen, sie sollten aber mit vielen echten Anwendungenszenarios korrelieren. Sie kommen in ähnlicher Form im weit verbreiteten SiSoft Sandra Benchmark zum Einsatz. Beide Memory Benchmarks werden im Batchmodus ausgeführt, um Durchschnittswerte über Durchläufe zu ermitteln. Alle speicherrelevanten Ergebnisse werden in MB/s angegeben, d.h. wie viele MB in 1 Sekunde übertragen werden können. [ala]

3.1.4 Parallele Messungen

Normaleweise lassen sich mehrere virtuelle Maschinen auf dem gleichen Rechner laufen. Also messen wir mit zwei und dann mit drei virtuelle Maschinen (mehr virtuellen Maschinen würden nicht gehen, da die Spezifikation vordefiniert wurde für einen einheitlichen Vergleich mit den Ergebnissen der anderen Virtualisierer). Bei solchen Messungen sieht man recht schnell wo die Vor- und Nachteile der Lösung liegen. Beispielsweise bricht bei voller CPU Auslastung der Netzwerkdurchsatz ein. Dann will man wissen, wie der Netzdurchsatz skaliert, wenn mehrere Rechner parallel über den gleichen Netzinterface kommunizieren. Man will also sehen, wie gut sich eine Lösung schlägt, bei diversen Arten der Auslastungen (CPU durch Linpack, oder 3 x Disk Benchmark, 3 x Netz Benchmark, oder Netz und Disk gleichzeitig). Um die Skalierbarkeit von Virtuozzo zu testen sind die nachfolgenden Messungen nötig. Die Spezifikation der parallele Messungen ist folgendermaßen definiert:

- Alle Parallele Messungen werden unter Windows 32 Bit durchgeführt.
- Linpack auf 2 und 3 virtuelle Maschinen laufen lassen.
- Iometer (20 m limitiert) auf 2 und 3 virtuelle Maschinen (Disk und Netz).
- Ramspeed auf 2 und 3 virtuelle Maschinen laufen lassen.
- Iometer (Netz) und die GUI läuft in eine weitere VM.
- Iometer (Netz) und auf 2 VM (Linpack).

3.2 Messmethodik

Die theoretische Performance einzelner Komponenten wird in dieser Arbeit gemessen. Alle Tests laufen unter gleichen Voraussetzungen: die Betriebssysteme werden im Auslieferungszustand installiert und es werden keine Updates eingespielt. Es gibt zwei Arten der Messungen. Die synthetischen Messungen testen, wie im Kapitel 3.1 ausführlich erklärt, die CPU, Disk, Netz und RAM Performance eines physischen Servers und einer virtuellen Maschine sowohl unter Linux als auch unter Windows Plattform und die Ergebnisse der Messungen werden verglichen. Die Faktoren, die bei dieser Messungen eine wichtige Rolle spielen, sind beim Linpack die Ausführungszeiten und beim restlichen Tests die Transferraten, die bei der Performance einer Maschine wichtig sind. Getestet wird, ob eine virtuelle Maschine im Vergleich zu einem physischen Servers genau so gut abschneiden kann oder auch nicht. Im Idealfall läge der Wirkungsgrad bei knapp hundert Prozent. Allerdings entspricht diese Situation nicht dem Normalfall, immerhin möchte man mehrere virtuelle Maschinen auf einem einzelnen Rechner betreiben. Die parallelen Messungen testen das Scheduling des Virtualisieres auf Herz und Nieren und zeigen, ob einzelne Komponenten CPU, Netz usw. optimal

zusammenarbeiten. Deswegen braucht man die parallelen Messungen die noch viel wichtiger als die einzelnen sind, weil man sich als Ziel setzt, nicht einzelne Komponenten zu verbessern, sondern ein optimales Zusammenspiel von Faktoren wie Prozessorzeit, Speicher, Netzperformance und anderen Faktoren zu erreichen. Beim parallelen Rechnen bekommt auch der Begriff des Durchsatzes eine andere Bedeutung als im einzelnen Bereich. Während im sequentiellen Teil Ausführungszeit und Durchsatz im Prinzip nur zwei verschiedene Ausdrucksformen für das gleiche Ergebnis waren, ergeben sich im parallelen Bereich Unterschiede zwischen diesen zwei Begriffen. Man ist hier daran interessiert möglichst viele Programme in möglichst geringer Zeit fertig gestellt zu haben, während die Ausführungszeit eines einzelnen Programms nicht mehr eine so große Rolle spielt wie bei der sequentiellen Berechnung. Es ist z.B. nicht sinnvoll die Ausführungszeit auf einem Prozessor zu verbessern, wenn dies zu erhöhter Berechnungszeit oder Ruhezeit anderer Prozessoren führt. Im Kapitel 6 werden verschiedene parallele Messungen vorgestellt, um zu zeigen wie die Performance bei parallelen Programmen bestimmt werden kann.

4 Versuchsaufbau

4.1 Hardware Vorrichtung

Die Rechner in dem die Tests danach laufen werden sind folgendermaßen konfiguriert:

- CPU: AMD ATHLON64 X2 4800+ 65W AM2.
- Motherboard: ASUS M2N-SLI Deluxe nForce570SLI.
- RAM: 4 x SASMUNG 1024MB DDR2-800 PC2-6400.(4 GB Speicher pro Rechner)
- Grafikkarte: Sapphire Radeon HD3450 256MB PCIe 2.0.
- Festplatten 2x 20 GB: Seagate Barracuda 7200.10 250 GB SATA-II NCQ 16MB LG GSA-H55N bulk black 20x.
- Ethernetadapter: Intel(R)PRO/1000 MT Desktop Adapter 2

Auf diese Messvorrichtung werden danach die Hostsysteme instaliert, die in den nächsten Kapiteln dargestellt werden.

4.2 Software Vorrichtung

Das Hostsystem für Virtuozzo 4.5 ist ein bereitgestellter Windows Server 2003 jeweils für 32- und 64-Bit. Für das Hostsystem wurden Treiber für Chipsatz: NForce Driver 15.46, Treiber für die Graphikkarte: Catalyst 9.4, Treiber für die Ethernetadapter: Network Adapter Drivers 14.7 und ein iSCSI Initiator Programm, das von der Microsoft Seite verfügbar war, installiert. Auf das Windows Server 2003 wurde das Gastsystem Virtuozzo 4.5 jeweils für 32-und 64-Bit installiert. Die Installation von Virtuozzo 4.5 inklusiv einen Parallels Infrastructure Manager und die Management Console wurde auf drei virtuelle Maschinen beschränkt, entsprechend der Aufgabestellung. Jede virtuelle Maschine besitzt nach Konfiguration eine virtuelle CPU und ein GB RAM. Die detallierte Installation von Virtuozzo 4.5 wird im Unterkapitel 2.4 dargestellt.

Das Hostsystem für OpenVZ ist ein Ubuntu 9.04 Server Edition auch jeweils für 32- und 64-Bit. Auf das Ubuntu 9.04 Server wurden keine Updates und zusätzliche Treiber nachinstalliert. Das Ubuntu 9.04 Hostsystem ist dann die Basis für das Gastsystem OpenVZ. Die genaue Installation und Einrichtung von OpenVZ findet man auch im Unterkapitel 2.3.

5 Synthetische-Messungen

Ziel der synthetischen Messungen ist die theoretische Performance einer virtuellen Maschine im Vergleich zum nativen Server zu untersuchen. Beim OpenVZ für Linux und Virtuozzo für Windows werden alle Messungen mit den Benchmark-Tools Linpack, Iometer und Ramspeed durchführen. Die Leistung entspricht nicht ganz einem echten Anwendungsfall nur noch die theoritische Performance. In diesem Abschnitt wird die Gesamtperformance sowohl eines nativen Servers als auch einer virtuellen Maschine untersucht und dann die Ergebnisse miteinander verglichen. Die Frage hier ist ob, die Gesamtperformance von einer virtuellen Maschine im Vergleich zu einem nativen Servers genau so effizient ist. Die Hostsysteme sind der Windows Server 2003 Standard Edition in der 32 und 64Bit Ausführung und der Ubuntu 9.04 Server Edition auch jeweils für 32-und 64-Bit. Jedes Gastsystem (virtuelle Maschine) bekommt eine logische CPU und steht eine GB RAM zur Verfügung.

5.1 CPU

Die Messungen der CPU Leistungsanalyse werden mit dem Linpack-Tool durchgeführt. Ermittelt wird hier die Leistung der Anlage beim Lösen eines linearen Gleichungssystems bei vorgegebener Komplexität. Ziel der Ermittlungen sind vier Matrizendurchläufe mit den standard (Matrizen in beiden zweier Dimensionen verdoppelt) Größen 1000x1000, 2000x2000, 4000x4000 und 8000x8000. Die Durchläufe werden sowohl unter Ubuntu 9.04 Server Edition(nativ) und Ubuntu-OpenVZ für Linux, als auch unter Windows Server 2003 R2(nativ) und Virtuozzo für Windows jeweils für 32 und 64-Bit realisiert. Wichtig ist es auch zu erwähnen, dass die Anzahl der Läufe auf dreißig Minuten beschrängt wurde. Gemessen wird die CPU Leistung beim Lösen der Matrizen für die oben angegebenen Größen von physikalischen im Vergleich zu virtuellen Server. Die Messungen helfen zu späteren Aussagen über die Betriebsystem Virtualisierung von OpenVZ-Virtuozzo im Vergleich zum virtuellen Server. Die folgenden Tabellen zeigen die Ergebnisse der OpenVZ Tests unter Linux und der Virtuozzo Tests unter Windows (virtuelle Maschine) und (native). Getestet werden Matrizen in Größen(1000x1000, 2000x2000, 4000x4000, 8000x8000) und die Ergebnisse der CPU Berechnung werden in den folgenden Tabellen separat für Linux 5.1 und Windows 5.2 protokolliert. Alle Angaben in den Tabellen stehen für die Messungen geteilt durch die Anzahl der Durchläufe (10 oder 4 Durchläufe in diesem Fall) und werden in Sekunden dargestellt. Die este Spalte der beiden Tabellen zeigt die Arraygröße, danach folgen die Durchläufe und die Ergebnisse der Messungen jeweils mit (v) für die virtuelle Maschinen und mit (n) für die nativen Server. Nach Abschluß aller CPU Tests ergeben sich bei der Performance keine nennenswerte Unterschiede zwischen den virtuellen Maschinen und den nativen Servern sowohl bei OpenVZ als auch bei Virtuozzo. Es gibt wirklich keine Performanceverlust einer virtuelle Maschine im Vergleich zu dem nativen Server. Im Gegenteil sind die virtuelle Maschinen bei allen Messungen um ein paar Sekunden schneller. Die Beobachtungen zeigen eine 50 Prozent Performanceverlust für Virtuozzo im Vergleich zu OpenVZ wie die Abbildung 5.1 protokolliert. Die Berechnungen werden unter OpenVZ fast 50 Prozent schneller durchgeführt und

zeigen die Stärke eines Linux-Plattforms. Die Messungen haben auch gezeigt, dass unter Windows die Matrizengröße 8000x8000 nur noch 4 anstatt 10 Durchläufe schaffen. Dies bedeutet, dass in den 30 Minuten Linpack nur noch viermal durchläuft und dann abbricht.

	Durchläufe	**Lin32(v)**	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
1000x1000	10	0,351	0,333	0,351	0,332
2000x2000	10	2,821	2,71	2,79	2,683
4000x4000	10	20,961	20,654	20,885	20,533
8000x8000	10	162,424	157,325	162,019	156,91

Tabelle 5.1: Linpack-Linux (Angaben in Sekunden)

	Durchläufe	**Win32(v)**	**Win64(v)**	**Win32(n)**	**Win64(n)**
1000x1000	10	0,808	0,846	0,784	0,823
2000x2000	10	5,997	6,242	5,816	6,12
4000x4000	10	46,56	48,53	45,245	47,694
8000x8000	4	359,1	383,11	354,7725	374,525

Tabelle 5.2: Linpack-Windows (Angaben in Sekunden)

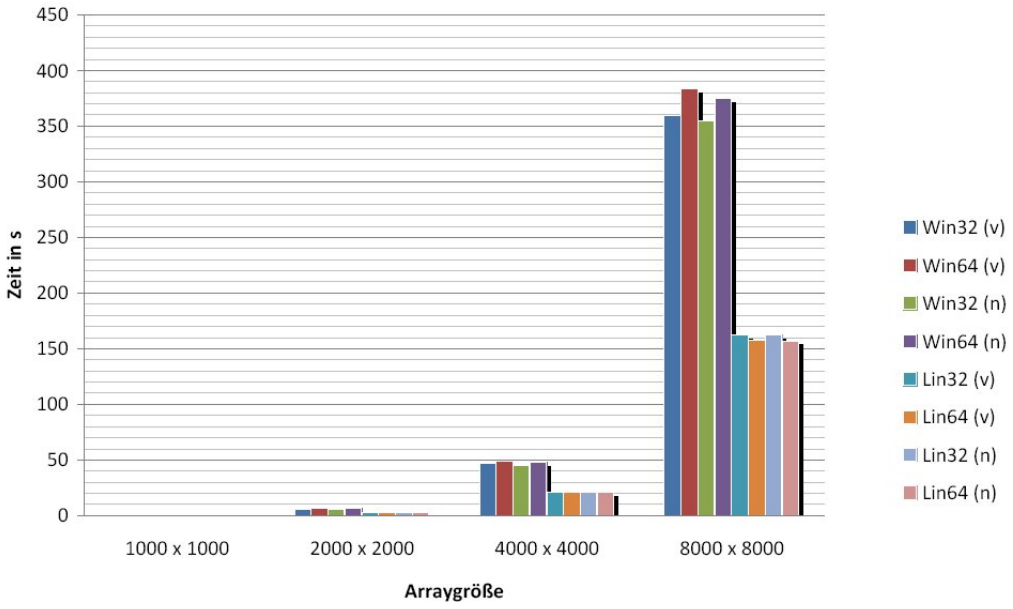

Abbildung 5.1: CPU Performance

5.2 Disk und Netzmessungen

Iometer ist ein Tool zur Analyse von Festplatten und Netzperformance. Die nächsten Tests werden mit dem Iometer-Tool durchgeführt. Die Iometer Software beinhaltet zwei Program-

me. Den Iometer und den Dynamo (auch Worker gennant). Die nächste Abbildung 5.2 zeigt das User Interface, was das eigentliche Programm von Iometer ist und dient zur Konfiguration und Verwaltung der Daten. Das User Interface hilft den sogenannten -Worker- zu konfigurieren, die Operationsparameter zu setzen und die Tests zu starten oder zu stoppen. Der Iometer gibt dem Dynamo vor was zu tun ist. Der Dynamo sammelt die Ergebnisse von den Tests und schreibt es in einem output File. Der Dynamo hat kein eigenes User-Interface und jedes Mal läuft maximal nur eine Kopie auf dem Server und jeweils eine Kopie in jedem Client.

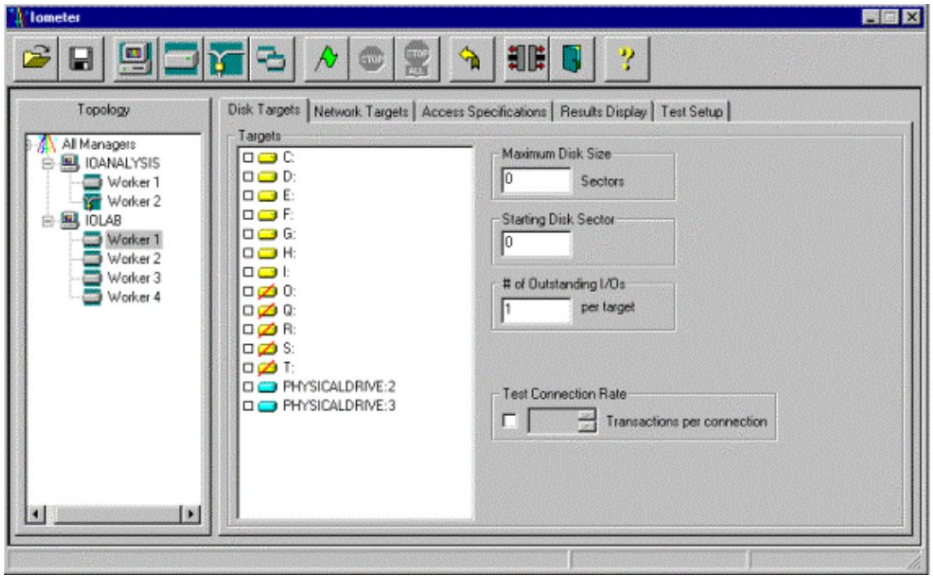

Abbildung 5.2: Iometer GUI

5.2.1 Disk

Zum Testen der Diskmessungen, wurden vier Parameterkombinationen untersucht: zufälliges Lesen und Schreiben und sequenzielles Lesen und Schreiben von festen Blöcken. Die Blockgröße beträgt 4 KB, 32 KB, 256 KB, 1 MB, 4 MB. Die virtuelle Maschine führt Schreib-/Leseoperationen aus. Die Zeitdauer für eine Messung beträgt 15 Minuten für sequenzielles Lesen, 15 Minuten für sequenzielles Schreiben, 15 Minuten für zufälliges Lesen und 15 Minuten für zufälliges Schreiben und die Gesamtdauer beträgt 1 Stunde. Somit kann jede Blöckgröße ca. 180 Sekunden getestet werden. Protokolliert wird die Anzahl der I/O Operationen pro Sekunde und der Durchsatz in MB. Jeder Test wird zweimal für 60 Minuten durchgeführt. Interessant sind hier die Transferraten für lesen und schreiben von verschiedenen Blockgrößen. Die Tabellen 5.3 und 5.4 dokumentieren in der ersten Spalte die sequenziellen und zufälligen Tests und danach folgen die Messungen zuerst der virtuellen Maschinen (v) und danach der nativen Server (n) jeweils für Linux und Windows. Die Messungen stehen für MB pro Sekunde. Bei den Diskmessungen können, wie die Abbildung 5.3 zeigt, zwei Schlüsse gezogen werden. Die native Maschinen schneiden generel etwas besser ab. Die OpenVZ Messungen kommen aber sehr nah an den virtuellen Maschinen. Beim Virtuozzo ist dies nicht der Fall. Hier hat die virtuelle Maschine beim sequenziellen Schreiben sowohl unter Windows 32 als auch unter Windows 64-Bit eine schlechtere Transferrate. Das könnte dran liegen, dass

unter Windows nicht die gesamte iSCSI Partition genutzt wurde, sondern eine 10 GB Image Festplatte angelegt wurde. Anders sieht es beim Ubuntu aus. Hier wurde die gesamte Partition in die virtuelle Maschine gemountet. Also auf die iSCSI Partition wird eine virtuelle Platte gelegt und auf die virtuelle Platte eine Iometer Imagedatei mit mit dem -dd- Tool erzeugt. Die Größe der Imagedatei ist 10 GB.

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
Write No Random	27,248	27,265	27,98	26,178
Read No Ranom	22,131	23,172	30,455	30,711
Write Random	18,289	19,843	13,822	13,802
Read Random	19,956	20,297	20,025	20,041

Tabelle 5.3: Disk-Linux (Angaben in MB/Sekunden)

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
Write No Random	16,36	14,412	28,745	28,785
Read No Random	24,179	24,151	23,059	23,089
Write Random	10,917	10,958	12,351	12,342
Read Random	15,968	15,866	18,264	18,108

Tabelle 5.4: Disk-Windows (Angaben in MB/Sekunden)

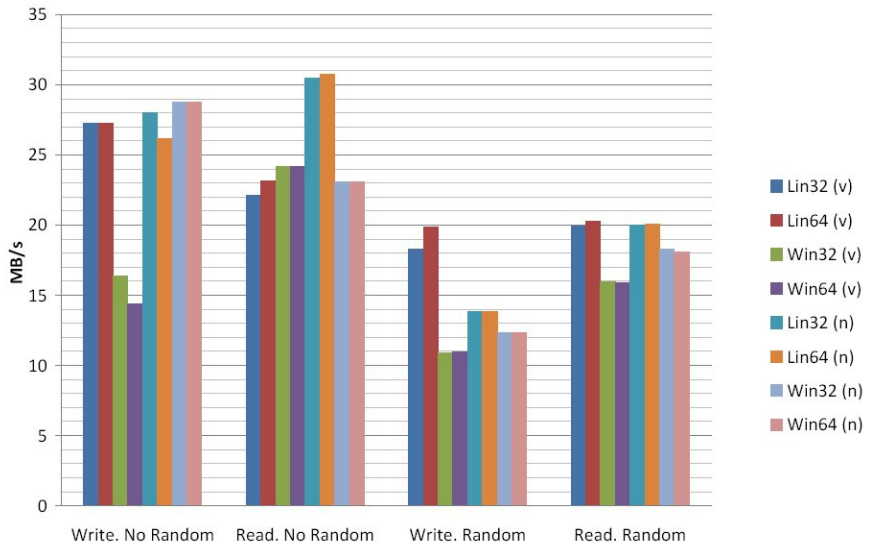

Abbildung 5.3: Disk Transferraten

5.2.2 Netz

Die Netzmessungen beruhen, genauso wie die Diskmessungen, auf dem Lesen und Schreiben von festen Blöcken. Die Blockgröße beträgt in diesem Fall wieder 4 KB, 32 KB, 256 KB, 1 MB, 4 MB. Die virtuelle Maschine führt Schreib-/Leseoperationen aus. Die Zeitdauer für eine Messung beträgt sowohl 15 Minuten für Lesen als auch 15 Minuten für Schreiben. Protokolliert wird die Anzahl der I/O Operationen pro Sekunde und der Durchsatz in MB. Jeder Test wird zweimal über 60 Minuten durchgeführt. Interessant sind hier die Transferraten zu beachten für lesen und schreiben. Die Tabellen dokumentieren in der ersten Spalte die Schreib- und Leseraten in MB/s. Zuerst werden die Messergebnisse der virtuellen Maschinen (v) aufgeführt, dann die der nativen Server (n) sowohl für Linux als auch Windows. Für die Netzdurchsatzmessungen wird eine dedizierte Leitung zwischen dem physischen Host und dem Router verwendet. Damit ist für die Gesamtdauer der Messungen volle Bandbreite verfügbar. Die Netzperformance sieht beim OpenVZ und Virtuozzo im Vergleich zu den

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
Write No Random	51,693	47,973	52,971	48,91
Read No Random	61,145	61,37	61,174	61,191
Write Random	51,783	48,056	52,666	48,974
Read Random	69,938	61,412	61,193	61,332

Tabelle 5.5: Netz-Linux (Angaben in MB/Sekunden)

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
Write No Random	51,876	51,803	52,582	52,669
Read No Random	60,712	60,56	64,594	64,551
Write Random	51,878	51,799	52,478	52,674
Read Random	60,693	60,533	64,642	64,548

Tabelle 5.6: Netz-Windows (Angaben in MB/Sekunden)

nativen Server, wie die Tabellen 5.5 und 5.6 zeigen, fast identisch aus. Das bedeutet, dass OpenVZ und Virtuozzo genau so viele Lese- und Schreibzugriffe in eine Sekunde operieren können und sind in der Lage gleich gute Ergebnisse zu erziehen wie ein nativer Server. Die virtuellen Maschinen unter Virtuozzo sowohl für 32-Bit als auch für 64-Bit hingegen können zwar beim Schreiben mithalten, verlieren aber an Performance solange eine virtuelle Maschine beim sequenziellen und zufälligen Lesen verwendet wird. Hier hat Virtuozzo einen Performanceverlust von ca. 7 Prozent. Der Grund könnte der installierte Netzwerktreiber sein. Beim Linux64 nativ und OpenVZ 64-Bit kann beobachtet werden, dass auch ein leichter Performanceverlust beim zufälligen und sequenziellen Schreiben existiert. Die Beobachtungen können in der Abbildung 5.4 nachgelessen werden. Als Grund kann hier der Reifegrad der synthetischen Treiber für Linux (Linux Integration Components) ausgeschlossen werden, weil die 32-Bit Maschinen unter OpenVZ und nativ keine solche Auffälligkeiten aufweisen.

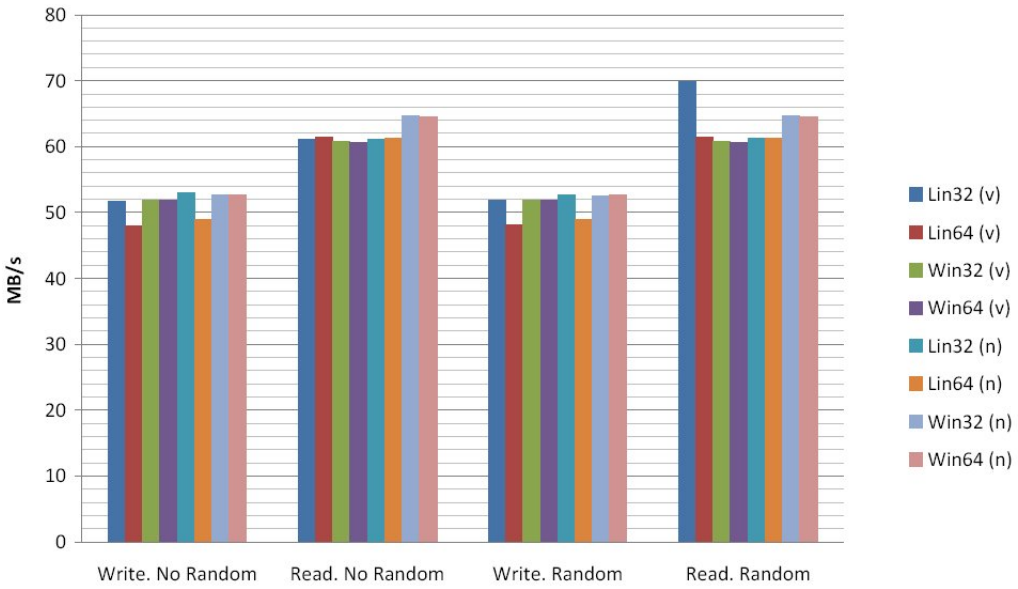

Abbildung 5.4: Netz Transferraten

5.3 RAM

Die CPU-Performance wird mit dem Open-Source-Kommandozeilen-Dienstprogramm RamSpeed ermittelt. Es unterstützt wie bereits ausgeführt drei Hardware-Plattformen (i386, AMD64, Alpha) und mehrere populäre Unix-ähnliche Betriebssysteme. Das Programm versucht den Arbeitspeicher voll auszulasten. Die Grundidee ist, dass synthetische Benchmarks mit solchen kombiniert werden, die die tägliche Arbeit am Rechner simulieren. Hierzu werden zwei synthetische Tests eingesetzt: INTmark und FLOATmark. Je nach Testart werden unterschiedliche Bereiche der CPU gefordert. Dabei werden Lese- und Schreibraten ermittelt indem sequentielle Datenströme durch die ALU- beziehungsweise FPU-Einheit geschleust werden. Zunächst wird der freier Speicherbereich alloziiert, der groß genug ist. Anschließend werden Speicherblöcke fester Größe gelesen bzw. geschrieben. Bei diesen Tests werden die Blöcke in zweier Dimensionen verdoppelt (1 KB, 2 KB, 4 KB, ..., 512 KB, 1 MB,..., bis zu 32 MB). Diese synthetische Benchmarks sind in der Praxis allerdings nicht von großer Bedeutung - sie zeigen lediglich die theoretischen Speichergeschwindigkeiten. Die nächsten Tabellen zeigen alle Messungen unter Linux 32 und 64-Bit und Windows 32 und 64-Bit sowohl für eine virtuelle Maschine als auch für einen nativen Server, die feste (wie die Tabellen zeigen) Blöcke schreiben/Lesen. Die Tabellen 5.7, 5.8, 5.9, 5.10 für die Intergeroperationen und 5.11, 5.12, 5.13, 5.14 für die Floatoperationen zeigen alle Durchschnittsergebnisse der Messungen getrennt für L1, L2-Cache und RAM. L1 Cache kommt zum Einsatz von 1 KB bis 64 KB, L2 Cache von 128 KB bis 512 KB. RAM wird ab 1 MB bis 32 MB gemessen. Das bedeutet natürlich ein Geschwindigkeitsverlust, wenn die Blochgrößen die 64 KB übersteigen und der L2 Cache oder der RAM zum Einsatz kommt. Die Ergebnisse des Windows Servers 2003 für

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
L1 AVERAGE	17103,86	28785,87	17146,52	28832,58
L2 AVERAGE	5782,50	6770,04	5635,32	6613,25
RAM AVERAGE	2477,22	2887,66	2446,67	2875,12

Tabelle 5.7: INTmark-Linux-Write (Angaben in MB/Sekunden)

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
L1 AVERAGE	18624,26	18350,114	18829,04	18721,474
L2 AVERAGE	6330,84	6337,143	6382,28	6387,083
RAM AVERAGE	2365,318	2374,445	2440,52	2402,261

Tabelle 5.8: INTmark-Windows-Write (Angaben in MB/Sekunden)

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
L1 AVERAGE	14533,86	36494,14	13533,76	32851,95
L2 AVERAGE	6723,29	6944,22	5225,98	5789,46
RAM AVERAGE	3103,86	3407,84	3101,05	3399,51

Tabelle 5.9: INTmark-Linux-Read (Angaben in MB/Sekunden)

32 Bit im Vergleich zu dem nativen Systems, wie die Abbildungen 5.5 und 5.6 zeigen, liegen

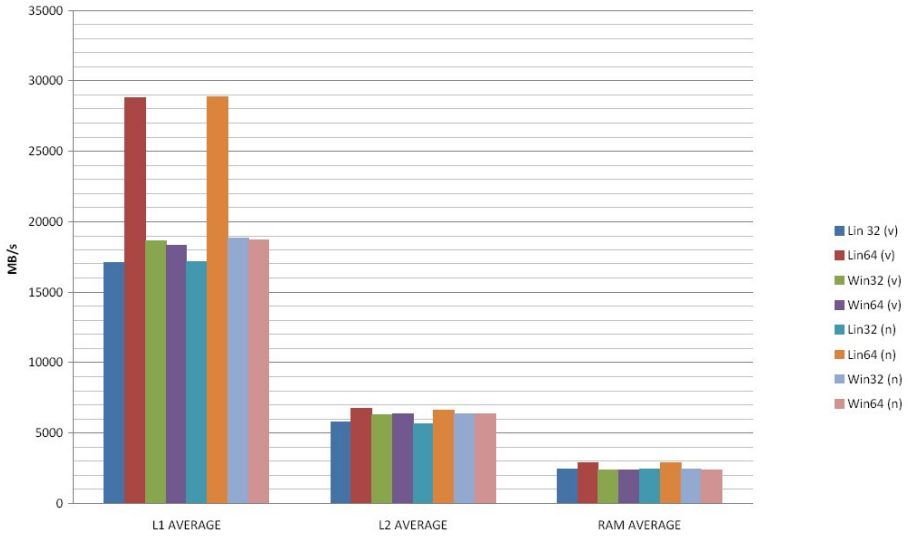

Abbildung 5.5: INTmark-Write Transferraten

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
L1 AVERAGE	18901,17	17418,06	17472,27	17547,29
L2 AVERAGE	7130,69	5860,17	5922,40	5924,25
RAM AVERAGE	3081,69	3074,66	3140,88	3125,73

Tabelle 5.10: INTmark-Windows-Read (Angaben in MB/Sekunden)

wie erwartet auf gleich hohem Niveau. Die geringfügig besseren Werte des 32-Bit Windows(v) Systems gegenüber des nativen Servers beim Lesen sind zu beachten. Der L1-Cache-Zugriff scheint bei den Integeroperationen die Achillesferse der 32-Bit-Ausgabe von Ubuntu-Server zu sein. Bei keinem anderen Benchmark fällt Ubuntu gegenüber den Windows-Systemen so deutlich ab, kann sich aber ansonsten recht gut behaupten. Der deutliche Vorsprung des Ubuntu Server x64 zeigt, dass das Benchmark stark von den zusätzlichen CPU-Befehlssätze profitiert. Dadurch werden die synthetischen Berechnungen deutlich beschleunigt. Beim

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
L1 AVERAGE	19949,73	36427,18	19996,5	36494,46
L2 AVERAGE	6948,82	6874,95	19956,94	37198,61
RAM AVERAGE	2868,43	2862,79	4616,42	4911,78

Tabelle 5.11: FLOATmark-Linux-Write (Angaben in MB/Sekunden)

Schreiben für die Floatoperationen, wie die Abbildung 5.7 zeigt, gilt das gleiche wie für die Integeroperationen. Beide Linux-Maschinen bei der 64-Bit Variante operieren fast fünfzig Prozent schneller als alle andere Distributionen beim Level 1. Die Zugriffe beim Level 2 Cache und RAM sehen anders aus. Hier sind alle virtuelle Maschinen im Vergleich zu den nativen um 60 Prozent für das Level 2 Cache und um 50 Prozent für RAM langsammer. Beim Lesen für die Floatoperationen wird in der Abbildung 5.8 ein ähnliches Spiel für den Level 1 Cache mit einer Ausnahme der Windows Maschinen beobachtet. Die haben einen schlech-

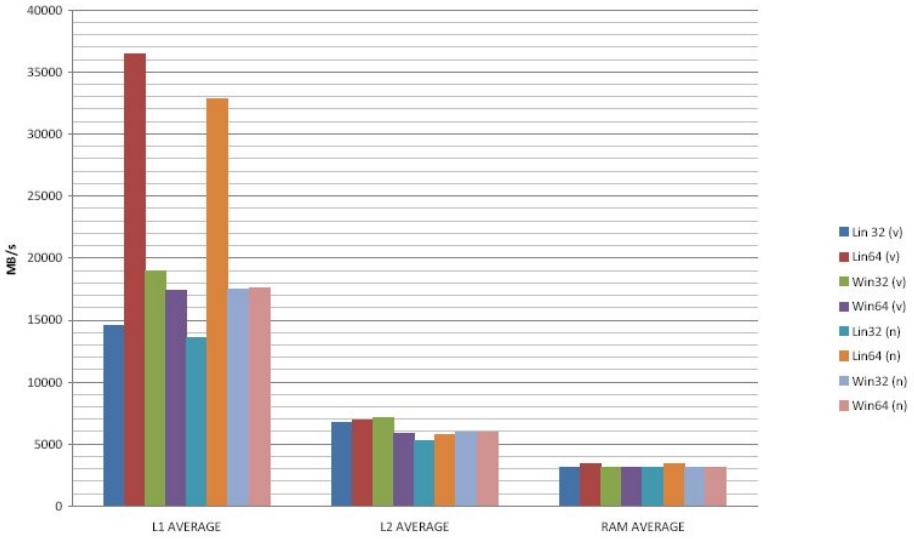

Abbildung 5.6: INTmark-Read Transferraten

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
L1 AVERAGE	17343,01	19583,77	20041,87	20012,05
L2 AVERAGE	7621,95	7595,18	20266,20	20007,63
RAM AVERAGE	2819,39	2811,80	5276,65	5257,54

Tabelle 5.12: FLOATmark-Windows-Write (Angaben in MB/Sekunden)

teren Performance gegenüber Linux. Die Zugriffe unter Level 2 Cache und RAM zeigen auch einen Vorteil der nativen Server gegenüber der virtuellen Maschinen.

Um die Lese- bzw. Schreib-Leistung bei reelen Anwendungen zu testen, werden daher weitere zwei Tests durchgeführt: INTmem sowie FLOATmem. Jeder von ihnen besteht aus vier Teilschritten: Copy, Scale, Add und Triad (Kopieren, Skalieren, Hinzufügen und Triad). Zwar handelt es sich dabei ebenfalls um synthetische Simulationen, sie sollten aber mit vielen echten Anwendungenszenarios korrelieren. Die nächsten Tabellen 5.15, 5.16, 5.17 und 5.18 zeigen alle INTmem und FLOATmem Messungen unter Linux 32 und 64-Bit und Windows 32 und 64-Bit sowohl für eine virtuelle Maschine als auch für einen nativen Server. Alle Tabellen besitzen eine erste Spalte, die die Durchschnittswerte für Copy, Scale, Add und Triad präsentiert und vier weitere Spalten mit den Messungen der virtuellen Maschinen (v) und den nativen Servern (n). Die Messungen stehen für MB pro Sekunden.

Auch die Integer-und Floatarithmetikoperationen (zum Beispiel die Operation Add) werden unter Ubuntu64-Bit schneller verarbeitet, aber nicht schneller als die entsprechenden nativen Systemen. Nach Beobachtungen der Durchschnittwerte für Integer-und Floatoperationen gilt allgemein, dass die virtuelle Maschinen unter Linux generell schneller sind. Die Windows nativen Maschinen haben einen leichteren Durchschnittsperformance im Vergleich zu den virtuellen Windows Maschinen, die Werte aber vernachläsichbar sind. Alle Beobachtungen werden mit den Abbildungen 5.9 und 5.10 verdeutlicht. Die Abbildungen zeigen nochmal deutlicher alle Distributionen jeweils für Copy, Scale, Add und Triad.

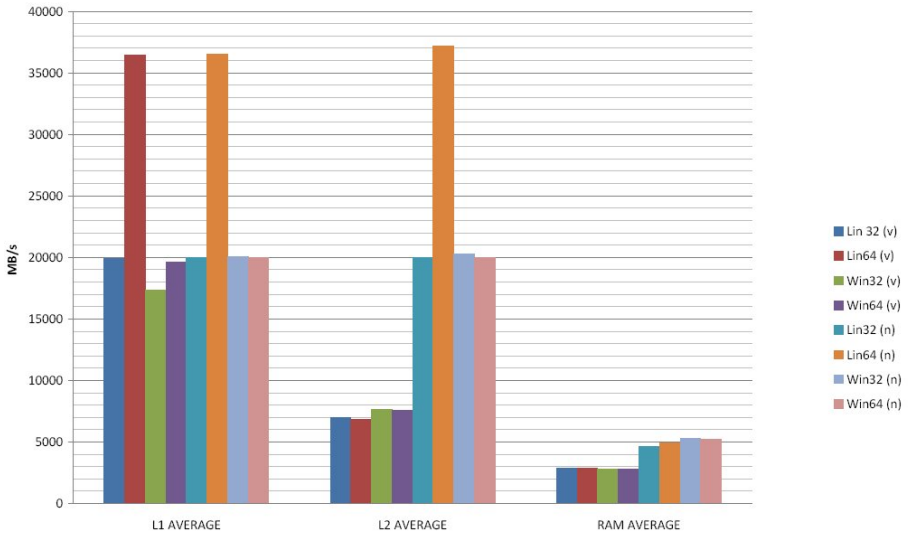

Abbildung 5.7: FLOATmark-Write Transferraten

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
L1 AVERAGE	19265,31	37800,29	14582,65	27909,55
L2 AVERAGE	6062,41	7496,11	15353,45	28195,36
RAM AVERAGE	3184,14	3413,26	4208,6	4461,88

Tabelle 5.13: FLOATmark-Linux-Read (Angaben in MB/Sekunden)

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
L1 AVERAGE	28243,64	25352,72	21221,42	21021,50
L2 AVERAGE	7345,7	6110,83	20912,73	21388,81
RAM AVERAGE	3313,8	3306,74	4768,95	4745,15

Tabelle 5.14: FLOATmark-Windows-Read (Angaben in MB/Sekunden)

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
Copy	2820,9	3005,9	2787,2	2882,7
Scale	2806,31	3022,07	2805,76	2980,7
Add	2965,8	3111,636	3318,18	3385,41
Triad	2964,42	2936,72	3301,31	3392,86
AVERAGE	2889,36	3019,08	3053,1	3160,42

Tabelle 5.15: INTmem-Linux (Angaben in MB/Sekunden)

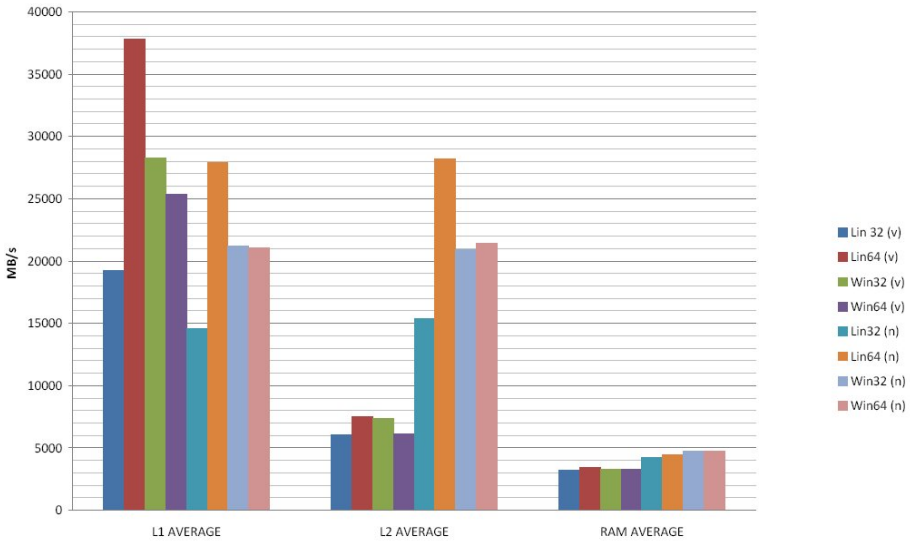

Abbildung 5.8: FLOATmark-Read Transferraten

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
Copy	2933,72	2917,78	3094,91	3082,14
Scale	2845,76	2842,94	2903,97	3002,42
Add	1814,26	1832,32	1911,85	1907,32
Triad	1859,83	1922,12	1993,95	1991,48
AVERAGE	2363,39	2378,79	2476,17	2495,84

Tabelle 5.16: INTmem-Windows (Angaben in MB/Sekunden)

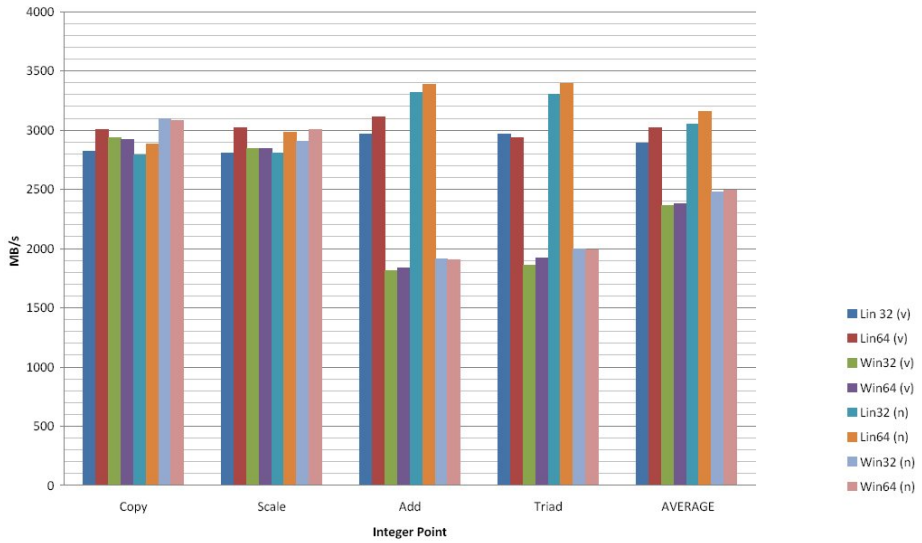

Abbildung 5.9: INTmem Transferraten

	Lin32(v)	**Lin64(v)**	**Lin32(n)**	**Lin64(n)**
Copy	2915,45	2892,89	2867,12	2917,92
Scale	2961,75	2798,95	3004,11	2966,02
Add	3465,81	3413,54	3472,96	3429,02
Triad	3425,20	3377,8	3421,27	3412,63
AVERAGE	3192,05	3120,82	3191,37	3181,40

Tabelle 5.17: FLOATmem-Linux (Angaben in MB/Sekunden)

	Win32(v)	**Win64(v)**	**Win32(n)**	**Win64(n)**
Copy	3174,40	3125,32	3200,42	3185,59
Scale	3142,47	3157,48	3199,83	3196,25
Add	3019,66	2983,31	3055,10	3043,92
Triad	3085,47	3062,73	3108,17	3104,42
AVERAGE	3105,50	3082,21	3140,88	3132,55

Tabelle 5.18: FLOATmem-Windows (Angaben in MB/Sekunden)

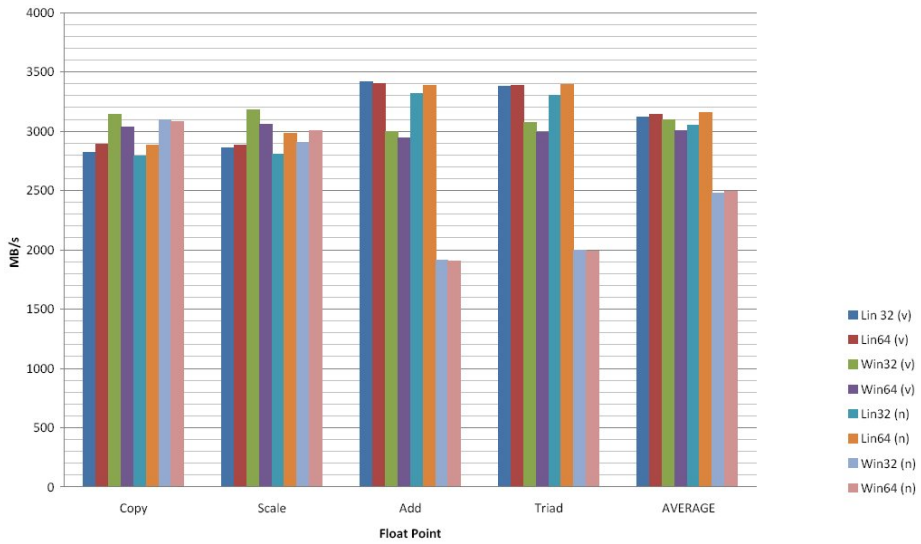

Abbildung 5.10: FLOATmem Transferraten

6 Parallele-Messungen

Die theoretische Performance einer virtuellen Maschinen im Vergleich mit den nativen Server bei OpenVZ für Linux und Virtuozzo für Windows wurde mit den Benchmark-Tools Linpack, Iometer und Ramspeed untersucht. In diesem Abschnitt wird die Fähigkeit zur Konsolidierung der beiden Virtualisierer unter die Lupe genommen. Die Frage ist hier, ob die Gesamtperformance mit mehr als eine virtuelle Maschine im Vergleich mit einer steigt. Um die Fähigkeit zur Konsolidierung von OpenVZ und Virtuozzo zu testen, werden zuerst zwei und dann drei virtuelle Maschinen parallel ausgeführt. Das Hostsystem ist Windows Server 2003 Standard Edition in der 32-Bit Ausführung. Jedes Gastsystem bekommt eine logische CPU und steht eine GB RAM zur Verfügung. Die Systempartitionsgröße beträgt 20GB. Für die Disk-Messungen stand eine Imagedatei mit einer Größe von 5 GB zur Verfügung. Diese Größe sollte genügen um ausreichend signifikante wahlfreien I/O-Operationen zu generieren um das I/O-Teilsystem richtig auszulasten. Für die parallele Messungen sieht die Spezifikation wie folgt aus:

- Linpack-Benchmark parallel für 2 und 3 virtuelle Maschinen
- Iometer-Disk-Benchmark parallel für 2 und 3 virtuelle Maschinen
- Iometer-Netz-Benchmark parallel für 2 und 3 virtuelle Maschinen
- Iometer-Netz/Disk-Benchmark parallel für 2 und 3 virtuelle Maschinen
- Ramspeed-Benchmark parallel für 2 und 3 virtuelle Maschinen
- IoMeter-Netz-Benchmark (GUI läuft in einer weiteren VM)
- IoMeter-Netz-Benchmark (in zwei weiteren VMs läuft je ein Linpack-Benchmark)
- IoMeter-Disk-Benchmark (in zwei weiteren VMs läuft je ein Linpack-Benchmark)

Diese Messungen erlauben eine recht zuverlässige Einordnung für die Perfomance einzelner virtueller Maschinen und zeigen auch die Konsolidierungsfähigkeit der Tools auf. Jeder Benchmark ist so ausgelegt, dass die jeweilige VM voll ausgelastet ist. Mit zusätzlichen VMs sinkt deswegen die Performance der einzelnen VMs, weil das Hostsystem seine Ressourcen aufteilen muss. Die Summe der Performance aller VMs sollte nichtdestotrotz ansteigen. In der Theorie ist die Konsolidierungsfähigkeit des Hosts erreicht wenn beim Hinzufügen einer weiteren virtuellen Maschine die Gesamtperformance nicht mehr ansteigt. In dieser Arbeit ist die Gesamtzahl der VMs auf 3 beschränkt.

6.1 Parallele Linpack Messungen

Wir starten zwei beziehungsweise drei virtuelle Maschinen und lassen das Linpack-Benchmark parallel laufen. Hier zeigt sich wie das CPU-Scheduling funktioniert. Zuerst zeigen wir die

CPU-Efizienz von Virtuozzo bei zwei virtuelle Maschinen: Die nächste Tabelle 6.1 zeigt die Messungen von zwei virtuellen Maschinen im Vergleich zu den Messungen einer einzelnen virtuellen Maschine. Die Zeit wird in Sekunden erfasst. Die Tabelle zeigt uns die Messungen der Arraygrößen von 1000x1000, 2000x2000, 4000x4000 geteilt durch die 10 Durchläufe und die Messungen der Arraygröße von 8000x8000 geteilt durch 4 Durchläufe. Das gleiche gilt auch für die Tabelle 6.2, die die Messungen von drei virtuelle Maschinen präsentieren. Die zwei Tabellen zeigen, eine unterschiedliche Vorgehensweise des Schedulers. Bei 2 VMs ist die Zuteilung der Rechenzeit wie bei einer einzelnen Maschine. Das bedeutet, dass beide virtuelle Maschinen einen sehr gute Performance haben, bezüglich der CPU betrifft. Bei 3 VMs tritt ein recht interessanter Effekt zum Vorschein. Die erste und die dritte virtuelle Maschine sind um Faktor zwei langsamer als die zweite VM. Parallels, der Hersteller von Virtuozzo, nennt das übrigens -fair share- Scheduling. Das stimmt, wenn nicht spezifiziert wird auf welche Systeme die Aussage bezogen ist. Jedenfalls erklärt dies, warum die VPS manchmal so langsam sind, wenn nicht auf eine faire Konfiguration achtet. Das hat damit zu tun, dass der Scheduler bei Virtuozzo auf 2 Ebenen arbeitet: zunächst wird entschieden, welche VM den CPU Takt erhält. Dabei werden die VMs in der Reihenfolge in der sie gestartet wurden an die CPU-Kerne zugewiesen: VM1 und VM3 an den 1. Kern, VM2 an den 2. Erst dann entscheidet der standardmäßige Windows-Planer, welcher Prozess in ausgewählten VE den CPU-Takt bekommen soll. Dabei werden wie immer die Standard-Prioritäten von Prozessen benutzt. Ansonsten gilt: Virtuozzo ist sehr schnell, da es einen gemeinsamen Kernel sowohl für Host als auch für alle Gäste benutzt, es werden keine zusätzlichen Systemaufrufe erstellt. Dies ist ein Vorteil bei höherer VM-Dichte. Die nächste Abbildung 6.1 zeigt deutlich, dass die zweite virtuelle Maschine um den Faktor zwei schneller rechnet.

	VM1	**VM2**	**einzelne VM**
1000x1000	0,813 s	0,807 s	0,808 s
2000x2000	6,266 s	6,11 s	5,997 s
4000x4000	48,686 s	47,22 s	46,56 s
8000x8000	377,015 s	368,94 s	365,37 s

Tabelle 6.1: Linpack-Parallel-2 VM (Angaben in Sekunden)

	VM1	**VM2**	**VM3**	**einzelne VM**
1000x1000	8,23 s	0,82 s	0,822 s	0,808 s
2000x2000	12,135 s	6,079 s	12,149 s	5,997 s
4000x4000	94,591 s	47,111 s	94,507 s	46,56 s
8000x8000	182,963 s	367,18 s	183,975 s	365,37 s

Tabelle 6.2: Linpack-Parallel-3 VM (Angaben in Sekunden)

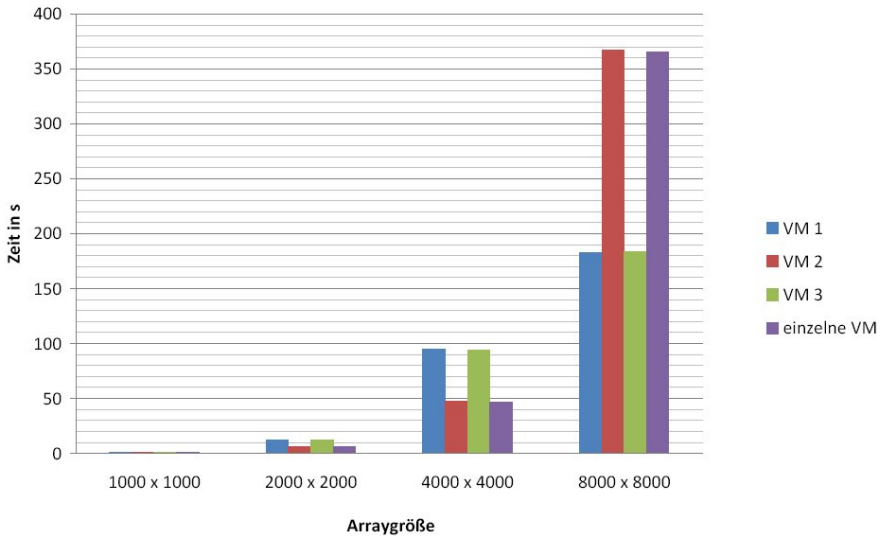

Abbildung 6.1: Linpack 3 VMs

6.2 Parallele Iometer Disk-Messungen

Anschließend messen wir den effektiven Festplattendurchsatz der virtuellen Maschinen bei Lese- bzw. Schreibzugriffen. Hierfür werden wie vorhin 2 bzw. 3 VMs untersucht. Das Ergebnis: beim sequentiellen sowie wahlfreien Zugriffen bewegt sich der Datendurchsatz auf dem Niveau einer einzelnen VM. Lediglich im sequentiellen Schreiben und Lesen arbeitet Virtuozzo mit hohem Datendurchsatz. Im Schnitt errechnen wir einen Geschwindigkeitsvorteil von zirka 3 Prozent. Die Transferrate nimmt beim Virtuozzo bei steigender Clientsanzahl zu. Wie die zwei Tabellen 6.3 und 6.4 auch gezeigt haben je mehr virtuellen Maschinen laufen lässt, desto höher wird die Transferrate. Bei den zufälligen Zugriffe ist sogar die Transferrate höher als die jenige der einzelne virtuelle Maschine. Das war aber auch zu erwarten, da Virtuozzo auf eine hohe Containerdichte optimiert ist. Die gleichmäßige Lastverteilung wird hier wegen den CFQ Scheduler (Completely Fair Queuing) verursacht. Der Sheduler basiert auf das Prinzip des Round Robin Algorithmus. Die nächste Abbildung 6.2 zeigt deutlich, dass beim zufälligen Zugriffen die Transferrate sogar ansteigt.

	VM1	**VM2**	**Summe**	**einzelne VM**
Write No Random	7,6 MB/s	7,71 MB/s	15,31 MB/s	16,36 MB/s
Read No Random	9,937 MB/s	9,817 MB/s	19,754 MB/s	24,179 MB/s
Write Random	5,809 MB/s	5,854 MB/s	11,664 MB/s	10,917 MB/s
Read Random	8,203 MB/s	8,067 MB/s	16,27 MB/s	15,968 MB/s

Tabelle 6.3: Iometer-Disk-Parallel-2 VM (Angaben in MB/Sekunden)

	VM1	VM2	VM3	Summe	einzelne VM
Write No Random	5,278 MB/s	5,274 MB/s	5,313 MB/s	15,866 MB/s	16,36 MB/s
Read No Random	6,759 MB/s	6,843 MB/s	6,768 MB/s	20,372 MB/s	24,179 MB/s
Write Random	5,125 MB/s	5,165 MB/s	5,543 MB/s	15,834 MB/s	10,917 MB/s
Read Random	6,787 MB/s	6,814 MB/s	6,693 MB/s	20,295 MB/s	15,968 MB/s

Tabelle 6.4: Iometer-Disk-Parallel-3 VM (Angaben in MB/Sekunden)

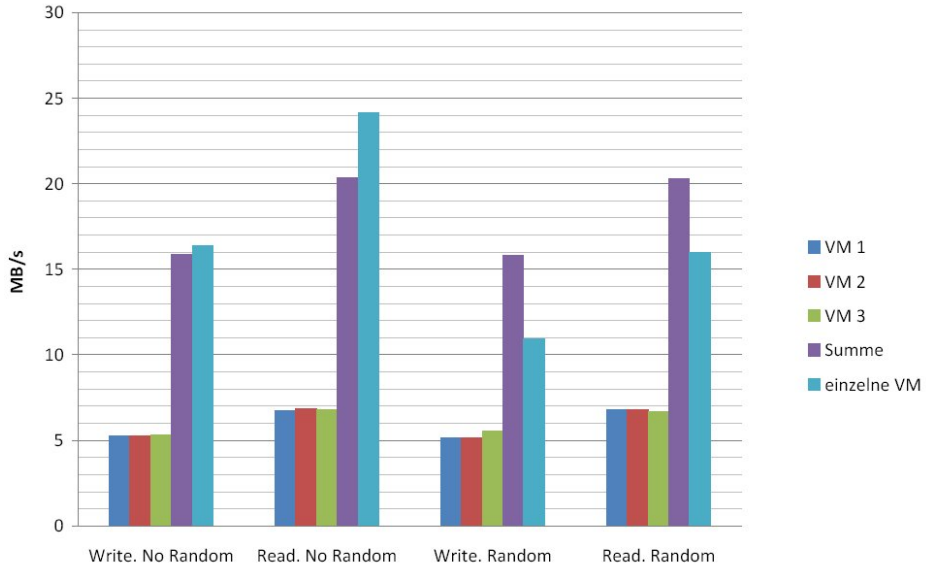

Abbildung 6.2: Iometer-Disk 3 VMs

6.3 Parallele Iometer Netz-Messungen

Bei den parallelen Netz-Messungen überprüft wird wie gut Virtuozzo den Zugriff auf eine Netzwerkschnittstelle realisiert. Die zwei Tabellen 6.5 und 6.6 zeigen, dass der gesamte Netzwerkdurchsatz gleich geblieben ist und die Bandbreite, die für die Messungen zur Verfügung gestellt wurde gleichmäßig unter zwei und drei virtuelle Maschinen aufgeteilt wird. Die parallele Zugriffe sowohl für zwei als auch für drei virtuelle Maschinen im Vergleich mit den einzelnen Messungen einer virtuelle Maschine sind sogar insgesamt einpaar Prozente besser. Diese Unterschiede machen sich aber in der Praxis kaum bemerkbar, wie die nächste Abbildung 6.3 eindeutig zeigt.

	VM1	VM2	Summe	einzelne VM
Write No Random	26,335	26,297	52,632	51,876
Read No Random	30,702	30,828	61,53	60,712
Write Random	26,341	26,293	52,634	51,878
Read Random	30,69	30,83	61,52	60,693

Tabelle 6.5: Iometer-Netz-Parallel-2 VM (Angaben in MB/Sekunden)

	VM1	VM2	VM3	Summe	einzelne VM
Write No Random	17,361	17,308	17,357	52,027	51,876
Read No Random	20,525	20,477	20,515	61,519	60,712
Write Random	17,315	17,316	17,363	51,995	51,878
Read Random	20,543	20,495	20,547	61,586	60,693

Tabelle 6.6: Iometer-Netz-Parallel-3 VM (Angaben in MB/Sekunden)

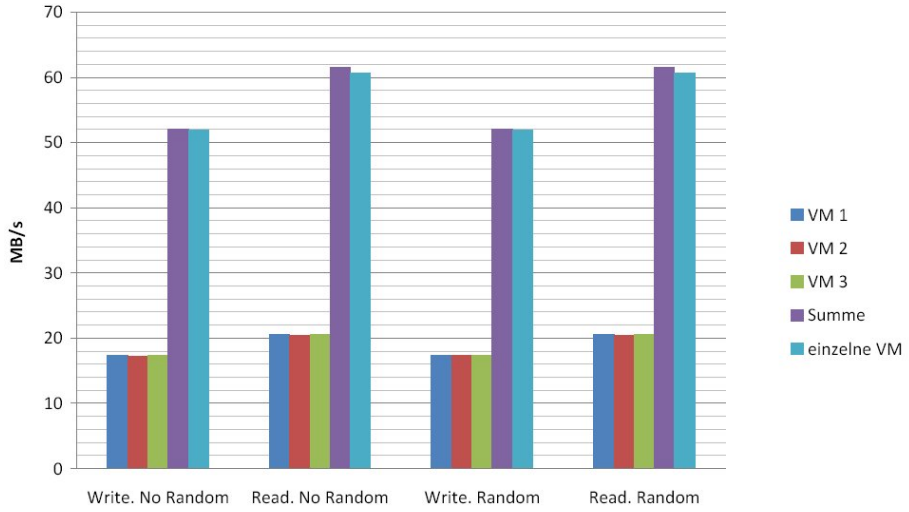

Abbildung 6.3: Iometer-Netz 3 VMs

6.4 Diskmessung unter Last

Der Hintergrund dieser Messung ist den Festplattedurchsatz zu untersuchen unter einer bestimmten CPU-Last. Ähnliches Szenario wie bei Netztests unter Last. Insgesamt operieren in diesem Test drei virtuelle Maschinen. Bei zwei der Maschinen läuft eine Instanz des Linpack-Benchmarks und auf der dritten Maschine der Iometer Worker, der den Festplattendurchsatz protokoliert. Wir wollen die zwei Prozessorkerne, die zur Verfügung stehen voll auszulasten und dabei auf die Festplatte Blöcke Schreiben und Lesen. Die Messungen werden in der nächsten Tabelle 6.7 und die Abbildung 6.4 dargestellt. Wie auch zu erwarten war, verursacht hier Virtuozzo wenig CPU Overhead bei den Disk Messungen, wenn die zwei Kerne beschäftigt sind. Dies ergibt sich, da Virtuozzo wie bereits erwähnt für eine hohe Containerdichte optimiert ist. Dabei ist die Lastverteilung fair und die Messung hat einen höheren Durchschittswert, zumindestens für die zufälligen Lese- und Schreibzugriffe als die einzelne virtuelle Maschine.

	VM1	einzelne VM
Write No Random	18,91	16,36
Read No Random	25,291	24,179
Write Random	15,255	10,917
Read Random	20,326	15,968

Tabelle 6.7: Disk-unter Last (Angaben in MB/Sekunden)

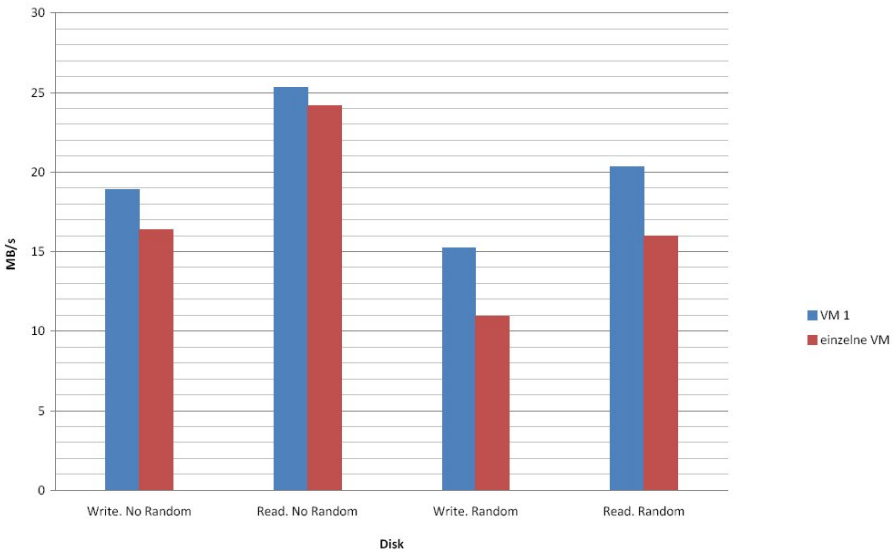

Abbildung 6.4: Disk unter Last

6.5 Netz unter Last

Es wird hier im Test wie bei den Diskmessungen unter Last ähnlich vorgegangen. Die Simulation einer hohen Clientdichte wird hier untersucht. Es wird getest ob, der Netzdurchsatz einbricht, sobald die Hostressourcen aufgebraucht sind. Insgesamt operieren drei virtuelle Maschinen. Auf den zwei läuft eine Instanz des Linpack-Benchmarks und auf der dritten Maschine der Iometer Worker. Die zwei CPU-Kerne stehen unter Last und dabei wird das Verhalten des Netzdurchsatzes im Vergleich zu den einzelnen virtuellen Maschine untersucht. Die Messungen zeigen die nächste Tabelle 6.8 und die Abbildung 6.5 hilft bei der Verständniss der Ergebnisse.

	VM1	einzelne VM
Write No Random	44,717	51,876
Read No Random	55,441	60,712
Write Random	44,755	51,876
Read Random	55,384	60,693

Tabelle 6.8: Netz-unter Last (Angaben in MB/Sekunden)

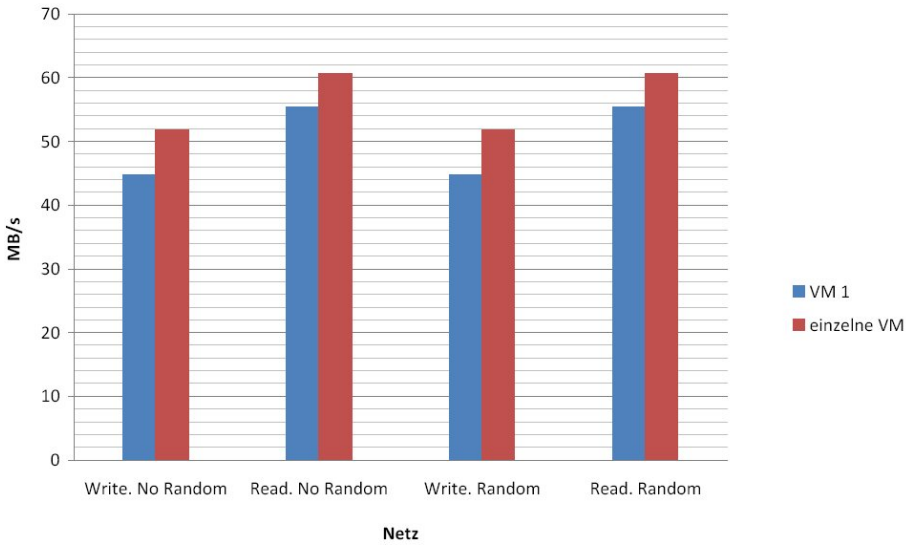

Abbildung 6.5: Netz unter Last

6.6 VM zu VM Messung

Diese Messung überprüft wie schnell Datenpakete zwischen zwei virtuelle Maschinen ausgetauscht werden. Um den Datendurchsatz zu messen musste in jedem virtuelle Maschine eine Instanz des Iometer-Workers installiert werden. Der Test soll theoritisch zeigen, dass die Daten schneller fließen als bei den Netzmessungen der einzelnen virtuellen Maschinen. Nach der Betrachtung der Tabelle 6.9 und der Abbildung 6.6 sind die Ergebnisse anderslautend. Bei dieser Messung sehen wir schlechte Transferraten des Virtualisierers anstatt einer besseren Leistung zwischen der zwei virtuellen Maschinen. Der Grund könnte die Isolierung der Container voneinander sein. Die Container besitzen keine Kenntnisse darüber, ob sie allein oder mit vielen Maschinen auf einem Hostsystem existieren.

	VM1	**einzelne VM**
Write No Random	12,226	51,876
Read No Random	9,256	60,712
Write Random	11,541	51,876
Read Random	9,505	60,693

Tabelle 6.9: VM-VM (Angaben in MB/Sekunden)

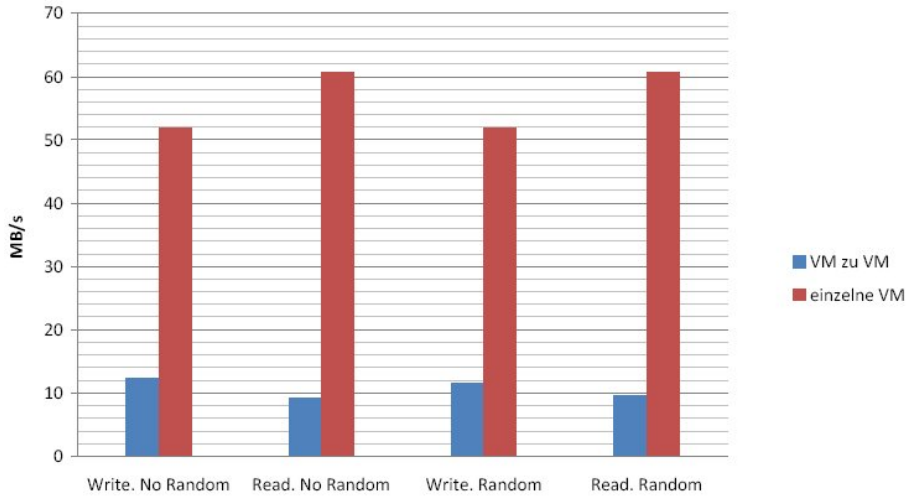

Abbildung 6.6: VM zu VM

6.7 Parallele Iometer Netz und Disk-Messungen

In diesem Abschnitt wird der maximale Datendurchsatz ermittelt indem sowohl gleichzeitig auf die lokale Festplatte gelesen beziehungsweise geschrieben und dabei zusätzlich aus dem Netz gelesen und geschrieben wird. Zuerst werden die zwei virtuelle Maschinen parallel untersucht und danach folgen die Ergebnisse der Messungen mit drei virtuelle Maschinen. Die Messungen zeigen eine überdurchschnittliche Performance von Virtuozzo. Die Summe der gesamten Performance ist sowohl beim Netz wie die Tabelle 6.10 zeigt als auch beim Disk wie die nächste Tabelle 6.11 präsentiert, höher im Vergleich zu den einzelnen synthetischen Messungen einer virtuelle Maschine, die unter Virtuozzo lief. Die gute Performance zeichnet sich in den beiden Abbildungen 6.7 und 6.8 auf. Anschließend werden die Messungen mit drei virtuellen Maschinen durchgeführt. In diesem Punkt geht es eingentlich um das gleiche wie bei den Messungen mit zwei virtuellen Maschinen. Untersucht wird der Virtualisierer auf seine Performancestärke. Die nächsten zwei Tabellen 6.12 und 6.13 protokolieren die Performance von Virtuozzo beim parallelen Platteoperationen und Netzdurchsatz. Bei drei laufenden virtuellen Maschinen hat Virtuozzo einen Nachteil beim Netzdurchsatz, laut Tabelle 6.12, von ungefähr fünf Prozent für sequenzielles und zufälliges Lesen gegenüber der einzelnen virtuellen Maschine während beim Diskmessung, wie die Tabelle 6.13 dokumentiert, der Nachteil noch für das sequenzielles Lesen gilt. Ansonsten ist es bei den Diskberechnungen so, dass Virtuozzo besser abschneidet, auch wenn es nicht mehr als 3 MB/s ist. Die Aussagen werden mit Hilfe der nächsten Abbildungen 6.9 und 6.10 nachvollziehbar.

	VM1	VM2	Summe	einzelne VM
Write No Random	26,144	26,125	52,267	51,858
Read No Random	30,904	30,805	61,709	63,997
Write Random	26,093	26,169	52,262	51,839
Read Random	30,916	30,78	61,696	63,993

Tabelle 6.10: Iometer-Full-Netz-Parallel-2 VM (Angaben in MB/Sekunden)

	VM1	VM2	Summe	einzelne VM
Write No Random	8,513	8,788	17,301	16,36
Read No Random	10,818	10,964	21,782	24,179
Write Random	7,116	6,797	13,913	10,917
Read Random	9,706	9,353	19,059	15,968

Tabelle 6.11: Iometer-Full-Disk-Parallel-2 VM (Angaben in MB/Sekunden)

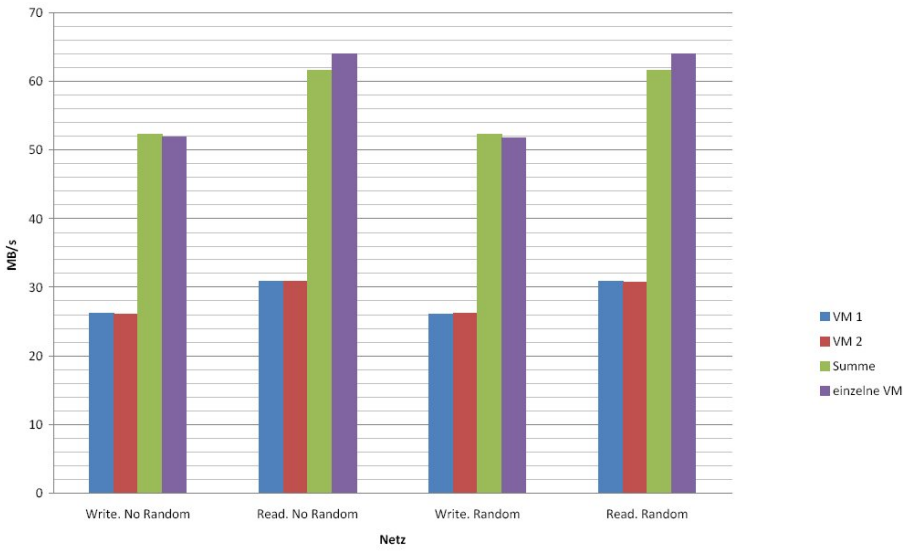

Abbildung 6.7: Iometer-Netz-Full 2 VMs

	VM1	VM2	VM3	Summe	einzelne VM
Write No Random	17	17,422	17,382	52,000	51,858
Read No Random	20,59	20,548	20,589	61,727	63,997
Write Random	17,421	17,419	17,394	52,234	51,839
Read Random	20,597	20,557	20,586	61,74	63,993

Tabelle 6.12: Iometer-Full-Netz-Parallel-3 VM (Angaben in MB/Sekunden)

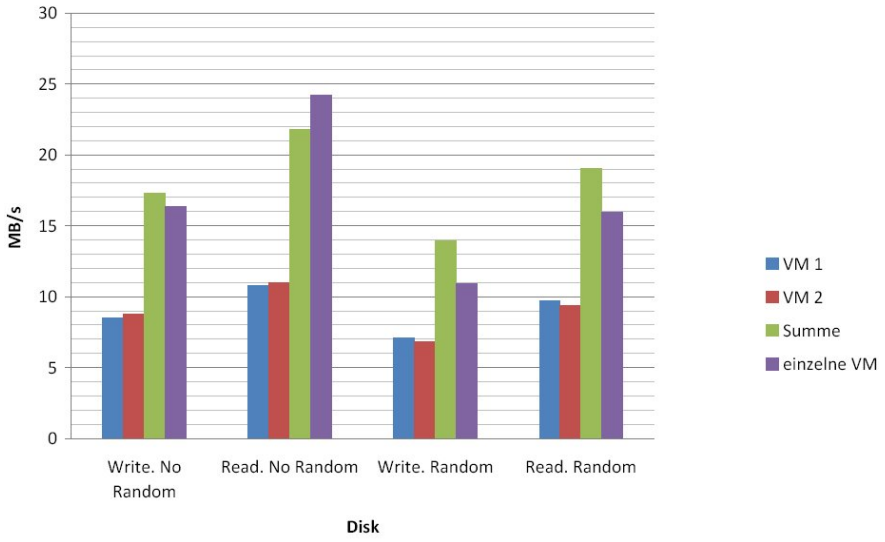

Abbildung 6.8: Iometer-Disk-Full 2 VMs

	VM1	**VM2**	**VM3**	**Summe**	**einzelne VM**
Write No Random	5,446	6,134	5,563	17,143	16,36
Read No Random	7,421	7,564	7,406	22,391	24,179
Write Random	4,448	4,403	4,642	13,493	10,917
Read Random	6,425	6,103	6,328	18,856	15,968

Tabelle 6.13: Iometer-Full-Disk-Parallel-3 VM (Angaben in MB/Sekunden)

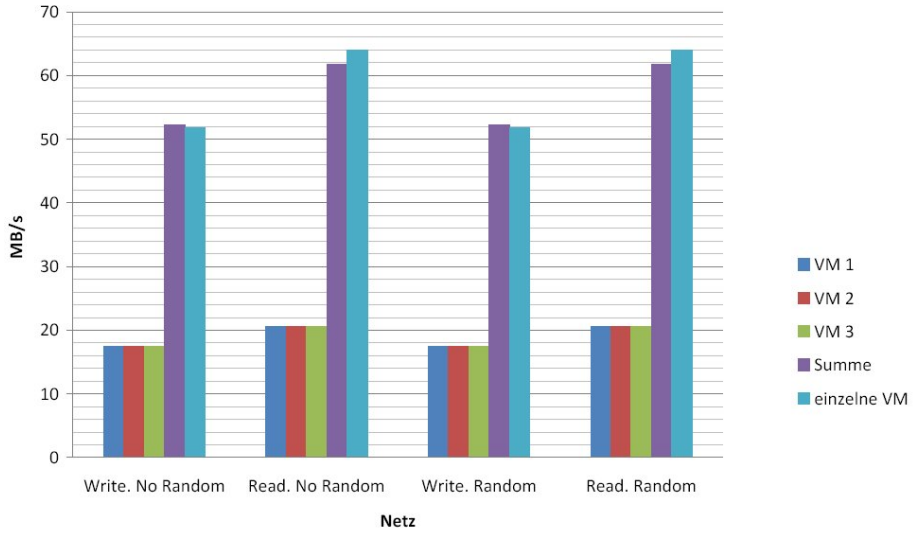

Abbildung 6.9: Iometer-Netz-Full 3 VMs

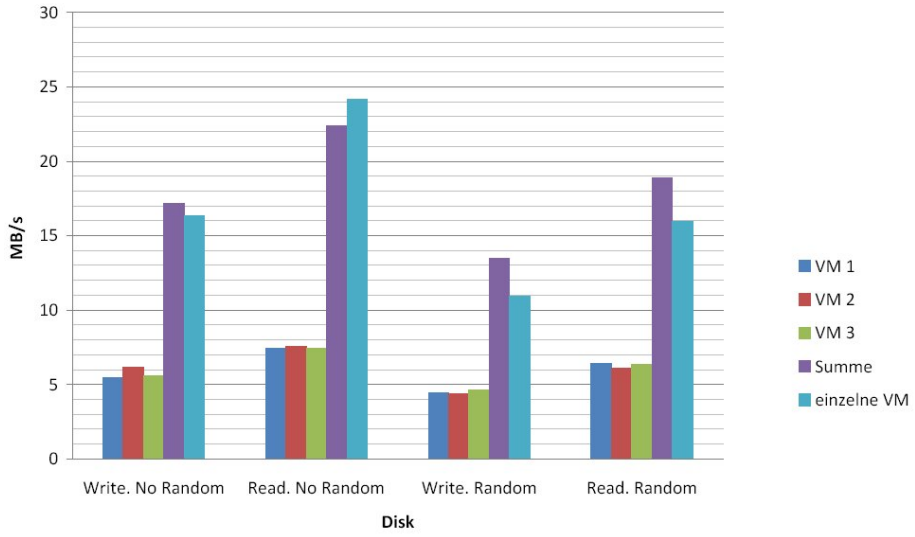

Abbildung 6.10: Iometer-Disk-Full 3 VMs

6.8 Ramspeed Parellel

In diesem Abschnitt wird mit dem Ramspeed Benchmark versucht den Arbeitsspeicher zu testen indem zuerst mit zwei virtuellen Maschinen und dann mit drei Maschinen parallel gearbeitet wird. Die Messungen werden ebenfalls in Tabellen präsentiert. Um die Ergebnisse dann auch deutlicher zu erfassen werden sie in Diagramme gezeigt. Alle Tabellen besitzen eine Spalte für die L1, L2-Cache und RAM Durchschnittswerte, die Messergebnisse von einer einzelnen virtuellen Maschine (die virtuelle Maschine unter Virtuozzo 32-Bit als Vergleichswert genommen), danach die virtuelle Maschinen, die parallel liefen (zwei oder drei), der Averagewert der parallelen Maschinen und die Summe der parallelen Maschinen wird in die letzte Spalte präsentiert. Die Beobachtungen unter den parallelen RAM-Messungen sehen für Integer-und Floatzugriffe unterschiedlich aus. Bei zwei virtuellen Maschinen sieht es sowohl bei den Integer als auch bei den Floatzugriffe sehr einheitlich aus. Beide Maschinen besitzen einen ähnlichen Performance. Jede Maschine greift auf dem Arbeitspeicher genau so schnell wie eine einzelne Maschine. Im Falle von Floatoperationen beim Lesen sind beim L1-Cache sogar die 2 virtuelle Maschinen minimal schneller als die einzelne Maschine. Die detalierte Informationen werden von den zwei Tabellen 6.16, 6.20 für die Interger und-Floatlesezugriffe präsentiert. Beim Schreiben sieht die Sache genau so aus wie beim Lesen. Die Floatzugriffe sind beim L1-Cache für die zwei virtuelle Maschinen mininal schneller als die einzelne Maschine wie die Tabellen 6.14 und 6.18 bestätigen können. Die Abbildungen 6.13, 6.11 und 6.17, 6.15 zeigen ebenfalls für Lesen und Schreiben, dass beide Maschinen genau so schnelle Zugriffe wie die einzelnen Maschinen haben und die Unterschiede für Virtuozzo wirklich minimal sind. Was jetzt die Performance unter 3 virtuelle Maschinen angeht, die parallel laufen, sieht es zumindestens bei den Intergeroperationen genau so aus wie bei den CPU Messungen der Fall war. Die erste und die dritte virtuelle Maschine sind um Faktor zwei langsamer wie die Tabellen 6.15 und 6.19 protokolieren. Das gilt aber nicht für die Floatoperationen. Die Tabellen 6.17 und 6.21 zeigen, dass alle drei virtuelle Maschinen genau so schnell laufen wie eine einzelne Maschine. Hier sind alle Lese-und Schreibzugriffe der drei virtuellen Maschinen genau so schnell wie von den einzelnen Maschinen und die zwei Abbildungen 6.12, 6.16 zeigen kein Unterschied für die Floatoperationen an. Als Schlußfolgerung ist zu beobachten, dass außer die Intergeroperationen beim Einsatz von drei virtuelle Maschinen skaliert hier Virtuozzo, sowohl unter parallelen Betrieb von zwei virtuellen Maschinen für Integer und Float als auch mit drei virtuelle Maschinen für die Floatzugriffe, sehr gut und erreicht Werte, die Virtuozzo unter Verwendung von einer virtuelle Maschine verzeichnet hat.

	einzelne VM	**VM 1**	**VM 2**	**Average**	**Summe**
L1 AVERAGE	18624,26	18316,11	18332,42	18324,27	36648,54
L2 AVERAGE	6330,84	6333,56	6113,43	6223,5	12447
RAM AVERAGE	2365,31	1752,15	1763,41	1757,78	3515,57

Tabelle 6.14: RAM-paral-INTWrite 2 VMs (Angaben in MB/Sekunden)

	einzelne VM	VM 1	VM 2	VM 3	Average	Summe
L1 AVERAGE	18624,26	9160,29	18175,21	12691,30	13342,27	40026,82
L2 AVERAGE	6330,84	3048,01	6274,81	3068,29	4130,37	12391,11
RAM AVERAGE	2365,31	1005,09	1935,15	1017,36	1319,20	3957,61

Tabelle 6.15: RAM-paral-INTWrite 3 VMs (Angaben in MB/Sekunden)

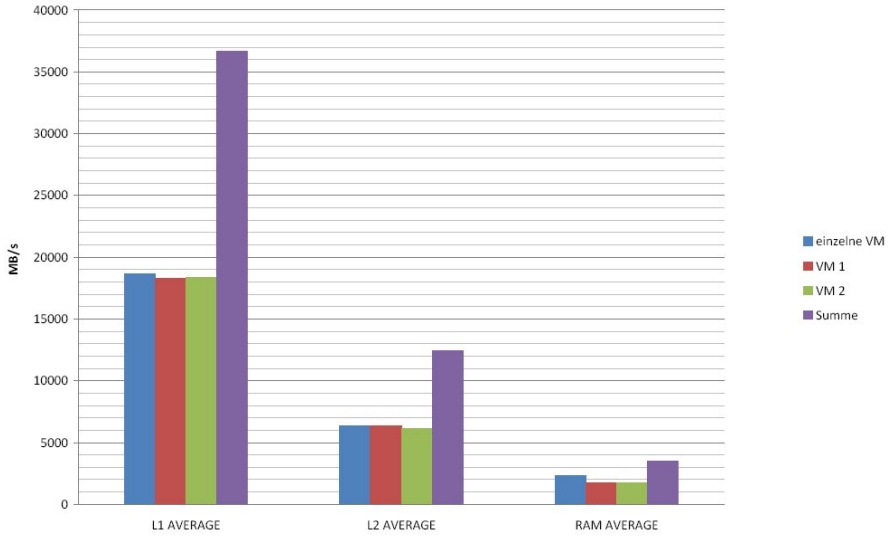

Abbildung 6.11: RAM Parallel-INT Write-2 VMs

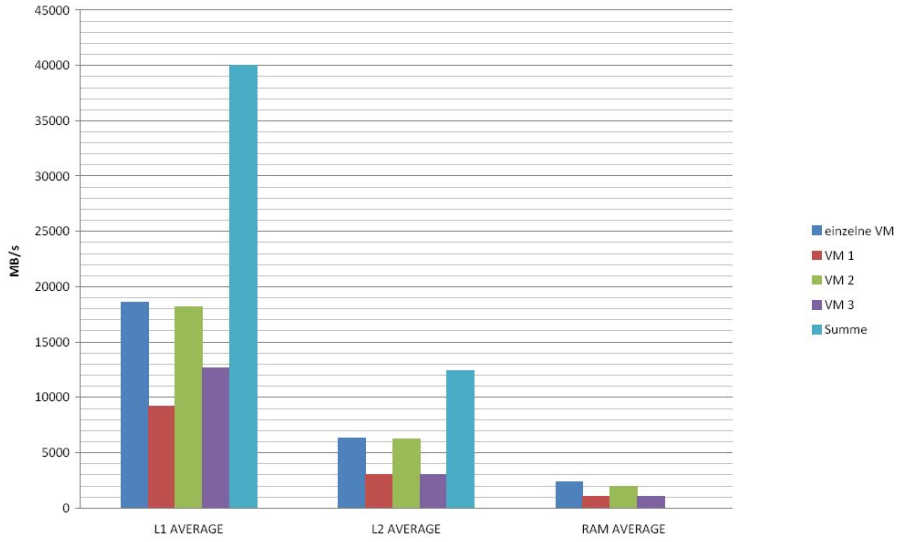

Abbildung 6.12: RAM Parallel-INT Write-3 VMs

	einzelne VM	VM 1	VM 2	Average	Summe
L1 AVERAGE	18901,17	18712,05	18630,55	18671,30	37342,60
L2 AVERAGE	7130,69	7200,29	6973,75	7087,02	14174,04
RAM AVERAGE	3081,69	2795,85	2693,36	2744,61	5489,22

Tabelle 6.16: RAM-paral-INTRead 2 VMs (Angaben in MB/Sekunden)

	einzelne VM	VM 1	VM 2	VM 3	Average	Summe
L1 AVERAGE	18901,17	9160,29	18175,21	12691,30	13342,27	40026,82
L2 AVERAGE	7130,69	3359,6	6625,14	3325,36	4436,70	13310,11
RAM AVERAGE	3081,69	1238,41	2274,73	1276,26	1596,47	4789,42

Tabelle 6.17: RAM-paral-INTRead 3 VMs (Angaben in MB/Sekunden)

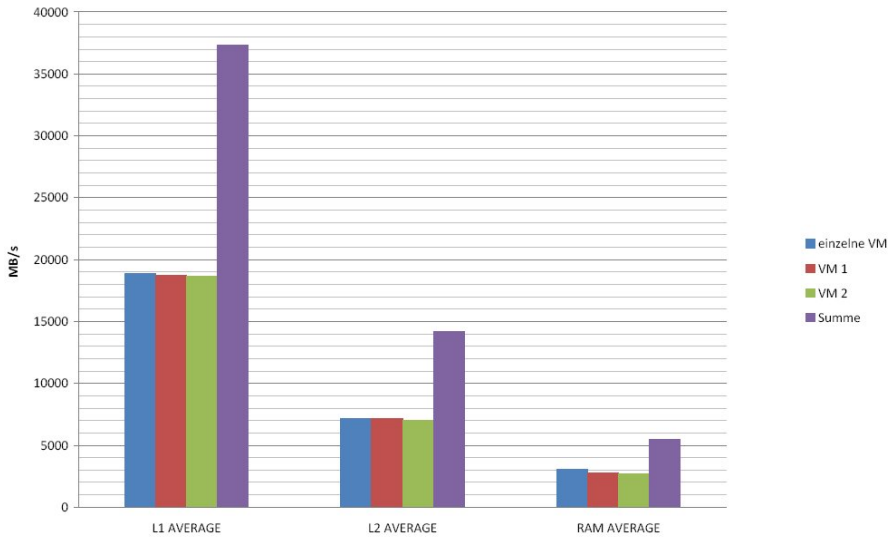

Abbildung 6.13: RAM Parallel-INT Read-2 VMs

	einzelne VM	VM 1	VM 2	Average	Summe
L1 AVERAGE	17343,01	19496,31	19468,50	19482,40	38964,81
L2 AVERAGE	7621,95	7526,72	7042,98	7284,85	14569,71
RAM AVERAGE	2819,39	2000,82	1971,92	1986,37	3972,75

Tabelle 6.18: RAM-paral-FLOATWrite 2 VMs (Angaben in MB/Sekunden)

	einzelne VM	VM 1	VM 2	VM 3	Average	Summe
L1 AVERAGE	17343,01	19808,86	19320,38	18395,87	19175,03	57525,11
L2 AVERAGE	7621,95	7489,36	6769,23	7379,17	7212,59	21637,78
RAM AVERAGE	2819,39	1833,82	2166,44	1814,35	1938,20	5814,62

Tabelle 6.19: RAM-paral-FLOATWrite 3 VMs (Angaben in MB/Sekunden)

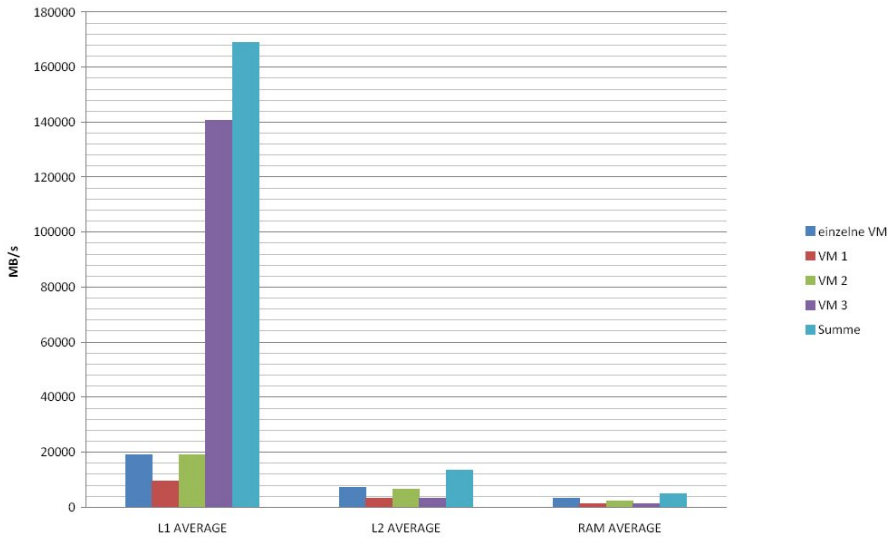

Abbildung 6.14: RAM Parallel-INT Read-3 VMs

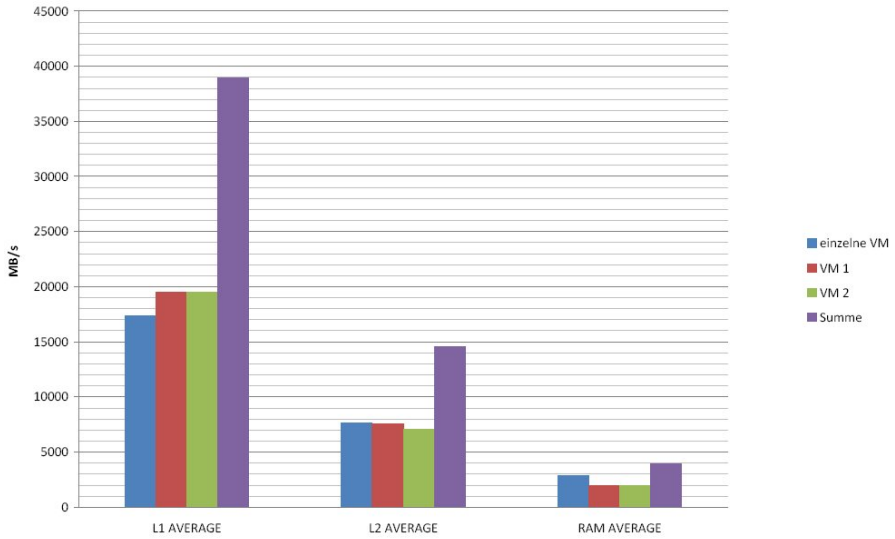

Abbildung 6.15: RAM Parallel-FLOATWrite-2 VMs

	einzelne VM	**VM 1**	**VM 2**	**Average**	**Summe**
L1 AVERAGE	28243,64	27637,43	27865,24	27751,33	55502,67
L2 AVERAGE	7345,7	7269,17	6686,33	6977,75	13955,5
RAM AVERAGE	3313,8	2988,53	2712,56	2850,54	5701,09

Tabelle 6.20: RAM-paral-FLOATRead 2 VMs (Angaben in MB/Sekunden)

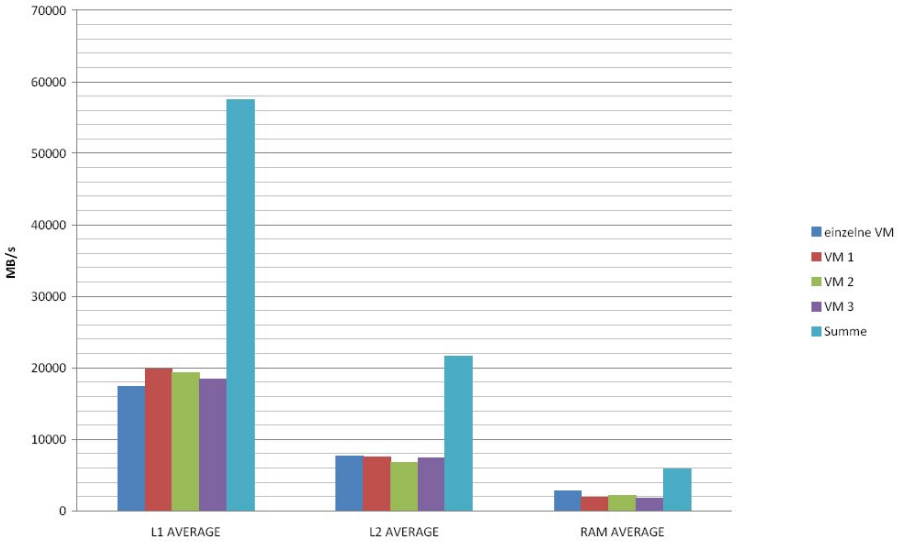

Abbildung 6.16: RAM Parallel-FLOATWrite-3 VMs

	einzelne VM	**VM 1**	**VM 2**	**VM 3**	**Average**	**Summe**
L1 AVERAGE	28243,64	27875,96	27498,97	26982,27	27452,40	82357,21
L2 AVERAGE	7345,7	7301,14	6631,58	7061,94	6998,22	20994,67
RAM AVERAGE	3313,8	2916,50	2513,32	2860,51	2763,44	8290,34

Tabelle 6.21: RAM-paral-FLOATRead 3 VMs (Angaben in MB/Sekunden)

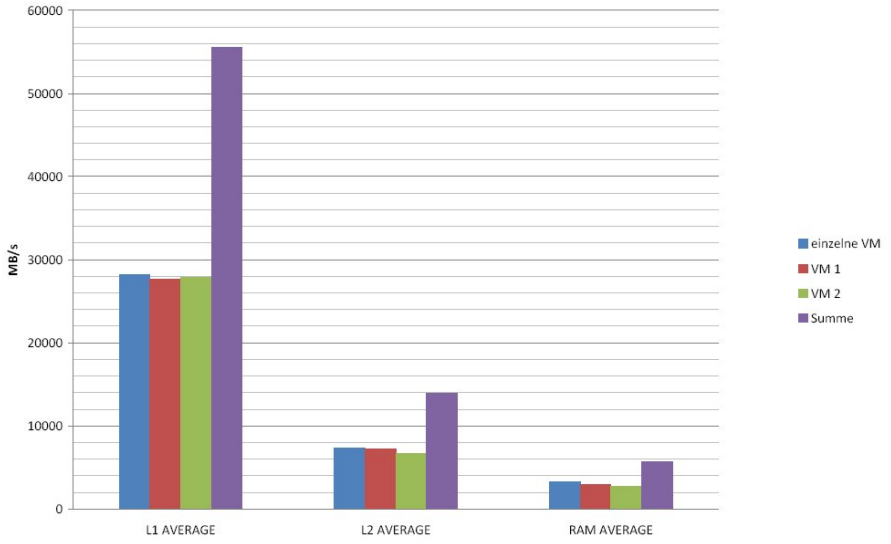

Abbildung 6.17: RAM Parallel-FLOATRead 2 VMs

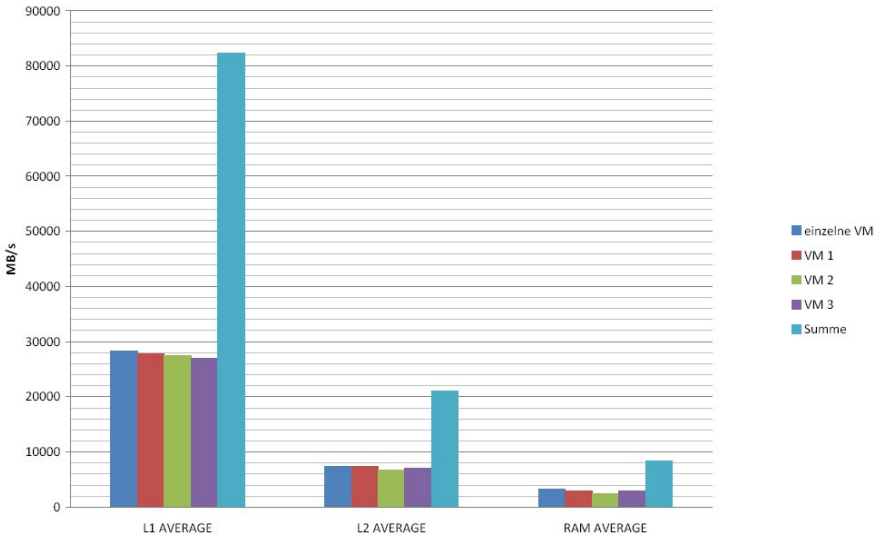

Abbildung 6.18: RAM Parallel-FLOATRead 3 VMs

7 Interpretation

Nach der Durchführung der Messungen ist es an der Zeit die zwei Fragen zu beantworten, die bei dieser Arbeit der Ausgangspunkt waren. Ist die Gesamtperformance einer virtuellen Maschine unter Verwendung von OpenVZ und Virtuozzo gegenüber eines nativen Servers genau so effizient? Die nächste Frage ist ob, die Gesamtperformance von mehr als eine gleichzeitig laufenden virtuellen Maschine im Vergleich steigt oder sinkt.

7.1 Messungen (synthetisch)

Die synthetische CPU-Messungen haben kleine Unterschiede gezeigt. Hier verlieren die native Systeme ca. 3 Prozent gegenüber den virtuellen Maschinen. Eine virtuelle Maschine ist um 3 Prozent besser als ein nativer Server. Die interessanteste Beobachtung ist, dass Virtuozzo einen Performanceverlust von 50 prozent im Vergleich zu OpenVZ aufweisst. Alle Berechnungen beim OpenVZ werden halb so schnell durchgeführt. Bei den Disk-Messungen schneiden die native Maschinen generell besser ab. Die OpenVZ Messungen kommen aber sehr nah an den nativen Maschinen, weil beim OpenVZ die Gastsysteme direkt auf den Festplattenbereich zugreifen, der vom Host-Kernel eingebunden und verwaltet wird. Beim Virtuozzo ist dies nicht der Fall. Hier hat die virtuelle Maschine beim sequenziellen Schreiben sowohl unter Windows 32 als auch unter Windows 64-Bit eine schlechtere Transferrate. Die Netzperformance ist beim OpenVZ und Virtuozzo im Vergleich zu den nativen Server vergleichbar. Das bedeutet, dass OpenVZ und Virtuozzo genau so viele Lese und Schreibzugriffe in eine Sekunde operieren können und in der Lage sind gleich gute Ergebnisse zu erziehen wie ein nativer Server. Die virtuelle Maschinen verlieren unter Virtuozzo ca. 7 Prozent beim Lesen aber sie können beim Schreiben mithalten. Die RAM-Messungen waren vier an der Zahl. Die INTmark-Messungen zeigen keine Unterschiede beim L1, L2-Cache und RAM zwischen virtuellen Maschinen und nativen Server. Der Vorteil liegt hier bei den Windows-Systemen für 32-Bit gegenüber Ubuntu 32-Bit zu sein. Der Vorsprung des Ubuntu Server 64-Bit zeigt, dass das Benchmark stark von den zusätzlichen CPU-Registern profitiert. Es werden doppelt so viele Variablen im Kern behalten, anstatt diese aus dem langsameren Zwischenspeicher zu holen. Dadurch werden die synthetischen Berechnungen deutlich beschleunigt. Beim Schreiben für die Floatoperationen gilt eigentlich das gleiche wie für die Integeroperationen. Deide Linux-Maschinen operieren fast fünfzig Prozent schneller als alle andere Distributionen beim Level 1. Die Zugriffe beim Level 2 Cache und RAM sehen anders aus. Hier sind alle virtuelle Maschinen im Vergleich zu den nativen um 60 Prozent für das Level 2 Cache und um 50 Prozent für RAM langsammer. Beim Lesen für die Floatoperationen werden ähnliche Ergebnisse erziehlt für das Level 1 Cache mit einer Ausnahme der Windows Maschinen. Die Windows Maschinen haben eine schlechtere Performance gegenüber Linux. Die Zugriffe unter Level 2 Cache und RAM zeigen auch einen Vorteil von 50 Prozent der nativen Server gegenüber der virtuellen Maschinen. Auch die Integer-und Floatarithmetikoperationen (zum Beispiel die Operation Add) werden unter Ubuntu64-Bit schneller verarbeitet, aber nicht schneller als die entsprechenden nativen Systemen. Nach Beobachtungen der Durchschnittwerte für

Integer-und Floatoperationen gilt allgemein, dass die virtuellen Maschinen unter Linux generell schneller sind. Die Windows nativen Maschinen haben einen geringfügig schwächere Durchschnittsperformance im Vergleich zu den virtuellen Windows Maschinen. Auf die Ausgangsfrage zurückkommend, ob die virtuelle Maschinen unter Virtuozzo und OpenVZ genau so Effizient sind, kann man mit Sicherheit sagen, dass es wirklich so ist. Alle virtuellen Maschinen waren sowohl unter OpenVZ als auch unter Virtuozzo genau so schnell oder fast so schnell wie ein nativer Server. Die einzige Unterschiede der Beobachtungen die erzielt wurden spielten sich nur noch zwischen den unterschiedlichen Distributionen. Generell gilt, dass OpenVZ eine bessere Effizienz als Virtuozzo aufweisen kann. Bei der CPU-Messungen sogar um 50 Prozent besser.

7.2 Messsungen (parallel)

Die parallele Messungen für Disk und Netz haben die gute Skalierbarkeit von Virtuozzo gezeigt. Hier bewegt sich Virtuozzo auf dem Niveau einer einzelner Maschine. Beim Disk nimmt der Durchsatz bei Virtuozzo bei steigender Anzahl der Clients zu. Im Schnitt errechnen wir einen Geschwindigkeitsvorteil von ca. 3 Prozent. Je mehr virtuelle Maschinen laufen, desto höher wird die Transferrate. Sogar beim Disk unter Last verursacht Virtuozzo wenig Overhead. Die Lastverteilung ist sehr fair und die Messung hat einen hohen Durchschnittswert gezeigt. Die Netz parallele Zugriffe sowohl für zwei als auch für drei virtuelle Maschinen im Vergleich mit den einzelnen Messungen einer virtuelle Maschine sind sogar insgesamt einige Prozentpunkte besser. Die virtuelle Maschine beim Netz unter Last schafft sogar Werte, die sehr nah an die einzelne Messung kommen. Wir können auf keinen Fall über einen Netzdurchsatzeinbruch sprechen. Das war aber auch zu erwarten ist, da Virtuozzo auf eine hohe Containerdichte optimiert ist. Bei der VM zu VM Messung sehen wir aber schlechte Transferraten des Virtualisierers statt einer erwartender besserer Leistung zwischen den zwei virtuellen Maschinen. Der Grund könnte die Isolierung der Container voneinander sein. Die Container besitzen keine Kenntnisse darüber, ob sie allein oder mit vielen Maschinen auf einem Hostsystem existieren. Die nächste parallele Messungen betraffen die CPU und den Arbeitspeicher. Die CPU-Zuteilung bei zwei virtuellen Maschinen ist wie bei einer einzelnen Maschine. Das bedeutet, dass beide virtuelle Maschinen eine sehr gute Performance aufweisen, was die CPU betrifft. Bei 3 VMs tritt ein recht interessanter Effekt zum Vorschein. Die erste und die dritte virtuelle Maschine sind um Faktor zwei langsamer als die zweite VM. Das hat damit zu tun, dass der Scheduler bei Virtuozzo auf 2 Ebenen arbeitet: zunächst wird entschieden, welche VM den CPU Takt erhält. Dabei werden die VMs in der Reihenfolge in der sie gestartet wurden an die CPU-Kerne zugewiesen: VM1 und 3 an den 1. Kern, VM 2 an den 2. Erst dann entscheidet der standardmäßige Windows-Planer, welcher Prozess in ausgewähltem VE den CPU-Takt bekommen soll. Dabei werden wie immer die Standard-Prioritäten von Prozessen benutzt. Die parallele RAM-Messungen zeigten beim Interger Lese-und Schreibzugriffe die gleichen Performance-Eigenschaften sowohl für zwei als auch für drei virtuelle Maschinen. Bei 2 VMs ist der Zugriff des Arbeitsspeichers wie bei einer einzelne Maschine. Das heißt, dass beide virtuelle Maschinen eine sehr gute Performance was L1, L2-Cache und RAM betrifft aufweisen. Bei 3 VMs tritt der gleiche Effekt zum Vorschein wie bei den CPU-Messungen. Die erste und die dritte virtuelle Maschine sind um Faktor zwei langsamer als die zweite VM. Hier kommt beim Virtuozzo wieder der -fair share Scheduling- zum Einsatz. Das hat wieder wie bei der CPU damit zu tun, dass der Scheduler

bei Virtuozzo auf 2 Ebenen arbeitet: zunächst wird entschieden, welches VM auf den RAM zugreift. Dabei werden die VMs in der Reihenfolge in der sie gestartet wurden auf den RAM zugreifen. Die parallele RAM-Messungen beim Float Lese-und Schreibzugriffe zeigen aber ein einheitlicheres Bild. Hier spielt es keine Rolle, ob zwei oder drei virtuelle Maschinen auf den L1, L2-Cache oder RAM zugreifen. In diesem Fall beweist Virtuozzo sein Vorteil bei höherer VM-Dichte. Die Skalierung der virtuelle Maschinen funktioniert in diesem Fall sehr gut. Mit Ausnahme der CPU und die Integer-RAM Messungen skaliert Virtuozzo sehr gut, wie unsere Ergebnissen gezeigt haben. Ansonsten gilt: Virtuozzo ist sehr schnell, da es einen gemeinsamen Kernel sowohl für Host als auch für alle Gäste benutzt. Es werden keine zusätzlichen Systemaufrufe zwischen Excess-Layern erstellt. Sein Vorteil liegt bei höherer VM-Dichte.

8 Fazit

Es hat sich ergeben, dass OpenVZ viele Vorteile in den Virtualisierungsverfahren für eine Linux-Serverumgebung bieten kann. So bietet OpenVZ eine optimale Ausnutzung von Ressourcen mit nahezu keinem Verlust so wie die Messungen gezeigt haben. Dem gegenüber steht jedoch die fehlende Möglichkeit für virtuelle Maschinen eigene Betriebssystem-Kernel zu verwenden. Dies beschränkt die Nutzung von OpenVZ ausschließlich auf eine Linux-Serverumgebung mit einem einzigen Kernel als Verwaltungsinstanz. Des weiteren ist es nicht möglich einem Gastsystem physikalische Ressourcen, abgesehen von Netzkarten, exklusiv zuzuweisen. Als enormen Vorteil ist jedoch die Möglichkeit einer simplen, einfachen und schnellen Installation von Gästen zu erwähnen, die es erlaubt innerhalb von wenigen Minuten einen neuen Gastsystem zu erstellen. Somit eignet sich OpenVZ speziell für den Einsatz in Umgebungen mit ähnlicher bzw. identischer Struktur der Gäste, wie beispielsweise eine Webserver-Farm. Virtuozzo für Windows ist die ideale Virtualisierungslösung, wenn mehrere virtuelle Server unter Windows benötigt werden. Neue Server können in weniger als drei Minuten erstellt werden, ohne Festplattenplatz zu verbrauchen. Dies macht Virtuozzo auch in Test- und Schulungsumgebungen sehr interessant. Der Ressourcenbedarf pro virtuelle Maschine ist um ein vielfaches geringer als der einer virtuellen Maschine bei Vollvirtualisierung. Die Performance, insbesondere bei Platten- und Netz-I/O, ist deutlich höher. Das macht Virtuozzo häufig zu einer besseren Alternative im Vergleich zu einer Vollvirtualisierung. Die Verwaltungstools via Management Konsole und Web-Interface sind sehr gut durchdacht und leicht und übersichtlich in der Bedienung. Die Funktionalität ist sowohl für Hoster, als auch für Intranet-Serveradministratoren ausgelegt. Das Virtuozzo-Filesystem sorgt dafür, dass Betriebssystem und Anwendungen nur einmal pro Hostsystem installiert werden müssen und dann von den VPSen mitbenutzt werden können. Als Schwachstellen sind zu nennen, dass zum einen das Erstellen von Templates für Anwendungen sehr aufwendig und kompliziert ist, zum anderen ist es nicht möglich in verschiedenen VPSen unterschiedliche Releasestände von Windows Server zu betreiben. Mit der Linux-Version von Virtuozzo (OpenVZ) ist letzteres problemlos möglich.

Abbildungsverzeichnis

Literaturverzeichnis

[ala] ALASIR. http://www.alasir.com/software/ramspeed/index.html.

[Bau07] BAUER, TOBIAS: *OpenVZ: Das kleine Handbuch*. Books on Demand GmbH, Norderstedt, 2007.

[Fuj09] FUJITSU: *Performance Report Hyper-V*, 2009. https://sp.ts.fujitsu.com/dmsp/docs/wp-pr-hyper-v-de.pdf.

[iom] IOMETER.ORG. http://iometer.cvs.sourceforge.net/*checkout*/iometer/iometer/Docs/Iometer.pdf.

[net] NETLIB.ORG. http://www.netlib.org/linpack/.

[ope] OPENVZ.ORG. http://openvz.org/download/Kernel.

[PG74] POPEK, GERALD J. und ROBERT P. GOLDBERG: *Formal Requirements for Virtualizable Third Generation Architectures*, 1974. http://portal.acm.org/citation.cfm?id=808061&dl=GUIDE&coll=GUIDE&CFID=57773902&CFTOKEN=60711586.

[PH09] PARALLELS HOLDINGS, LTD: *Parallels Virtuozzo Containers for Windows,Version 4.5*, 2009. http://download.swsoft.com/pvc/45/win/docs/en/VzWindowsUG.pdf.

[Sol] SOLTESZ, PÖTZL, FIUCZYNSKI BAVIER PETERSON: *Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors*. http://www.cs.princeton.edu/~mef/research/vserver/paper.pdf.

[SWs] SWSOFT. http://download.openvz.org/doc/OpenVZ-Users-Guide.pdf).

[Tho08] THORNS, FABIAN: *Das Virtualisierungs-Buch*. C und l Computer- U. Literaturverlag, September 2008.